AccessWear: Making Smartphone Applications Accessible to Blind Users

Prerna Khanna
pkhanna@cs.stonybrook.edu
Stony Brook University, USA

Shirin Feiz
sfeizdisfani@cs.stonybrook.edu
Stony Brook University, USA

Jian Xu
jianxu1@cs.stonybrook.edu
Stony Brook University, USA

IV Ramakrishnan
ram@cs.stonybrook.edu
Stony Brook University, USA

Shubham Jain
jain@cs.stonybrook.edu
Stony Brook University, USA

Xiaojun Bi
xiaojun@cs.stonybrook.edu
Stony Brook University, USA

Aruna Balasubramanian
arunab@cs.stonybrook.edu
Stony Brook University, USA

Abstract

In this paper, we present *AccessWear*, a system that improves the accessibility of smartphone touchscreen interactions for blind users using smartwatch gestures. Our system design is human-centered, namely, it incorporates the design goals that were learned from a formative user study with 9 blind participants. The formative study showed that blind users liked the idea of using smartwatch gestures as an alternative: 4 participants liked that when using smartwatch gestures, they did not have to bring their expensive phones out in public and 6 participants liked that smartwatch gestures can be performed with one-hand, as the other hand is usually occupied in holding a cane or a guide dog. Even though there are several advantages to smartwatch gestures, our study also shows that gestures performed by blind users have different patterns compared to sighted users, making gesture recognition more challenging. To this end, AccessWear makes two contributions. The first is a gesture recognition system that works specifically for blind users that is lightweight and does not require per-person training. The second is a near-zero-effort gesture replacement system that does not require any changes to the original application. AccessWear uses input virtualization techniques so that a given gesture can replace the touchscreen input seamlessly. We implement AccessWear on an Android smartphone and Android watch. We perform a quantitative and qualitative study with 8 blind participants. Our study shows that AccessWear can recognize gestures with a 92% accuracy and the end-to-end latency when using an alternate gesture was 53 msec on average. The qualitative study shows that when participants perform a task, consisting of a series of gestures, the system is robust, does not have perceived delays, and does not add physical or mental load on the users.

CCS Concepts

• **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**; **Accessibility systems and tools**; **Gestural input**.

Keywords

Gesture recognition, Accessibility, Mobile Systems, Virtualization

ACM Reference Format:
Prerna Khanna, Shirin Feiz, Jian Xu, IV Ramakrishnan, Shubham Jain, Xiaojun Bi, and Aruna Balasubramanian. 2023. AccessWear: Making Smartphone Applications Accessible to Blind Users. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3570361.3592495

1 Introduction

There are over 1 million blind users in the US [12]. Blind users interact with their phones using touchscreen, voice, external keyboard, braille keyboard [23, 45] or use the edge of the phone as a guide [43]. However, the most typical interaction mode for information-seeking and content-exploration tasks is to use touchscreen gestures.

Unfortunately, touchscreen interactions are challenging—some gestures (i.e. two or three-finger swipe) require both hands for interactions which is hard when holding a cane or a guide dog [1], users are susceptible to shoulder-surfing attacks [54], and gestures can become overloaded [32]. Blind users do use voiced interactions as an alternative, largely for text entry tasks. But voice interactions are unreliable in noisy environments, can compromise the privacy of a user, and are prone to errors [8].

In response, we design AccessWear, a system that allows blind users to flexibly and universally use alternate gestures to replace touchscreen gestures, especially for content seeking tasks. For example, a user can flick their wrist to the right instead of taking their phones out and swiping right on the screen. A different user may use a different gesture, such as moving their forearm up, to swipe right on the screen. To achieve this goal, we design (1) a new gesture recognition system specifically designed for blind users, and (2) a near-zero effort system so that any user can replace a touchscreen gesture with an alternate gesture, without requiring any changes to the application.

Existing gesture recognition systems require large amounts of training data [22, 56], need personalized learning [52], or require specialized hardware or high-end computation [30, 52] (see Table 4). Further, our formative study, with 9 blind participants, found that blind users perform gestures differently compared to sighted users, and existing gesture recognition systems do not work well for blind users.

AccessWear captures short hand and wrist gestures with IMU sensors, available in all smartwatches. IMU sensors are made up of gyroscope, accelerometer, and magnetometer sensors. We find that accelerometer and magnetometer sensors introduce additional noise. Gyroscope, on the other hand, is less noisy for short gestures and are ideal to capture hand and wrist movements which are primarily rotational. When analyzing data from the gyroscope, we observe that even when the overall gestures are different across different users, there is a distinct nucleus that remains the same. We isolate this distinct nucleus buried inside the gesture using lightweight algorithms and compare its similarity with predefined nucleus templates to identify the correct gesture in real-time. This low-cost gesture recognition system works with only gyroscope data, and does not require per-user model training or large amounts of training data. Further, our evaluations show that gyroscope alone is sufficient to identify short-range hand and wrist gestures, without the need for expensive sensor fusion techniques that combine information from all three IMU sensors (see Figure 13).

For zero-effort gesture replacement, we design an *input virtualization* technique that decouples the application interaction from the application logic. Most mobile operating systems are designed such that applications receive sensor inputs from a single sensor service. AccessWear creates a virtual sensor service that can virtually generate a gesture and feed it to the application as though it was generated locally. This allows AccessWear to work without requiring changes to the application or design specially handcrafted applications [33]. AccessWear then uses a simple metaprogram specification to specify alternate gesture mappings.

We implemented AccessWear on an Android smartphone and an Android smartwatch. In our first set of evaluations, we measured the quantitative and qualitative performance of AccessWear based on a user study with 8 blind participants (8 out of the 9 participants from the formative study returned for the evaluation study). During the user study, we asked the participants to map five smartwatch gestures to the smartphone touchscreen gestures of their choice. They then performed an exploratory task requiring them to search for an object on the screen using the alternative smartwatch gestures. AccessWear end-to-end latency between performing a gesture on the watch to observing the result on the smartphone was 53 msec on average, well under the reaction time of users [21]. The accuracy of gesture recognition in controlled settings is 92% across all gestures and all blind users. Based on an exit survey, all users found that AccessWear was robust, easy to use, and was not physically or mentally demanding.

Performing real-world, long-duration, studies with blind users is challenging. Instead, we conduct a user study with 5 sighted participants in real-world settings to evaluate the technical capabilities of AccessWear. The study shows that the false positive rate when using AccessWear during everyday activities is low and AccessWear works with several common applications without requiring application changes.

2 Formative study

We conducted an IRB-approved study to characterize the challenges faced by blind users when interacting with their smartphones. As part of the study, we also wanted to understand if smartwatch gestures are a good replacement for touchscreen interactions. To this end, we (i) solicit feedback from blind users on the use of alternate smartwatch gestures, and (ii) capture smartwatch gestures performed by blind users (and also sighted users for comparison) to analyze how well we can recognize smartwatch gestures.

Blind users interact with smartphones predominantly using the touchscreen, especially for content-seeking tasks, and they use screen readers (such as TalkBack [27] for Android and VoiceOver [7] for iPhone) to read out the content of the screen. To this end, we focused on touchscreen interaction using screen readers for this study.

Participants: We recruited 9 blind participants between the ages of 22-61 (17 male and 8 female). Table 7 in Appendix 1 shows the demographics and other details about the participants. All blind participants were completely blind and knew how to use a screen reader. None of the participants had any motor impairments that affected their interaction with smartwatch gestures.

In addition, we recruited 16 sighted participants who were only asked to perform smartwatch gestures, to compare gestures performed by blind vs sighted users. All other studies were conducted only for blind participants.

Apparatus: During our training, we used iPhone 8 plus to demonstrate the screen reader (VoiceOver) actions and touchscreen actions [6]. All users were asked to wear Fossil Gen 5 smartwatch. When the participants performed gestures, the IMU data was streamed via Bluetooth to the phone. It took 1.5 hours to conduct the user study and each blind participant was paid $100 for their time.

Design: The experiments for the first part of the study (with only blind participants) consisted of asking the participants to interact with their smartphones using touchscreen gestures. During the study, we asked the participants their experience in using touchscreen interactions and their preferences when using smartwatch gestures. In addition, we designed a comparative experiment in which we studied how participants (both blind and sighted) perform smartwatch gestures. We divided our sighted participants (16 in total) into two random groups of size 8 with participants of one group receiving a visual demonstration of the gestures during the training while the second group received a verbal description of the gestures. We repeated the same experiment with the blind participants all of whom received verbal descriptions similar to the ones given to the sighted participants.

We asked each participant to perform 10 gestures in a counterbalanced order, repeating each gesture 3 times. The gestures were forearm directional gestures, shape-related gestures, finger tapping gestures, and wrist gestures. We selected these gestures because they are common gestures used in similar studies [19, 52].

Procedure: The study starts with a survey regarding the smartphone and smartwatch interaction habits of the participants. See Appendix 1, Table 7 for a summary of user responses. After conducting the gesture experiments described above, a semi-structured interview was conducted.

