Hand Gesture Recognition for Blind Users by Tracking 3D Gesture Trajectory

Prerna Khanna
pkhanna@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

Hand gestures provide an alternate interaction modality for blind users and can be supported using commodity smartwatches without requiring specialized sensors. The enabling technology is an accurate gesture recognition algorithm, but almost all algorithms are designed for sighted users. Our study shows that blind user gestures are considerably different from sighted users, rendering current recognition algorithms unsuitable. Blind user gestures have high inter-user variance, making learning gesture patterns difficult without large-scale training data. Instead, we design a gesture recognition algorithm that works on a 3D representation of the gesture trajectory, capturing motion in free space. Our insight is to extract a micro-movement in the gesture that is user-invariant and use this micro-movement for gesture classification. To this end, we develop an ensemble classifier that combines image classification with geometric properties of the gesture. Our evaluation demonstrates a 92% classification accuracy, surpassing the next best state-of-the-art which has an accuracy of 82%.

CCS CONCEPTS

• **Human-centered computing → Gestural input**; **Accessibility design and evaluation methods**; **Systems and tools for interaction design**.

KEYWORDS

Gesture recognition, Sensing, Accessibility, Blind users

ACM Reference Format:
Prerna Khanna, IV Ramakrishnan, Shubham Jain, Xiaojun Bi, and Aruna Balasubramanian. 2024. Hand Gesture Recognition for Blind Users by Tracking 3D Gesture Trajectory. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3613904.3642602

1 INTRODUCTION

Hand gestures made with the arm, wrist, and fingers, are being increasingly used as an alternate modality for interacting with smartphones. Hand gestures can support various interaction modes including one-handed interactions [44], cross-device interactions [18], and in-air gesture typing [53].

One important use case of this alternate interaction modality is in improving accessibility for blind users. Blind users interact with their smartphone devices using a touchscreen or voice input, but numerous studies have shown that these interactions are challenging [4, 32, 59]. Instead, hand gestures support an intuitive, flexible, and versatile interaction platform for blind users [15]. Importantly, with the increasing popularity of hand-worn commodity devices such as smartwatches, users can leverage alternate gesture interactions *without* the need for specialized sensors or hardware.

However, enabling these hand gesture interactions requires a highly accurate gesture recognition algorithm. A gesture recognition algorithm tracks motion sensors that are available in commodity smartwatches and uses the sensor data to classify gestures. There has been considerable work on designing highly accurate gesture recognition algorithms [19, 57, 61], but almost all of these algorithms have been designed for sighted users.

In this work, our goal is to design a gesture recognition algorithm that works well for blind users. Our user study shows that there are considerable differences between how blind users perform hand gestures compared to their sighted counterparts, and these differences have an implication on the gesture recognition algorithm. We conducted a comparative study with 10 blind and 16 sighted users, where each user performed 15 different hand gestures. We study gesture-specific features including gesture variations, jerk, jitter noise, and gesture velocity. Based on these features, we find that blind user gestures exhibit (i) higher inter-user variation, (ii) have higher noise in terms of jerks and jitters, and (iii) have lower velocity and more pauses.

These properties make existing gesture recognition algorithms unsuitable for blind users. Many gesture recognition algorithms [19, 61] work by learning the gesture pattern using large amounts of training data. However, it is time-consuming and expensive to collect training samples from blind users at such a scale. In addition, the higher inter-user variation means that the same gesture performed by different blind users is considerably different from each other, which makes training challenging. We find that even techniques such as few-shot learning with base models from sighted users [61] are ineffective (see § 6.2). Further, most gesture recognition algorithms rely on the accelerometer sensor to track linear motion [2, 58, 60], but there is higher noise in the accelerometer data due to jerks and jitters. AccessWear [27] is the closest related gesture recognition work that has similar goals to this work. AccessWear is designed for blind users and does not require extensive training data. However, AccessWear is designed to work only for simple forearm gestures and does not work for more complex gestures (see §6.1). Our work understands this gap and designs a gesture recognition system for blind users to recognize a set of 15 simple and complex gestures including forearm, compound, and shape gestures (see Table 1).

Instead, we present a new algorithm that addresses the unique challenges of blind user gesture recognition. Our intuition is that, even if the complete gesture signature looks different for different users (due to the high intra-user variability), there should exist a nucleus of the gesture that is user invariant. The goal then is to identify this user-invariant *micro-movement* in the gesture that remains consistent across users, which enables learning even with limited training data. Our gesture recognition algorithm relies only on gyroscope sensors because the accelerometer sensor is highly noisy. Since hand gestures consist of not only lateral movements but also rotational movements, a gyroscope sensor (which tracks rotational movement) provides sufficient signal for gesture recognition.

Our gesture recognition algorithm works on a 3D representation of the gesture trajectory to accurately capture gestures performed in free space. As a first step, we use a simple signal processing technique that tracks energy change points to identify an *approximate* gesture nucleus in the 3D trajectory. The goal then is to classify the gesture based on this approximate gesture nucleus. Unfortunately, the classification problem is more challenging in the 3D space compared to 2D primarily because of a higher degree of freedom. In free space, different users can perform gestures in different planes and different orientations. In 2D space, two gestures can be matched using Dynamic Time Warping (DTW) techniques that align the gesture temporally [31]. For instance AccessWear [27] uses this DTW matching for gesture recognition. However, this technique cannot be extended beyond simple 2D gestures.

Our contribution is a novel ensemble classifier that combines two models to accurately classify 3D gesture nucleus, even if gestures are performed in different planes and have different orientations. The first model formulates the gesture classification problem as a multi-view image classification problem. To this end, we design a multi-view CNN model that takes as input the gesture nucleus from 6 different angles, to account for the different planes and orientations of the gesture, and classifies the gesture. Importantly, we show how this CNN model can be trained with limited training data from blind users. The second model learns a classifier based on the geometric properties of the gesture nucleus such as curvature, torsion, and Centroid Distance Function (CDF). Geometric properties such as curvature and CDF are invariant to plane and orientation differences, which makes them well-suited to classify gestures performed in free space. Because this second model is learned over already extracted features, it does not require large amounts of training data. The final classifier combines the multi-view CNN model and the geometric properties-based model.

In this research, we make the following contributions:

- We examine hand gestures performed by both blind and sighted users, quantifying variations in gesture characteristics across dimensions such as inter-user distance, jerk, jitter, velocity, and pause duration. Analyzing these differences, we explore their implications in the context of designing gesture recognition algorithms. Subsequently, we establish design guidelines specifically tailored for gesture recognition systems intended for blind users. Our recommendations include the design of systems capable of effective performance with limited training data, reliance on gyroscope sensors exclusively, and the ability to capture subtle gestures.
- We develop an ensemble classifier to classify complex gestures performed by blind users across 3 categories: forearm, compound, and shape gestures. The classification algorithm can detect gestures performed in free space by capturing the 3D representation of the gesture. The classifier only requires limited training samples from the user and works well even when there is large inter-gesture variation amongst user gestures.
- We perform a real-world evaluation study to validate our gesture recognition system for 15 gestures across 10 blind users. We compare our classifier to state-of-the-art gesture recognition systems designed for sighted users, including TapNet [19] which trains a generalized deep learning model, Serendipity [57] which trains a personalized model, and a few-shot learning approach based on Xu et. al [61]. Our system achieves an accuracy of 92%. In comparison, the next best-performing classifier achieves an accuracy of 82%. We also compare our system with AccessWear [27] which is designed specifically for blind users and does not require training. AccessWear only achieves an accuracy of 68% since it is designed to work for only simple gestures.

2 BACKGROUND AND RELATED WORK

2.1 Gesture recognition

There has been considerable work in recent years on recognizing hand gestures using Inertial Measurement Units (IMUs). An IMU is a low-cost sensor, consisting of an accelerometer, a gyroscope, and sometimes a magnetometer. It is commonly available in commodity devices such as smartwatches, smartphones, and modern earable devices. The sensors track the various motion artifacts of a gesture—the accelerometer sensor measures the linear and gravitational acceleration, the gyroscopes measure the angular (rotational) velocity and the magnetometer measures the strength of the magnetic field intensity.

State-of-the-art gesture recognition systems learn the gesture performed by the user by tracking one or more sensors available in the IMUs. These algorithms can be broadly classified into two categories: (1) signal processing-based algorithms and (2) learning-based algorithms. Early signal processing-based algorithms used the fusion of accelerometer and gyroscope sensors to recognize gestures [6, 20, 42] using Kalman filters [7, 33, 45], Hidden Markov Models (HMMs) [48, 54], or template matching algorithms such as Dynamic Time Warping (DTW) [35, 36, 62].

