SYNCHRONOUS INTERFACES FOR WEARABLE COMPUTERS

A Dissertation
Presented to
The Academic Faculty

By

Jason Wu

In Partial Fulfillment
of the Requirements for the Degree
B.S. in Computer Science from the
School of Interactive Computing

Georgia Institute of Technology

April 2018

SYNCHRONOUS INTERFACES FOR WEARABLE COMPUTERS

Approved by:

Dr. Thad Starner, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Thomas Ploetz
School of Interactive Computing
Georgia Institute of Technology

Date Approved: May 4, 2018

ACKNOWLEDGEMENTS

My decision to pursue research throughout my undergraduate career started as a way to get involved in interesting projects and gain exposure to cool technology. However, it has grown into a lifetime goal of pursuing answers to the unsolved and exploring the unknown. After graduating from Georgia Tech, I will attend graduate school for a PhD in Human-Computer Interaction to build devices that can act as our companions, able to interpret our intent and make our lives better. In the future, I aspire to become a research professor so that I can not only explore different research fields but also contribute back to them. For that, I have many people to thank.

Firstly, my family as helped me realize the importance of dreams and education. Dream big, they would tell me, and failure is the mother-in-law of success (a translation of a Chinese proverb). Years of hammering these cliched maxims into my head combined with their unwavering support have led me to dream big while realizing that overcoming setbacks is a part of the achievement. Most importantly, my family has given me the freedom to choose my own future and work toward it.

In addition, I have my friends and lab-mates to thank for their support. Through my research work, I have met many other students exploring their own passions and contributing to their own fields of study. Many of these students are already in graduate school and have encouraged me to do the same. Other undergraduate researchers are just as excited as I am to explore their interests by applying for a PhD. Specifically, I'd like to thank Cooper Colglazier, Adhi Ravishankar, Yuyan Duan, and Yuanbo Wang for their direct involvement in my research projects. I'd also like to thank the researchers in the Contextual Computing Group and GT Ubicomp lab for creating a productive and cooperative environment for research. Outside of the lab, I have friends who are pursuing their dreams in other ways. Some have joined the Peace Corps, while others have started businesses or have become accomplished musicians. I am in constant awe of their achievements and ambition, and I

seek to follow their example.

Finally, my mentors have not only taught me a great deal about the field of HCI and research, they have been a constant source of motivation and inspiration. When I approached my first research advisor, I was not sure if I could contribute to the project or even be useful. However, through the guidance and mentorship of my mentors, I became familiar and accustomed to the exploratory yet disciplined nature of research. I'd like to thank my advisors, Dr. Rosa Arriaga, Dr. Gabriel Reyes, Dr. Gregory Abowd, Dr. Thad Starner, and Dr. Thomas Ploetz for the research opportunities, advice, and skills they have given me. In short, I have been very fortunate to have professors and mentors dedicated to my success as a student and a researcher. In the future, I hope I can be as educational and inspiring to others as they were to me.

Achieving my goals will undoubtedly demand a great deal of dedication, involve a lot of setbacks, and require skills and experience I have yet to obtain. My advisors have cautioned me that the road to professorship is a long and arduous one but that they couldn't imagine a more fulfilling and stimulating job. I still have a long way to go, but when I finally get there, I won't be alone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Synchronous interfaces provide a new input modality for wearable devices requiring minimal user learning and calibration. We present SeeSaw, a synchronous gesture interface for commodity smartwatches to support rapid, one-handed input with no additional hardware. Our algorithm introduces methods for minimizing false-trigger events while facilitating fast and expressive input. Results from a live evaluation of the system as a one-handed notification response gesture show comparable speed and accuracy to two-handed touch-based interfaces on smartwatches. The SeeSaw input interaction is also evaluated as an input interface for smartwatches and head-worn display systems, showing that the interface enables rapid and accurate interaction. Thus, we find that the SeeSaw synchronous gesture offers a compelling alternative to existing input methods on wearable computers. Finally, a suite of demo applications are presented to show SeeSaw's support of binary, multi-target, and activation input.

CHAPTER 1
INTRODUCTION

Wearable devices such as smartwatches and head-mounted computers provide convenient, readily available access to content. However, a primary obstacle for the widespread adoption of wearables is the lack of input techniques appropriate for the types of interactions common with these devices. Just as the keyboard and mouse are not suitable for smartphones, existing technologies such as the touchscreen and voice-based commands are not ideal for many wearable devices. Touch interfaces are hindered by problems of finger occlusion on small form-factor devices, especially smartwatches[1]. Meanwhile, voice interactions can be effective for issuing commands and inputting text but are not discrete and not socially acceptable. An effective input modality for wearable devices supports microinteractions with fast access time while maintaining an acceptable level of expressiveness[2].

Synchronous interfaces have been explored to address some of the shortcomings with traditional gesture-based interactions, including their memorizability and intuitiveness. Instead of expressing user intent by performing multiple discrete gestures, a single motion is performed in synchrony with the target stimulus. Thus, minimal user learning and memorization is required[3]. This concept has been explored recently for smartwatch input and multi-user systems, using eye-gaze tracking, camera-based hand motion tracking, and magnetic ring motion correlation [1, 3, 4, 5]. However, these systems currently require additional external hardware, making it unsuitable for use on many commodity smartwatches. In this work, we present SeeSaw, a rapid, one-handed synchronous gesture interface that supports expressive and subtle input on commodity smartwatches with no extra hardware.

SeeSaw is a synchronous gesture interface that uses only the gyroscope sensor present on smartwatches. The resulting synchronous gestures can be performed with the user's wrist and hand and require no additional hardware or software modification. The system is

also designed to facilitate subtle interaction, minimizing attention to the user when interacting with the wearable computer. Results from user evaluations show one-handed dismissal speed of 4.5s - 5.5s for smartwatch-only interaction and 3.6s for HWD interaction. We show that SeeSaw enables effective interaction with wearable computers in many common use-cases and provides a compelling alternate input modality to replace or augment traditional touch interfaces.

CHAPTER 2
RELATED WORK

Many wearable computers, especially smartwatches and head-worn computers, are primarily meant to facilitate quick microinteractions between the user and the wearable system[6, 2]. Despite this, the default input modalities included with most smartwatches and head mounted computers, mainly the touch-screen and speech to text input systems, do not fully meet these requirements, as many contexts impose constraints on two-handed input and social acceptability. Thus, much research has been done to explore alternative input modalities for these devices.

2.1 One-handed Input

One-handed input is preferable in many everyday use scenarios where the user's second hand is occupied, or a more discreet mode of interaction is needed. Serendipity is an example of such a finger gesture recognition system capable of recognizing 5 fine-motor gestures[7]. To expand the capabilities of gesture recognition systems, additional software and hardware modifications are often introduced to allow for a larger gesture set and higher detection accuracy. The ViBand input system uses a custom smartwatch software kernel to allow for increased sampling from sensors, which enables more accurate and expressive user input and the ability to sense external objects through touch[8]. Hardware modifications are also possible, as demonstrated by numerous projects such as WristWhirl, WristFlex and Tomo, that allow for a larger gesture set and continuous input[9, 10, 11].

2.2 Rhythmic Input

While many discrete gesture recognition systems rely on classifying windows of sensor input, rhythmic patterns rely on the temporal dimension to recognize user intent. Often, this results in an intuitive and easy to use input system. This is shown by the Whack gesture system, an input modality for mobile devices that allows users to whack their mobile phone with the open palm or heel of the hand in rhythmic succession to show intent[12]. Ghomi explores the optimal vocabulary size of the rhythmic patterns and feedback method, finding that a 30-pattern vocabulary can be recognized with a 94% recognition rate[13]. In addition to discrete selection tasks, rhythmic patterns have also been applied to search filtering. By allowing users to tap a song's rhythm on the device touchscreen or body, the Finding My Beat system is able to filter a musical library to find songs that match the user's input[14].

