Watchpoint: Freehand Pointing with a Smartwatch in a Ubiquitous Display Environment

Keiko Katsuragawa[*], Krzysztof Pietroszek[†], James R. Wallace[‡*] and Edward Lank[*]

[*]Cheriton School of Computer Science, University of Waterloo
Waterloo, Ontario, Canada
{kkatsura, lank}@uwaterloo.ca

[†]California State University Monterey Bay
Seaside, California, USA
kpietroszek@csmb.edu

[‡]School of Public Health and Health Systems, University of Waterloo
Waterloo, Ontario, Canada
james.wallace@uwaterloo.ca

ABSTRACT
We describe the design and evaluation of a freehand, smartwatch-based, mid-air pointing and clicking interaction technique, called Watchpoint. Watchpoint enables a user to point at a target on a nearby large display by moving their arm. It also enables target selection through a wrist rotation gesture. We validate the use of Watchpoint by comparing its performance with two existing techniques: Myopoint, which uses a specialized forearm mounted motion sensor, and a camera-based (Vicon) motion capture system. We show that Watchpoint is statistically comparable in speed and error rate to both systems and, in fact, outperforms in terms of error rate for small (high Fitts's ID) targets. Our work demonstrates that a commodity smartwatch can serve as an effective pointing device in ubiquitous display environments.

CCS Concepts
• **Human-centered computing→Interaction devices**.

Keywords
wearable, pointing, large displays, smartwatch

1. INTRODUCTION
The ability to point and gesture at nearby objects is a natural and intuitive method of interaction, widely used by humans as they communicate with one another. Alongside its use in interpersonal communication, freehand interaction, i.e. pointing and gesturing, has been frequently explored as a modality for interacting with ubiquitous computing objects [2] [9] [19], a result of its ability to communicate target and actions of interest. Given the widespread availability of computation – both in displays and in augmented smart devices – an interaction modality that communicates both target and intent remains an important goal of interaction research.

While pointing and gesturing seems a desirable interaction modality for ubiquitous computing environments, the technical ability to track hand movements has proven challenging, and no 'gold standard' solution has yet to be established. For example, vision-based systems have shown potential by tracking users in front of large displays, but can suffer from poor lighting and occlusion [6][17]. Vision-based systems also require augmentation of the environment with specialized sensing to support interaction, adding cost to their deployment. Specialized devices, such as Nintendo's Wiimote, or Thalymic Lab's Myo, can facilitate interactions with nearby devices but require users to purchase and setup those devices prior to interaction.

AVI'16, June 07–10, 2016, Bari, Italy.
 ACM 978-1-4503-4131-8/16/06…$15.00.
DOI: http://dx.doi.org/10.1145/2909132.2909263

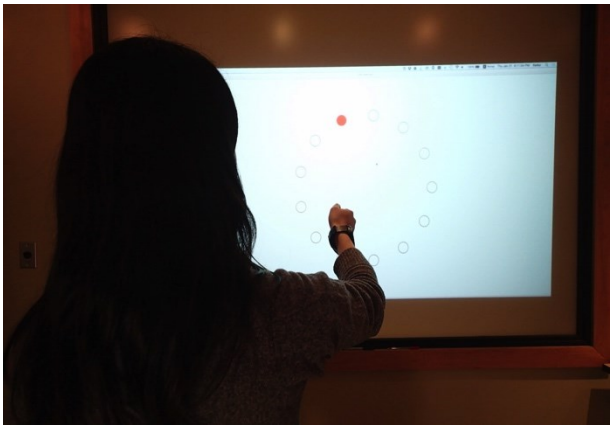

Figure 1. Watchpoint enables freehand interaction with nearby ubiquitous displays via a user's smartwatch.

Alongside specialized devices and augmented environments, generic personal devices such as smartphones and smartwatches represent platforms of convenience to access computing in everyday contexts [13]. Specifically within the space of pointing and gestural interaction, smartwatches represent a particularly accessible companion for sensing user input. They are 'always on' and ready, are worn on the wrist, are able to communicate wirelessly with nearby devices, and contain an evolving set of sensors (cameras, inertial measurement units) that sense device movement. Finally, because the cost of these devices is relatively low and because a device is uniquely assigned to a single user, smartwatches overcome both the cost of instrumenting an environment and the challenge of identifying a specific user of interest from the surrounding context.

While it may be desirable to solve all challenges associated with ubiquitous input, this paper focuses specifically on the paradigm of pointing and selecting in ubiquitous display environments. We restrict ourselves to this paradigm both because it is a natural extension of the familiar components of WIMP-interaction into the ubiquitous context and because pointing and selecting are foundational activities in the articulation of intent in communicative interactions. In other words, pointing and clicking are useful interactions in any reactive environment because they

leverage both a user's familiarity with computer interaction and are a natural and demonstrative way of selecting the desired target of interaction and activating its available services.

In this paper, we present the design and validation of Watchpoint, a system that supports pointing and clicking on a nearby large display via a smartwatch (Figure 1). Through early pilot studies we identify challenges associated with target acquisition and design cursor stability functions in both moving and clicking to support rapid interaction. We also show through a controlled Fitts's Law study that our prototype provides performance comparable to other freehand pointing techniques for pointing time. Finally, we present optimizations for target selection that allows Watchpoint to outperform both Vision-based Vicon-system [18] and specialized device-based Myo freehand pointing input [8] in terms of error rate for small targets. Our work demonstrates that a commodity smartwatch can serve as an interaction device for pointing in ubiquitous display environments.