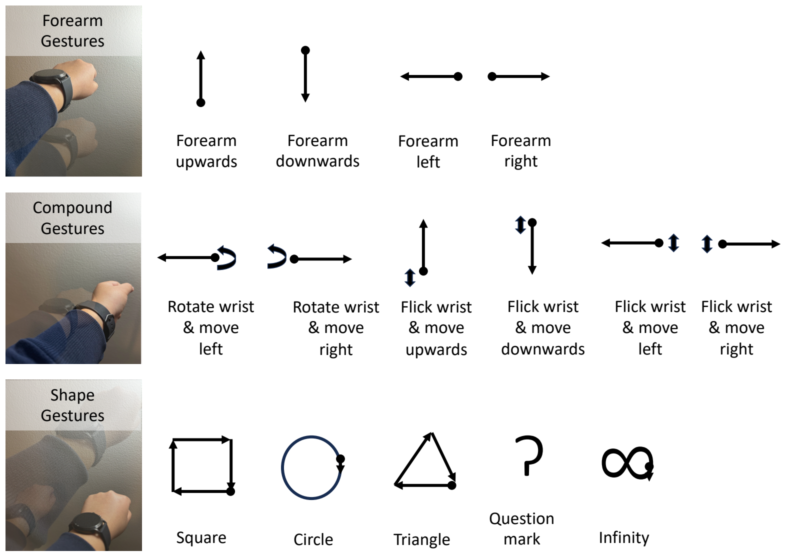

Figure 1: The 15 hand gestures performed by the users under three categories: forearm gestures, compound gestures, and shape gestures.

With the advances in machine learning, the more recent gesture recognition algorithms are learning-based, most often using deep learning techniques. The algorithms use architectures such as convolutional neural networks (CNNs) [22, 37], recurrent neural networks (RNNs) [39, 52] and Long Short-Term Memory (LSTM) [3, 30, 64], to learn the gesture performed by the user. These systems can learn generalized patterns in gestures, and perform gesture customization. However, these systems require a large amount of training data.

External and on-body cameras have also been used for hand tracking and gesture recognition. Sharp et. al [46] and Ma et. al. [38] used depth cameras to track hand motion in real-time, while modern VR systems like Quest 2 [1] have also used cameras for hand tracking. Additionally, wearable cameras have been investigated, with WristCam [9] and Chen [8] et. al. using wrist-worn RGB cameras for hand gesture recognition. Despite their robustness and accuracy, these systems necessitate line-of-sight, specific sensors and lighting conditions, and significant computational resources, and raise considerable privacy issues.

There are also several other gesture recognition algorithms that use specialized sensors including proximity detectors [10, 16], barometric pressure sensors [23, 49], electrical impedance tomography [21, 66] and WiFi-based gesture recognition [25]. However, these works require specialized sensors that are not always accessible to users.

2.2 Gesture-based interactions for blind users

Blind users interact with their computing devices using a screen reader [26], using touchscreen gestures. Several studies [4, 32, 59] have shown that these touchscreen gestures are challenging because users cannot perform one-handed interactions, the gestures are often overloaded, and the users are subject to shoulder-surfing attacks.

Instead, there has been recent work on using alternate interaction modalities to replace touchscreen gestures. Dim et. al. [13] study the preferences for mid-air TV gestures of blind users through user-elicitation and choice-based user studies. Previous works have explored ring-based gesture interactions [34] and deformable surface interactions [14] as alternate modalities. Blind users also use voice for interactions. But voice interactions are unreliable in noisy environments, can compromise the privacy of a user, and are prone to errors [4].

Hand gesture interactions using smartwatches are an attractive alternative because users do not have to use specialized sensors. Malu et al. [40] explore smartwatch interactions for users with upper body impairments including blind users. They establish that the physical abilities limit the users' ability to perform tap gestures on small target areas that need precision. Hand gestures on a smartwatch have been used to identify "wet floor" signs [43] for blind users. Recent work, AccessWear [27] finds that blind users prefer smartwatch gestures as an alternate interaction. They design a gesture recognition algorithm that uses simple template matching. However, this technique does not work well for more complex shape and compound gestures. Our work finds that complex gestures are better represented in 3D space and the 2D template matching techniques presented in AccessWear do not scale up the complex gestures (see § 6.2)

3 CHARACTERIZING BLIND USER GESTURES

We conduct an IRB-approved comparative experiment with blind and sighted user populations to understand how the different populations perform hand gestures. The main motivation of this study is to derive design principles to develop an effective hand gesture recognition system for blind users.

To this end, we study gesture-specific properties including range of motion, jerks, jitter noise, and speed, and characterize the differences in gestures performed by blind users versus sighted users. We

Category	Gestures	Median SEQ Rating	Average Rating
Forearm	1. Forearm upwards	7	7
	2. Forearm downwards	7	
	3. Forearm horizontally left	7	
	4. Forearm horizontally right	7	
Compound	1. Rotate wrist by 90 deg and move right	6.5	6.66
	2. Rotate wrist by 90 deg and move left	6.5	
	3. Flick and forearm upwards	6.5	
	4. Flick and forearm downwards	6.5	
	5. Flick and forearm horizontally left	7	
	6. Flick and forearm horizontally right	7	
Shape	1. Square	6.5	6.4
	2. Circle	7	
	3. Triangle	7	
	4. Question mark	5.5	
	5. Infinity	6	

Table 1: The 15 hand gestures performed by the users under three categories: forearm gestures, compound gestures, and shape gestures. The table shows the corresponding Single Ease Question (SEQ) rating given by blind users. (7: high preference, 1: low preference). All the gestures scored high on user preference.

analyze the implications of these differences with an eye towards designing a better gesture recognition system for blind users.

3.1 Participants

We recruited 10 blind participants between the ages of 38-64 (5 male and 5 female). Out of the 10 users, 8 participants were blind since birth and 2 participants were blind since ages 3 and 11. None of the participants had any motor impairments that affected their ability to perform hand gestures. In addition, we recruited 16 sighted participants between the ages of 22-36 (10 male and 6 female) to compare gestures performed by blind vs sighted users.

3.2 Apparatus

All users were asked to wear a Fossil Gen 5 smartwatch. When the participants performed gestures, the IMU data was streamed at 100Hz via Bluetooth to the Pixel 3 XL smartphone using a custom data logger application. It took 1 hour to conduct the user study and each blind participant was paid $75 for their time. The offline data analysis is performed on a MacBook M1 laptop.

3.3 Design

We designed a comparative experiment in which we studied how participants (both blind and sighted) perform smartwatch gestures. We asked each participant to perform 15 gestures in a counterbalanced order, repeating each gesture 10 times. We also asked each participant about their gesture preference. The gestures included five forearm directional gestures, five compound gestures (i.e., wrist gesture followed by a directional gesture), and shape-related gestures. Figure 1 shows the gestures and Table 1 shows the details of the gestures and the preferences of blind users to use these gestures. We selected this set of gestures because they are widely recognized and commonly employed in various gesture recognition studies [29, 55, 57, 61, 65]. Moreover, each of these gestures received a favorable response from the blind users, with all scoring high on the Single Ease Question scale (SEQ rating > 6). Users found all these gestures easy to perform.

3.4 Gesture Characterization

We estimate different gesture properties (Table 2) to characterize gestures performed by both blind and sighted users. To estimate these properties we collect time series sensor data from accelerometer and gyroscope. Previous works use a similar set of gesture properties to characterize touchscreen gestures of blind versus sighted users [24]. This work finds that there is a quantitative difference in how blind users perform touchscreen gestures and how sighted users perform the same gestures. Feiz et al. [15] explore the differences between how visually impaired users do smartwatch gestures compared to their sighted peers. AccessWear [27] makes an observation that gestures performed by sighted and blind users are different and that the accelerometer sensor has more noise for blind users' gestures. Going beyond AccessWear, in this work, we quantitatively compare the difference in gestures of blind and sighted users across dimensions such as inter-user distance, jerk, jitter, velocity, and pause duration. We infer the implications of these differences with respect to designing gesture recognition algorithms for blind users and distill a set of design guidelines. We discuss the differences in the following subsections.

3.4.1 High inter-user variance. We estimate the inter-user variation while performing gestures using Dynamic Time Warping (DTW). DTW is the standard way to measure variations between two time series wherein two gesture time series are aligned and then the Euclidean distance between them is estimated [41]. We use the significant axis of gyroscope sensor data (the axis that has the highest rotational movement) for comparing the gestures of the users. We measure the similarity of a gesture by estimating the distance between each gesture and a predefined gesture template across users.

If we use one of the blind user's gestures as the template and compare it with the same gestures for all other users, the average distance is 78.21, 166.27, and 72.45 for forearm, compound, and shape gestures respectively. In contrast, the distance between the

Gesture Property	Definition
Inter-user variation	Measure of the Euclidean distance between each gesture and a predefined gesture template across users.
Jerk	Sudden changes in the acceleration of a signal with respect to time (m/sec^3).
Jitters (Accelerometer and Gyroscope)	Measure of low-frequency noise in the accelerometer and gyroscope (<10Hz) readings
Velocity	Cumulative integral of acceleration (m/sec).
Pause duration	Time period in a gesture where the linear acceleration and angular velocity are near zero (sec).

Table 2: Quantitative gesture properties used to compare the gestures performed by blind and sighted users.