2.3 Synchronous Gesture Interfaces

Synchronous gestures are similar to rhythmic patterns in that they both allow the user to express intent over time, but in the case of synchronous gestures, the stimulus is presented to the user indicating the expected gesture or pattern[15]. Motion correlation has been implemented successfully in many camera-based systems, often allowing for robust, multi-user selection on large displays[15, 4, 16]. Recently, synchronous gestures have also been explored using smooth pursuit tracking, allowing users to interact using gaze. AmbiGaze is such a system that allows users to interact with ambient devices and trigger corresponding actions by performing correlation with relative eye movement[17]. Orbits is another smooth pursuit based tracking system used to interact with smartwatches with high accuracy[1]. However, pursuit tracking requires the use of specialized eye-tracking hardware and may interfere with the user's ability to comfortably view on-screen content. FingOrbits seeks to replace pursuit-tracking with finger movement by using a specially-designed thumb ring[3]. The FingOrbits system is implemented using an IMU and contact microphone connected to a laptop running a FFT-based detection algorithm. Recent work by Reyes et al. has investigated removing the need for additional external powered hardware for synchronous gesture interfaces by constructing a thumb ring with a passive rare-earth magnet, showing viable accuracy and speed for notification response applications on smartwatches[5]. While all of these input interactions show the effectiveness and advantages of synchronous gesture interfaces, they all require external hardware and are thus not suitable for out-of-the-box operation with commodity smartwatches.

2.4 Subtle Interfaces

In addition, the usability of gesture systems often depend on their social acceptability, as interfaces often require users to adopt disruptive or embarrassing behaviors to interact with the system[18]. Research has shown that gestures that were subtle or utilized everyday movements were more likely to be socially acceptable[18]. Some gesture systems aim to minimize the motion required to trigger the system. An example that uses this approach measures the electromyographic (EMG) signal, allowing for subtle and intimate interaction[19]. Another approach for facilitating subtle interaction is requiring movement of unseen or hidden body parts such as the inside of the mouth or the jaw and inner ear. Systems like Bitey shows that tooth click gestures are viable for subtle interaction[20] while Stick it in your ear shows that an Outer Ear Interface (OIE) can be constructed for a variety of applications such as gesture detection, jaw movement, and even heart rate monitoring[21]. Finally, subtle gesture systems can be created by disguising gestures as everyday activities such as foot/table tapping or scratching one's nose. Prior work shows that gestures such as table tapping and foot tapping have social acceptability in public settings[18], and the Itchy Nose interface is able to facilitate accurate, subtle gestures using EOG sensors[22].

CHAPTER 3
INTERFACE DESIGN

3.1 Gesture Interaction

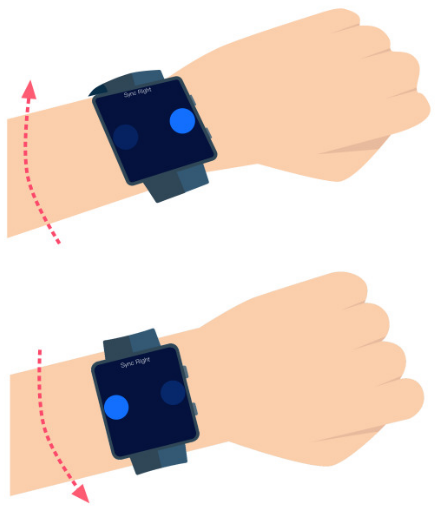
The SeeSaw input interaction is a synchronous gesture interface that allows users to perform input through synchronous wrist rotations. The primary input interaction used by SeeSaw is a repetitive back and forth tilting of the wrist in a sinusoidal motion. When a stimulus signal is provided, it is rendered as a flashing target where the brightness of the target corresponds to the stimulus signal or as a haptic vibration. Syncing with a target is achieved by tilting the wrist away from the body when the target is illuminated and tilting the wrist back towards the body when the target is dimmed (Figure 3.1). When a haptic stimulus is used, the user times the tilting motion so that the vibration occurs when the wrist is away from the body. The user is able to provide both a positive and negative response to the stimulus by performing the synchronous gesture in-sync or out-of-sync. SeeSaw also supports detecting sinusoidal synchronous gestures without a stimulus. These gestures are performed by repeatedly tilting the wrist inward and outward at a constant frequency.

3.2 User Interface

3.2.1 Binary Input

Many microinteractions are meant to be fast and simple. Examples of these interactions include notification dismissal and basic navigation. These Tasks that require only binary input can be easily integrated with the synchronous gesture interface by assigning both selections to a pair of synchronous elements with alternating blinking patterns so that the correlation value and direction from a single detector can be used to determine intent.

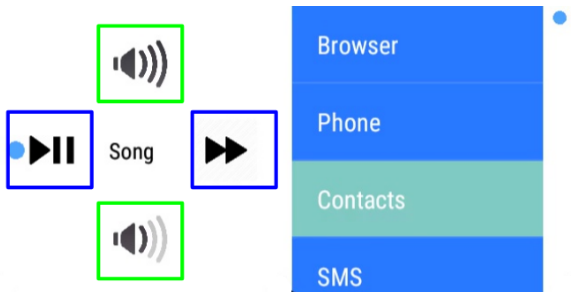
Figure 3.1: SeeSaw Wrist-Tilting Gesture Interaction



3.2.2 Multi-Target Selection

While binary selection is sufficient for most notification-type microinteractions, its expressiveness is too limited for most application input. To enable multi-target selection, the application designer can organize UI elements into *control groups* consisting of pairs of related controls and introduce a separate gesture to cycle or switch between control groups. Another approach is to use two separate detectors to construct a blinking cursor interface. One detector is used to navigate a list of items and move a blinking cursor on a selection while the other detector is used to confirm a selection.

Figure 3.2: Multi-Target Selection using Control Groups (Left) and Blinking Cursor (Right)



3.2.3 Activation Gesture

Activation gestures are used by application designers to enable or activate new controls or interaction modes. While it is possible to have an always-visible blinking element for activation gestures, it is impractical due to the limited screen space available on most smartwatches and wearable displays. Alternatively, designers can embed the synchronous stimulus in existing UI elements (i.e. blinking cursor in a digital clock watch-face) or allow users to perform a 1Hz gesture without any UI stimulus.

CHAPTER 4
TECHNICAL IMPLEMENTATION

The SeeSaw algorithm is a motion correlation algorithm designed to facilitate rapid synchronous gesture detection and to minimize false-triggering. The current implementation is written as a Java library that can be included in Android apps for live use and on the desktop for programmers to perform offline analysis for activity-dependent frequency tuning. In addition to its gesture detection capabilities, the library provides mechanisms for providing live user feedback. The SeeSaw gesture detection algorithm is implemented as a multi-stage pipeline where sensor input from the smartwatch is processed in multiple steps: Signal Preprocessing, Synchrony Detection, Lag Adjustment, and Output Processing (Figure 4.1).

4.1 Signal Preprocessing

4.1.1 Feature Extraction

The primary gesture interaction for SeeSaw involves repeatedly tilting the smartwatch back and forth along a single axis. This implies the existence of a dominant axis of rotation affected by the wrist motion. Because the dominant axis of rotation is closely aligned with the gyroscopic x-axis of the smartwatch used for testing (Figure 4.2), the x-component of the gyroscope is used as the gesture detection feature. For smartwatches or other devices that use an alternate sensor layout, it is possible to identify the dominant axis or artificially construct one using Principal Component Analysis (PCA).

The samples are stored in a sliding window, X_W. The length of the sliding window is set to 1.5 s using results from previous work on synchronous gestures and through empirical testing [5]. Given knowledge of the gesture motion and the dominant axis, it is possible to prevent false triggering by ensuring at least a factor $\alpha_{\min}$ of the total variance between all 3 orthogonal axes is from the dominant axis, $\sigma_x \geq \alpha_{\min}\sigma_{\text{total}}$. The variance factor, $\alpha_{\min} = 0.4$ is set empirically. Knowledge of the gesture is also used to set additional bounds, $\sigma_{\min} = 0, \sigma_{\min_{\text{activation}}} = 1, \sigma_{\max} = 12$.

4.1.2 Resampling

Timestamps from non-realtime operating systems installed on most commodity smartwatches do not guarantee even or consistent timings. To account for the possibility of data overflow or underflow and to transform the sensor readings into a more useful time-series representation, the sensor values are first resampled. Readings are requested from the gyroscope sensor at 100 Hz and are downsampled to 10 Hz using an interval-based subsampler. The subsampler stores incoming readings in a buffer and returns the mean value every 100 ms. Upon emitting an output, the subsampler's buffer is cleared.

4.1.3 Signal Detrending

To minimize the effects of sensor drift, environmental noise, and external low-frequency movement, a detrending procedure is applied to the signal. Linear regression is performed on the samples in the current window to compute a line of best fit, $y = mx + b$. The best-fit line is then subtracted from each sample, resulting in a zero-centered signal with no overall temporal correlation.