2.1 Key Takeaways

1: Need for alternate gestures: Our study largely corroborates related literature regarding smartphone interaction challenges for blind users [16, 54]. All of the users found at least one touchscreen interaction difficult to perform and more than half found at least one of the touchscreen interactions difficult to remember. For example, 2 participants found selecting the first and the last item on the screen difficult and 2 of them found finger swipes confusing.

"Gives me the confidence to not take out the phone, everyone carries a watch, if I wave no one will know I am using a phone"
"In public, I don't feel comfortable with the phone, I don't want to draw much attention, swipes and all gestures are less noticeable"
"More security, keep phone inside the pocket."
"Convenient, easy access"

Table 1: Some user responses to the question "The advantages of performing gestures using a smartwatch?"

Forearm and Wrist	Upwards, Right, Left, Downwards, Flick wrist	7
Shape	In air square, Circle	5
Finger	Pinch, finger taps	7
Wrist rotations	Left rotate, Right rotate	3

Table 2: Median preference scores for different gesture categories (7: high preference, 1: low preference).

When specifically asked about the advantages of using a smartwatch to interact with applications, 4 participants mentioned that they feel more secure if they don't have to take their expensive phones out in public places and 6 participants liked that they can perform one-handed interactions using the smartwatch. Table 1 summarizes a few additional responses. When specifically asked about perceived disadvantages of using a smartwatch, 4 participants did not find any disadvantage, 3 did not want to perform elaborate gestures that will attract attention to them, 1 participant was concerned about false positives, and 1 participant was concerned about forgetting to wear the watch.

2: Gesture preference: Users prefer finger and forearm gestures over shape gestures and wrist rotation gestures. Table 2 shows the median ratings given for different categories of gestures, where 7 represents high preference and 1 represents low preference. The interquartile range (IQR) of scores for these gestures was low, i.e. 0, 1, 0, 0 respectively, indicative of less variance across the users. A non-parametric Friedman test of differences was performed to compare the ratings for the four gesture categories. There was a significant difference between the categories (${X_r}^2(3) = 23.73, p < 0.001$). We use the five forearm and wrist gestures in Table 2 when implementing and evaluating AccessWear because they were highly preferred by participants in our study.

3: Blind users perform gestures differently: Figure 1 shows the same gesture (flick) performed by 2 sighted users with visual training and 2 blind users with verbal training. Visually, the gestures performed by sighted users are similar, but those performed by blind users are different. We measure the similarity between the gestures by estimating the distance between each gesture and a predefined gesture

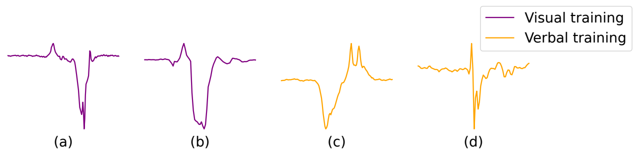


Figure 1: Gyroscope data of flick gesture for two sighted users (a), (b) who were shown the gesture visually, and two blind users (c) and (d). There is a larger variation in how gestures are performed by blind users compared to sighted users.

template. The distance is measured using Dynamic Time Warping (DTW) that aligns the two gestures and then estimates the euclidean distance between the two temporal sequences [39].

The average distance between the gestures performed by blind users and the standard template is 65.2. Even among blind users, there are large differences. If we use one of the blind user's gestures as the template and compare it with all others, the average distance is 77.05. In contrast, the distance between the gestures performed by sighted users (with visual training) is 29.8. A significance test (ANOVA) confirms that there is significant dis-similarity between gestures of blind and sighted users with visual training. The reason for this difference is likely because of the difference between verbal and visual descriptions. For sighted users who were only given a verbal description of the gesture, the distance between the individual gesture and the gesture template increases to 66.3.

The takeaway is that existing gesture recognition techniques [22, 56] that use training data from sighted users are unlikely to work well for blind users.

4: Users have different mapping preferences: Finally, we asked the participants to map alternate smartwatch gestures to replace touchscreen interactions based on their preferences. There was large variability in how users mapped gestures. For example, for selecting the previous item on the screen, typically performed using the touchscreen left swipe with one finger, 44% of blind participants wanted to use a "forearm left" gesture while 22% of the participants wanted to use the "forearm right" gesture. Table 3 shows the gesture mappings picked by 9 blind participants for 5 commonly used screen reader actions.

3 AccessWear Challenges

The goal of AccessWear is to make it significantly easy for blind users to interact with smartphone applications and screen readers using smartwatch gestures. AccessWear focuses on content-seeking applications such as browsing or listening to music which are popular among blind users (see Table 7 Appendix 1). Content creation applications such as

Screen Reader action	Gestures
1. Select and speak an item	Flick wrist (44%, 4/9), Finger: pinch/tap (33%, 3/9), Forearm: up/ down (22%, 2/9)
2. Select previous item	Forearm: left (44%, 4/9), Forearm right: (22%, 2/9), Finger: left (22%, 2/9), Flick wrist (11%, 1/9)
3. Select next item	Forearm: right (55%, 5/9), Finger: right (22%, 2/9) Others (22%, 2/9)
4. Scroll up a page	Forearm: up (55%, 5/9), Flick wrist (22%, 2/9), Finger: up (11%, 1/9) Forearm: down (11%, 1/9)
5. Scroll down a page	Forearm: down (44%, 4/9), Flick wrist (33%, 3/9), Finger: down (11%, 1/9) Forearm: up (11%, 1/9)

Table 3: Variety of gesture mappings chosen for performing screen reader actions across 9 blind users.

	Sensors	No T	No P	M
ViBand [30], [36]	A	✗	✗	✗
uWave [34], [13]	A	✗	✗	✓
[2], [28], [29]	A	✗	✓	✗
[37]	A	✗	✓	✓
Serendipity [52]	A, G	✗	✗	✗
RisQ [40]	A, G, M	✗	✓	✓
ArmTrak [47]	A, G, M	✓	✗	✗
MUSE [46]	A, G, M	✓	✓	✗
[53]	A, G	✗	✗	✗
LimbMotion [60]	A, G, Acoustic	✗	✗	✓
ArmTroi [35], [61], [56]	A, G	✗	✗	✓
Tapnet [22], [4]	A, G	✗	✓	✓
[25]	A, G	✗	✓	✗
AccessWear	**G**	✓	✓	✓

Table 4: Previous works in gesture recognition using IMU. A: accelerometer, G: gyroscope, M: magnetometer. No T: No training, No P: No personalization, and M: Runs on mobile

writing emails or messages require keyboard input and are not a focus of this work. There are two main challenges in designing AccessWear:

Gesture recognition challenges. Despite considerable work in gesture recognition, real-time gesture recognition on resource-constrained devices is still an open problem. Table 4 shows the related work on gesture recognition that uses IMU sensors readily available on smartwatches (the related work is expanded in §9). These works have one or more of the following limitations—they require large amounts of training data, require per-person training, or cannot run on a mobile device.

For example, the state-of-the-art gesture recognition system TapNet [22] requires large amounts of data training data (135K samples). In fact, we trained TapNet with data collected from blind users and found that they work poorly with limited training data (§7.2). Collecting data for blind users on a large scale to train deep learning models is expensive and time-consuming. Further, our formative study shows that blind user gestures have a large variance; therefore using training data is likely insufficient. Other works such as Serendipity [52] are trained with relatively smaller amounts of data (600 samples from each sighted user), but require per-person training before the system can be used. Per-user training has a huge overhead in terms of designing a data collection system catered to work with blind users.

Besides the above challenge, AccessWear's gesture recognition system should be lightweight so that it can run on mobile devices. Previous works [46, 52] that use intensive signal processing techniques or compute heavy particle filters cannot run on mobile devices.

Gesture replacement challenges. Related works that design alternate gestures for smartphone interactions [19, 33] require that applications be rewritten to work with alternate gestures. This severely limits the applicability and usefulness of the system. Our goal in AccessWear is to design a near-zero-effort gesture replacement, where users can specify how they want to map alternate smartwatch gestures to touchscreen gestures, but the application itself does not need to be modified. The key problem is that in smartphone applications, the application logic and the interactions are coupled, making it hard to change the interaction without modifying the application.

To this end, in AccessWear, we design (i) a robust gesture recognition system that works well with simple sensors available on a smartwatch with no per-user training or large training datasets, and (ii) an input virtualization system that allows a user to flexibly use the smartwatch gestures to interact with unmodified applications.

4 AccessWear Gesture Recognition

We focus on gesture recognition for forearm and wrist gestures. In §8 we show that the techniques can be extended to other gestures.

4.1 Insights

1. Using gyroscope data only: AccessWear's gesture recognition uses Inertial Measurement Unit or IMU sensors that are commonly available in most devices including smartwatches. IMUs consist of three sensors: gyroscope, magnetometer, and accelerometer, and many gesture recognition systems fuse information from all three sensors. However, magnetometer readings are known to be affected by ferromagnetic interference [46]. In our experimental environment, the standard deviation of magnetic field density variation reaches over 24 μT, which can result in a tracking error of over 50 degrees. Similarly, the accelerometer data is noisy compared to gyroscope data because it captures both linear and gravity motion. Figure 2 shows the accelerometer and gyroscope data when a flick gesture was performed by three blind users; visually, the accelerometer data across users is more dissimilar compared to the gyroscope data.