This paper is organized as follows. First, we explore related work in pointing and target selection in ubiquitous display environments. Next, we describe the interaction supported by Watchpoint, including optimizations that we identified through pilot testing. Finally, we present our summative evaluation of Watchpoint and discuss the implications of our work.

2. RELATED WORK

Given the ubiquity of displays in our everyday lives, the idea of supporting interaction with these displays is of interest [2] [9] [19]. While some displays that we encounter naturally support interaction with attached input device such as keyboards and mice or touchscreens, many displays only support more distant interactions or do not support any interactions because of the cost and the hardware restriction. The compelling rationale around supporting freehand gestural input to ubiquitous displays is that interaction can be supported in a way that places few restrictions on the placement of the device or the availability of user-centric specialized hardware. In other words, freehand gestural input enables, from an end-user perspective, a true 'walk-up-and-use' experience with ubiquitous devices. However, the desirability of freehand interaction is offset by the requirement of user movement tracking. The lack of reliable tracking results frequently in both high-cost and a high error rate.

In this section we explore, in turn, the use of handheld devices and special purpose devices to interact with ubiquitous displays, the efficacy of freehand pointing techniques to interact with ubiquitous displays, and smartwatch-based multi-display interactions.

2.1 Distant interaction with personal devices

Since the mid-1990s, the use of handheld devices to interact with external, ubiquitous displays has been an ever-expanding focus of research in human-computer interaction [2][13][15][16][19]. The Pebbles project leveraged early handheld computers to explore and provide input to external computer displays [13]. Follow-on work included systems such as peephole [20] and, more recently, systems such as Smartcasting [15] and Tiltcasting [16]. The benefit of using general purpose handheld computers to interact with ubiquitous displays is that the nature of these devices increases the likelihood that they are available: Users are more likely to have with them a general purpose device than some form of specialized hardware which is only useful in a specific computational context. Drawbacks of using a general purpose computational platform includes the fact using a device designed for one purpose for another to which it may be poorly suited. Another drawback is that, even if a handheld computer is carried by the user, the act of pulling it out of one's pocket and turning it on does introduce a lack of fluidity into interaction.

To enhance the speed and accuracy of pointing, one can always design hardware that is optimized for a specific task. One example of a special purpose pointing device is the Wii-remote, a game controller that includes both a handheld input device containing an inertial measurement unit (IMU) and camera and emitter systems that support more accurate tracking [19]. More recently, developers of devices such as the Myo have exploited novel sensing [9], potentially improving gesture input and recognition.

2.2 Vision-based freehand interaction

While one option to support pointing is to use a personal device or a specialized device, the ideal point-and-click interaction in public environments is simply to point at the display. The most obvious way to support freehand input to ubiquitous displays is to track user input as users interact in surroundings that contain ubiquitous displays. Two common approaches to tracking as a mechanism for computer input are, first, to leverage high-end accurate movement tracking systems. These highly precise motion capture systems represent a gold standard to support gestural input on large displays.

While precise, high end trackers are costly for end users, and this high cost has given rise to a set of commodity tracking systems including devices such as the Microsoft Kinect and the Leap Motion hand gesture sensor. These lower-end systems compromise on either precision or range, but do support many useful gestural interactions; if neither the cost nor the potential for theft of vandalism is a concern, these systems can be deployed cheaply and rapidly in many environments.

Regardless of whether one uses a highly precise and costly motion capture system or one uses less costly commodity devices, there exist other drawbacks to computer-vision-based gestural tracking systems, particularly when deployed in real world contexts. For example, motion capture systems like the Vicon typically require users to position tracking points on their limbs to aid more accurate identification; the use of these tracking points is a barrier to public deployments because users will not, typically, be equipped with augmentations to aid in tracking in their everyday activities. Even if one neglects the complexity of augmenting users, occlusion and distractors (e.g., non-users within the tracking region) frequently cause problems for these systems. Furthermore, even with problems of augmentation and occlusion solved, range issues continue to persist, with tracking systems frequently being restricted to a relatively small region. Finally, even if all other problems are solved and one assumes fully reliable tracking, it is sometimes desirable to preserve the identity of a user from one session to the next, and this is challenging for motion capture systems as they are designed to capture movement and not to recognize faces or other salient features of an individual.

2.3 Smartwatch

As an alternative device for distant interaction, we were motivated to explore the smartwatch as a platform of convenience based on four advantages over handheld computers such as smartphones [2], specialized devices such as the Myo [9], and camera-based techniques such as Vicon-based systems [19]. First, a smartwatch is always on and always ready to be used, and, unlike a smartphone does not need to be removed from a user's pocket and unlocked. Second, a smartwatch is cost effective (versus, e.g. Vicon), and, unlike a Myo armband or Tobii eyetracking glasses,

a smartwatch is intended to be used on a daily basis to support a number of tasks, and not as a specialized input device. Third, use of a smartwatch obviates the need for other sensing hardware, as opposed to computer vision based systems where each display must be configured with cameras to detect gestures, and the effectiveness of the technique relies upon proper lighting and limited occlusion. Finally, because a smartwatch is designed as a personal device and has a unique ID, it provides a means of identifying a user during a sequence of ubiquitous interactions – a noted limitation of camera-based techniques [8].

Given these many benefits, the question remains whether a current generation, off-the-shelf smartwatch can match the pointing performance afforded by other technologies. With this question in mind, we created a prototype smartwatch-based ubiquitous freehand pointing technique, called Watchpoint.