Property	Blind Users (Mean ± Std dev)	Sighted Users (Mean ± Std dev)
Inter-user distance	78.21 ± 34.19	44.20 ± 20.39
Jerk (m/sec^3)	25.32 ± 14.93	7.79 ± 1.62
Jitter gyroscope	3.26 ± 2.30	2.93 ± 1.63
Jitter accelerometer	5.70 ± 3.81	3.10 ± 1.85
Velocity (m/sec)	5.07 ± 1.96	7.35 ± 3.04
Pause duration (sec)	0.51 ± 0.035	0.20 ± 0.052

Table 3: Comparing gesture properties of blind and sighted users for forearm gestures.

Property	Blind Users (Mean ± Std dev)	Sighted Users (Mean ± Std dev)
Inter-user distance	166.28 ± 63.25	96.37 ± 63.30
Jerk (m/sec^3)	37.05 ± 10.74	4.04 ± 4.80
Jitter gyroscope	9.44 ± 6.73	6.54 ± 5.06
Jitter accelerometer	14.48 ± 3.44	9.56 ± 5.33
Velocity (m/sec)	4.57 ± 2.33	7.12 ± 4.15
Pause duration (sec)	0.882 ± 0.04	0.11 ± 0.0147

Table 4: Comparing gesture properties of blind and sighted users for compound gestures.

Property	Blind Users (Mean ± Std dev)	Sighted Users (Mean ± Std dev)
Inter-user distance	72.45 ± 37.13	49.35 ± 42.16
Jerk (m/sec^3)	14.13 ± 4.57	4.98 ± 2.04
Jitter gyroscope	2.75 ± 2.47	1.97 ± 0.95
Jitter accelerometer	10.50 ± 4.11	5.44 ± 3.33
Velocity (m/sec)	6.67 ± 2.30	11.36 ± 3.46
Pause duration (sec)	0.79 ± 0.17	0.18 ± 0.028

Table 5: Comparing gesture properties of blind and sighted users for shape gestures.

gestures performed by sighted users is 44.19, 96.37, and 49.34 respectively (see Table 3, 4, 5). An Independent two-sample t-test was performed ($t_{24} = 6.399, p = 0.00076$), indicating a highly significant difference between the DTW distances for blind and sighted users. To illustrate this difference, Figure 2 shows how different users perform the square gesture. This figure shows the gyroscope time series data of 5 different blind (a-e) and sighted users (f-j). We can see that the gestures of sighted users follow a similar pattern, while the gestures of blind users are different from each other.

Implication: The gestures performed by blind users exhibit a greater variation among individual users. This significant diversity makes learning-based approaches more challenging to design. Having lots of training data is the key when high variance is observed, but a large amount of training data is difficult to obtain from the blind user population.

3.4.2 *More jerk and jitter.*

Jerk and jitter are two gesture features that quantify the noise in a gesture. We use these features to get an estimate of the noise recorded in the motion sensor data. Jerk measures changes in the acceleration of a signal with respect to time. The higher the jerk, the more sudden the change in acceleration. Mathematically, Jerk is calculated as a third derivative of the acceleration data estimated over time On average, blind users had 136% more jerky gestures than sighted users. (see Table 3, 4, 5). An Independent two-sample t-test was performed ($t_{24} = 10.41, p = 0.000019$), indicating a highly significant difference in the jerk for blind and sighted users' accelerometer data.

Jitter estimates the low-frequency noise present in the gestures induced due to motion artifacts contributing to the sensor noise. To calculate jitter we smoothen the raw accelerometer and gyroscope data using a 10 Hz smoothing filter. We then calculate the Normalized Mean Squared Error (NMSE) between the raw and smoothened signal to obtain the jitter.

On average, blind users' accelerometer data had 51.56% more jitter than sighted users' accelerometer data. An independent two-sample t-test demonstrated ($t_{24} = 15.93, p = 0.00073$) significantly more jitters in accelerometer data of blind than sighted users Figure 3 visually compares the jerk in the accelerometer data of a blind user and a sighted user.

Whereas blind users' gyroscope data had only 29.91% more jitter than sighted users' gyroscope data. An Independent two-sample t-test was performed ($t_{24} = 0.75, p = 0.43$), which signifies that more jitters in blind users' gyroscope data than sighted users' is not statistically significant.

Implication: Most existing gesture recognition systems fuse time-series data from both the accelerometer and the gyroscope sensors (or in some cases, use the accelerometer alone [58, 60]). However, high jerk and jitter in the accelerometer data indicates that the accelerometer data (which captures lateral motion) for blind users is noisy as compared to sighted users. This makes the accelerometer sensor less reliable for blind users.

In fact, we show in our evaluation that using even a small percentage of data from the accelerometer sensors considerably reduces the gesture recognition performance. In other words, gesture recognition systems that either use an accelerometer alone or in combination with gyroscope [19, 57] will work poorly for blind users.

3.4.3 *Lower velocity and more pauses.*

We next estimate the velocity of gestures and the duration of pauses during the gesture.

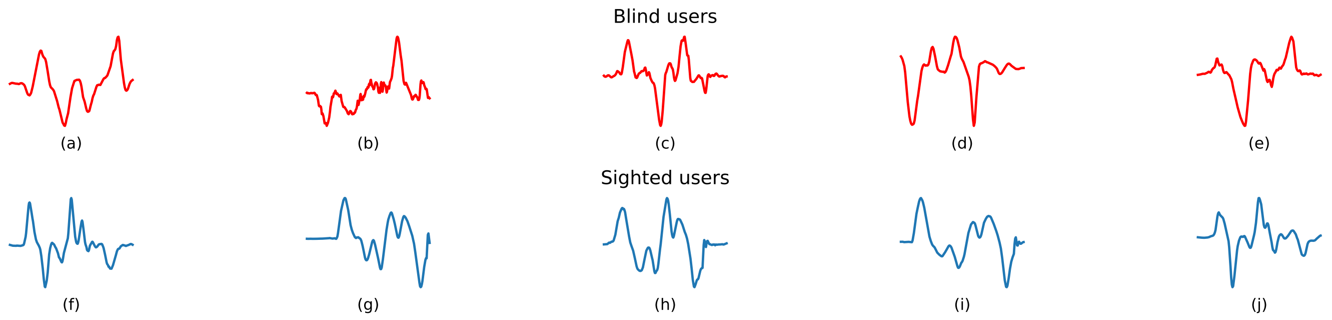

Figure 2: High variation is observed in the blind user population while performing the same gesture as compared to the sighted user population. The figure shows the time series signal of the significant axis of the gyroscope data for 5 different blind users and sighted users while they perform a square gesture.

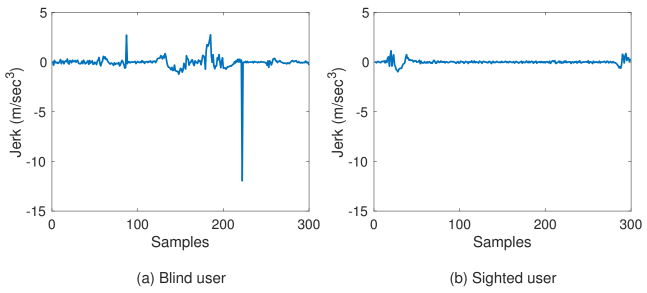

Figure 3: Measure of jerk for two sample users that illustrates that the magnitude of jerk observed for a blind user (a) is higher than a sighted user (b).

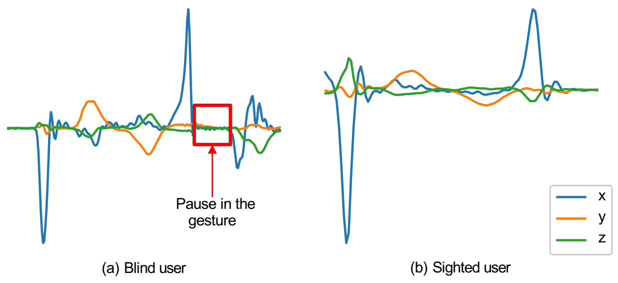

Figure 4: Accelerometer readings shown for two sample users to illustrate pauses in gestures. A sample blind user (a) takes an extended pause to complete a compound gesture (pause duration is marked by the red box). No such pause is observed for this sample sighted user (b).

Velocity is calculated as the cumulative integral of acceleration over time. On average sighted users performed a gesture 35.06% faster than blind users. An independent two-sample t-test was performed ($t_{24} = 10.10, p = 0.0006$), which signifies that the gestures of blind users are slower than sighted users and this difference is statistically significant.

The pause duration is estimated as the amount of time (sec) where the linear acceleration and angular velocity were near zero. These instances are those when the user takes pauses to complete a single gesture. On average pause duration for blind users was 126.6% more than sighted users, meaning blind users took more extended pauses to complete a gesture. An independent two-sample t-test was performed ($t_{24} = 3.94, p = 0.0019$), which signifies that the gestures of blind users had significantly more pauses than sighted users. Figure 4 shows that the blind user took an extended pause to complete the compound gesture; while no such pause was observed for the sighted user.

