SeleCon: Scalable IoT Device Selection and Control Using Hand Gestures

Amr Alanwar
University of California, Los Angeles
alanwar@ucla.edu

Moustafa Alzantot
University of California, Los Angeles
malzantot@ucla.edu

Bo-Jhang Ho
University of California, Los Angeles
bojhang@ucla.edu

Paul Martin
University of California, Los Angeles
pdmartin@ucla.edu

Mani Srivastava
University of California, Los Angeles
mbs@ucla.edu

ABSTRACT

Although different interaction modalities have been proposed in the field of human-computer interface (HCI), only a few of these techniques could reach the end users because of scalability and usability issues. Given the popularity and the growing number of IoT devices, selecting one out of many devices becomes a hurdle in a typical smarthome environment. Therefore, an easy-to-learn, scalable, and non-intrusive interaction modality has to be explored. In this paper, we propose a *pointing* approach to interact with devices, as pointing is arguably a natural way for device selection. We introduce SeleCon for device selection and control which uses an ultra-wideband (UWB) equipped smartwatch. To interact with a device in our system, people can point to the device to select it then draw a hand gesture in the air to specify a control action. To this end, SeleCon employs inertial sensors for pointing gesture detection and a UWB transceiver for identifying the selected device from ranging measurements. Furthermore, SeleCon supports an alphabet of gestures that can be used for controlling the selected devices. We performed our experiment in a 9m-by-10m lab space with eight deployed devices. The results demonstrate that SeleCon can achieve 84.5% accuracy for device selection and 97% accuracy for hand gesture recognition. We also show that SeleCon is power efficient to sustain daily use by turning off the UWB transceiver, when a user's wrist is stationary.

CCS CONCEPTS

•**Computer systems organization** →**Embedded systems;** *Redundancy;* Robotics; •**Networks** →Network reliability;

KEYWORDS

Pointing, IoT, hand gestures.

ACM Reference format:
Amr Alanwar, Moustafa Alzantot, Bo-Jhang Ho, Paul Martin, and Mani Srivastava. 2017. SeleCon: Scalable IoT Device Selection and Control Using Hand Gestures. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI 2017),* 12 pages.
DOI: http://dx.doi.org/10.1145/3054977.3054981

IoTDI 2017, Pittsburgh, PA USA

DOI: http://dx.doi.org/10.1145/3054977.3054981

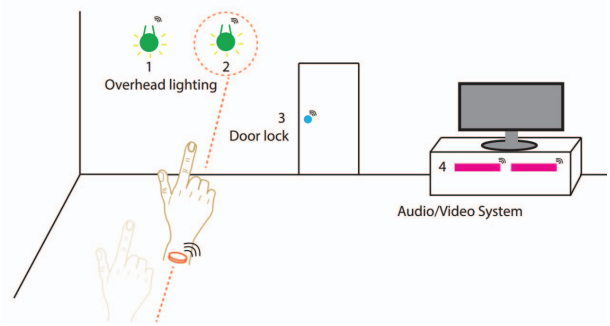

Figure 1: Gesture based IoT device selection using wearable devices.

1 INTRODUCTION

There has been a proliferation of smart devices in the last decade. These devices penetrate every aspect in our daily lives in many forms including mobile phones, smartwatches, thermostats, and door locks. In addition, smart home controllers, e.g. Amazon Echo [1] and Google Home [2], represent another recent wave of smart devices that are catching a lot of attention and gaining remarkable popularity. The development of smart devices has been fueled by research progress in several directions, such as improved network connectivity, reliability, and availability. The advancement in machine learning and data analysis also allow us to make semantics inferences from the sensory data streamed from these devices. However, making an easy and natural control interface for many surrounding devices at home remained unsolved problem. It is desirable that human interaction with machines in daily use should be intuitive and simple. Thus, much effort has been invested in Human-Computer Interface (HCI) domain to enable more natural forms of interactions between humans and devices.