3. WATCHPOINT DESIGN

Development of the Watchpoint prototype involved capturing motion data from the smartwatch hardware, interpretation of captured motion data, and relaying pointer events to an associated ubiquitous display. This section presents the design of the overall system, including hardware, interaction, and calibration or cursor to display location.

3.1 Hardware & Sensors

Watchpoint hardware consists of an Android smartwatch, a smartphone, and a personal computer. The smartwatch detects a user's hand movements using its built-in gravity and rotation vector sensors. In our current prototype, the watch then relays raw sensor input data to a connected smartphone via Bluetooth. The smartphone then forwards movement data to a personal computer via USB. The computer interprets sensor input into on-screen mouse movements and depicts interaction on an attached display, typically a data projector projecting on a rear-projection display to mimic a ubiquitous public display. Note that, within the hardware setup, the Android smartphone only exists due to programming constraints in the Android Wear ecosystem; we will be able to eliminate the smartphone from the data input stream once this programming constraint is relaxed.

Watchpoint uses an Android smartwatch's gravity and orientation sensor. One requirement of the smartwatch is that it must incorporate a 'nine-axis' inertial measurement unit (IMU), i.e. it must incorporate an IMU that includes an accelerometer (movement and gravity), a gyroscope (movement and orientation) and a magnetometer (movement and orientation) to provide sufficient input accuracy. The specific hardware device we use in our implementation is an LG G Watch R connected via Bluetooth to a Nexus brand smartphone device, but any smartwatch containing a 9-axis IMU will support Watchpoint interaction.

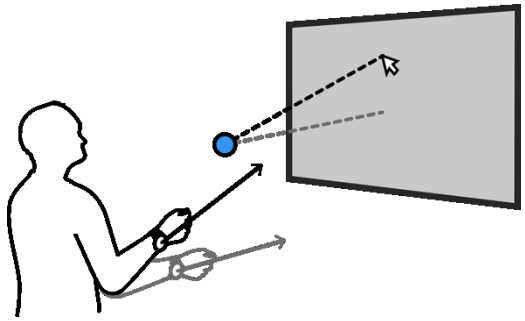

Figure 2. Fixed-point ray-casting.

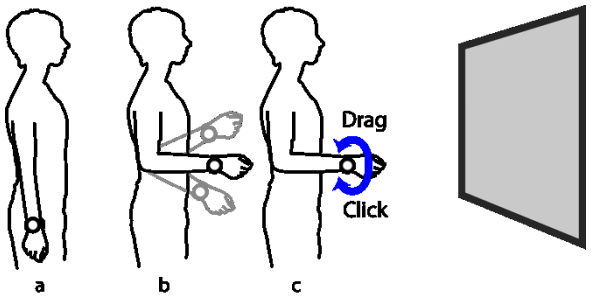

Figure 3. From the Inactive mode (a), the user can activate Watchpoint by raising their arm (b,c). When active, the user can control the cursor and pointer (b) and drag and click(c) on the nearby display.

We currently sample both orientation and gravity at a rate of 50Hz with a latency of less than 100ms. This frequency is high enough to track hand movement smoothly and also gives a sufficient data stream to support smoothing, an issue discussed later in this section.

3.2 Controlling the Cursor

3.2.1 Activating and Deactivating Cursor Movement

To support cursor input, Watchpoint leverages the ray-casting metaphor [12], where a user controls a cursor on a nearby display by pointing at it, and the position of the cursor on the display is determined by the intersection of a 'cast' ray intersecting with the screen. Watchpoint employs fixed-point ray-casting, where the origin of the ray relative to the display is fixed (Figure 2) regardless of the user's position. This process simplifies interaction because the user does not need to be tracked; instead, cursor movement on-screen maps to device movement, but the actual position of the on-screen cursor is a result of relative movement of the fixed origin ray. Fixed-origin ray-casting has been shown to be an effect method of supporting ray-casting with a personal device [15].

Watchpoint's interaction model (Figure 3 and 4) consists of four states: Inactive, Tracking, Clicked, and Mouse down. The cursor is initially placed in an inactive state with the user's arm resting at their side (Figure 3a). When the user raises his or her arm, the system switches into the Tracking state (Figure 3b), and a cursor appears on the external display allowing the user to move a cursor around the screen. From the Tracking state, the user can invoke a click event by rotating the wrist outwards (watch face down), and a drag event by rotating their wrist inwards (watch face up), (Figure 3c). In each of the Tracking, the Clicked, and the Mouse down states a user can manipulate the cursor position by moving their arms up/down and left/right.

It is common to represent WIMP-based interaction using a 3-state model attributable [4]. In this work, we separate the drag state into two different states, click and drag. The rationale for the separation between click and drag states is as follows. We had originally decided to include only three states (Inactive, Tracking and Mouse down), but pilot testing revealed that it was difficult for users to return to the Tracking state (Mouse up) from a Mouse down state without dragging. Therefore, we decided to separate the Click and Drag functionality into two separate states.