Implication: The slow motion and extended gesture pauses make the gestures of blind users more subtle compared to sighted user gestures. Gesture recognition algorithms designed for sighted users can miss these subtle movements and classify them as noise. This can lead to poorer gesture recognition performance.

4 GESTURE RECOGNITION SYSTEM FOR BLIND USERS

Our user study uncovers characteristics of blind user gestures that require rethinking the design of gesture recognition algorithms. Below we describe the design guidelines we distill from the user study.

4.1 Design guidelines

Overcoming inter-user variability using limited training data. The high inter-user variance in gestures indicates that each user has a different style of performing a gesture, which makes learning a generalized gesture pattern difficult. A common approach to learning gesture patterns is to use large amounts of training data. For example, state-of-the-art gesture recognition work [19] requires 135K training samples. Collecting large-scale data from blind users is expensive and time-consuming.

Another approach is to train the model on sighted users' data and then use few-shot learning approaches [61] to personalize the model for blind users. However, we find that this approach does not work well (§6.2) because of the difference between how blind users and sighted users perform gestures. Our design goal, therefore, is to design a gesture recognition system that works well with limited training data.

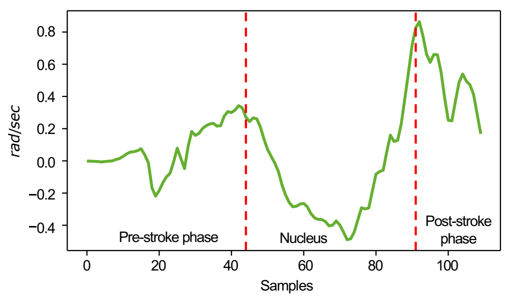

Figure 5: Three phases of a gesture: pre-stroke, nucleus, and the post-stroke phase. The nucleus (marked in red) is the phase, where the actual gesture is performed. There is typically less user variance in the nucleus compared to the rest of the gesture.

Designing gesture recognition with gyroscope data only. Most existing gesture recognition systems fuse data from accelerometer and gyroscope or use accelerometer alone. Unfortunately, our user study shows that the jitter in blind user accelerometer data is high, and is jerky, making accelerometer data highly noisy. The third sensor in IMUs, namely the magnetometer, is known to work poorly in environments with large ferromagnetic interference; for example, in the presence of computer machines. To this end, our system leverages the gyroscope sensor alone. This is based on the insight that all arm movements have a significant rotational component since the motion is restricted to the surface of a sphere that is centered at the shoulder, elbow, or wrist. One challenge is that gyroscope data suffer from large drifts that add noise over time [47].

Capturing subtle gestures. Gestures performed by blind users are slower and have more pauses. These signals could be mistaken to be noise by gesture recognition systems. To avoid this, we should use a generous first-cut filter that has lower thresholds for noise. In other words, the system should trade off processing more (possibly) spurious gesture signature for missing subtle gestures.

4.2 Insight: Identifying micro-movements in 3D representation

Gestures comprise of three phases [5, 63]: (i) a pre-stroke phase, where the user positions their hand to perform a gesture, (ii) the nucleus which is when the gesture is performed, and (iii) post-stroke phase, where the user takes the hand back to resting position after performing a gesture. For example, Figure 5 shows the gyroscope time series as a user is performing the flick gesture. The figure marks the pre-stroke, nucleus, and the post-stroke phase. Here we use the lightweight change point detection algorithm designed for simple gestures [27] to mark the different phases. The algorithm calculates the root-mean-square (RMS) energy of the signal over a window and calculates the windows with the highest change in RMS.

Previously, researchers have shown that even though the entire gesture signature is different for different users, small micro-movements in the gesture (i.e., the nucleus) can be user-invariant [27]. For example, Figure 6 shows the gyroscope trace as 5 blind users perform the forearm-up gesture. Even though the trace looks different, the nucleus (marked in red) is visually similar.

The key challenge is that, while the user-invariance is established for simple gestures, it is not clear if such a user-invariant micro-movement can be identified for more complex gestures. We find that for complex gestures such as shape and compound gestures, the micro-movement is hard even to identify. For example, Figure 7 shows 5 blind users performing the square gesture. Visually, it is hard to identify a consistent nucleus in this time series. Our analysis reveals that finding such user-invariant micro-movements is challenging across the more complex shapes and compound gestures.

Instead, we turn our attention to the 3D representation of complex gestures. The complex gestures are performed in free space that spans across multiple planes, unlike 2D gestures performed on a touchscreen. For example, Figure 8 shows the approximate extracted nucleus for 2 blind users when they are performing the shape gesture, but this time represented in 3D (See § 5.2 for details on how to extract the 3D representation and identify the approximate nucleus for complex gestures.) Compared to Figure 7 which is a 2D representation of the same gesture, visually, one is able to match the two nucleus when viewed in 3D.

One additional advantage of leveraging these short micro-movements to identify gestures is that it naturally overcomes the gyroscope drift problem. Recall that gyroscope sensors can drift over time; but since the micro-movements are short, we do not encounter problems with drift. In fact, our experiments reveal that the average duration of micro-movements is 620 msec where the gyroscope drift is negligible. Additionally, the intermittent nature of gestures allows for sensor re-calibration between gestures.

5 GESTURE RECOGNITION BY TRACKING 3D GESTURE TRAJECTORY

We present a gesture recognition system that works well for blind user gestures based on the insights described above. Figure 9 presents the system overview. The input to the system is a trace of gyroscope data (x, y, and z-axis). The system identifies the approximate (potential) nucleus in the data. The input and the nucleus detection are marked "Pre-processing" in the figure. The approximate nucleus is detected using a low threshold filter, to ensure that subtle movements are not missed.

The nucleus is fed to the classification module (marked "Classifier"). This module uses the 3D representation of the gesture and an ensemble classifier model to accurately classify the gesture. Below, we first describe the classification module and then describe the rest of the system.

5.1 Classification based on 3D representation

The central idea is to classify a gesture based on the (approximate) nucleus data, where the nucleus is represented in 3D space. Recall that the nucleus is only a small part of the entire gesture and the nucleus exhibits higher similarity across users. In contrast, the entire gesture signature is dissimilar across different users.

The challenge is that users can perform gestures in different planes and orientations when performing the gesture in free space. Classically, when gestures are performed/represented in 2D, the

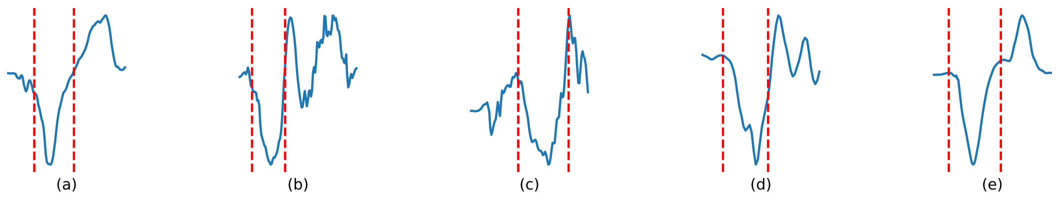

Figure 6: The nucleus (marked in red) when a forearm up gesture is performed by 5 different blind users. The pre-stroke and post-stroke phases look visually dissimilar, but the nucleus is visually consistent across users. The figure shows the significant axis of gyroscope data.

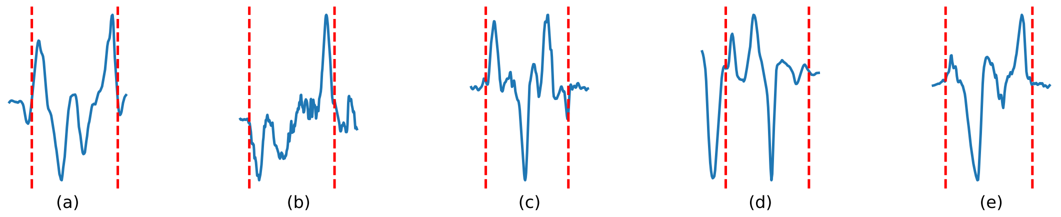

Figure 7: The figure shows the significant axis of gyroscope data when a shape gesture (square) is performed by 5 different users. In this figure, it is difficult to identify a nucleus. In general, identifying the nucleus of complex gestures such as shape gestures and compound gestures is difficult in 2D space.

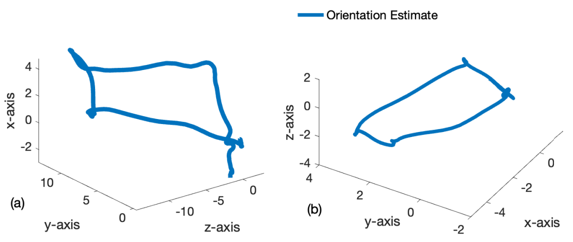

Figure 8: The figure shows the approximate nucleus for a square gesture performed by 2 blind users, but this time represented in 3D space. Visually, one can match the two nucleus.

time series data is matched using techniques such as DTW [41], that align the gestures temporally. However, DTW techniques do not work well when gestures can be performed in different planes and orientations.