4.2 Synchrony Detection

SeeSaw supports gesture detection both with and without a stimulus. While both modes of operation rely on motion correlation, the algorithm can choose to correlate the sensor signal with itself or a *reference signal* depending on the context.

Including a stimulus is useful for *notification gestures*, *command gestures*, and more expressive input. The stimulus allows the user to provide a positive or negative response by performing the synchronous gesture in-phase or out-of-phase. Multiple stimuli can be used to construct more complex application interfaces.

The algorithm is also able to function without a stimulus by performing autocorrelation. Simple repetitive synchronous gestures can be recognized by detecting temporal self-similarity within the sensor data. This allows the synchronous gesture interface to be used for *activation gestures* or *initiation gestures*.

4.2.1 With Stimulus

Most synchronous gesture interfaces require a stimulus to be presented to the user in order to determine intent. These stimuli include rotating, blinking, or oscillating visual elements that correspond to different available input choices [1, 5, 23]. Complex synchronous gesture interfaces for multi-user and many-target applications have explored the use of geometric shapes or paths as stimuli. However, it has also been shown that simple stimuli are necessary for providing fast, expressive input for smartwatches [5]. SeeSaw provides a similar mode of interaction by displaying flashing targets on the screen that can each accept two responses.

While the synchrony detection algorithm does not require that the reference signal be periodic, I choose to use a periodic sinusoidal wave due to its simplicity and performability. Thus, the reference signal is defined completely by its amplitude, A, period, T, and starting timestamp, t_0. The signal is generated using the simple harmonic motion model.

$$x(t) = A \sin(\omega t + \varphi), \omega = \frac{2\pi}{T}, \varphi = \arcsin A^{-1} - \omega t_0 \tag{4.1}$$

The synchrony detection is based on correlation, so the signal amplitude, A, does not affect detector output and is set to a constant value of 1.

The Pearson correlation coefficient, ρ, is chosen as a measurement of similarity between the reference signal and motion signal.

$$\rho_{X,Y} = \frac{\mathrm{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{4.2}$$

The Pearson correlation coefficient measures the linear correlation between X, and Y, as a value between -1 (total negative correlation) and 1 (total positive linear correlation). This property allows the correlation coefficient to be easily thresholded to detect gestures with a certain accuracy or speed. A positive response is registered when $\rho_{X,Y} \geq \rho_{\text{thresh}}$ and a negative response is registered when $\rho_{X,Y} \leq -\rho_{\text{thresh}}$.

4.2.2 Without Stimulus

Without the presence of a stimulus signal, the algorithm lacks a reference signal to compute motion correlation. When an explicit stimulus is not provided, the algorithm assumes a simple, periodic synchronous gesture modeled by Equation 4.1 will be performed as an activation gesture.

The autocorrelation function is used to detect the presence of a synchronous gesture by calculating the correlation of a signal with itself at various time shifts. Similar to Equation 4.2, the autocorrelation function is normalized so that its output is bounded between -1 and 1.

$$R(\tau) = \frac{\mathrm{E}[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2} \tag{4.3}$$

The output of the autocorrelation function is sampled at the expected signal period, T. Under the assumption that synchronous gesture is periodic and its motion is well-modeled by simple harmonic motion, $R(T)$, will return a high value for signals of period T when compared to noise. Thus, $R(T)$ is thresholded such that an activation event is triggered when $R(T) \geq R_{\text{thresh}}$.

4.3 Lag Adjustment

To account for display latency and human response speed for stimulus-based syncing, lag adjustment is performed on the reference signal. Lag adjustment involves determining the time delay between the reference signal and the motion signal and regenerating a better-aligned signal for correlation. Previous work has shown that time delay analysis using cross-correlation has been successfully used to measure and correct for these lag factors [5]. However, the standard formulation, which finds the maximum of the cross-correlation function, does not account for negative responses where the user intentionally performs the synchronous gesture out of phase. Furthermore, constraints are introduced to limit the range of possible time-shifts to plausible values, prevent an increase in the false positive rate, and reduce computation time.

$$\tau_{\text{delay}} = \underset{t,-n\leq t\leq n}{\arg\max}(|(f \star g)(t)|) \tag{4.4}$$

The time delay is used to generate a adjusted reference signal by adjusting the offset term, φ, in Equation 4.1. The new offset shifts the reference to generate facilitate better alignment. $\varphi' = \arcsin A^{-1} - \omega(t_0 + \tau_{\text{delay}})$.

To further minimize the false positive rate, a separate sliding window is kept of lag adjustments made to the previous $\frac{|X_W|}{2}$ windows of sensor data. An additional condition for syncing is added to ensure the lag adjustments are correcting for input lag instead of external noise, $\sigma_\tau^2 \leq \sigma_{\max}^2$. The size of the lag adjustment window and value of $\sigma_{\max}^2 = 2000$ are made empirically.

4.4 Output Processing

The synchrony detection algorithms described in Section 4.2 is triggered when the correlation output of the motion signal reaches a certain threshold. To prevent false-triggering caused by unintentional or momentary movements, the correlation output is further processed before being compared to the thresholds.

Exponential Weighted Moving Average (EWMA) is used to smooth the correlation output to remove short-term fluctuations. EWMA is a type of moving average that applies exponentially decreasing weights to preceding terms in a time-series. It is widely used in computer science for measuring metrics such as CPU utilization and network latency.

$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha \cdot Y_t + (1-\alpha) \cdot S_{t-1}, & t > 1 \end{cases} \tag{4.5}$$

EWMA is chosen due to its infinite impulse response, simplicity, and low computational cost relative to signal filters. The decay factor, $\alpha = 0.35$, is set empirically.

Figure 4.1: SeeSaw System Overview Diagram

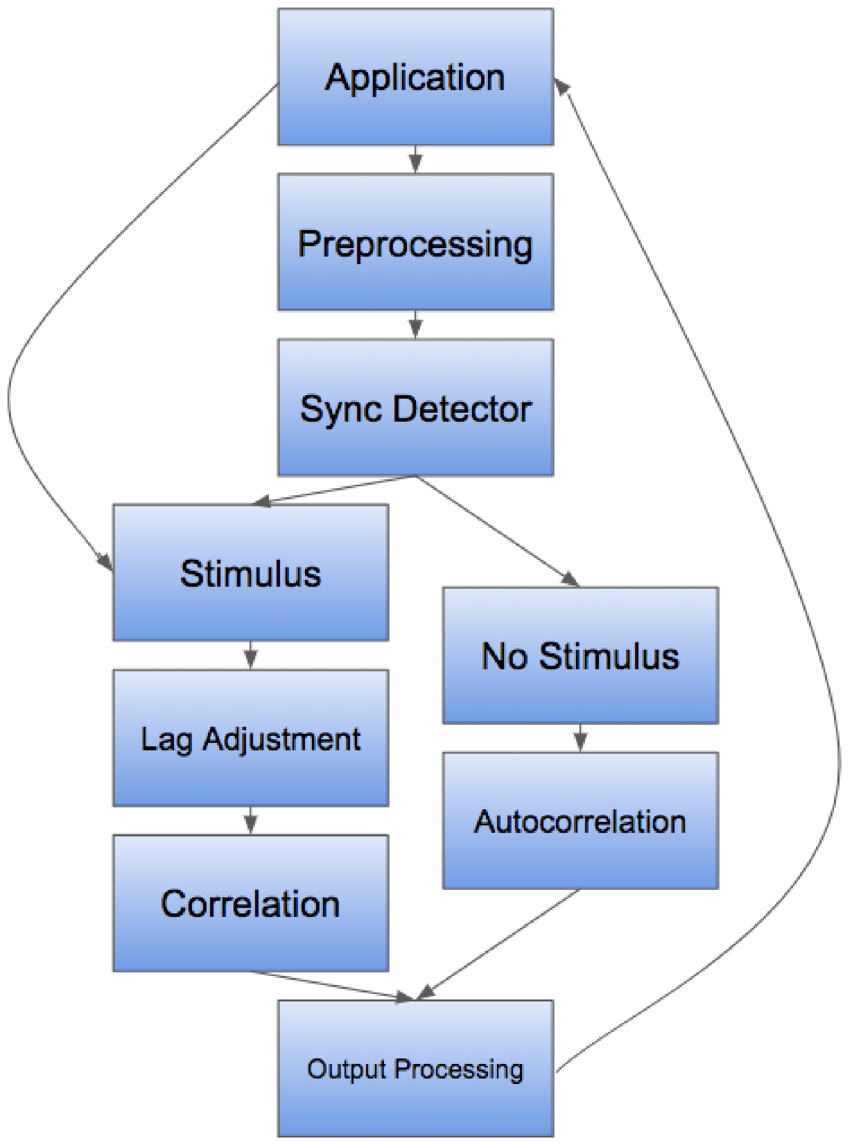
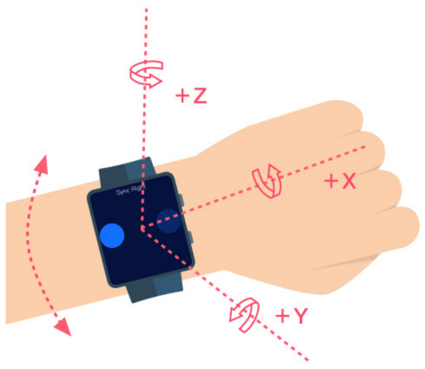


Figure 4.2: SeeSaw Dominant Axis Visualization



CHAPTER 5
SMARTWATCH-ONLY EVALUATION

In this section we describe the evaluation of SeeSaw as an interaction for applications installed and displayed on the smartwatch.