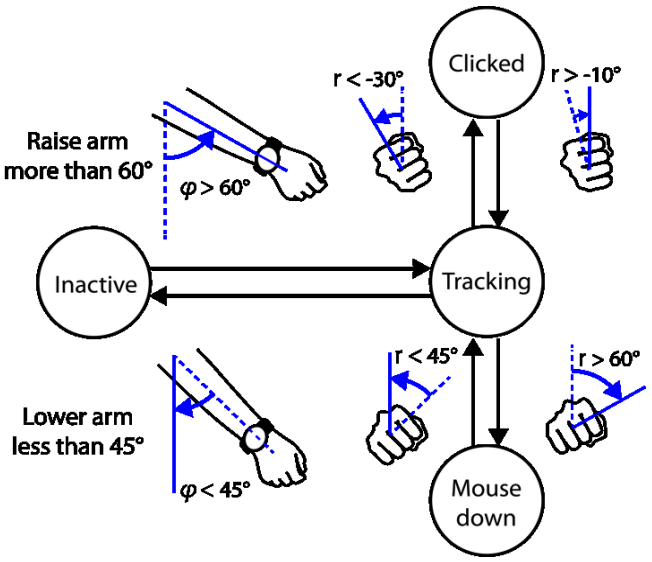

Figure4. This figure summarizes empirically determined thresholds used to support Watchpoints transitions from Inactive to Tracking to Clicked and Mouse down states.

A challenge identified in early pilot testing was avoiding unintended clicks. As shown in Figure 4, the Tracking state is represented by the user holding her arm flexed with the watch face vertical. To avoid unintentional multiple clicks by the wrist, Watchpoint uses a 300ms threshold for continuous clicks [9].

Another issue is unintentional transitions between the Mouse down and Tracking states. To avoid these transitions, we use an asymmetric wrist angle for activation and deactivation. To move from the Tracking state to the Mouse down state, one must rotate the wrist 60º, and then back to less than 45º to return to the Tracking state (Figure 4). Similarly, for Clicked, Watchpoint uses an activation angel of -30º and a deactivation angel of -10º. We identified problems of accidental activation/deactivation during pilot testing, empirically set thresholds for activation and deactivation. These thresholds seem to work well: they are well within the normal range of movement of a user's wrist rotation range, but are sufficiently separated that the distinction between activation and deactivation is non-negligible.

3.2.2 Cursor Movement

As noted earlier, Watchpoint uses a smartwatch's IMU to detect movement. X-axis and Y-axis movement are detected in two different ways. For x-axis movement, the rotation sensor is used to detect the angle of a user's forearm in the horizontal plane. The rotation sensor on a 9-axis IMU combines readings from the magnetometer and gyroscope to sense position and movement. As a result, one challenge is to identify an appropriate initial reading for the location of the display.

To map horizontal movement onto x-axis movement on an external display, Watchpoint maps 60º (π/3 radians) of horizontal forearm movement to the range spanning the width of the screen, a movement range calibrated during pilot testing to be sufficiently wide so as to allow precise targeting and sufficiently narrow that users did not need to hyperextend to access the entire screen. Specifically, given a horizontal screen resolution of W and a rotation reading θ_i from the smartwatch, the x-coordinate of the cursor at the given time i, denoted by X_i can be obtained by the following formula:

$$X_i = \frac{W}{2} + \frac{3\theta_i}{\pi} \quad (1)$$

Watchpoint uses Android's gravity sensor to determine the y-coordinate of the cursor and detect wrist rotation. The gravity sensor measures the magnitude of gravity in three dimensions through a fusion of accelerometer and gyroscope data [1]. Thus, data from the gravity sensor is similar to accelerometer readings, but without linear acceleration. When the users rest their arm at their side, the y- and z- components are 0 and the x-component is equal to the force of gravity (9.8m/s^2). When a user raises their arm horizontally to point at a display, keeping the watch face approximately vertical relative to the floor, the y-component will be equal to the gravity and x and z will be zero, as shown in Figure 5.

Note in Figure 5 that the x-component of the smartwatch's coordinate system is aligned with the user's arm. Leveraging the fact that the x-coordinate system of the smartwatch is aligned with the arm, the angle of the user's arm can be calculated using the x-component of gravity as follows:

$$\varphi_i = \cos^{-1}\frac{Gravity_x}{9.8} \quad (2)$$

Using this angle, given a vertical display resolution of h, the y-coordinate of the cursor Y_i at a given time i can be calculated as follows:

$$Y_i = \frac{h}{2} + \varphi_i \frac{3h}{\pi} \quad (3)$$

Finally, we can again leverage the force of gravity to sense clicking and dragging. When the user tilts the watch by rotating the wrist to the outside, the z component of gravity will increase. When the watch tilt angle r_i obtained from the z-component exceeds 30 degrees outwards, Watchpoint triggers the click state; likewise, exceeding 60 degrees inwards fires a change of state to the drag state.

3.2.3 Calibration

One important consideration in Watchpoint is cursor calibration. Specifically, because we are using absolute orientation readings from a 9-axis IMU, we need some way to indicate where the middle of the display is, i.e., to align the center of the display with the user. We use the transition from the Inactive to the Tracking states as a calibration step. When the user lifts his or her arm (Figures 3 and 4) to activate cursor movement, the cursor location is initialized as the mid-point of the width of the display, i.e. x = displaywidth/2. Users would interact with the display, allow their arms to drop to rest, and then begin to interact with the display again. Since our calibration step is very subtle, it can naturally merge into this activating process. Our Inactive to Tracking state

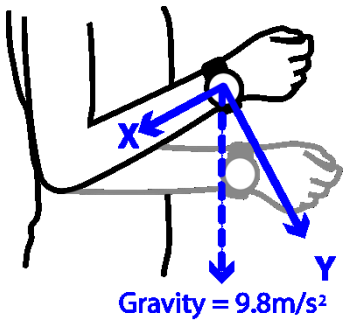

Figure 5. XY-components of gravity.

transition allows us to repeatedly calibrate the overall angle, supporting a reset of interaction and preventing cursor drift without requiring users to spend extra time for calibration. These savings are one of the benefits of absolute position mapping.