We address this challenge by designing an ensemble classifier that consists of two models. For the first model (multi-view CNN), we formulate the gesture classification problem as an image classification problem by converting the gesture trajectory to an image. Image classification is a well-studied problem with sophisticated and accurate classifiers. To account for the different planes and orientations, we design a multi-view CNN model that takes as input the trajectory image from multiple angles.

However, reducing the gesture trajectory to an image means that the geometric features of the trajectory are lost. These geometric features such as curvature and Centroid Distance Function (CDF) capture the shape of the gesture and are agnostic to plane and orientation. To this end, the second model (geometric-property-based) extracts geometric properties and uses a 1-D CNN model to classify the gestures based on the properties. The ensemble classifier combines both the multi-view model and the geometric-property-based model for the final classification. Below, we describe the two models and our approach to train the models with limited data from blind users.

5.1.1 Multi-view CNN model.

The gesture trajectories produced by blind users resemble 2D hand-drawn images but within a 3D spatial context. We cast the gesture classification problem to an image classification problem, but look at the image from multiple angles to take into account the different planes and orientations of the gesture. For example, Figure 10 shows images of the gesture trajectory from different views.

We then use a multi-view CNN to classify these images. Multi-view CNN is a well-studied problem [50] with models attaining high accuracy. However, these models require large amounts of training data that is not available

Instead, our approach is to build on a pre-trained classifier. Specifically, we use the *EfficientNetB0* CNN architecture [51] pre-trained on *ImageNet* [11] as a foundational feature extractor for each view. We modify this architecture by removing its top layers and appending a 2D convolution layer, followed by a subsequent max-pooling layer. We use the blind user data to train the weights for the 2D convolution and pooling layers, but the weights of the pre-trained model remain the same. We note here that are able to use the pre-trained weights because we work with the gesture nucleus and not the entire gesture. We discuss why this works better than state-of-the-art transfer learning approaches in §6.1.

5.1.2 Geometric property-based model.

In three-dimensional (3D) geometry, curvature, torsion, and centroid distance function (CDF) concepts describe the shape and orientation of curves or

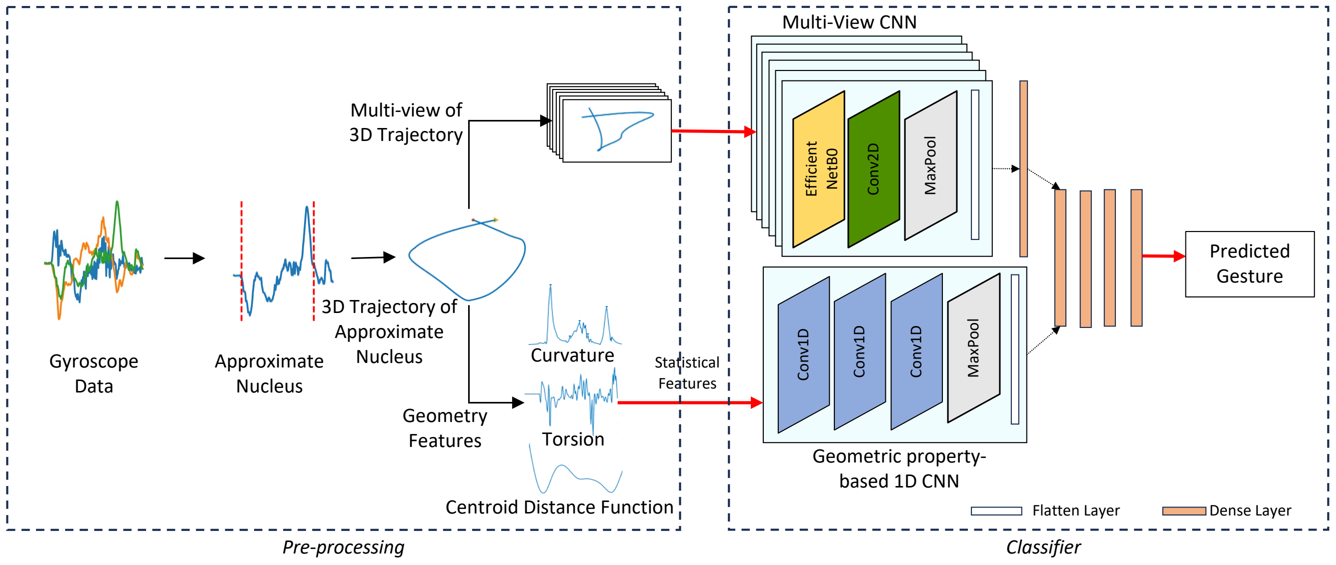

Figure 9: The end-to-end design of the gesture recognition algorithm.

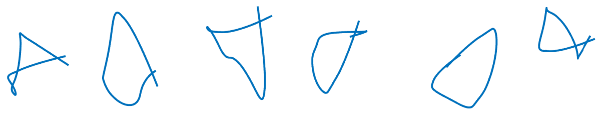

Figure 10: The gesture trajectory of a square gesture performed by a blind user, viewed as an image from 6 distinct angles.

paths in 3D space. Curvature and CDF are considered to be invariant across different orientations and planes. We calculate these properties on the estimated 3D nucleus trajectory and use these properties to learn a classifier.

Curvature measures how much the gesture deviates from being a straight line at a particular point on the curve. Curvature can be thought of as a measure of how "curved" or "bent" a trajectory is at a specific point.

Torsion measures how much a gesture trajectory twists or rotates as it moves along its path in 3D space. Detecting torsion is valuable for recognizing compound gestures that involve not only changes in direction but also rotational motion.

Centroid Distance Function (CDF) measures how far a specific point on the trajectory is from this "center" or "centroid." For hand gestures, CDF identifies the moments in the gesture where the hand is positioned closer to or farther away from the "average" (resting) hand position. High CDF values correspond to points where the hand is extended away from the resting position, while low CDF values indicate points where the hand is closer to the resting position.

Our goal is to build a gesture classifier using these three geometric properties. Figure 11 shows the curvature, torsion, and CDF plots for a square gesture. To this end, we design a 1D CNN model to learn the geometric property-based model. We then extract the statistical features of the three properties to feed as an input to the 1D CNN pipeline. The statistical features used are minimum, maximum, range, mean, median, standard deviation, coefficient of variation, zero-crossing rate, skewness, kurtosis, area under the curve, energy, inter-quartile range, energy, and first 10 FFT coefficients. This model comprises of three 1D convolutional layers, a max-pool layer followed by a dropout layer to enhance generalization. Since this model is learned over already extracted features, it does not require large amounts of training data.

5.2 End-to-End system

We discussed the classifier above. In this section, we discuss the rest of the system including the parameters used for learning.

The input to the classifier is the 3D orientation estimate over time of the approximate nucleus of the gesture. We first determine the gyroscope axis with the highest rotational variation (referred to as the significant axis). To approximate the nucleus boundaries, we employ a change-point detection algorithm that assesses the empirical change in root-mean-square (RMS) values along the significant axis of the gesture sequence [28]. We use a more liberal threshold (empirically chosen value: 10) in the RMS change-point algorithm to accommodate the subtle gestures made by blind users. This can trigger false positives for the gesture instances, but these false positives are filtered by the subsequent classifier block. Figure 7 shows the approximate nucleus boundaries detected for the square gesture.

Since the gesture is performed in a 3D space, we convert the extracted approximate nucleus to 3D orientation estimate using only the gyroscope data. This process involves utilizing the boundaries identified by the change-point algorithm and extracting the 3-axis gyroscope data within these limits. Gyroscopes measure the rate of rotation (angular velocity) around 3 axes (X, Y, Z), and to obtain

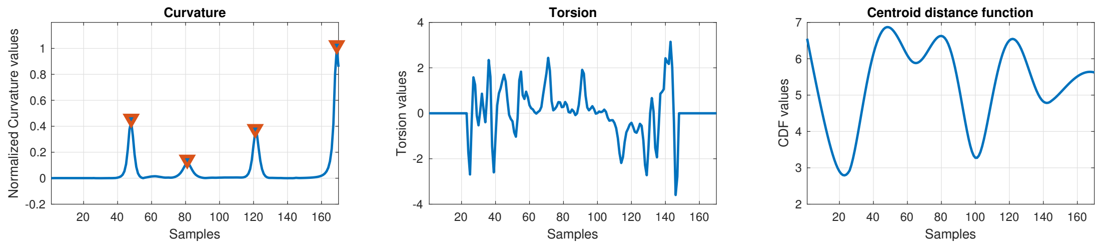

Figure 11: Curvature, torsion, and the centroid distance function (CDF) plots for a square gesture. The user makes 4 sharp changes in direction while making a square gesture, this is represented by the 4 peaks in the curvature curve. The torsion curve shows how the user's hand rotated over time. CDF curve shows how the user's hand moved farther and closer from the resting hand position.

the 3D orientation (angular position) from this data, we integrate the angular velocity with respect to time. In mathematical terms, if $\omega(t)$ represents the angular velocity at time t, then the orientation $\theta(t)$ at a given time t is obtained by integrating the angular velocity:

$$\theta(t) = \int_0^t \omega(\tau)\, d\tau$$

In practice, to represent orientation quaternion integration using Euler's method is used [12]. Estimated orientation provides insight into how the hand is positioned in three-dimensional space at the specific moment t. Figure 8 shows the Euler angle based orientation estimate [56] over time of the approximate 3D nucleus, which is input to the model.