5.1 Smartwatch-Only System

A commodity Sony Smartwatch 3 SWR50 Android smartwatch is used for development and testing. The smartwatch features a quad-core 1.2 GHz ARM processor, 512 MB of RAM, and a 9 DoF IMU. Android 6.0.1 and Android Wear 1.5 was installed on the device. The SeeSaw synchronous gesture detector was implemented as an Android library which was used to construct Java applications that ran unmodified on the default operating system.

5.2 Evaluation

5.2.1 User Study - Frequency Tuning

The usability of the synchronous interface is largely dependent on the user's ability to accurately and consistently sync with the desired targets. Previous work has shown that 1 Hz is the optimal frequency for synchronous gestures performed with the thumb [5]. The SeeSaw interaction involves a larger motion performed using a different body part. To determine the optimal frequency for the syncing stimulus, a study is conducted comparing the effectiveness of three frequencies while performing two distraction tasks.

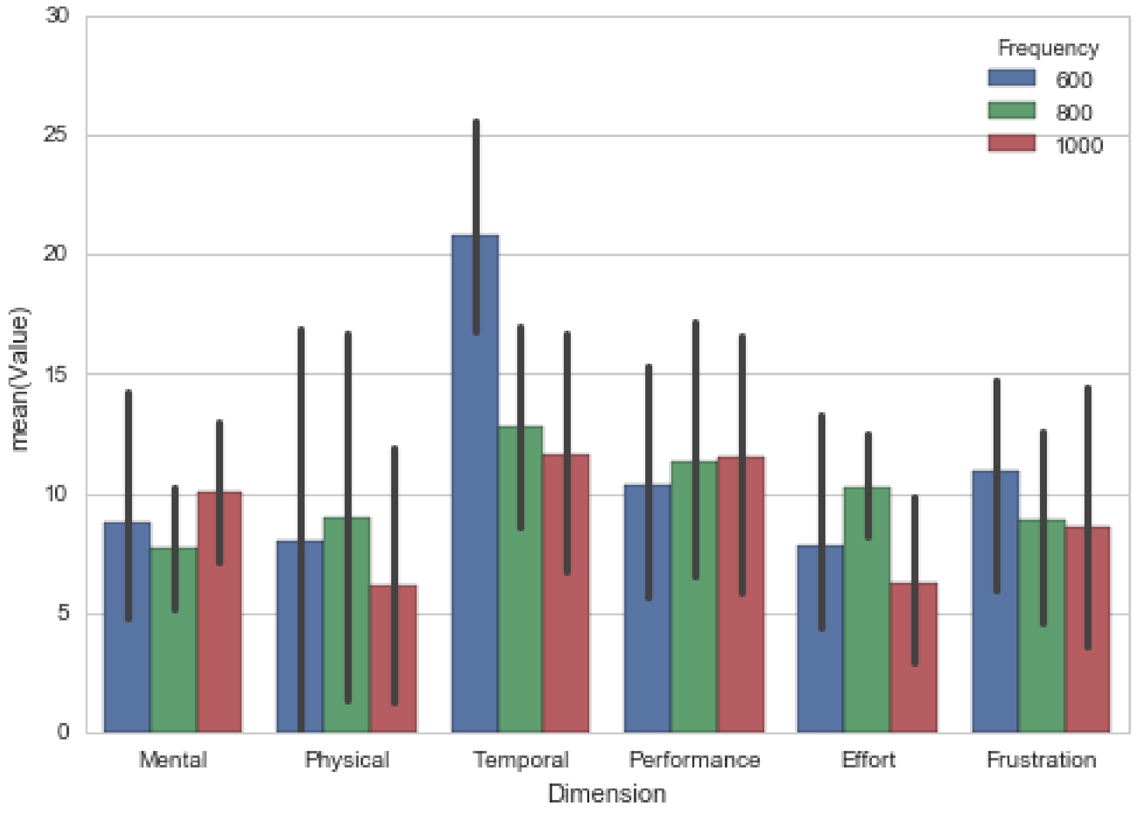
The study was conducted with 6 participants (5M/1F, ages 19-26) in our institution's usability lab. Participants were compensated $10 for the study, which lasted approximately 90 minutes. The frequency tuning study was a within-subjects study with a total of 6 experimental conditions with three frequencies (1 Hz, 1.25 Hz, and 1.67 Hz) and two activities (sitting and walking), similar to the study design previously conducted by Reyes et al. [5]. During the sitting condition, participants drive a car in a driving simulator with a Logitech steering wheel, and during the walking condition, participants are asked to walk along a fixed path. Experimental conditions are balanced between participants using a 6x6 Latin square. Participants were asked to dismiss incoming notifications as quickly as possible while wearing the smartwatch on their dominant hand to increase noise motion and variation. The smartwatch was set to provide both visual and haptic stimuli to allow users to react to notifications using their preferred modality. During the study, the trigger threshold was set to a high value $\rho_{thresh} = 0.85$ to avoid early false-triggering. For each frequency, the participant was given a prep session consisting of 25 notifications without any distraction. At the end of the study, participants filled out a NASA TLX assessment for each of the 3 frequencies. Participants are also asked about the level of driving (less cognitive load from distraction task) and asked to rate the level of musical experience that they have (improved rhythmic ability).

Table 5.1: Stimulus Frequency Notification Dismissal

	1 Hz	1.25 Hz	1.67 Hz
Overall Dismissed	60%	55%	60%
Overall Time	4.5s	5.5s	4.9s
Drive/Music Dismissed	70%	74%	73%
Drive/Music Time	4.7s	5.2s	5.2s
No D/M Dismissed	12%	14%	26%
No D/M Time	6.0s	5.5s	5.6s

We evaluate each frequency by the user dismissal rate, average dismissal speed, and task load. There was not a strong relationship between notification dismissal rate ($r_1 = 60\%$, $r_{1.25} = 55\%$, $r_{1.67} = 66\%$), and the mean notification dismissal times for the dismissed notifications was similar ($t_1 = 4.6$s, $t_{1.25} = 5.5$s, $t_{1.67} = 4.9$s). Among the 4 participants who had either driving or musical experience, the mean dismissal rates was much higher ($r_{1exp} = 70\%$, $r_{1.25exp} = 74\%$, $r_{1.67exp} = 73\%$), but the mean dismissal time was similar

Figure 5.1: Stimulus Frequency NASA TLX



($t_{1exp} = 4.7$s, $t_{1.25exp} = 5.2$s, $t_{1.67exp} = 5.2$s). The best-performing participant, P2, who had both driving and musical experience, performed much better than average in terms of notification dismissal rate ($r_{1P2} = 91\%$, $r_{1.25P2} = 87\%$, $r_{1.67P2} = 97\%$) but not for dismissal speed ($t_{1P2} = 4.2$s, $t_{1.25P2} = 5.7$s, $t_{1.67P2} = 4.7$s). 2 participants had neither driving nor musical experience and performed much more poorly in terms of dismissal rate ($r_{1nexp} = 12\%$, $r_{1.25nexp} = 14\%$, $r_{1.67nexp} = 26\%$) and dismissal speed ($t_{1nexp} = 6.0$s, $t_{1.25nexp} = 5.5$s, $t_{1.67nexp} = 5.6$s). The worst performing participant, P4, dismissed 0% of 1 Hz notifications, 0% of 1.25 Hz notifications, and 35% of 1.67 Hz notifications at 5.8 s ($r_{1.67P4} = 35\%$, $t_{1.67P4} = 5.8$s). P4 mentioned that he/she found the task of syncing to the rhythmic stimulus very difficult, even during the prep sessions.