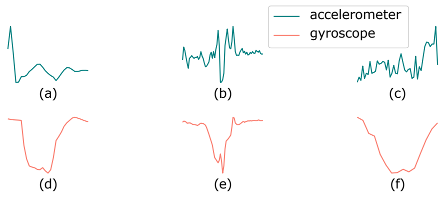


Figure 2: Accelerometer and gyroscope data for the flick gesture across 3 different users. Visually, the accelerometer data is considerably different across users compared to the gyroscope data.

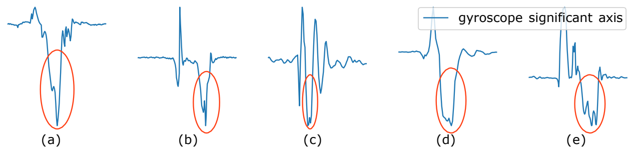


Figure 3: The significant axis of gyroscope data when the flick wrist gesture is performed by 5 different users. The nucleus looks similar (marked in red), but Pre-stroke and Post-stroke look visually dissimilar.

To address the shortcomings of the accelerometer and magnetometer, our system relies only on the gyroscope sensor. The insight here is that human arm and hand movements are primarily rotational since their motion is restricted to the surface of a sphere that is centered at the shoulder, elbow, or wrist. This motion can be captured using a gyroscope even when the gesture is seemingly linear. Further, by doing so we avoid the computational and power overhead of sensor fusion techniques.

The one disadvantage of the gyroscope is the drift. However, our goal is to identify short-duration gestures and not track gestures over time, making drift a non-issue.

2. Identifying nucleus: Each individual has a unique style in which they move their arm and hands. Upon observing the gyroscope data of different users at a more granular level, we find that despite the overall differences in gesture patterns, each gesture exhibits a distinct signature. A gesture can be divided into three phases [10, 58]: i) pre-stroke, ii) nucleus, and iii) post-stroke. The pre-stroke refers to the initial movement in a gesture where a user is preparing to perform the gesture, for example, moving their arm to a starting position. The post-stroke is at the completion of the gesture where, for example, the arm moves away from the gesture position. The nucleus is the core of the gesture, following the pre-stroke.

In Figure 3, these phases can be observed in the gyroscope data captured when the flick gesture is performed by 5 different users. The nucleus (marked by a red oval) is visually similar across users while pre-stroke and post-stroke phases vary, likely due to differences in how users approach the start and end of a gesture. AccessWear leverages this nucleus similarity in its gesture recognition pipeline.

4.2 AccessWear Gesture Recognition Pipeline

Figure 4 shows the gesture recognition pipeline. This section elaborates on each component of the pipeline.

4.2.1 Gating Function. Continuously streaming data from the smartwatch to the phone and running the gesture recognition pipeline is expensive. To keep the power and compute costs in the system low, we need to know when a user is potentially performing a gesture. We introduce a gating function that is triggered when significant arm movement is detected. This gating function ensures that data is sent from the watch to the smartphone only when a gesture is likely performed. If the magnitude of gyroscope data exceeds an empirically determined threshold along any axis, the gating function is set to *true* and triggers the rest of the gesture recognition pipeline. We chose the threshold value to be $0.5dps$, which is able to capture most movements performed on the smartwatch. We picked a conservative threshold to avoid missing a possible gesture.

4.2.2 Nucleus Detection. Gesture recognition starts by identifying the nucleus of the gesture. We first determine the axis with the highest rotational variation (hereafter referred to as the significant axis, g_s) and calculate the energy. Next, we use a sliding window and compute the root-mean-square (RMS) energy in this signal window. Empirically we use a sliding window size of 20. For each window, RMS energy (E) is: $E = \sqrt{\frac{1}{N}\sum_n |g_s(n)|^2}$

We implement a lightweight change point detection algorithm to examine the difference in RMSE in consecutive windows. If the RMSE values are close, we discard the new window and continue using the first window as a reference window. When the difference in energy between the reference and the new window exceeds an empirically determined threshold, the change points are noted. The nucleus is determined by two change points. Figure 5 shows the detected nucleus in a gesture.

In our experiments, we use a threshold value of 0.4; i.e., if the difference in energy between two windows exceeds 0.4, we mark it as a change point. This value was based on experiments conducted using different thresholds (see Appendix 2).

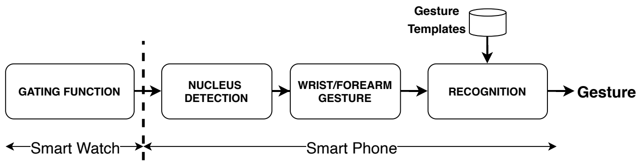


Figure 4: AccessWear gesture recognition pipeline.

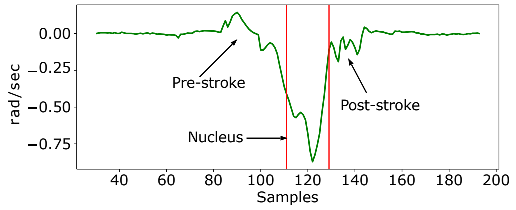


Figure 5: Nucleus boundaries detected based on the change in the RMS energy. Red lines show the change points.

The threshold is robust across the population studied in this work, as shown by the high accuracy of our gesture recognition technique (§7.2). But, a (future) longitudinal study is needed to assess the robustness of this threshold across a large population.

4.2.3 Distinguishing forearm and wrist movements. Based on our data analysis, we observe that the nucleus of a forearm and a wrist gesture look similar. However, wrist gestures are performed very quickly and have a lower range of motion as compared to forearm gestures. The nucleus of a wrist gesture is narrow, whereas the nucleus of a forearm gesture is wider (longer duration). Figure 6 (a), (b) shows the nucleus of a forearm and wrist gesture respectively.

The frequency response of a forearm gesture has lower energy, as seen in Figure 6 (c), while that of the wrist gesture has high energy in certain frequencies, as seen in Figure 6(d). Wrist and forearm gestures can, therefore, be distinguished based on the frequency response. However, translation to the frequency domain using techniques like Fourier transform can induce latency and is not feasible in a real-time system [41]. Instead of frequency domain features, we closely examine the time-series data from the nucleus of the wrist gesture to extract features. We observe that due to the impulse-like nature of the wrist gesture, jitters are observed in the nucleus. We identify these jitters in real-time using a lightweight peak detection algorithm. Based on the nucleus duration and jitters, we classify each gesture as a wrist or forearm gesture.

4.2.4 Recognizing the gestures. The final step is to classify the nucleus. To this end, the detected nucleus is matched against a library of pre-defined gesture templates. In contrast to existing approaches that require large amounts of data, this library contains one template for each gesture. We decided not to use training data from blind users, and

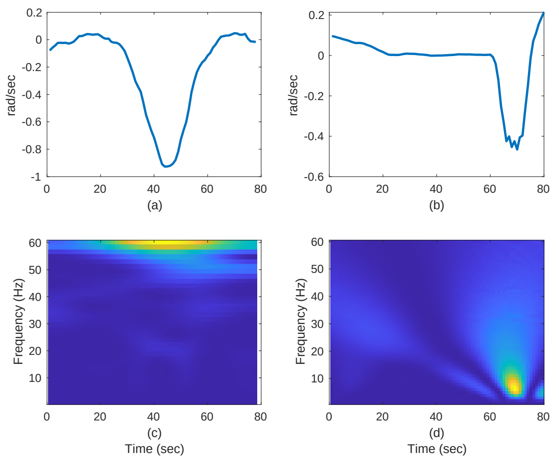


Figure 6: (a) Nucleus of a forearm gesture, (b) nucleus of a wrist gesture that shows jitters, (c) frequency response of a forearm gesture that shows the lack of a high energy component, (d) frequency response of a wrist gesture that shows a high energy component.

therefore, the templates are obtained from a sighted user. The detected nucleus is matched against potential gestures using Fast-DTW [44], which is an approximation of DTW that requires linear time complexity and is appropriate for real-time systems. If the distance returned by the Fast-DTW template matching for a probable gesture occurrence is below an empirically determined threshold, we confirm it as a gesture. We choose this threshold to be a distance of 50. In our evaluations, we find that when a gesture is performed, the distance value is considerably less than 50. A lower distance signifies a high similarity between the template and the gesture. This lightweight algorithm is capable of running in real-time on mobile devices.

4.2.5 Lock/Unlock gesture. Even with the gating function, there is a chance for high false positives because arm/hand movements performed during day-to-day activities may be mistaken for a gesture. Instead, we propose one gesture that can lock and unlock AccessWear; a user simply unlocks AccessWear when needed and lock it again. However, how does one recognize the lock/unlock gesture? We design a separate, lower power pipeline to recognize the lock/unlock gesture. We choose a *turn wrist* gesture as the lock/unlock gesture. To detect the lock/unlock gesture, we use a similar template matching using Fast DTW [44], but do not differentiate between forearm or wrist gestures.