3.3 Correction and filtering

While the above design provides sufficient functionality to control a mouse cursor on a nearby display and interact with on-screen data, we identified several areas for improvement during pilot testing. In particular, the mouse cursor often 'jumped' when users rotated their wrist to select a target, and we noticed difficulty in pointing at small targets due to a high level of jitter. This section describes how we addressed these issues in our final design.

3.3.1 Clicking Correction

A drawback of using wrist rotation to trigger selection is that when an individual rotates their arm, the watch may initially detect horizontal movements. While the smartwatch's lateral movement is minimal, we found that in practice the on-screen mouse cursor would move enough to cause 'missed' selections, a phenomenon known as the Heisenberg Effect [3]. We found this issue was particularly frequent when selecting smaller targets.

To address the issue, Watchpoint calculates the direction of wrist rotation at each sensor signal input event and applies a simple threshold. When the speed of the rotation is greater than 5 degrees per second, the direction of the watch rotation is defined as right or left depending on its direction of movement. If the wrist rotation is less than 5 degrees/second, it is defined as neutral. When Watchpoint detects a wrist rotation that would lead to a state change (i.e., from neutral to left/right or vice versa), the mouse cursor's on-screen position is stored in memory for later use during a click event.

When a click event is triggered by the user rotating their wrist to an angle exceeding 30º outward (Click) or 60º inward (Mouse down), Watchpoint checks if a cursor position was saved less than 1 second ago. If this is the case, the cursor's position for the click action is moved back to the saved position. This simple book-keeping operation is highly effective at stabilizing cursor position during clicking – so effective, in fact, that our error rate exceeds Vicon-based tracking of clicking gestures as implemented in past systems [19].

3.3.2 Cursor Acceleration

Freehand pointing at small targets with any technique is challenging because of natural hand tremors and sensor noise. In-line with other research [9][19] our pilot testing revealed that these factors made it difficult for users to select small targets, as it was difficult to maintain the mouse cursor's position long enough to trigger a selection. For example, in our initial design, one degree of horizontal movement corresponds to about 28.5mm (30.3px) of cursor movement. Thus, to select a target with a diameter of less than 20mm, a user needs to keep their arm within a range of a single angular degree.

To address this issue, we introduced a cursor acceleration function to slow down the cursor movement in these cases. Watchpoint scales the cursor's speed depending on the speed of the smartwatch's physical movement. We experimented with a number of cursor acceleration functions but, ultimately, we implemented the following step acceleration function:

- When the speed of the watch movement is faster than 10 degrees/second, the cursor moves at the normal speed (30.3 px/degree). This corresponds to direct input.
- When the watch speed is slower than 5 degrees/second, the speed of cursor movement is 15.15px/degree, i.e. half of the normal speed.
- When the watch speed is between 5 degrees/second and 10 degrees/second, a smooth power function is used to map IMU data to cursor speed.

Other cursor acceleration functions exist, including a detailed exploration of cursor function by Nancel et al. [14]. But an important difference between our implementation and previous work is that we use absolute positional input to assign cursor movement to orientation (Equations 1, 2, and 3). In our pilot testing we found that this simple acceleration function allowed us to preserve absolute positioning for high speed, while maintaining the ray-casting metaphor, and limited the extent of relative positioning during fine-tuning.

One drawback of combining absolute and relative positioning in the interface is that the center position of the cursor may be shifted from the original center angle after a series of slow movements. To correct for this offset, Watchpoint reverts to absolute positioning immediately after any click event and upon speed up to a speed greater than 10 degrees/second.

The drawback of cursor repositioning on click or speed up is that this does result in a cursor jump. On speed-up, participants in pilot studies were unaware of the cursor jump; however, on click, participants could see the cursor jump, but the reassurance of the event firing correctly reassured participants in our pilot studies that, at the very least, input was correctly interpreted.

3.3.3 Jitter correction

The final issue we address is jitter. Although Android's gravity and rotation vector data are smoothed, they still include some noise, which affects pointing performance, particularly for small targets. We originally used a 50 ms moving average as a filter, but found this was insufficient to suppress jitter and caused latency in input. To address this issue, we implemented the 1 € filter [5] with a minimum cutoff set to 3.0Hz, delta cutoff set to 1.0 Hz and beta set to 0.01.

4. EXPERIMENTAL VALIDATION

We experimentally validated Watchpoint's performance using a Fitts's Law task [7][11]. Because our initial motivation was to determine if a smartwatch can support pointing performance comparable to camera- and device-based techniques, our goal was to identify representative techniques and perform a comparative study with Watchpoint.

Haque et al. [9] provides a useful baseline by which we can evaluate freehand pointing techniques: in their evaluation of Myopoint, they find that their technique provides similar pointing performance to a Vicon-based technique. We use the same criteria in our evaluation, and find that Watchpoint provides comparable pointing speed to both Vicon and Myopoint. Replicating the experimental setup of Myopoint [9] and a Vicon-based freehand pointing evaluation by Vogel and Balakrishnan [19], a between-subjects analysis showed Watchpoint is statistically comparable in performance and outperforms in terms of error rate for small (high ID) targets as compared to both Vicon systems and to Myopoint.

4.1 Participants

We collected interaction data from 10 participants (2 female, 1 left-handed). The participants used their dominant hand for the experiment. Each participant received $5 remuneration, and each session lasted 20 to 30 minutes.