The results from the NASA TLX assessment are shown in Figure 5.1. The overall TLX showed that participants generally perceived slower frequencies to be less cognitively demanding ($M_1 = 43$, $M_{1.25} = 63$, $M_{1.67} = 66$). Mainly, participants found that slower frequencies induced less temporal demand ($M_{1T} = 11.6$, $M_{1.25T} = 12.8$, $M_{1.67T} = 20.9$). Thus, the 1 Hz frequency is chosen for subsequent studies.

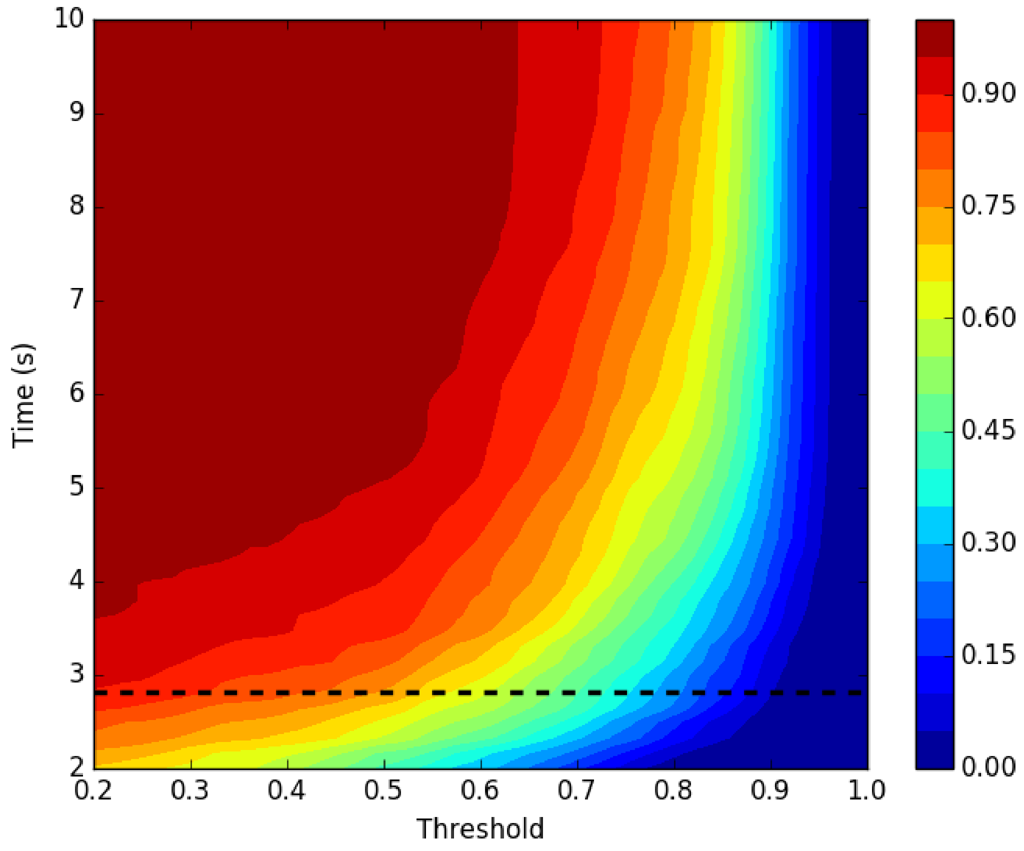
5.2.2 User Study - Gesture Comparison

A within-subjects study was conducted with a total of 4 experimental conditions (balanced by 4x4 Latin Square) to compare the synchronous gesture with a traditional swipe gesture in a semi-controlled environment. Similar to the frequency tuning study, participants performed two distraction tasks while dismissing incoming notifications - driving a car with a Logitech steering wheel in a driving simulator and walking while holding a filled cup of water. Participants are given a prep session of 25 notifications before each syncing session for them to become accustomed to a novel gestural input while performing the distraction task. The system is evaluated by its accuracy and detection speed in notification response scenarios with 12 participants (7M/5F, ages 20-26) in our institution's usability lab. The trigger threshold was set to a high value ($\rho_{thresh} = 0.85$) to enable post-hoc analysis of lower thresholds and to avoid early false-triggering. All participants were paid \$10 to complete the study, which lasted approximately one hour. Following participation in the notification response study, participants are asked to complete a NASA TLX assessment for both the syncing gesture and the swiping gesture.

The results of the notification response evaluation for each activity are shown in Figure 5.2 and 5.3 in contour plots showing the relationship between the precision (true-positive rate), time, and correlation threshold. The average time for the swiping gesture is shown as a black dotted line. Noise data is collected from the smartwatch during both activities when the user was not dismissing a notification.

Most participants found that notification dismissal with both gestures was more difficult during the sitting condition, due to the driving task. Swiping took 9.2% longer ($t_{sit} = 2.81$s, $t_{walk} = 2.57$s) and it took more time to reach the same accuracy for the syncing gesture.

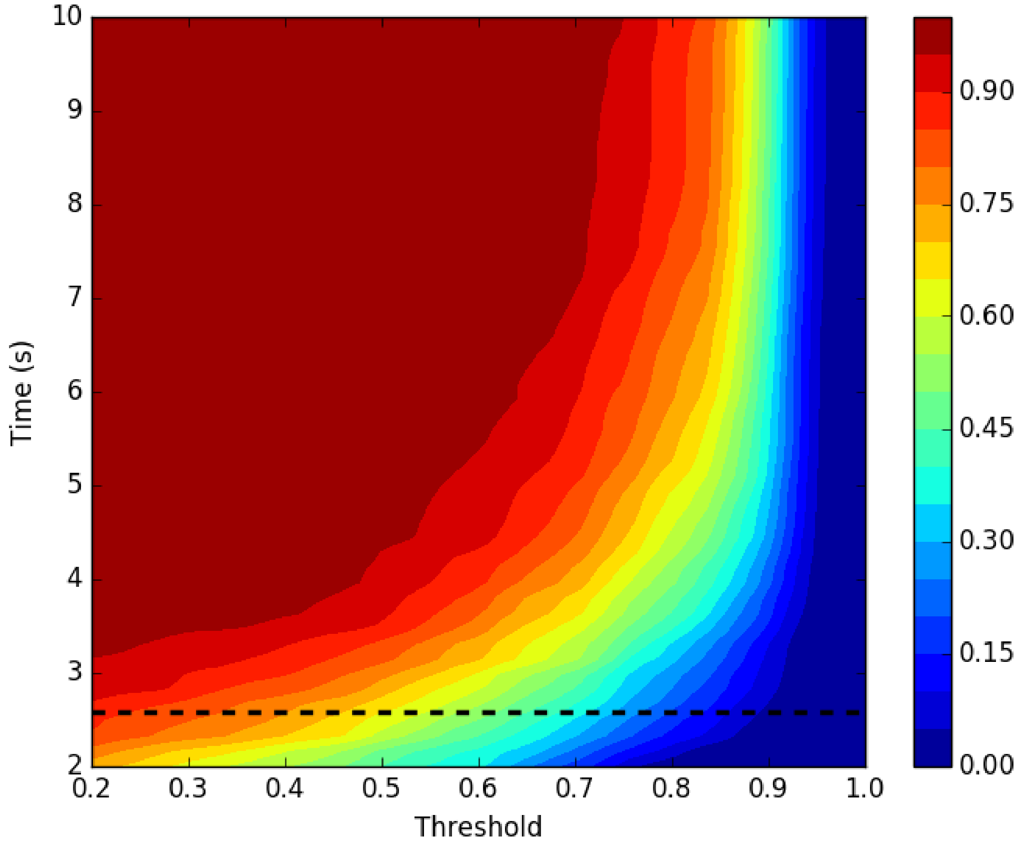
Figure 5.2: Sitting Precision vs Threshold vs Sync Time



However, the detector's false positive rate was much lower for the sitting condition (Figure 5.5), as the user did not perform any periodic motion. Using a threshold of $\rho_{sit} = 0.57$ with a false positive rate of 22 per hour, notifications can be dismissed in 4 seconds with 85% accuracy. While users found it easier to dismiss notifications while walking due to the less cognitively demanding distraction task and the availability of gaze attention, the motion of walking and swinging arms introduced many more false positives. Using a threshold of $\rho_{walk} = 0.73$ with a false positive rate of 37 per hour, notifications can be dismissed in 5.5 seconds with 85% accuracy.