Different forms of interaction have been proposed, such as speech recognition [10], face recognition [29], gaze/eye tracking [31], and hand gesture tracking [15]. However, despite this tremendous effort, we are still far from having a natural way to control and interact with devices. Vision-based methods (e.g., [34]) present a serious invasion of the user's privacy and they work only when sufficient

lighting is provided in the room assuming all objects are in the view without obstruction. Similarly, approaches based on speech recognition [1, 2] also invade privacy because they contentiously record audio and release it to remote cloud servers to interpret users' commands.

Hand gesture is a natural and effective communication method. Hand gesture recognition has received much attention especially in the HCI domain. Different sensing modalities have been proposed to recognize hand gestures, including cameras [34], depth sensors [3], Wi-Fi signals [6, 39], and body-worn inertial sensors [16, 25, 46]. The last approach, in particular, fits well into the smart home scenario because of the wide adoption of smartwatches and other wearable devices that are equipped with inertial sensors. However, only few existing gesture-based control systems have reached end users because no scalable and practical solutions that fit into everyday life, yet. For example, a typical smart home may have tens of devices connected to each other, including lights, thermostats, locks, and other appliances. Current smart devices typically require every single family member to install applications for controlling these devices. It might additionally burden the users to assign semantic labels for each device such as "living room light 2" or "northeast door." With the increasing number of devices in users' surroundings, this process becomes cumbersome.

Existing hand gesture recognition methods do not well address the device selection problem. Although a body of literature has proposed different gesture recognition solutions [7–9, 12–16, 19, 25, 26, 33, 39], none of these techniques can *select* a device and *control* it without increasing the appliance installation overhead. For example, if a user wants to turn on the light in the living room, how does a smart home system know which device is intended? In fact, without augmenting the previous gesture recognition techniques with a position estimation technique coupled with predetermined locations of all smart devices, none of the existing techniques can be used to directly control a specific device based on human gestures unless a special gesture is assigned to each individual device. This motivates us to develop a new technology to enable accurate and scalable IoT device selection and control.

Towards achieving a scalable and practical architecture for selecting and giving commands to smart home devices, we design SeleCon, a gesture-based system that aims to provide a natural device selection and control method for users to interact with smart IoT devices. A user can simply point his arm towards the target device to select it, as shown in Figure 1. SeleCon is able to identify which IoT device is selected by monitoring the direction of the wrist movement. The user then draw a gesture in the air to give a command to the selected device.

As smart devices can be placed in arbitrary locations and users can move over time, inertial sensors alone are not sufficient to identify the intended target. Therefore, we designed and implemented a smartwatch prototype equipped with an ultra-wideband (UWB) transceiver, and we use pair-wise ranging measurements between the smartwatch and the IoT devices to identify the target. The intuition behind our device selection process is that when a user points towards a given device, the smartwatch attached to the user's wrist will get closer to the chosen device after the transition of the pointing event. We use different machine learning algorithms to verify our hypothesis. We also develop machine learning models for recognizing hand gestures for giving commands to target devices. One major challenge is that UWB is known to be power hungry compared to inertial sensors. To address this problem, we use the low power inertial sensors to implement a motion-based triggering module so that UWB ranging is turned on only after potential pointing actions are detected. Therefore, SeleCon can effectively reduce the operating time of the UWB transceiver to save energy.

We summarize our four main contributions:

- We introduce SeleCon which provides a practical and scalable method of IoT device selection and control using pointing and hand gestures.
- We designed and implemented a hardware prototype of smartwatch equipped with a UWB radio and inertial sensors. This prototype is used to evaluate the performance of SeleCon.
- We develop machine-learning models for device selection and hand gesture recognition from UWB ranging and inertial sensors data.
- We develop a module for pointing event detection that relies only on inertial sensors. As a consequence, the UWB transceiver on the smartwatch can be turned off 92% of the time for energy saving without affecting the system accuracy.