4.2 Apparatus

Participants completed the experimental task while wearing an LG G Watch R smartwatch, with sensor data forwarded to an Android Nexus 5 smartphone. The Nexus 5 was connected to a MacBook Air laptop computer. Participants stood 2m from a projector screen. The projection area was 1.71m by 1.05m (1920 by 1080 px). Although Vogel and Balaksishnan [19] and Haque et al. [9] used a wider dsplay with a 32:9 aspect ratio, we used a standard 16:9 projector screen. We chose to use a display with a 1.06 px-per-mm density to fall between those used in the evaluation of Myopoint [9] (0.83 px-per-mm) and the Vicon-based technique [19] (1.23 px-per-mm). The study took place in a closed experimental space on campus at our university laboratory.

4.3 Experimental Task and Design

We used the ISO 9241 Part 9 standard multi-directional Fitts's Law pointing task [7][11], with independent variables for Target WIDTH and DISTANCE. To enable comparisons with previous evaluations, we included target WIDTHs of 16, 54, and 144 mm. We also included 34mm targets to provide a greater opportunity to evaluate pointing performance for small targets. We included three target DISTANCES of 960mm, 640mm and 320mm, which when combined with the 4 target WIDTHs created 12 ID values ranging from 1.68 to 5.93. Our dependent variables were MOVEMENT TIME and ERROR RATE.

Our study differed from previous work in the method used to control Fitts's IDs. In the previous two studies, the researchers increased ID by increasing DISTANCE; however, the wider display aspect ratios used in their study prohibited the use of a fully multi-directional test, and their targets were, instead, separated only horizontally. We feel that this is a significant drawback to past research as it prevents comparison of input using the standard ISO multi-directional Fitts's Law pointing task. We addressed this limitation in our study by controlling both target WIDTH and DISTANCE, evaluating values of WIDTH and DISTANCE which permit a fully multi-directional test.

In summary, given our experimental task, participants completed 13 trials in each of 4 WIDTH × 3 DISTANCE conditions in a series of 12 blocks with one WIDTH-DISTANCE combination per block. The order of the 12 blocks was randomized. Overall, we collected 4 WIDTHS × 3 DISTANCES × 13 Repetitions × 10 Participants for a total of 1560 trials.

4.4 Procedure

Participants were first welcomed, completed a brief demographic questionnaire and informed consent form, and the experimental task was explained and demonstrated. Participants next completed two practice blocks and were given an opportunity to ask questions about the task before starting the experimental trials. Once ready, participants were instructed to work as quickly and as accurately as possible, to continue without trying to correct errors, and to rest as desired between blocks.

4.5 Data Collection and Analysis

All cursor movements were logged to data files. Trials judged to be errors, where the 'click' event fell outside of a target, were counted as errors and included in error rate analysis but were excluded from our analyses of movement time. Comparisons between Watch and Myo techniques for MOVEMENT TIME and ERROR RATE were performed using Repeated Measures Analysis of Variance (RM-ANOVA) tests, with an alpha of .05.

4.6 Result

As noted, error trials were excluded from Movement Time analysis. Overall error rate was 14.1% of trials. It was 17.9% in the Myo study and 18.4% in the Vicon study.

4.6.1 Movement Time

There was no significant effect of technique on Movement Time between Watchpoint, Myopoint, and Vicon. Qualitatively, the target selection time of Watchpoint was between Myopoint and Vicon. Figure 6 depicts movement time plots for Watchpoint (blue), Myopoint (red), and Vicon (green). Line fitting equations and R^2 values are as follows:

Watchpoint:	$R^2 = .96$ MT = 106.13 + 587.37 × ID
Myopoint:	$R^2 = .97$, MT = 171.59 + 609.36 × ID
Vicon:	$R^2 = .87$, MT = 28.93 + 528.32 × ID

4.6.2 Error Rate

Watchpoint had a significantly lower error rate ($F(1,22)=17.80$, $p<0.001$) than Myopoint (Figure 7). As expected, there was significant effect of target WIDTH on error rate ($F(2,22)=349.43$, $p<0.001$). Post hoc analyses revealed an interaction effect between TECHNIQUE and target WIDTH ($F(2,22)=4.8$, $p=0.012$), and a simple main effects analysis showed that Watchpoint had a significantly lower error rate than Myopoint for 48 mm targets ($p<0.038$ at 48mm) and than both Myopoint and Vicon for 16mm targets ($p<0.001$ at 16mm). Vicon outperformed for large (144mm) targets. No differences were found between Watchpoint and Myopoint for large targets ($p=0.585$). Analysis of main effects revealed a significant difference between WIDTH with both Watchpoint and Myopoint (Watch $p=0.0017$, Myo $p<0.001$).

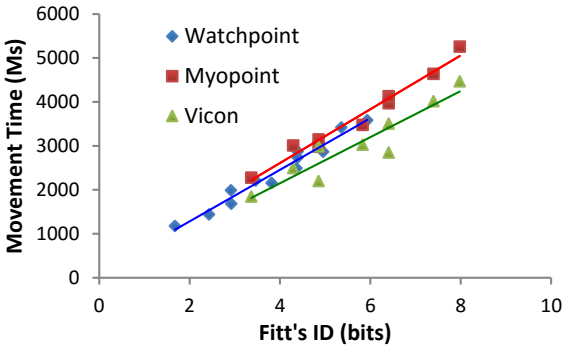

Figure 6. Movement time by ID.