5 AccessWear Input Virtualization

The remaining challenge is—how can we *flexibly* replace the original touchscreen gesture with an alternate gesture *without* requiring changes to the application?

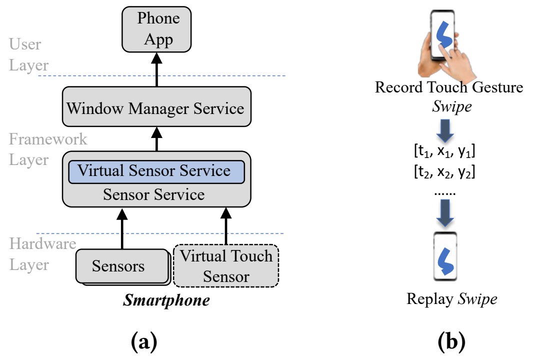


Figure 7: (a) The VirtualSensorService layer generates virtual touch events as though generated by a user locally, (b) Record-and-Replay mechanism to record and replay touch gestures.

AccessWear supports near-zero-effort gesture replacement using three techniques: (1) A metaprogram specification that specifies which smartwatch gesture should replace which touchscreen interaction on the smartphone, (2) A record-and-replay method that recreates any touchscreen gesture on the smartphone screen virtually, and (3) An Input virtualization technique that puts them all together.

When an alternate gesture is identified on the smartwatch, AccessWear uses the metaprogram to map the smartwatch gesture to the touchscreen interaction. AccessWear then replays the touchscreen interaction on the smartphone using record-and-replay, which is automatically delivered to the application as is usual. The application runs as though the touchscreen interaction was generated by a user and performs the necessary action in response to the interaction.

5.1 Metaprogram specification

Let's say a user wants to replace the touchscreen gesture, swiping right on the screen, with a forearm right gesture on the smartwatch. This information is encoded as a metaprogram which stores the mapping between smartwatch to phone gestures. Figure 9 (a) shows an example of such a metaprogram. We assume that a default set of mappings will be available for a user, but a user can customize the mappings as needed. To make the metaprogram specification easier, we design a simple GUI to specify, view, or change gesture mappings as illustrated in Figure 9 (b).

5.2 Record-and-replay

AccessWear uses a record-and-replay technique to generate the touchscreen gestures virtually. The intuition here is that the number of touchscreen gestures are finite and enumerable. Accordingly, we assume that the user performs common touchscreen gestures on the phone. When doing so, we record the coordinates of the gesture and then replay it at run-time. Figure 7(b) shows the record-and-replay. This technique can be used to generate gestures/interactions such as swiping left or right, scrolling, etc.

However, this alone is not sufficient. A user may click on a button or other UI (User Interface) elements. To replay these interactions, one needs to dispatch the touch event to the specified UI element. AccessWear leverages the Virtual Sensor Service to achieve this. While replaying the touch event, each event contains time series coordinates. This intermediate data can precisely locate the target UI element. In effect, the Virtual Sensor Service doesn't need to understand the semantics of the UI element but can still trigger a touch event. One additional issue is that users may want to interact with the phone when the phone screen is switched off. However, when the screen is switched off, the applications no longer function and cannot receive inputs [55]. We use existing techniques [54, 55] to allow applications to continue functioning even when the screen is switched off.

Of course in some rare cases, an application may support a touch interaction at a specific location on the screen; for example, a long press at the corner of the screen where there is no UI element. To replay such an interaction, we expect that the user would record the touch interaction at the same location.

5.3 Input Virtualization

Most mobile operating systems are designed such that all hardware sensors (touchscreen, IMU, etc) go through a sensor service that dispatches the sensor data. Since all applications receive inputs from the sensor service, this provides a natural modular boundary to abstract the application logic from the user interaction. AccessWear implements a virtual sensor service that intercepts the sensor dispatcher and adds a new sensor redirect mechanism. This redirection allows the application to receive input events from AccessWear's virtual sensor as though the touch event was generated locally. In Android, this is achieved by modifying the *InputDispatcher* class, but the general idea is applicable to other mobile OSes. Figure 7 (a) shows the design.

6 Implementation

We implemented AccessWear on Android OS 7.1.2 and Android Wear 2.0 smartwatch OS. Other mobile operating systems such as iOS, have a similar architecture for sensor framework design, and it is possible to replace the Android watch with an Apple watch by installing the watch proxy.

6.1 Run-time Implementation

The system consists of an AccessWear proxy running on the smartwatch and an AccessWear service running on a phone paired via Bluetooth. Figure 8 shows the run time overview of the system. The AccessWear watch proxy polls the gyroscope data at 100 Hz and sends the 3-axis data in

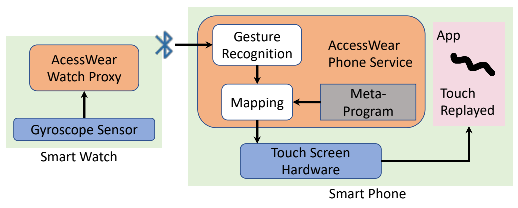
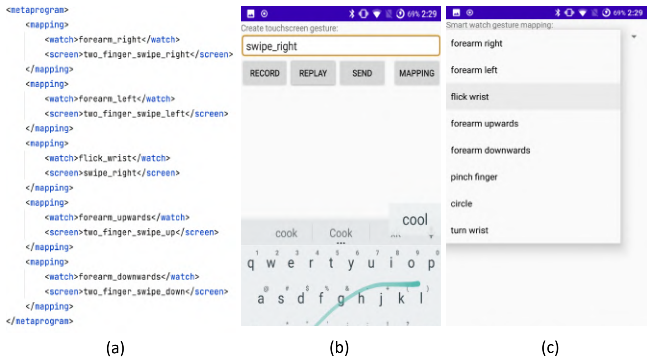


Figure 8: Run-time system implementation.

batches of 10 to the phone. Batching reduces the communication overhead [3].

The AccessWear phone service runs as a system-level thread at the Framework Layer. The phone service performs two tasks at run-time: (1) Gesture recognition, and (2) Replay touch gestures. We implement the gesture recognition pipeline using gyroscope data streamed over Bluetooth as described in Section 4.2.

6.2 Metaprogram Implementation



```
<metaprogram>
    <mapping>
        <watch>forearm_right</watch>
        <screen>two_finger_swipe_right</screen>
    </mapping>
    <mapping>
        <watch>forearm_left</watch>
        <screen>two_finger_swipe_left</screen>
    </mapping>
    <mapping>
        <watch>flick_wrist</watch>
        <screen>swipe_right</screen>
    </mapping>
    <mapping>
        <watch>forearm_upwards</watch>
        <screen>two_finger_swipe_up</screen>
    </mapping>
    <mapping>
        <watch>forearm_downwards</watch>
        <screen>two_finger_swipe_down</screen>
    </mapping>
</metaprogram>
```

Create touchscreen gesture: swipe_right

Smart watch gesture mapping:

Figure 9: Gesture Mappings options. (a) Example XML file used to represent the metaprogram. (b) and (c) User interface used to record touchscreen inputs and change gesture mappings.

Before runtime, AccessWear creates a metaprogram specification for gesture mapping and records touchscreen interactions. We implement the metaprogram specification as an XML file and we provide a GUI to specify and change the metaprogram. Figure 9 shows an example.

7 Evaluation

We conducted a real-world, IRB-approved, user study with *8 blind participants* and a trace-driven study using the data collected. The 8 participants also participated in the formative study and returned for the evaluation study. We summarize the findings from our system evaluation:

- **Real-time performance:** The end-to-end latency between performing a smartwatch gesture and triggering an application response is an average of 57 msec. Further, a user can interact with an application, requiring multiple gestures, with a 95.8% accuracy.

- **Accuracy of gesture recognition:** The AccessWear gesture recognition is 34% more accurate than state-of-the-art gesture recognition systems including Serendipity [52], TapNet [22], and other sensor fusion-based gesture recognition systems.
- **Usability:** An exit survey showed that blind users found AccessWear easy to use and robust.

Devices: All evaluations are performed on an LG Nexus 5 smartphone and Fossil Gen 5 smartwatch. The Nexus 5 phone and the Fossil Gen 5 smartwatch run the AccessWear service as shown in Figure 8.

7.1 Study Setup with Blind Users

Participants: The 8 participants (5 male and 3 female) age ranges from 34-61. Since the participants were part of the formative study, they were familiar with the motivation. It took 1.5 hours to conduct the study and each blind participant was paid $150 for their time.

Study Design: We adopted a repeated measures within-subject study design [49]. In the study, participants map smartwatch gestures to touchscreen gestures and then use the smartwatch gestures to interact with the application instead of using the touchscreen. We currently support 5 smartwatch gestures (Figure 10). The participants map it to the five most popular touchcreen gestures, where the popular touchscreen gestures were obtained from the formative study. The default mapping is shown in Table 5.