Compared to the swiping gesture, the syncing gesture had a slightly higher overall score on the NASA TLX assessment ($M_{sw} = 44.2$, $M_{sy} = 50.7$). Notably, users found that swiping was less mentally ($M_{swM} = 3.2$, $M_{sy_M} = 8.9$) and temporally demanding ($M_{swT} = 6.0$, $M_{sy_T} = 11.1$), but was less physically demanding ($M_{swP} = 14.3$, $M_{sy_P} =$

Figure 5.3: Walking Precision vs Threshold vs Sync Time



7.3). Other dimensions were similar for both gestures. For users with more rhythmic ability (through playing musical instruments) or experience with the synchronous gesture, we hypothesize that the gesture will require less overall concentration.

5.2.3 Discussion - Smartwatch Only Interaction

We evaluate the SeeSaw interface in a best-case sitting activity with little movement and a worst-case walking activity with large amounts of periodic movement. Results show that using different correlation thresholds, the SeeSaw detector can be used for rapid, one-handed, and gaze-free notification dismissal for both. A simple pose-detector using the smartwatch orientation can be used to determine the current activity and adjust the correlation threshold value.

While the swipe gesture was faster for both tasks, this can be attributed to most users'

Figure 5.4: Gesture Comparison NASA TLX

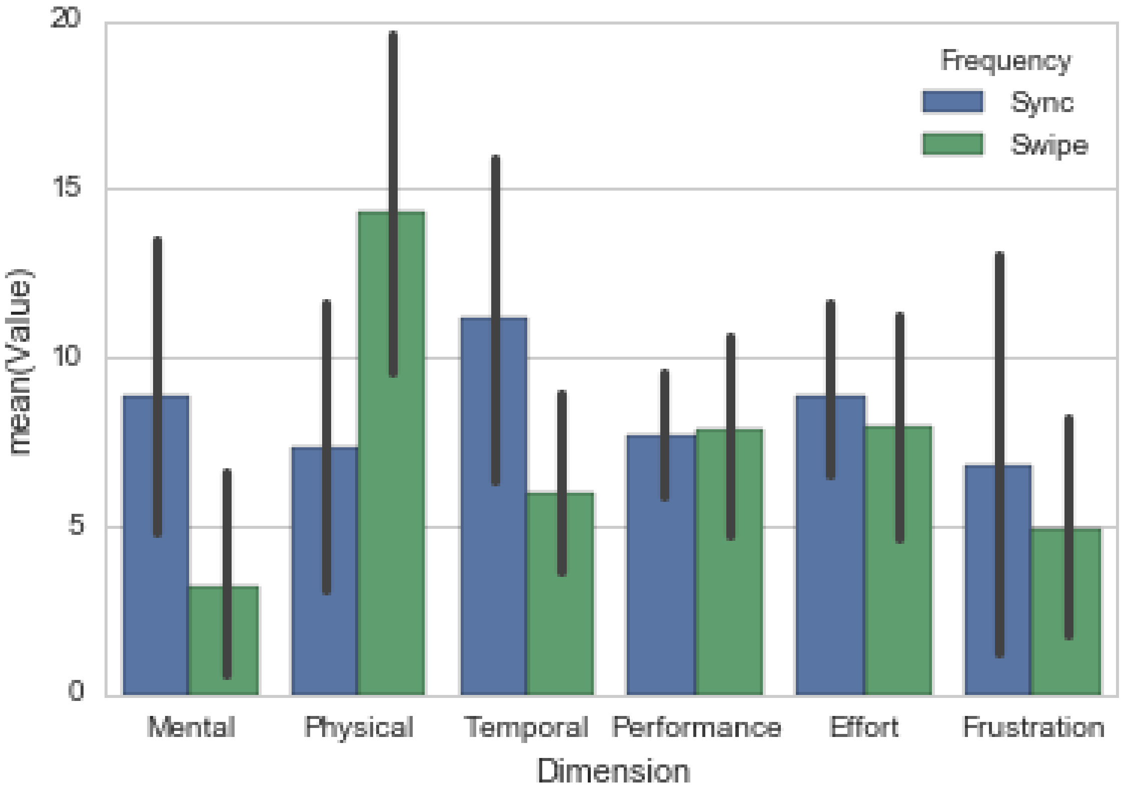
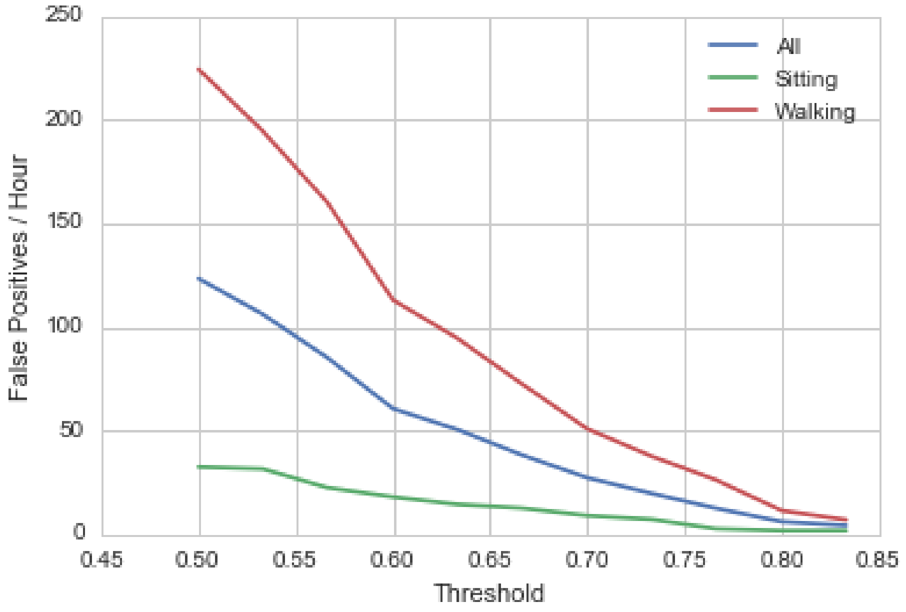


Figure 5.5: False Positive Rate for Correlation



familiarity with the gesture and touch-screen interfaces. SeeSaw is able to address many limitations of traditional systems. Sometimes, when holding the cup of water while walking, participants spilled some of the water from the cup. The amount of water spilled by each participant was not recorded, but when water was spilled on the touchscreen, participants noted that the accuracy of the swipe applications suffered dramatically. Participants who became familiarized with the haptic stimulus were able to dismiss notifications without needing to change focus to the watch screen. Overall, the SeeSaw interaction provides an effective alternative to traditional swipe interfaces on smartwatches. SeeSaw can be used to replace or augment touch-interfaces for notification-style applications and micro-interactions.

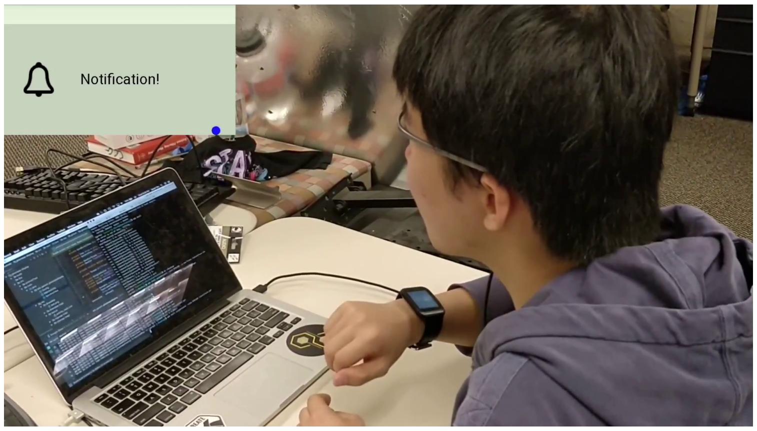
CHAPTER 6
SMARTWATCH & HWD EVALUATION

In this section we describe the evaluation of SeeSaw as an interaction for head-worn displays (HWD).