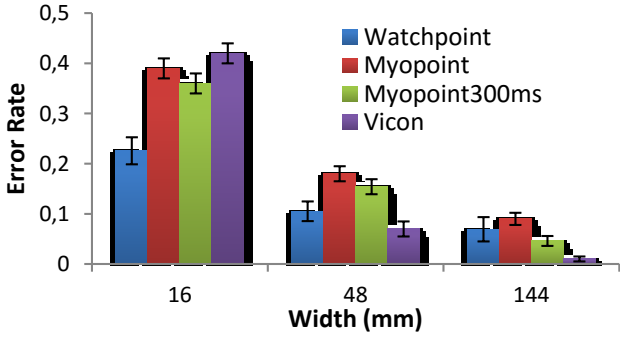

Figure 7. Error rate by target WIDTH. Error bars are SEM.

4.7 Discussion

Our results suggest that Watchpoint provides robust support for freehand pointing, matching the selection time performance of comparable techniques. Our RM-ANOVA comparison of the three techniques revealed no significant difference for MOVEMENT TIME. Analysis of effect size indicates that the observed difference between techniques is quite small. Given that Vicon-based pointing is often regarded as a 'gold standard' for freehand pointing [19], we consider this level of performance indicative that Watchpoint is suitable for supporting a variety of applications in ubiquitous settings.

Further, our results suggest that Watchpoint supports freehand pointing with a lower error rate than Myopoint ($p = .002$). These differences were particularly evident for small (16mm) targets where the error rate for Watchpoint was nearly half that of the Vicon and Myo-based techniques. In their evaluation of Myopoint, Haque et al. reported average error rates for such targets in excess of 35%. With Haque et al., we note that error seems a function of target size, not ID, indicating that, potentially, techniques may be more limited by visual acquisition and Control-Display gain settings than by form factor.

4.8 Limitations

Watchpoint represents a first-step towards developing a ubiquitous freehand pointing technique. As a first step, it demonstrates the feasibility and promise of such techniques, but is also necessarily limited in scope. In particular, we envision next steps in this research to address three of these limitations:

4.8.1 State-Transition Errors

During the course of our study, we noticed a few cases where users were not sure of Watchpoint's current state. For example, as shown in Figure 4, to avoid unintentional clicks, we designed the state transition diagram such that the required angle to return to the Tracking state is steeper (10 degrees) than the angle to invoke the click event (30 degrees). A consequence of this asymmetry is that once users are aware of the exact angle required to invoke the click event, they tend to rotate their wrist as little as possible to minimize effort, and occasionally do not rotate enough to return to the Tracking state.

While users rapidly became aware of their error during the study and did discover how to accommodate for state transitions, we also recognize these difficulties as an opportunity to provide additional feedback to users regarding the current state of the mouse cursor, and to refine Watchpoint's gesture recognition. For example, most smartwatches include haptic feedback hardware that could be leveraged to indicate when state-transitions occur. As well, in terms of state transition recognition, we used the naive approach of applying thresholds to the absolute angle of the user's wrist rotation. We expect that a state model that relies on relative wrist movements may further help to address these minor usability issues.

4.8.2 Hardware Limitations

Our current prototype was intentionally limited to running on a single display and relied upon smartphone and laptop computers for connectivity. While this design enabled us to empirically validate the performance of current-generation smartwatch hardware, it does not implement the vision of a fully-connected, ubiquitous interaction technique. At the time of this writing, smartwatch SDKs are becoming available that enable more powerful applications to run on the smartwatch CPUs, rather than being offloaded to a smartphone or laptop. It also allows the smartwatch to connect a network without a smartphone. We also expect that wearable devices will have improved connectivity in the near future, and, thus, be able to connect to a display directly through protocols such as Bluetooth HID.

4.8.3 Transitioning between Absolute and Relative Modes of Input

Our work addresses many challenges related to freehand pointing, such as the selection of small targets; however, there is room for improvement in the implementation of our cursor acceleration algorithm. In particular, as users transition between pointing at targets they transition between relative and absolute modes of interaction as the acceleration algorithm is activated and deactivated. While slowing of the cursor is handled elegantly in our system, under our current implementation the cursor 'jumps' as a user moves away from a target and the cursor's rate of movement returns to normal. Methods of 'smoothing' this transition are an important next step.

5. IMPLICATIONS FOR DESIGN

Our results suggest that current generation, off-the-shelf smartwatch hardware is capable of enabling interactions in ubiquitous environments, and that a modern smartwatch is well-positioned to serve as a gateway device for interaction with a computationally augmented world. However, upon reflecting on our work, our results suggest implications for the design of freehand pointing techniques, regardless of the facilitating technology. In particular, our results provide insight into the effectiveness of cursor acceleration functions and the need for simple calibration and subtle gestures.

5.1 Cursor Acceleration Functions

Our evaluation suggests that Watchpoint provides comparable selection times to other techniques, but with a significantly reduced error rate for small targets. However, achieving this level of performance required some fine-tuning on a number of levels. For example, our cursor acceleration algorithm was designed to improve pointing performance for small targets, and data collected during our study suggests the algorithm was effective. Similarly, our work on position correction and jitter correction is applicable to other freehand pointing techniques, particularly those that wish to include Watchpoint's 'twist to select' mechanism.

There is a benefit to additional research to identify optimal cursor acceleration and filtering parameters that can be applied to all freehand pointing techniques. As wearable devices continue to evolve and become more affordable, they are positioned to become a personal gateway to ubiquitous environments. Furthermore, vision-based systems will also continue to improve. We feel it is important to have a common experience across all freehand devices. In summary, there is a need for research, such as ours, that develops solutions to common issues such as jitter that can be applied across a range of underlying technologies.