Task: We designed an item selection task in which the participants are asked to find an item on the screen and select it. The participant selects an item from a 3×3 grid, where different rows have different category items (colors, countries, fruits). The participant repeats this selection three times, and in each round the order of categories and items are randomized. While the participant is performing the task, we record a video as ground truth for our evaluation (not including face) and log the events on the phone.

A task requires that the participant performs a series of gestures to choose an item; on average, each participant performed 15 gestures to complete all the tasks. In effect, this experiment evaluates the ability of a participant to (i) recall the right gesture and (ii) perform the gesture seamlessly even when performing a task that requires higher cognitive engagement.

Procedure: We begin the study with an introductory survey about phone interactions and smartwatch gestures. This is followed by a briefing session where the participant is introduced to the AccessWear system. The experimenter then asks the participant to select a mapping between a set of watch gestures and phone gestures. The experimenter inputs the selected mapping and asks the participant to practice the mapping on a training app that reads out the gesture performed and plays the corresponding mapping. Once the user is satisfied with the mapping we move to the task. The participants practiced the mappings for about 5 minutes on average.

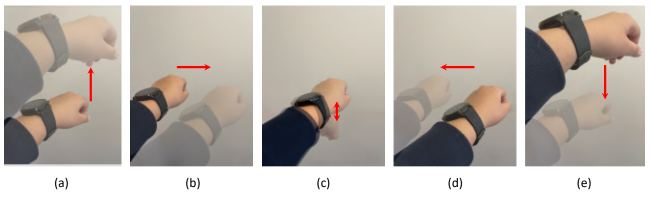


Figure 10: Gestures for Accesswear evaluation with blind users. (a) Forearm upwards, (b) Forearm right, (c) Flick wrist, (d) Forearm left, (e) Forearm downwards.

Gesture	Mapping
Forearm upwards	Two finger swipe up
Forearm right	Two finger swipe right
Flick wrist	Swipe right
Forearm left	Two finger swipe left
Forearm downwards	Two finger swipe down

Table 5: Default mappings between the smartwatch and touchscreen gestures provided by AccessWear. Users were able to change the mapping as needed.

Trace-based study. During the user study, the experimenter also asks the participants to perform the 5 gestures 3 times in a counterbalanced order. When the gestures are performed, the timestamps and sensor data from the accelerometer, linear accelerometer, gyroscope, magnetometer, and orientation sensor are logged at 100 Hz. This trace allows us to do repeatable comparison studies to compare different gesture recognition systems. Traces are collected while the user is sitting as users typically do not perform these gestures while walking.

7.2 Baselines

We compare AccessWear's gesture recognition with the following state-of-the-art.

- Serendipity [52] is a gesture recognition system that uses personalized SVM models to classify gestures for each user. For our implementation, we use the features described in the paper and train an SVM classifier for each user.
- TapNet [22] is a CNN-based tap detection system for mobile phones. We change the TapNet multi-task network to perform a single task by modifying the tap/no-tap classifier to a 5 gesture classifier. We split our data (8 users, 5 gestures performed three times by each user) into training (70%) and testing (30%) sets to train a generalized model. The model does not converge as the training data we have is just 84 samples as opposed to $135K$ training samples used in the TapNet paper.

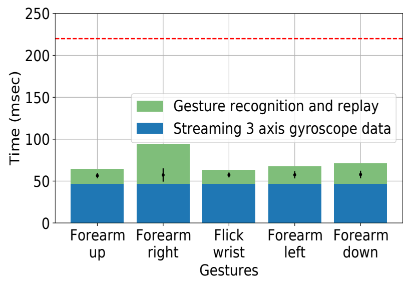
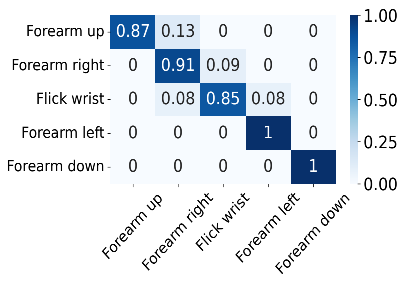


Figure 11: Average latency between the smartwatch gesture to application reaction.



	Forearm up	Forearm right	Flick wrist	Forearm left	Forearm down
Forearm up	0.87	0.13	0	0	0
Forearm right	0	0.91	0.09	0	0
Flick wrist	0	0.08	0.85	0.08	0
Forearm left	0	0	0	1	0
Forearm down	0	0	0	0	1

Figure 12: Confusion matrix for AccessWear gesture recognition across 8 blind users.

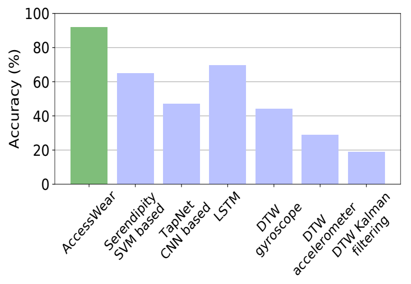


Figure 13: Accuracy comparison for different gesture recognition systems across 8 blind users.

- LSTM-based models are commonly used for tasks such as activity recognition. We develop an LSTM-based architecture, which consists of Bi-Directional LSTM layers, followed by Batch normalization, and a classifier. We split our data into training (70%) and testing (30%) to train a generalized model.
- DTW-variants is a commonly used technique [39] to compare the similarity between gestures. This technique is used to compare time series data that have varying speeds. We implement DTW similarity measures using an accelerometer, gyroscope, and a fusion of multiple sensors.

7.3 Results

End-to-end latency: Figure 11 shows the end-to-end delay of the AccessWear system for the five gestures in Figure 10. The latency is calculated as the time taken between performing the gesture on the watch to the application reaction to the input. We use NTP [38] to synchronize the clocks of the watch and the phone.

Across all five gestures and users, the average reaction time to the gesture is 57 msec, much smaller than the 220 msec reaction time that users can tolerate [21]. Quantitatively, this means that AccessWear works near real-time. Broken down further, the mean time spent in data streaming is 46 msec. It only takes 11 msec to recognize the gestures.

Task completion: Each of the 8 users performed the task 3 times. All but one participant was able to complete the task with an average task completion accuracy 95.8%. One use took an exploratory approach to randomly find items in the screen, and therefore, could not complete the task.

Gesture recognition accuracy: We next evaluate the accuracy of gesture recognition using the collected traces. Figure 12 shows a confusion matrix. The overall accuracy is 92% which shows that even though AccessWear's gesture recognition runs efficiently on mobile devices, that doesn't come at a cost of accuracy. We observe that due to a lack of visual feedback, participants do not necessarily perform a gesture as described, and can sometimes move their arms up/down or left/right, triggering some misclassifications. However, these misclassifications are rare and do not affect the overall user experience, as discussed later in this section.

Comparison with baseline: Figure 13 shows the comparison between AccessWear's gesture recognition and that of baseline techniques. AccessWear outperforms all other approaches with the LSTM-based model (70%) and the Serendipity-like model (65%) performing better than Tapnet's CNN model (50%). TapNet requires large amounts of data for training which is difficult to obtain for the blind user population. This result further shows that AccessWear can work well with the limited amount of data.

Further, we trained an LSTM-based model using the data we collected during our sighted user study with 16 users. The accuracy of the model, when tested with sighted users (70% training set and 30% test set), is 89% but this value drops to 48% when tested with blind users. This result corroborates our observation that blind users perform gestures differently compared to sighted users.

Questions	Mode of Score (IQR)
Did you find the system reliable?	7 (1)
Did you find the system robust?	7 (1.25)
Did you find the system responsive?	7 (0.25)
Is the system physically demanding?	1 (1.25)
Is the system mentally demanding?	2 (1)
Does the system meet your expectations?	7 (0)

Table 6: Mode of Likert scale score across 8 blind users. The score ranges from 1 to 7 where 1 indicates strongly disagree and 7 indicates strongly agree. The table also shows the IQR or Interquartile Range. Lower IQR suggests less variance in user responses.

We also compare the accuracy with DTW approaches that, similar to AccessWear, compare the distance to a template. We see an accuracy drop of nearly 52% and 68% for gesture recognition using DTW on gyroscope and accelerometer data respectively. IMU tracking papers [46, 47] fuse all three sensor readings accelerometer, gyroscope, and magnetometer for gesture tracking. To evaluate the performance of sensor fusion techniques, we employ Kalman filter [24] (a sensor fusion algorithm) based on template matching. We still see low accuracy as the magnetometer and accelerometer data have significant noise. In affect, using gyroscope alone is sufficient in recognizing arm and wrist gestures performed by blind users.

Qualitative Feedback: We ask the participants about their experience using AccessWear. Users were asked to answer questions about the system performance on a scale of 1 to 7, with 1 being 'strongly disagree' and 7 being 'strongly agree'. Table 6 shows the mode of scores. All users find AccessWear reliable and responsive and using alternate gestures was not mentally or physically demanding. We also conducted a SUS (System Usability Scale) survey which showed similar results regarding the usefulness and robustness of AccessWear. We present those results in Appendix 3.