6.1 Smartwatch & HWD System

A smartwatch & HWD system is constructed to allow applications on a head-worn computer to receive input from a smartwatch. A Sony Smartwatch 3 SWR50 smartwatch is used to collect and process sensor data, and a Google Glass is used as a head-worn display (Figure 6.1). Data and commands are sent wirelessly between the devices using a UDP connection. A flashing stimulus is displayed on the HWD and the user moves the wrist in sync to dismiss the notification. Sensor data is collected and processed on the watch using the autocorrelation algorithm. Once the threshold is reached, the smartwatch sends a signal to the HWD application.

Figure 6.1: SeeSaw Interface for HWD



6.2 Evaluation

6.2.1 User Study - Notification Response

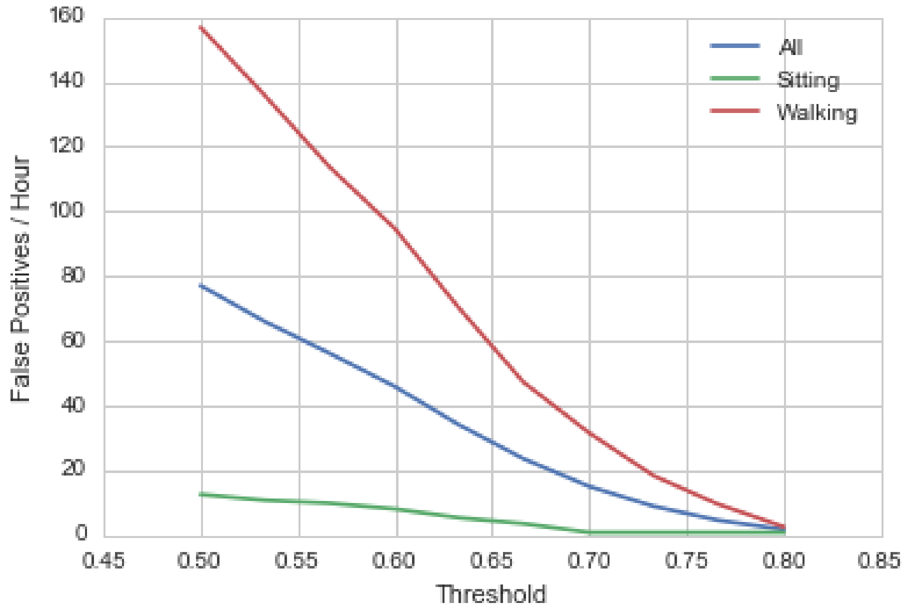
A total of 6 expert participants were recruited from the pool of participants in the watch-only notification response study. The notification response study for HWD interaction took around 45 minutes and participants were paid $10. The study was conducted with the sitting/driving condition used in the previous frequency tuning and gesture comparison studies, and participants were given two sessions of 30 notifications randomly triggered every 20 seconds. The first session was used for practice and familiarization with the system while the second was used for evaluation. The trigger threshold was set to $\rho_{thresh} = 0.8$ to avoid early false-triggering.

Table 6.1: Smartwatch & HWD Notification Dismissal

	HWD Dismissed	HWD Time
P1	93.3 %	2.7s
P2	90.0 %	3.2s
P3	93.3 %	3.5s
P4	100 %	4.7s
P5	100 %	3.2s
P6	93.3 %	4.0s
Overall	95.0 %	3.6s

Table 6.1 shows the notification dismissal rate and mean dismissal time for the study. All participants achieved over 90% accuracy on notification dismissal, and the overall mean dismissal time is around 1 second less than the smartwatch-only system. Although the system was only evaluated for the sitting condition, noise data collected from both activities from the gesture comparison study was used to calculate the false positive rate for different thresholds (Figure 6.2).

Figure 6.2: False Positive Rate for Autocorrelation



6.2.2 Discussion - Smartwatch & HWD Interaction

Overall, the smartwatch & HWD setup is very effective for rapid notification dismissal and one-handed interaction. The notification response evaluation shows a high dismissal rate of 95% and mean dismissal speed of 3.57 s. The fastest participant, P1, dismissed notifications more quickly than the mean swipe dismissal time from the previous study ($t_{glass\,P1} = 2.69$s, $t_{swipe} = 2.81$s).

Some participants reported that the addition of a HWD was not helpful for performing the repetitive wrist-tilting gesture, but that it helped them notice and respond to notifications faster. Other participants commented that the addition of the HWD display made the syncing gesture easier, as tilting the wrist did not move the visual stimulus away, out of view.

Unlike the stimulus correlation algorithm used for the smartwatch-only interaction, the smartwatch & HWD system uses autocorrelation. While this is less expressive and does not support binary selection by syncing in-phase and out-of-phase, it is still sufficient for notification dismissal and has some notable advantages. The correlation algorithm relies on little or no latency between the stimulus rendering and sensor processing. Since the stimulus and data are processed on two different devices, the additional network delay could worsen performance. The no-stimulus autocorrelation algorithm correlates a window of sensor data with the previous window, stored on the same device. Moreover, the autocorrelation algorithm was more forgiving in the range of motions it accepted, as it does not require the users to tilt their wrist in a smooth, sinusoidal motion and allows any periodic movement - even gross and abrupt jerks, to reach a high threshold.

CHAPTER 7
CONCLUSION

7.1 Limitations and Future Work

SeeSaw is well-suited as a complementary input method for smartwatches and HWDs. Depending on the activity and environmental noise, the SeeSaw detector system is able to achieve different levels of precision and false-positive rate. During the user studies conducted, the detector was set to trigger at a constant threshold. Additional work can be done by building a pose-detector or activity recognizer that could dynamically adjust the triggering threshold. This would be aided by evaluating the detector across a wider range of noise data from an in-the-wild data set.

Moreover, we intend to explore the use of the autocorrelation algorithm for smartwatch-only notification dismissal, given the good performance and low false positive rate of the algorithm for the smartwatch & HWD system. The expressivity of the HWD system can also be extended using the correlation algorithm.

A primary advantage of the syncing gesture is its ability to be performed subtly with one-hand. In addition to evaluating the gesture's effectiveness in different scenarios by examining the false positive data, we intend to explore the social acceptability of the gestures in different scenarios using a user perception study or a social acceptability rating scale. Finally, we intend to further explore the usefulness of SeeSaw as an activation gesture by pursing its application as a gaze-free interaction. We intend to explore gaze-free interaction by using a user-defined stimulus signal that allows users to simultaneously define the gesture stimulus on body part and perform the syncing motion on another. We hypothesize that the resulting gesture interaction will require lower concentration and achieve higher accuracy.

7.2 Conclusion

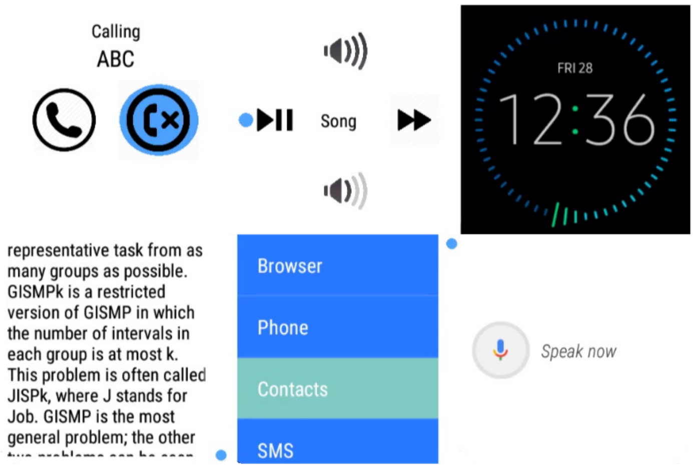
In this paper, we presented SeeSaw, a synchronous input interface for smartwatches that enables rapid, one-handed input for commodity smartwatches without any hardware or software modifications. In contrast to many gesture interfaces that detect user intent through gesture classification, the synchronous gesture interface uses flashing or vibrating stimuli to allow users to select UI elements or respond to notifications. We evaluated SeeSaw as a smartwatch-only interaction and found that users were able to dismiss incoming notifications from 4.5s - 5.5s using the one-handed, gaze-free interaction. During our evaluation of the smartwatch and HWD system, we found SeeSaw to provide excellent accuracy (95%) and dismissal speed (3.6s) with a low false positive rate. Finally, we demonstrate the integration of SeeSaw in a suite of example applications.