5.2 Zero Calibration

One of the benefits of using a smartwatch's gravity and rotation sensors is that they provide an absolute reference point. For vertical positioning, Watchpoint does not require calibration since a user's horizontal arm position is mapped to the center of the display. For horizontal positioning, the angle of the watch is used only when it switches to the active mode. Since we do not use the actual position of the screen, the calibration process is robust and simple. Using this technique, the calibration process is naturally

merged into normal pointing interactions with a nearby screen.Subtle Gestures

Camera-based tracking systems, such as the Vicon-based system in our study, often rely upon users making exaggerated gestures for input. For example, pointing at a large display requires a user to fully extend their arms and point with an outstretched finger. However, recent research [18] has confirmed that in reality, individuals tend towards more limited movements over time to reduce fatigue; a phenomenon widely recognized by those who have played Nintendo's Wii. However, as individuals shift towards these less exaggerated gestures, camera-based systems become less effective at detecting interactive gestures.

A benefit of Watchpoint, and other wearable-based techniques, is that they do not require these exaggerated movements, and work equally well with a person standing in front of a display or sitting in a chair with their arm on an armrest. Given other limitations of camera-based techniques such as occlusion, lighting constraints, and difficulties in identifying users, we suggest that device-based interactions are likely to replace camera-based techniques for many applications.

6. CONCLUSIONS

We present the design and validation of Watchpoint, a system that supports pointing and clicking on a nearby large display via a smartwatch. We show through a controlled Fitts's Law study that our prototype provides performance comparable to other freehand pointing techniques for pointing time, and outperforms those systems in terms of error rate. Our work demonstrates that a commodity smartwatch can serve as an interaction device for pointing in ubiquitous display environments. Because no vision-based tracking system is involved, there is no occlusion problem. The user can interact with displays in a ubiquitous environment even when there is an obstacle between the user and the display (ie. audience can interact with the screen in the movie theater from behind the other people.) We envision that Watchpoint may illustrate a potential 'killer app' for the emerging smartwatch market.

7. ACKNOWLEDGEMENTS

The authors thank the Natural Sciences and Engineering Research Council of Canada and Google for funding this research.

8. REFERENCES

[1] Android API Guides, Motion Sensors. Retrieved January, 2016, from http://developer.android.com/guide/topics/sensors/sensors_motion.html

[2] Ballagas, R., Borchers, J., Rohs, M., and Sheridan, J. 2006. The Smart Phone: A Ubiquitous Input Device. IEEE Pervasive Comput. IEEE Pervasive Computing, 5(1), 70-77.

[3] Bowman, D., Wingrave, C., Campbell, J., and Sy, V. 2001. Using pinch gloves ™ for both natural and abstract interaction techniques in virtual environments. Proceedings of HCI internationall 2001.

[4] Buxton, W. 1990. A Three-State Model of Graphical Input. In D. Diaper et al. (Eds), Human-Computer Interaction - INTERACT '90. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 449-456.

[5] Casiez, G., Roussel, N., and Vogel, D. 2012. 1 € filter. Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems - CHI '12.

[6] Chen, X., and Davis, J. 2000. Camera Placement Considering Occlusion for Robust Motion Capture. Stanford University Computer Science Technical Report, CS-RT-2000-07

[7] Fitts, P. M. 1954. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement Journal of Experimental Psychology, 47 (6), 381–391.

[8] Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., and Wang, M. 2011. Proxemic interactions. Interactions, 18(1), 42.

[9] Haque, F., Nancel, M., and Vogel, D. 2015. Myopoint. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15.

[10] Huang, E. M., Mynatt, E. D., and Trimble, J. P. 2007. When design just isn't enough: The unanticipated challenges of the real world for large collaborative displays. Personal and Ubiquitous Computing, 11(7), 537-547.

[11] ISO, 2002, Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non-keyboard input devices. (ISO 9241-9)

[12] Jota, R., Nacenta, M., Jorge, J., Carpendale, S. and Greenberg, S. 2010. A Comparison of Ray Pointing Techniques for Very Large Displays. GI '10 Proceedings of Graphics Interface 2010

[13] Myers, B. A. 2001. Using handhelds and PCs together. Communications of the ACM Commun. ACM, 44(11), 34-41.

[14] Nancel, M., Chapuis, O., Pietriga, E., Yang, X., Irani, P. P., & Beaudouin-Lafon, M. 2013. High-precision pointing on large wall displays using small handheld devices. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13.

[15] Pietroszek, K., Kuzminykh, A., Wallace, J. R., and Lank, E. 2014. Smartcasting. Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures the Future of Design - OzCHI '14.

[16] Pietroszek, K., Wallace, J. R., and Lank, E. 2015. Tiltcasting. Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15.

[17] Raskar, R., Nii, H., deDecker, B., Hashimoto, Y., Summet, J., Moore, D., Zhao, Y., Westhues, J., Dietz, P., Barnwell, J., Nayar, S, Inami, M., Bekaert, P., Noland, M.Branzoi, V. and Bruns, E. 2007. Prakash:Lighting aware motion capture using photosensing markers and multiplexed illuminators. ACM Transactions on Graphics, 26(3), 36.

[18] Ruiz, J., and Vogel, D. 2015. Soft-Constraints to Reduce Legacy and Performance Bias to Elicit Whole-body Gestures with Low Arm Fatigue. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15.

[19] Vogel, D., and Balakrishnan, R. 2005. Distant freehand pointing and clicking on very large, high resolution displays. Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology - UIST '05.

[20] Yee, K. 2003. Peehole displays. Proceedings of the Conference on Human Factors in Computing Systems - CHI '03.