8 Technical capabilities of AccessWear

In the previous section, we evaluate the performance of AccessWear with blind users in controlled settings. Evaluating some of the technical capabilities of the system requires longer duration studies that are difficult to perform with blind users. Instead, we perform a set of user studies with 5 sighted users, also on the LG Nexus 5 smartphone and the Fossil Gen 5 smartwatch.

8.1 Energy Consumption

Methodology: We compare the total energy consumption when using smartwatch gestures for the same task as described in the previous section. Total energy includes polling and streaming of gyroscope data on the watch. On the phone, total energy accounts for running the gesture recognition pipeline, replaying the touch gestures, and playing audio and vibratory feedback when applicable. We perform this experiment separately so it does not affect the integrity of the task. The participant performs a random set of all 5 gestures repeated 10 times for one minute. When the gestures are being performed, AccessWear runs the gesture recognition pipeline and the application responds to the gesture based on the metaprogram specification (we use the default mapping). We use the Batterystats tool [5, 31] tool to measure energy consumption on both the watch and the phone. Other common tools such as Monsoon Power Monitor could not be used on modern phones since the battery cannot be removed.

Figure 14a shows the total energy for different gestures averaged over all runs. The flick wrist gesture consumes more energy because the application vibrates in response to the gesture, which adds to the energy consumption. To put the energy numbers in context, we compare the energy consumed by AccessWear with the energy consumed to scroll an Instagram page for 1 minute on the phone. AccessWear consumes less than half the energy compared to Instagram scrolling. We also compare the power consumed by AccessWear on the watch to a heart rate monitor running for 1 minute and find that the energy consumption is comparable. We note that the energy consumption on the watch and the phone when AccessWear is locked is considerably lower (see Figure 14c). All the energy numbers reported in this experiment are the delta over the standby energy consumed by the phone or watch. Standby energy for the phone is 8 mAh and for the smartwatch is 4 mAh over a period of 1 minute.

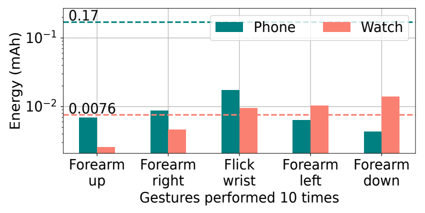
In a separate experiment, we compare the energy consumed by the smartphone when using the alternate gesture vs using the touchscreen gestures. AccessWear consumes 20% more energy when using the alternate gestures, due to the gesture recognition pipeline and replaying the touchscreen gestures. We omit the figure due to space constraints.

8.2 False positive detection in real-world settings

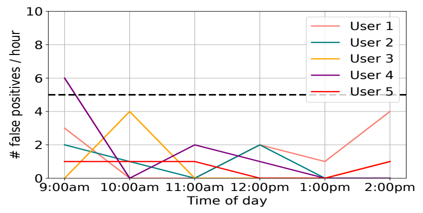
Methodology: The goal of this study is to evaluate the number of times a smartwatch falsely thinks a gesture was performed when a user is wearing their AccessWear-enabled smartwatch. Lower the false positives, the better. For these experiments, the participants wear the AccessWear-enabled watch for 5 hours while going about their daily lives. AccessWear is locked during the 5 hours; we estimate the number of times AccessWear is mistakenly unlocked and the gesture recognition pipeline is triggered. Participants are asked to maintain a log of the activity they did every half hour. The participants were not provided any information about AccessWear except demonstrating the lock gesture to them.

Figure 14b shows the false positive rate per hour; i.e., the number of times an AccessWear gesture was triggered over the 5 hours. The gestures were triggered only once per hour on average, except for two instances. In these two instances, the gesture was triggered—6 times and 4 times respectively within an hour. When mapped with the activity log, during this time the participant was washing dishes and opening a bottle. AccessWear reduces the per-hour false positive rate by 31.5% when compared to the previous false positive detection studies in real-world settings for gesture recognition [26].

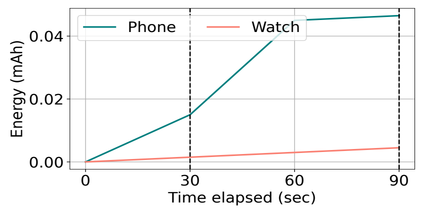
By designing a lock mechanism, AccessWear is able to reduce the probability of a user inadvertently triggering AccessWear gestures. Figure 14c shows the cost of the locking mechanism in terms of energy consumed. In this experiment, we unlock AccessWear 30 seconds into the experiment and lock it again after 1 minute. When locked, AccessWear consumed only 0.014 mAh energy on the phone and 0.0012 mAh



(a) Total energy consumed by AccessWear on the phone and watch. Dotted lines show the energy consumed by Instagram (on phone) and heart monitor (on watch) for comparison.



(b) The number of false positive gestures recorded in a 5-hour trace for 5 different users. In all but three cases, the number of false positives is less than 3 times an hour.



(c) Energy consumed when the system is locked for 30 seconds, and then unlocked for 1 minute. AccessWear only consumed 0.014 mAh energy (phone) and 0.0012 mAh (watch) when locked.

Figure 14: Evaluating technical capabilities of AccessWear.

energy on the watch. When unlocked, the energy increases to 0.046 mAh (phone) and 0.004 mAh (watch).

One limitation with this study is that a blind user's cane or guide dog could potentially trigger false positives. We will require a longitudinal study with blind users to evaluate this aspect further.

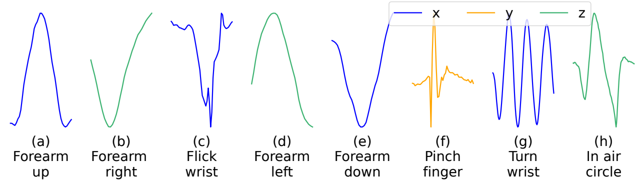
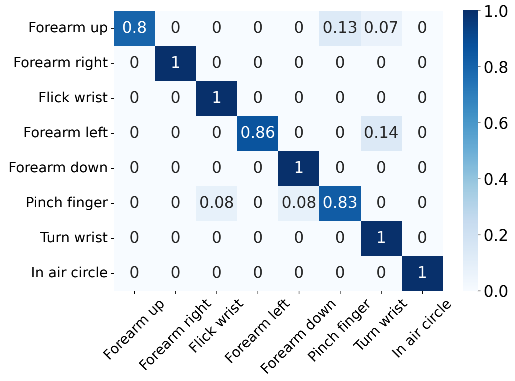


Figure 15: Nucleus for different gestures. x,y, and z are the axes of gyroscope data.

8.3 Extending AccessWear to more gestures

So far, we evaluated AccessWear for 5 forearm and wrist gestures. However, AccessWear's gesture recognition technique can support more gestures. Adding a new gesture requires determining the nucleus and creating a template of the nucleus. One criterion for adding a new gesture is that the nucleus needs to be sufficiently different from the nucleus of other gestures (Pearson correlation coefficient < 0.5). We add three new gestures to AccessWear: circle in the air, pinching a finger, and turning wrist. The nucleus of all eight gestures are shown in Figure 15.

We evaluate the accuracy of the gesture recognition of the 8 gestures with data collected from 5 sighted users. Each user performs each gesture 3 times. Figure 16 shows that the accuracy across the 8 gestures is 93%, showing that AccessWear can recognize more than just forearm and wrist gestures. In contrast to ML-based baseline techniques, adding a new gesture to the system does not require any additional data collection overhead.



	Forearm up	Forearm right	Flick wrist	Forearm left	Forearm down	Pinch finger	Turn wrist	In air circle
Forearm up	0.8	0	0	0	0	0.13	0.07	0
Forearm right	0	1	0	0	0	0	0	0
Flick wrist	0	0	1	0	0	0	0	0
Forearm left	0	0	0	0.86	0	0	0.14	0
Forearm down	0	0	0	0	1	0	0	0
Pinch finger	0	0	0.08	0	0.08	0.83	0	0
Turn wrist	0	0	0	0	0	0	1	0
In air circle	0	0	0	0	0	0	0	1

Figure 16: Confusion matrix for gesture recognition accuracy across 5 sighted users and 3 additional gestures. The result shows that AccessWear's gesture recognition can be extended beyond arm and wrist gestures.

8.4 Interacting with unmodified applications

The key feature of AccessWear is that it does not require any modifications to the smartphone application. To evaluate the user experience when interacting with unmodified applications using AccessWear's alternate gestures, we asked the 5 sighted users to interact with Spotify, YouTube, Gallery, and a Browser. The users used the 8 gestures and mapped each of them to a touchscreen gesture of their choice. The experimental setup is similar to the study in §7.

We then ask the participants to rate their user experience, perceived delay, and robustness on a scale of 1 to 5, with 1 being a great experience and 5 being a poor experience. In terms of perceived delays, the mode across all 5 users and all apps was 1, which indicates that there is no perceived delay. In terms of user experience, the mode across all 5 users for Spotify, Gallery, and Browser, was 4 or 5 indicating good user experience. However, the mode for YouTube was

3. One reason was that the “swipe right" gesture was used by the participants to go to the next video, but the application instead forwarded the video due to the placement of the touchscreen replay. This is a limitation in AccessWear; when replaying a touchscreen gesture, the gesture may interact with the elements on the screen and result in unexpected application behavior.