The final classifier combines the multi-view CNN model and the geometric properties-based model. We perform a 3-fold cross-validation split to make test and train sets, with 1050 samples in the train and 450 samples in the test set for each fold.

We jointly train the top of the multi-view CNN and the complete geometric properties-based model using a learning rate of 0.0001 on the Adam optimizer. ReLU activation was used for all the layers, and soft-max activation with L2-regularization was used on the output classifier layer. The loss function used is sparse-categorical-cross-entropy. We trained the model for 100 epochs.

6 EVALUATION AND RESULTS

We evaluate our gesture recognition algorithm using real gesture traces collected from blind users. We conducted a real-world, IRB-approved user study with *10 blind participants* where we collect IMU sensor data as users perform 15 gestures (described in §3). We stream this sensor data to our gesture recognition algorithm and evaluate the accuracy, false positives, and false negatives, against ground truth. We collected a video (of only the hands) as users perform gestures to obtain the ground truth.

We use the same traces to compare our algorithm to the state-of-the-art and perform ablation studies.

6.1 Accuracy of gesture recognition

Figure 12 shows the overall accuracy of the system for 15 gestures across 10 blind users. We perform 3-fold stratified cross-validation with 70-30 train-test split and the average accuracy across all the folds is 92% (precision: 92%, recall: 91%). The confusion matrix

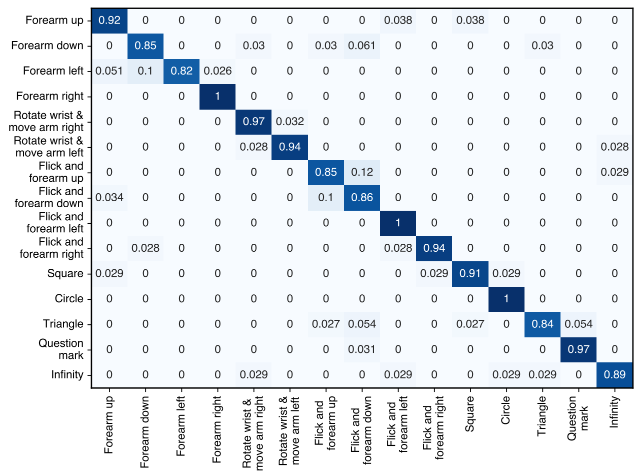

Figure 12: Confusion matrix for the gesture recognition system across 10 blind users for 15 gestures. Overall average accuracy is 92%

shows the accuracy of gesture recognition, as well as the percentage of the time the algorithm misclassified the gesture.

Some of the gestures such as *circle* and *forearm right* are recognized with a 100% accuracy. In general, the gestures with relatively lower accuracy are those that can be confused with other gestures because of how the gestures are performed. For example, *forearm up* gesture is recognized with an accuracy of 92%, because the algorithm gets it confused with the *flick and forearm left* gesture. This is because the *flick* gesture looks like a *forearm up* gesture for some users.

This high accuracy is achieved even when users perform gestures differently. For example, for the shape gestures, some users performed the gesture using their fingers to trace the shape, while others used their fists. The algorithm is able to classify the gestures accurately despite these differences.

For the previous experiment, we train a single classifier to classify gestures across all three categories. We next look at the accuracy of gesture recognition if we build a different classifier for each category of gesture: forearm, shape, and compound. Figure 13 shows the results. The forearm gesture model has an accuracy of 94% in recognizing forearm gestures, the compound gesture model achieves 91% accuracy, and the shape gesture model achieves 92%

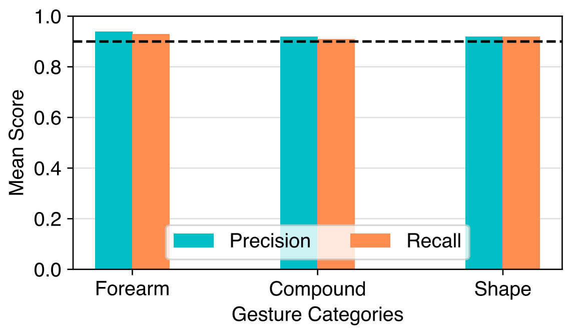

Figure 13: The precision and recall if we train a separate model for each gesture category: forearm, shape, and compound. The average accuracy across all models is 92.3%. This is close to the 92% accuracy we achieved by training a single model to recognize gestures across all categories.

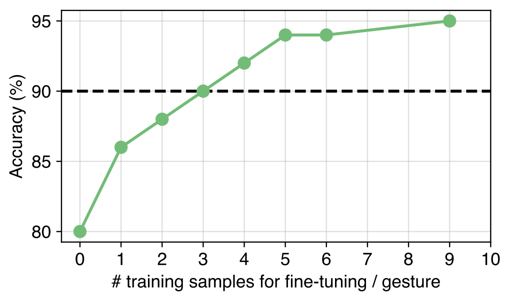

Figure 14: The average leave-one-user-out accuracy is 80%. Fine-tuning the model with 3 samples of each gesture from a user increases the accuracy to 90%.

accuracy. Overall average accuracy is 92.3%, compared to the 92% accuracy achieved when training a single model. In other words, our algorithm is able to achieve similar accuracy by training a single model, rather than train a separate model for each gesture category.

To evaluate the generalization capabilities of the model, we perform the leave-one-user-out evaluation, a cross-validation technique involving the training of the model on nine users while testing on the omitted user iteratively. This method assesses the model's performance across diverse users by leaving out one user at a time during training and evaluating on that user separately. The average leave-one-user-out accuracy for blind users is 80%. However, just by adding 3 samples of each gesture per user for fine-tuning, the model achieves an accuracy of 90%. Figure 14 shows the increase in average accuracy as the model is fine-tuned with some samples from a new user. The accuracy saturates at 3 samples from a new user. Thus, with less than 1 minute of training data from a new user, the model can recognize gestures with high accuracy.

6.2 Comparing with the State-of-the-art gesture recognition systems

We compare the performance of our gesture recognition algorithm with three state-of-the-art (SOTA) learning-based models and one gesture recognition technique that is designed for blind users, that does not require extensive training data.. Each baseline model represents a different framework: TapNet [19] is the SOTA deep learning model and Serendipity [57] is the SOTA model that uses personalization. The third baseline is the SOTA few-shot learning-based model that closely follows Xu et al. [61]. All of these models are designed for sighted users. The fourth baseline, AccessWear [27] is SOTA gesture recognition for blind users that is designed to recognize simple gestures.

	Accuracy	**Precision**	**Recall**
Our system	**92%**	**92%**	**91%**
TapNet [19] (CNN based)	77%	77%	77%
Modified TapNet (without accelerometer data)	80%	81%	82%
Serendipity [57] (personalized SVM based)	82%	82%	82%
Modified Serendipity (without accelerometer data)	86%	86%	87%
Few shot learning approach (closely following Xu et al. [61]	45%	45%	47%
AccessWear [27] (simple gesture recognition for blind users)	68%	65%	68%

Table 6: Comparing our gesture recognition algorithm to three state-of-the-art learning models and one model for blind users. The three learning based models are (i) TapNet [19] which learns a generalized deep learning model, (ii) Serendipity [57] which learns a personalized model for each user, (iii) a model that uses few-shot learning, based on Xu et al [61]. AccessWear [27] is a simple gesture recognition for blind users.

Table 6 shows the results across all the baseline comparisons; we present the results of the three baselines individually below.

TapNet [19] (SOTA deep learning model): This is a CNN-based tap detection model designed for mobile phones. TapNet runs both accelerometer and gyroscope data through a multi-task network to classify between tap and not tap and to learn the location of the tap. We modified the TapNet model so that it will take the same input, but will classify gestures. We split our data (10 users, 15 gestures performed 10 times by each user) into training (70%) and test (30%) sets to train a generalized model.

The accuracy of this baseline against the blind user gesture data is 77%, compared to the 92% accuracy achieved by our algorithm. The original TapNet model is trained with 135K samples, but such a large training set is not available in this scenario. Coupled with the high inter-user variance observed in our dataset, TapNet is not able to learn the gesture patterns for accurate classification. Since the accelerometer data is noisy in our dataset, we trained a modified TapNet model that used only gyroscope data. This improved the accuracy to 80% which is still considerably lower compared to our algorithm.