Appendices

APPENDIX A
DEMONSTRATION APPLICATIONS

We present a suite of demonstration applications to showcase the capabilities of the SeeSaw synchronous gesture interface. In particular, we categorize common smartwatch interactions into three main categories, as described in Section 3.2 and create applications for each case using the SeeSaw interface. The applications included are inspired by demonstrations by prior work in motion correlation and synchronous gesture interfaces[1, 5].

Figure A.1: Demo Applications for Binary Input (Left), Multi-Target Input (Center), and Activation (Right)



representative task from as many groups as possible. GISMPk is a restricted version of GISMP in which the number of intervals in each group is at most k. This problem is often called JISPk, where J stands for Job. GISMP is the most general problem; the other

A.1 Binary Input

Dismiss Phone Call: This phone call dismissal application simulates an incoming call notification and allows the user to take the call by opening a calling app on a paired smartphone or to dismiss the call by sending it to voicemail. A pair of flashing icons allows the user to use a *notification gesture* to select an option. Ignoring the notification for a certain amount of time triggers the default behavior of sending the call to voicemail.

Text Viewer: In addition to allowing users to react to simple notifications, binary selection is sufficient for displaying and navigating text content and basic media. In this application, a text viewer application displays a large block of text inside of a scrollable text element. A pair of flashing targets is displayed vertically on the application scrollbar. Scrolling up or down is achieved by syncing wrist movement to either of these targets as a *command gesture*. Text navigation can also be implemented by mapping the Next Page and Previous Page commands to each target, for a potentially more intuitive ”page flip” metaphor.

A.2 Multi-Target Selection

Music Player: A music player application is created with four controls - Play/Pause, Next Song, Volume Up, and Volume Down. Similar to the the Text Viewer application, *command gestures* are used to control the application, but a single pair of flashing elements is only able accommodate a maximum of two controls. In this case, controls in the music player are organized into two control groups arranged in a cross - the playback group (Play/Pause and Next Song) and the volume group (Volume Up and Volume Down). A separate wrist-flick gesture, which is integrated into the Android operating system, is used to toggle between these two control groups. The location of the flashing targets are used to indicate the control group that is currently enabled.

Application Menu: An application menu allows the user to launch an application or perform an operation from a list of commands. The menu contains four controls (Browser, Phone, Contacts, and SMS) that are vertically arranged in a list. A cursor blinking at 0.5 Hz is used to indicate the item in the list that is currently selected. To confirm the current selection, the user performs a synchronous gesture using the blinking cursor as a stimulus. A separate pair of flashing elements (1 Hz) is displayed vertically on the application sidebar. Similar to the Text Viewer application, moving the cursor is achieved by syncing to either of these two targets.

A.3 Activation Gesture

Voice Assistant Activation: Many smartphones and smartwatches feature voice assistants that allow users to interact with applications using spoken natural language. A major drawback of voice assistants is their need for an ”always listening” feature to detect certain trigger phrases. This privacy concern can be addressed by replacing the voice based activation mechanism used by most voice assistants with a synchronous gesture interface. Furthermore, an activation mechanism based on a synchronous gesture interface would consume less energy, as a much lower sampling rate is required. The Voice Assistant Activation application features a watch face that serves as the default screen for the smartwatch. Performing a synchronous gesture at 1 Hz at the default screen will launch the default voice assistant application and cause the device to begin listening for speech input.

Application Menu Activation: The Application Menu Activation demo shows that is possible to construct a fast, one-handed workflow for smartwatches using a synchronous gesture interface. Similar to the Voice Assistant Activation application, the Application Menu Activation application features a watch face that is used as the smartwatch's default screen. Performing a synchronous gesture at 1 Hz launches the Application Menu which also uses a synchronous gesture interfaces to launch applications on the smartwatch or a smartphone.

REFERENCES

[1] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen, “Orbits: Gaze interaction for smart watches using smooth pursuit eye movements,” in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, 2015, pp. 457–466.

[2] D. L. Ashbrook, *Enabling mobile microinteractions*. Georgia Institute of Technology, 2010.

[3] C. Zhang, X. Wang, A. Waghmare, S. Jain, T. Ploetz, O. T. Inan, T. E. Starner, and G. D. Abowd, “Fingorbits: Interaction with wearables using synchronized thumb movements,” in *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ACM, 2017, pp. 62–65.

[4] M. Carter, E. Velloso, J. Downs, A. Sellen, K. O’Hara, and F. Vetere, “Pathsync: Multi-user gestural interaction with touchless rhythmic path mimicry,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 2016, pp. 3415–3427.

[5] G. Reyes, J. Wu, N. Juneja, M. Goldshtein, W. K. Edwards, G. D. Abowd, and T. Starner, “Synchrowatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 4, 158:1–158:26, Jan. 2018.

[6] D. L. Ashbrook, J. R. Clawson, K. Lyons, T. E. Starner, and N. Patel, “Quickdraw: The impact of mobility and on-body placement on device access time,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008, pp. 219–222.

[7] H. Wen, J. Ramos Rojas, and A. K. Dey, “Serendipity: Finger gesture recognition using an off-the-shelf smartwatch,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 2016, pp. 3847–3851.

[8] G. Laput, R. Xiao, and C. Harrison, “Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers,” in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ACM, 2016, pp. 321–333.

[9] J. Gong, X.-D. Yang, and P. Irani, “Wristwhirl: One-handed continuous smartwatch input using wrist gestures,” in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ACM, 2016, pp. 861–872.

[10] A. Dementyev and J. A. Paradiso, "Wristflex: Low-power gesture input with wrist-worn pressure sensors," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*, ACM, 2014, pp. 161–166.

[11] Y. Zhang and C. Harrison, "Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, 2015, pp. 167–173.

[12] S. E. Hudson, C. Harrison, B. L. Harrison, and A. LaMarca, "Whack gestures: Inexact and inattentive interaction with mobile devices," in *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, ACM, 2010, pp. 109–112.

[13] E. Ghomi, G. Faure, S. Huot, O. Chapuis, and M. Beaudouin-Lafon, "Using rhythmic patterns as an input method," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2012, pp. 1253–1262.

[14] D. Boland and R. Murray-Smith, "Finding my beat: Personalised rhythmic filtering for mobile music interaction," in *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, ACM, 2013, pp. 21–30.

[15] E. Velloso, M. Carter, J. Newn, A. Esteves, C. Clarke, and H. Gellersen, "Motion correlation: Selecting objects by matching their movement," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 24, no. 3, p. 22, 2017.

[16] C. Clarke, A. Bellino, E. Velloso, H. Gellersen, *et al.*, "Tracematch: A computer vision technique for user input by tracing of animated controls," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 2016, pp. 298–303.

[17] E. Velloso, M. Wirth, C. Weichel, A. Esteves, and H. Gellersen, "Ambigaze: Direct control of ambient devices by gaze," in *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, ACM, 2016, pp. 812–817.

[18] J. Rico and S. Brewster, "Usable gestures for mobile interfaces: Evaluating social acceptability," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2010, pp. 887–896.

[19] E. Costanza, A. Perdomo, S. A. Inverso, and R. Allen, "Emg as a subtle input interface for mobile computing," in *International Conference on Mobile Human-Computer Interaction*, Springer, 2004, pp. 426–430.

[20] D. Ashbrook, C. Tejada, D. Mehta, A. Jiminez, G. Muralitharam, S. Gajendra, and R. Tallents, “Bitey: An exploration of tooth click gestures for hands-free user interface control.,” in *MobileHCI*, 2016, pp. 158–169.

[21] A. Bedri, D. Byrd, P. Presti, H. Sahni, Z. Gue, and T. Starner, “Stick it in your ear: Building an in-ear jaw movement sensor,” in *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ACM, 2015, pp. 1333–1338.

[22] J. Lee, H.-S. Yeo, M. Dhuliawala, J. Akano, J. Shimizu, T. Starner, A. Quigley, W. Woo, and K. Kunze, “Itchy nose: Discreet gesture interaction using eog sensors in smart eyewear,” in *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ACM, 2017, pp. 94–97.

[23] C. Zhang, X. Wang, A. Waghmare, S. Jain, T. Ploetz, O. T. Inan, T. E. Starner, and G. D. Abowd, “Fingorbits: Interaction with wearables using synchronized thumb movements,” in *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ser. ISWC ’17, Maui, Hawaii: ACM, 2017, pp. 62–65, ISBN: 978-1-4503-5188-1.