9 Related Work

IMU-based Gesture Recognition. There has been considerable work on using IMUs for gesture recognition. In Table 4, we summarized these works. The state-of-the-art models are TapNet [22], Serendipity [52], and gesture customization [56]. We discuss TapNet and Serendipity in our evaluation; the former needs a large training set and the latter needs personalization. The gesture customization [56] work is more recent. This work focuses on sighted users only but also uses gyroscope data similar to AccessWear and proposes few shot learning using a small amount of personalized data. However, the work requires extensive training data to train the base model, which is difficult to obtain for blind users.

ViBand [30] shows that oversampling the accelerometer sensor can capture fine-grained gestures. uWave [34] shows that with one sample from a user, accelerometer data can capture the motion gesture patterns. However, we find that accelerometer data contains a mixture of linear and gravity acceleration, making it noisy. The state-of-the-art in gesture tracking MUSE [46], ArmTrak [47], ArmTroi [35] are able to track gestures, rather than only recognize gestures. They use sensor fusion to compensate for drift over time. The problem is that these works rely on heavy signal processing or require point cloud generation, both of which are computationally heavy for a mobile phone.

Gesture Recognition. Other works have used different sensors such as an array of proximity sensors [19], barometric pressure sensors [59], and electrical impedance tomography [48] for gesture recognition. WristLens [57] enables single-handed gestures on surfaces using an optical motion sensor embedded in a wrist strap, allowing the user to leverage any proximate surface, including their own body, for input and interaction. However, these sensors are not readily available on smartwatches. Microphone [50, 51] has been used for gesture recognition but it suffers from the drawback of not being able to work in the presence of external acoustic noise. WiFi-based gesture recognition [26] has shown good accuracy, but only works in smart building type of environments.

Finally, most gesture recognition works only recognize gestures, but do not design a gesture replacement. WristWhirl [19] is one of the few works that use gesture recognition (using proximity sensors) to replace existing smartphone gestures. However, it requires that the application be rewritten to work with alternate gestures.

Smartwatch Gestures for Accessibility. Smartwatches are used in many scenarios for visually impaired users. One of the earliest works uses smartwatch gestures in an app that identifies "wet floor" signs [42]. Smartwatches are also used for interactions in navigation [18], writing [11], map exploration [9, 20], and Braille reading [14]. In this work, we leverage the smartwatch for alternate gestures. Despite previous research which suggests that there are differences between how visually impaired users do gestures compared with sighted peers [16], there is not any recent work on gesture recognition for visually impaired users.

Another important direction in accessibility research is to provide alternative interaction for the screen reader. For instance, related work creates a ring [17, 33] or a deformable surface [15] as input modalities that can replace screen reader actions on smartphones. However, these works need specialized hardware and require handcrafted applications to work with alternate gestures.

10 Conclusion

Motivated by the needs of blind users, we explored how commodity smartwatches can be leveraged to improve the accessibility of smartphone interactions. We developed AccessWear, wherein we design a practical smartwatch-based gesture recognition system that runs on resource-constrained mobile devices in real-time. These gestures are then mapped to touchscreen gestures on the smartphone through a novel input virtualization mechanism. AccessWear allows blind users to interact with their devices conveniently via a smartwatch, alleviating their concerns about shoulder-surfing and phone safety. AccessWear's design enables the extension to multiple gestures and can support any mobile application without the need to modify it. In a study with 8 blind users, we find that AccessWear is accurate with 92% gesture recognition accuracy, and can run in real-time.

Acknowledgments

We sincerely thank our anonymous shepherd and reviewers for their insightful comments and suggestions. This work was supported in part by a Google Research Scholar award, the National Science Foundation under award numbers 2110193, 2153056, 2125147, 2113485 and the National Institutes of Health under award numbers R01EY030085, R01HD097188.

References

[1] Ali Abdolrahmani, Ravi Kuber, and Amy Hurst. 2016. An empirical investigation of the situationally-induced impairments experienced by blind mobile device users. In *Proceedings of the 13th International Web for All Conference*. 1–8.

[2] Ahmad Akl, Chen Feng, and Shahrokh Valaee. 2011. A novel accelerometer-based gesture recognition system. *IEEE transactions on Signal Processing* 59, 12 (2011), 6197–6205.

[3] Ardalan Amiri Sani, Kevin Boos, Min Hong Yun, and Lin Zhong. 2014. Rio: a system solution for sharing i/o between mobile systems. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. 259–272.

[4] Edwin Valarezo Añazco, Seung Ju Han, Kangil Kim, Patricio Rivera Lopez, Tae-Seong Kim, and Sangmin Lee. 2021. Hand gesture recognition using single patchable six-axis inertial measurement unit via recurrent neural networks. *Sensors* 21, 4 (2021), 1404.

[5] Android. [n.d.]. Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/topic/performance/power/setup-battery-historian

[6] Apple. [n.d.]. Learn VoiceOver gestures on iPhone. https://support.apple.com/en-in/guide/iphone/iph3e2e2281/ios

[7] Apple. [n.d.]. VoiceOver Getting Started Guide. https://support.apple.com/en-in/guide/voiceover-guide/welcome/web

[8] Shiri Azenkot and Nicole B Lee. 2013. Exploring the use of speech input by blind people on mobile devices. In *Proceedings of the 15th international ACM SIGACCESS conference on computers and accessibility*. 1–8.

[9] Sandra Bardot, Marcos Serrano, and Christophe Jouffrais. 2016. From tactile to virtual: using a smartwatch to improve spatial map exploration for visually impaired users. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 100–111.

[10] Gibran Benitez-Garcia, Muhammad Haris, Yoshiyuki Tsuda, and Norimichi Ukita. 2020. Continuous finger gesture spotting and recognition based on similarities between start and end frames. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[11] Syed Masum Billah, Vikas Ashok, and IV Ramakrishnan. 2018. Write-it-yourself with the aid of smartwatches: A wizard-of-oz experiment with blind people. In *23rd International Conference on Intelligent User Interfaces*. 427–431.

[12] CDC. [n.d.]. Fast Facts of Common Eye Disorders. https://www.cdc.gov/visionhealth/basics/ced/fastfacts.htm.

[13] Casey A. Cole, Bethany Janos, Dien Anshari, James F. Thrasher, Scott Strayer, and Homayoun Valafar. 2020. Recognition of Smoking Gesture Using Smart Watch Technology. https://doi.org/10.48550/ARXIV.2003.02735

[14] Aritra Dhar, Aditya Nittala, and Kuldeep Yadav. 2016. TactBack: Vibrotactile braille output using smartphone and smartwatch for visually impaired. In *Proceedings of the 13th International Web for All Conference*. 1–2.

[15] Matthew Ernst, Travis Swan, Victor Cheung, and Audrey Girouard. 2017. Typhlex: Exploring deformable input for blind users controlling a mobile screen reader. *IEEE Pervasive Computing* 16, 4 (2017), 28–35.

[16] Shirin Feiz and IV Ramakrishnan. 2019. Exploring feasibility of wrist gestures for non-visual interactions with wearables. In *Proceedings of the 16th International Web for All Conference*. 1–4.

[17] Catherine Feng. 2016. Designing wearable mobile device controllers for blind people: a co-design approach. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*. 341–342.

[18] Ombretta Gaggi, Claudio E Palazzi, Matteo Ciman, and Armir Bujari. 2018. Stepbywatch: A smartwatch-based enhanced navigation system for visually impaired users. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 1–5.

[19] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. Wristwhirl: One-handed continuous smartwatch input using wrist gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 861–872.

[20] William Grussenmeyer, Jesel Garcia, and Fang Jiang. 2016. Feasibility of using haptic directions through maps with a tablet and smart watch for people who are blind and visually impaired. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 83–89.

[21] Henry Hamburger. 1969. Donald RJ Laming. Information, theory of choice-reaction times. New York: Academic Press, 1968.

[22] Michael Xuelin Huang, Yang Li, Nazneen Nazneen, Alexander Chao, and Shumin Zhai. 2021. TapNet: The Design, Training, Implementation, and Applications of a Multi-Task Learning CNN for Off-Screen Mobile Input. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–11.

[23] Mohit Jain, Nirmalendu Diwakar, and Manohar Swaminathan. 2021. Smartphone usage by expert blind users. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[24] Rudolph Emil Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering* 82, Series D (1960), 35–45.

[25] Peiqi Kang, Jinxuan Li, Bingfei Fan, Shuo Jiang, and Peter B Shull. 2021. Wrist-worn Hand Gesture Recognition while Walking via Transfer Learning. *IEEE Journal of Biomedical and Health Informatics* (2021).

[26] Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota. 2014. Bringing gesture recognition to all devices. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 303–316.

[27] Brian Kemler. 2021. Our all-new talkback screen reader. https://blog.google/products/android/all-new-talkback

[28] Minwoo Kim, Jaechan Cho, Seongjoo Lee, and Yunho Jung. 2019. IMU sensor-based hand gesture recognition for human-machine interfaces. *Sensors* 19, 18 (2019), 3827.