SeleCon achieves 84.5% accuracy in detecting the target device, and 97% in recognizing the hand gesture commands. The rest of the paper is organized as follows: Section 2 provides an overview of SeleCon system architecture. We then go through SeleCon module by module. Pointing event detector is illustrated in Section 3. Section 4 introduces a simplified formulation for pointing gesture recognition problem, highlights the challenges in pointing gesture recognition, and introduces device selection algorithm using pattern matching. The language of gestures is introduced in Section 5. Section 6 evaluates SeleCon. Section 7 summarizes the related work. Current limitations and future work are shown in Section 8. Finally, Section 9 concludes this paper.

2 SYSTEM OVERVIEW

Figure 2 illustrates the different stages in the SeleCon processing pipeline. Broadly speaking, our system can be divided into three different stages in the following chronological order: the *listening stage*, the *pointing stage*, and the *command stage*. In *listening stage*, SeleCon detects all potential wrist motion events using the low-power inertial sensors. After a hand motion event is detected, SeleCon moves into the *pointing stage*. Our system uses the inertial sensor to verify whether the user has started pointing to a target device. Concurrently, SeleCon turns on the UWB transceiver to perform distance ranging between the smartwatch and the surrounding smart devices for a predefined time window. If the pointing action is verified by our system, SeleCon performs machine learning algorithms over UWB time series to determine which device is selected. The last step is the *command stage*. SeleCon again leverages the inertial sensors, exploits machine learning algorithms to classify the hand gestures. Figure 8 shows the gestures supported in our system.

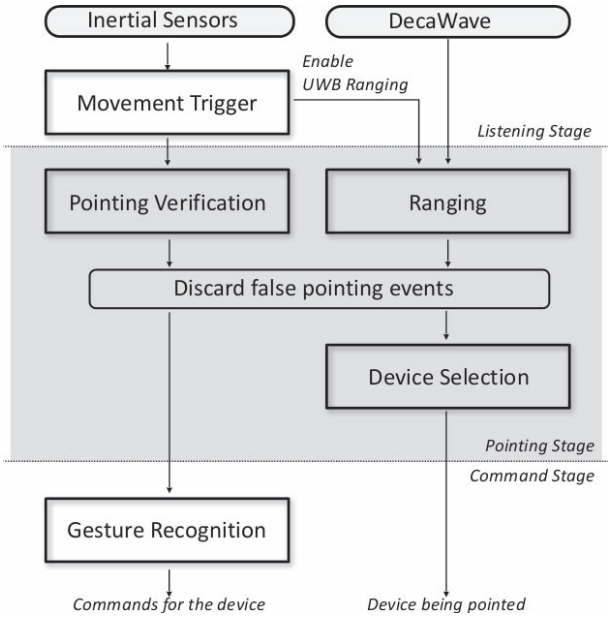

Figure 2: SeleCon system overview.

Figure 3 shows an example of inertial sensor data over a valid device interaction session. A session includes two parts, a pointing event followed by a gesture to give the command to a device. Note that the entire process is done within 2 seconds.

3 POINTING EVENT DETECTION

A major reason that we serve the pointing gesture as the preamble of one device interaction session is to save energy. Since UWB is $10x$ more power consuming than inertial sensors, it is infeasible to turn on the UWB for device selection for a long time. To cope with this issue, we use wrist motion as a trigger and enables the UWB only when a user tries to select a device. The *movement trigger* module in the *listening stage* aims at using low power inertial sensors to detect local wrist movements. As demonstrated in Figure 3, the motion of pointing is fast and the duration is no longer than 0.5 seconds. Thus, as long as a user *starts* moving her wrist, SeleCon enters the *pointing stage* and turns on the UWB transceiver. All the wrist movements are then passed to the *pointing verification* module, which checks whether the motion is a pointing gesture based on the inertial time series data. Below we provide the details of the *movement trigger* module and *pointing verification* module.