Serendipity [57] (SOTA personalized model): This work trains an SVM classifier for each user individually. Accordingly, we trained a personalized model for each blind user using the features described in the paper. The features are extracted from the time-series

Nucleus detection	Pipeline A	Pipeline B	Accuracy
Nucleus detection	Multi-view CNN	Geometric property 1D CNN	92%
No nucleus detection	Multi-view CNN	Geometric property 1D CNN	85%
Nucleus detection	Multi-view CNN (no base model)	Geometric property 1D CNN	55%
Nucleus detection	Multi-view CNN (with accelerometer)	Geometric property 1D CNN	71%
Nucleus detection	No multi-view CNN	Geometric property 1D CNN	78%
Nucleus detection	Multi-view CNN	Geometric property (LSTM)	78%
Nucleus detection	Multi-view CNN	Geometric property (Transformer)	72%
Nucleus detection	Multi-view CNN	No geometric property 1D CNN	66%

Table 7: Ablation study that removes/modifies one the three components of the algorithm: nucleus detection, multi-view CNN model, and geometric property based 1-D CNN model.

data obtained from both accelerometer and gyroscope and take into consideration the orientation in which the gesture was performed.

The mean accuracy across all the user's personalized models is 82%, which is compared to the 92% accuracy achieved by our work. As before, we train a modified Serendipity model by removing accelerometer data because of its noise. This modified model has an improved accuracy of 86%, but this is still lower than our approach.

Personalized models do perform better than generalized models such as TapNet because it can capture the unique characteristics of the user. Nevertheless, our algorithm performs better than the personalized models, because it is able to learn an accurate general classifier with relatively small amounts of training data.

Few shot learning based on Xu et al [61] The Xu et al. work introduces a deep learning-based gesture recognition model that uses a few-shot learning approach. A base model is trained with data from 512 participants, and the model is fine-tuned for each user's custom gesture with a small amount of personal data (only 3 samples). The model uses accelerometer and gyroscope data filtered at various frequencies as the input.

Since we do not have access to the base model, we train our own model with data from both sighted users and 70% of blind user data. We then augment this data set using the data augmentation techniques described in the paper. We implement the data synthesis pipeline described in the paper to simulate the natural motion variance of the gestures. This pipeline is a self-supervised encoder-decoder model. Finally, we implement a few-shot learning approach over the base model.

However, the accuracy of this approach is only 45%. This poor performance is due to multiple reasons. First, we find that data augmentation approaches do not work for blind users' gestures as they create even larger variance in the data. Further, there is a large difference between data from sighted users and data from blind users. Therefore, a base model using sighted user data does not work.

AccessWear [27] (SOTA gesture recognition for blind users): AccessWear uses pre-defined templates and matches the single axis of gyroscope data (one with the highest rotation) to this template. However, this technique is designed for simple forearm gestures that can be identified in 2D and does not work well for more complex gestures performed in 3D space like shape and compound gestures.

We implemented the AccessWear pipeline for blind users and the accuracy of this baseline is 68%. If separate models are trained for forearm, compound, and shape gestures, AccessWear achieves an accuracy of 91%, 37%, and 23% respectively. Templates are unique for forearm gestures owing to higher classification accuracy. For more complex gestures performed in the 3D space, the simple micro-movement templates do not scale up. Figure 7 shows the nucleus for the square gesture and due to high variance, the template matching technique does not scale up for such gestures.

6.3 Ablation Study

To evaluate how each module of the gesture recognition system contributes to the accuracy, we conducted an ablation study. Our gesture recognition pipeline consists of three components (see Figure 9): (A) Approximate nucleus detection (B) Multi-view CNN pipeline, and (C) Geometric property-based 1-D CNN model. We describe the effect of removing each of these components. Table 7 summarizes the results of the study (the first row represents the complete algorithm with all three components).

Approximate nucleus detection: As a first step, we remove the approximate nucleus detection and input the full gesture sequence to the ensemble classifier. In this case, the accuracy drops to 85%. Gestures have high inter-user variance in the pre-stroke and post-stroke phases, while the nucleus is more consistent across users. Therefore, learning is harder when using the entire gesture signature rather than using the gesture nucleus.

Multi-view CNN: Our multi-view CNN pipeline uses a baseline model that is pre-trained using ImageNet. If we remove this model, the accuracy drops to 55%. This is because we have limited training samples from blind users and learning a generalized model with a small number of samples is difficult.

We note that using a pre-trained model in our algorithm is similar to the transfer learning approach (Xu et.al. [61]) described in the previous section, which worked poorly. However, our work uses the pre-trained model to learn the gesture nucleus. The nucleus of the gesture remains fairly consistent across all users. However, the SOTA transfer learning model [61] uses the pre-trained weights to learn the entire gesture; since the gestures of sighted users and blind users vary considerably, their approach does not work well in our setting.

Recall that we do not use accelerometer data in our learning pipeline because it is noisy. In the next ablation study, we look at how accelerometer data will affect the accuracy. We use a standard sensor fusion technique called complementary filtering [17] to combine the gyroscope and accelerometer data. The rest of the pipeline remains the same. The accuracy drops to 71% because of the noise in the accelerometer. Finally, if we remove the multi-view CNN model completely from the algorithm, the accuracy drops to

78%. The multi-view CNN model is able to classify gestures based on powerful image classification techniques, which is critical to the accuracy of the algorithm.

Geometric-property based 1D CNN: The next pipeline is a 1D CNN model that classifies gestures based on geometric properties. As a first ablation study, we use different architectures to learn the classifier instead of the 1D CNN model. Specifically, we use an LSTM and Transformer architecture instead of the 1D CNN. In each case, the accuracy drops to 78% and 72% respectively.

Finally, if we remove the geometric property-based model, the accuracy drops to 66%, showing the importance of including geometric properties in the classification algorithm.

We chose the 1D CNN model as it has the maximum accuracy (92%) when combined with the Multi-view CNN pipeline, among all the other models tested.

7 LIMITATIONS AND FUTURE WORK

This work shows that 3D representation can be used to recognize free-form gestures performed by blind users, without requiring extensive training data. However, one limitation of our work is that our evaluations are based on a user study with only 10 blind users. In the future, we will conduct a more comprehensive, longitudinal study, involving a larger and more diverse group of blind users. Further, our leave-one-user-out evaluation shows that we only need a small number of samples (three in our case) from a given user to recognize a gesture with high accuracy. We will expand this work to more (possibly user-defined) gestures to evaluate the effectiveness of our approach to recognizing more free-form gestures with a limited number of samples. Another avenue of research is to make the gesture recognition algorithm more generalizable so that even a small number of samples are not needed.

Finally, the ideas presented in this work can be used to recognize gestures not only for blind users but also for sighted users. While large companies may be able to collect training samples with a large number of users, this is not practical for others. Designing a gesture recognition system that requires limited training data can help with deployability and with adding new gestures. Our initial pilot study shows that indeed our approach is able to recognize gestures performed by sighted users with high accuracy. However, we need to perform extensive experiments and comparison studies to evaluate the efficacy of our gesture recognition algorithm for sighted users.

8 CONCLUSION

We design a hand gesture recognition algorithm that can recognize blind user gestures using a commodity smartwatch with high accuracy, without requiring extensive training data. In our formative user study with 10 blind users and 16 sighted users, we find that hand gestures made by blind users are significantly different from those of sighted users for a range of gesture properties including inter-user variance, jitter, jerk, and velocity. The implications of our findings is that existing gesture recognition algorithms, almost all of which are designed for sighted users, are not well-suited to recognize blind user gestures. The principle problem is that blind user gestures exhibit high inter-user variance, which makes learning gesture patterns extremely challenging. This problem is exacerbated by the limited training data available from blind users. Instead, our algorithm uses the insight that there are micro-movements within the gesture that are more consistent across users. We extract the micro-movements and design an ensemble classifier to classify the gestures from the micro-movements. The classifier works on a 3D representation of the gesture trajectory; 3D helps better capture gestures performed in free space. The classifier combines a multi-view CNN model with geometric properties of the gesture trajectory. The resulting algorithm achieves 92% classification accuracy and has an improvement in accuracy compared to the next best state-of-the-art which has an accuracy of 82%.

ACKNOWLEDGMENTS

We sincerely thank the reviewers for their insightful comments and suggestions. This work was supported in part by a Google Research Scholar award, Google Inclusion Research Award, the National Science Foundation under award numbers 2153056, 2125147, 2113485, 2110193, and the National Institutes of Health under award numbers R01EY030085, R01HD097188, R01EY035688.