[29] Jonathan Knighten, Stephen McMillan, Tori Chambers, and Jamie Payton. 2015. Recognizing social gestures with a wrist-worn smartband. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 544–549.

[30] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 321–333.

[31] Sunjae Lee, Hoyoung Kim, Sijung Kim, Sangwook Lee, Hyosu Kim, Jean Young Song, Steven Y Ko, Sangeun Oh, and Insik Shin. 2022. A-mash: providing single-app illusion for multi-app use through user-centric UI mashup. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 690–702.

[32] Barbara Leporini, Maria Claudia Buzzi, and Marina Buzzi. 2012. Interacting with mobile devices via VoiceOver: usability and accessibility issues. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*. 339–348.

[33] Guanhong Liu, Yizheng Gu, Yiwen Yin, Chun Yu, Yuntao Wang, Haipeng Mi, and Yuanchun Shi. 2020. Keep the Phone in Your Pocket: Enabling Smartphone Operation with an IMU Ring for Visually Impaired People. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–23.

[34] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition

and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675.

[35] Yang Liu, Zhenjiang Li, Zhidan Liu, and Kaishun Wu. 2019. Real-time arm skeleton tracking and gesture inference tolerant to missing wearable sensors. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services.* 287–299.

[36] David Mace, Wei Gao, and Ayse Coskun. 2013. Accelerometer-based hand gesture recognition using feature weighted naïve bayesian classifiers and dynamic time warping. In *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent user interfaces companion.* 83–84.

[37] Antigoni Mezari and Ilias Maglogiannis. 2018. An easily customized gesture recognizer for assisted living using commodity mobile devices. *Journal of Healthcare Engineering* 2018 (2018).

[38] David L Mills et al. 1985. Network time protocol (NTP). (1985).

[39] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.

[40] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. 2014. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services.* 149–161.

[41] Chun-Su Park. 2017. Guaranteed-Stable Sliding DFT Algorithm With Minimal Computational Requirements. *IEEE Transactions on Signal Processing* 65, 20 (2017), 5281–5288. https://doi.org/10.1109/TSP.2017.2726988

[42] Lorenzo Porzi, Stefano Messelodi, Carla Mara Modena, and Elisa Ricci. 2013. A smart watch-based gesture recognition system for assisting people with visual impairments. In *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices.* 19–24.

[43] Right-Hear. [n.d.]. How Do Blind People Use Smartphones? https://youtu.be/IkQk8ZbToNo?t=54.

[44] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[45] Barbara Šepić, Abdurrahman Ghanem, and Stephan Vogel. 2015. BrailleEasy: one-handed braille keyboard for smartphones. In *Assistive Technology.* IOS Press, 1030–1035.

[46] Sheng Shen, Mahanth Gowda, and Romit Roy Choudhury. 2018. Closing the gaps in inertial motion tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking.* 429–444.

[47] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I am a smartwatch and i can track my user's arm. In *Proceedings of the 14th annual international conference on Mobile systems, applications, and services.* 85–96.

[48] Peter B Shull, Shuo Jiang, Yuhui Zhu, and Xiangyang Zhu. 2019. Hand gesture recognition and finger angle estimation via wrist-worn modified barometric pressure sensing. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27, 4 (2019), 724–732.

[49] Lisa M Sullivan. 2008. Repeated measures. *Circulation* 117, 9 (2008), 1238–1243.

[50] Wei Wang, Alex X Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking.* 82–94.

[51] Yanwen Wang, Jiaxing Shen, and Yuanqing Zheng. 2020. Push the limit of acoustic gesture recognition. *IEEE Transactions on Mobile Computing* (2020).

[52] Hongyi Wen, Julian Ramos Rojas, and Anind K Dey. 2016. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the 2016 CHI conference on human factors in computing systems.* 3847–3851.

[53] Chao Xu, Parth H Pathak, and Prasant Mohapatra. 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications.* 9–14.

[54] Jian Xu, Syed Masum Billah, Roy Shilkrot, and Aruna Balasubramanian. 2019. DarkReader: bridging the gap between perception and reality of power consumption in smartphones for blind users. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility.* 96–104.

[55] Jian Xu, Qingqing Cao, Aditya Prakash, Aruna Balasubramanian, and Donald E Porter. 2017. UIWear: Easily adapting user interfaces for wearable devices. In *Proceedings of the 23rd annual international conference on mobile computing and networking.* 369–382.

[56] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, et al. 2022. Enabling hand gesture customization on wrist-worn devices. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems.* 1–19.

[57] Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, Alejandro Samboy, Hideki Koike, Woontack Woo, and Aaron Quigley. 2020. WristLens: Enabling Single-Handed Surface Gesture Interaction for Wrist-Worn Devices Using Optical Motion Sensor. In *Proceedings of the Augmented Humans International Conference.* 1–8.

[58] Ying Yin and Randall Davis. 2014. Real-time continuous gesture recognition for natural human-computer interaction. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, 113–120.

[59] Yang Zhang, Robert Xiao, and Chris Harrison. 2016. Advancing hand gesture recognition with high resolution electrical impedance tomography. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology.* 843–850.

[60] Han Zhou, Yi Gao, Xinyi Song, Wenxin Liu, and Wei Dong. 2019. Limbmotion: Decimeter-level limb tracking for wearable-based human-computer interaction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–24.

[61] Peide Zhu, Hao Zhou, Shumin Cao, Panlong Yang, and Shuangshuang Xue. 2018. Control with gestures: a hand gesture recognition system using off-the-shelf smartwatch. In *2018 4th International Conference on Big Data Computing and Communications (BIGCOM).* IEEE, 72–77.

Appendix

Appendix 1: Demographics of the participants in the user study

ID	Gender	Studies Participated	Commonly used apps on phone
S1	M	P, E	Banking, video streaming, games, maps
S2	M	P, E	Calling, YouTube
S3	M	P, E	Weather, iMessage, E-mail, banking
S4	M	P, E	Radio, maps, financial, money reader, Netflix
S5	F	P, E	YouTube, Amazon, social media, games
S6	F	P, E	Social media, maps, browser, book-reader, blind square, cash reader
S7	F	P, E	Kindle, amazon, video calling
S8	M	P, E	Social media, calling, Spotify, pandora
S9	F	P	Weather, iMessage, E-mail, calling, blind square

Table 7: User study participants demographics. P: participated in pilot study, E: participated in the evaluation study.

The exploratory study consisted of 9 blind participants; 8 participants returned for the evaluation study (one participant dropped out). Table 7 shows the demographics of the 9 participants. 5 out of the 9 participants wore the watch on their left hand. As part of the introductory survey, we asked the participants the most common applications they used on their phones. Every participant mentioned one or more content seeking application, such as video streaming and browsing. It is important that these content seeking applications are made accessible for blind users.

Appendix 2: Nucleus threshold for gesture recognition pipeline

The AccessWear gesture recognition pipeline uses a threshold to recognize change points (§ 4.2.2). If the change point, defined as the difference in RMS energy between two sliding windows, is greater than a threshold, it is recorded as the boundary of the nucleus. We evaluate the robustness of the threshold. Figure 17 shows the gesture recognition accuracy across blind users at different nucleus threshold values. Based on this experiment, we choose 0.4 as the threshold.

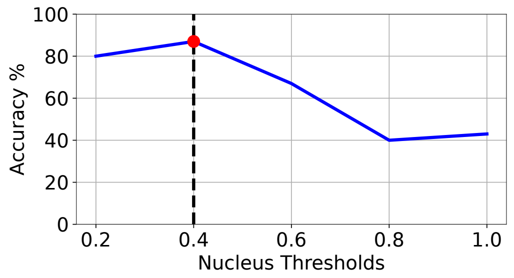


Figure 17: AccessWear gesture recognition accuracy at different nucleus thresholds.

Appendix 3: SUS survey responses after using the AccessWear system.

After conducting a task-driven evaluation of AccessWear (§7), we conducted a System Usability Scale (SUS) survey to characterize the usability of AccessWear. Table 8 shows the mode of SUS scores provided by the 8 blind participants for the ten SUS questions. The survey results show that AccessWear is easy to use.

SUS Questions	Mode of Score
I think that I would like to use this AccessWear frequently.	4
I thought AccessWear system was easy to use.	5
I found the various functions in AccessWear were well integrated.	5
I imagine that most people would learn to use this system very quickly.	5
I felt very confident using the AccessWear system.	5
I found the AccessWear unnecessarily complex.	1
I think that I would need the support of a technical person to be able to use AccessWear.	1
I thought there was too much inconsistency in AccessWear.	1
I found AccessWear very cumbersome to use.	1
I needed to learn a lot of things before I could get going with this system.	1

Table 8: Results of a SUS survey to quantify the usability of AccessWear. The table shows the mode of the Likert scale score as provided by the 8 participants for each question. The score ranges from 1 to 5, where 5 represents strongly agree and 1 represents strongly disagree.