REFERENCES

[1] Diar Abdlkarim, Massimiliano Di Luca, Poppy Aves, Sang-Hoon Yeo, R Chris Miall, Peter Holland, and Joseph M Galea. 2022. A methodological framework to assess the accuracy of virtual reality hand-tracking systems: A case study with the oculus quest 2. *BioRxiv* (2022), 2022–02.
[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] Sara Ashry, Reda Elbasiony, and Walid Gomaa. 2018. An LSTM-based descriptor for human activities recognition using IMU sensors. In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, Vol. 1. 494–501.
[4] 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.
[5] 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).
[6] K Abhijith Bhaskaran, Anoop G. Nair, K Deepak Ram, Krishnan Ananthanarayanan, and H.R. Nandi Vardhan. 2016. Smart gloves for hand gesture recognition: Sign language to speech conversion system. In *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. 1–6. https://doi.org/10.1109/RAHA.2016.7931887
[7] Ting Kwok Chan, Ying Kin Yu, Ho Chuen Kam, and Kin Hong Wong. 2018. Robust hand gesture input using computer vision, inertial measurement unit (IMU) and flex sensors. In *2018 IEEE International Conference on Mechatronics, Robotics and Automation (ICMRA)*. IEEE, 95–99.
[8] Feiyu Chen, Jia Deng, Zhibo Pang, Majid Baghaei Nejad, Huayong Yang, and Geng Yang. 2018. Finger angle-based hand gesture recognition for smart infrastructure using wearable wrist-worn camera. *Applied Sciences* 8, 3 (2018), 369.
[9] Feiyu Chen, Honghao Lv, Zhibo Pang, Junhui Zhang, Yonghong Hou, Ying Gu, Huayong Yang, and Geng Yang. 2018. WristCam: A wearable sensor for hand trajectory gesture recognition and intelligent human–robot interaction. *IEEE Sensors Journal* 19, 19 (2018), 8441–8451.
[10] Heng-Tze Cheng, An Mei Chen, Ashu Razdan, and Elliot Buller. 2011. Contactless gesture recognition system using proximity sensors. In *2011 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 149–150.
[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848
[12] James Diebel et al. 2006. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix* 58, 15-16 (2006), 1–35.
[13] Nem Khan Dim, Chaklam Silpasuwanchai, Sayan Sarcar, and Xiangshi Ren. 2016. Designing mid-air TV gestures for blind people using user-and choice-based elicitation approaches. In *Proceedings of the 2016 ACM conference on designing interactive systems*. 204–214.
[14] 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.

[15] 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.

[16] 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.

[17] Pengfei Gui, Liqiong Tang, and Subhas Mukhopadhyay. 2015. MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion. In *2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2004–2009.

[18] Steven Houben and Nicolai Marquardt. 2015. Watchconnect: A toolkit for prototyping smartwatch-centric cross-device applications. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 1247–1256.

[19] 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.

[20] Yangjian Huang, Weichao Guo, Jianwei Liu, Jiayuan He, Haisheng Xia, Xinjun Sheng, Haitao Wang, Xuetao Feng, and Peter B Shull. 2015. Preliminary testing of a hand gesture recognition wristband based on emg and inertial sensor fusion. In *Intelligent Robotics and Applications: 8th International Conference, ICIRA 2015, Portsmouth, UK, August 24-27, 2015, Proceedings, Part I 8*. Springer, 359–367.

[21] Dai Jiang, Yu Wu, and Andreas Demosthenous. 2020. Hand gesture recognition using three-dimensional electrical impedance tomography. *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, 9 (2020), 1554–1558.

[22] Weibin Jiang, Xuelin Ye, Ruiqi Chen, Feng Su, Mengru Lin, Yuhanxiao Ma, Yanxiang Zhu, and Shizhen Huang. 2021. Wearable on-device deep learning system for hand gesture recognition based on FPGA accelerator. *Mathematical Biosciences and Engineering* 18, 1 (2021), 132–153.

[23] Pyeong-Gook Jung, Gukchan Lim, Seonghyok Kim, and Kyoungchul Kong. 2015. A wearable gesture recognition device for detecting muscular activities based on air-pressure sensors. *IEEE Transactions on Industrial Informatics* 11, 2 (2015), 485–494.

[24] Shaun K Kane, Jacob O Wobbrock, and Richard E Ladner. 2011. Usable gestures for blind people: understanding preference and performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 413–422.

[25] 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.

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

[27] 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 *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[28] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *J. Amer. Statist. Assoc.* 107, 500 (2012), 1590–1598.

[29] Sven Kratz and Michael Rohs. 2010. The $3 recognizer: simple 3D gesture recognition on mobile devices. In *Proceedings of the 15th international conference on Intelligent user interfaces*. 419–420.

[30] Daniel Lauss, Florian Eibensteiner, and Phillip Petz. 2022. A Deep Learning based Hand Gesture Recognition on a Low-power Microcontroller using IMU Sensors. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 733–736.

[31] Hansheng Lei and Venu Govindaraju. 2004. Direct image matching by dynamic warping. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, 76–76.

[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] Tzuu-Hseng S. Li, Min-Chi Kao, and Ping-Huan Kuo. 2016. Recognition System for Home-Service-Related Sign Language Using Entropy-Based K -Means Algorithm and ABC-Based HMM. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, 1 (2016), 150–162. https://doi.org/10.1109/TSMC.2015.2435702

[34] 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.

[35] Siyu Liu, Jian Chen, Cheng Wang, and Lin Lin. 2023. Ultrasonic positioning and IMU data fusion for pen-based 3D hand gesture recognition. *Multimedia Tools and Applications* (2023), 1–19.

[36] Yuqi Luo, Jiang Liu, and Shigeru Shimamoto. 2021. Wearable air-writing recognition system employing dynamic time warping. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 1–6.

[37] Yuntao Ma, Yuxuan Liu, Ruiyang Jin, Xingyang Yuan, Raza Sekha, Samuel Wilson, and Ravi Vaidyanathan. 2017. Hand gesture recognition with convolutional neural networks for the multimodal UAV control. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, 198–203.

[38] Ziyang Ma and Enhua Wu. 2014. Real-time and robust hand tracking with a single depth camera. *The Visual Computer* 30 (2014), 1133–1144.

[39] Oleg Makaussov, Mikhail Krassavin, Maxim Zhabinets, and Siamac Fazli. 2020. A low-cost, IMU-based real-time on device gesture recognition glove. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3346–3351.

[40] Meethu Malu, Pramod Chundury, and Leah Findlater. 2018. Exploring accessible smartwatch interactions for people with upper body motor impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

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

[42] Chaithanya Kumar Mummadi, Frederic Philips Peter Leo, Keshav Deep Verma, Shivaji Kasireddy, Philipp M Scholl, Jochen Kempfle, and Kristof Van Laerhoven. 2018. Real-time and embedded detection of hand gestures with an IMU-based glove. In *Informatics*, Vol. 5. MDPI, 28.

[43] 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.

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

[45] Ionut Cristian Severin, Dan Marius Dobrea, and Monica Claudia Dobrea. 2020. Head gesture recognition using a 6dof inertial IMU. *International Journal of Computers Communications & Control* 15, 3 (2020).

[46] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 3633–3642.

[47] 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.

[48] Guangyi Shi, Yuexian Zou, Yufeng Jin, Xiaole Cui, and Wen J Li. 2009. Towards HMM based human motion recognition using MEMS inertial sensors. In *2008 IEEE International Conference on Robotics and Biomimetics*. IEEE, 1762–1766.

[49] 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.

[50] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*. 945–953.

[51] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[52] 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.

[53] Tijana Vuletic, Alex Duffy, Laura Hay, Chris McTeague, Gerard Campbell, and Madeleine Grealy. 2019. Systematic literature review of hand gestures used in human computer interaction interfaces. *International Journal of Human-Computer Studies* 129 (2019), 74–94.

[54] Danping Wang, Jina Wang, Yang Liu, and Xianming Meng. 2023. HMM-based IMU data processing for arm gesture classification and motion tracking. *International Journal of Modelling, Identification and Control* 42, 1 (2023), 54–63.

[55] Zhengjie Wang, Yushan Hou, Kangkang Jiang, Wenwen Dou, Chengming Zhang, Zehua Huang, and Yinjing Guo. 2019. Hand gesture recognition based on active ultrasonic sensing of smartphone: a survey. *IEEE Access* 7 (2019), 111897–111922.

[56] Eric W Weisstein. 2009. Euler angles. *https://mathworld. wolfram. com/* (2009).

[57] 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.

[58] Renqiang Xie and Juncheng Cao. 2016. Accelerometer-based hand gesture recognition by neural network and similarity matching. *IEEE Sensors Journal* 16, 11 (2016), 4537–4545.

[59] 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.

[60] Ruize Xu, Shengli Zhou, and Wen J Li. 2011. MEMS accelerometer based nonspecific-user hand gesture recognition. *IEEE sensors journal* 12, 5 (2011), 1166–1173.

[61] 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.

[62] Chan-Yun Yang, Pei-Yu Chen, Te-Jen Wen, and Gene Eu Jan. 2019. Imu consensus exception detection with dynamic time warping—a comparative approach. *Sensors* 19, 10 (2019), 2237.

[63] 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.

[64] Guan Yuan, Xiao Liu, Qiuyan Yan, Shaojie Qiao, Zhixiao Wang, and Li Yuan. 2020. Hand gesture recognition using deep feature fusion network based on wearable sensors. *IEEE Sensors Journal* 21, 1 (2020), 539–547.

[65] Yu Zhang, Tao Gu, Chu Luo, Vassilis Kostakos, and Aruna Seneviratne. 2018. FinDroidHR: Smartwatch gesture input with optical heartrate monitor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–42.

[66] Yang Zhang and Chris Harrison. 2015. Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 167–173.