Movement Trigger Module: The *movement trigger* module aims to identify all the possible pointing events based on inertial sensors. However, if we make the module over sensitive, the UWB transceiver will wake up more frequently, causing the energy concern. In our design, we make the *movement trigger* module responsive enough so that even slow pointing gestures can be captured. Whenever a possible pointing event is captured, the *movement trigger* mobile turns on the UWB transceiver, records both inertial data and UWB ranging data for a couple seconds, and then passes the collected data to the *pointing verification* module which verifies whether the detected motion is a pointing gesture.

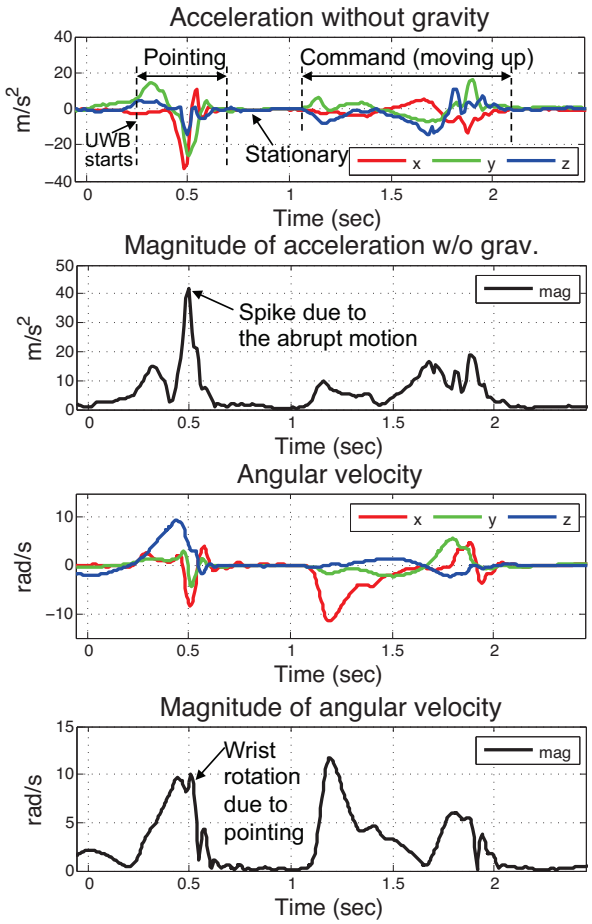

Figure 3: An example of inertial data: A user points to a device (TV) and conduct a moving up gesture (raise volume.)

Pointing Verification Module: Since the *movement trigger* module is set to capture small wrist movements, many false events are likely to be included. The goal of the *pointing verification* module is to keep those events that are intended to point to objects. The *pointing verification* module combines a number of heuristics to verify the pointing gestures from inertial measurements. Figure 3 shows both accelerometer and gyroscope data collected within a valid pointing event. Note that the gravity has been removed from the accelerometer data, processed in the hardware level. Since pointing gesture is an fast action, it will cause high acceleration showing as a spike in accelerometer data (e.g., at $t = 0.5$). On the other hand, though people have different habits to point towards objects, people naturally rotate their elbow joints and fully stretch their arms, making the smartwatch mounted on the wirst facing a different orientation. The orientation difference is reflected on gyroscope data which measures the angular velocity, demonstrated in Figure 3. We pick an acceleration and an angular velocity magnitude threshold such that true pointing events can be distinguished from incorrect ones.

Though the magnitude of inertial data is a good indicator of pointing gestures, it is still not robust enough to remove all the false events. We have the following observation. Users usually holds their arms for a short period (i.e., a couple hundred milliseconds) after pointing to an object. Therefore, if inertial sensors are stable for a predefined length, we consider this as a pointing event.

4 IOT DEVICE SELECTION

After detecting a pointing event, the system determines which device is selected by processing the UWB ranging measurements in the *device selection* module. Assume that we have a network of N IoT devices and a single user in a room. We denote the position of each device n_i for $i \in 1, ..., N$ by $p_i \in \mathbb{R}^3$. Similarly, the position of the smartwatch n_u (i.e., user's wrist) is denoted by $p_u(t)$ at time t. SeleCon gets the ranging measurements between the smartwatch n_u and any smart device n_i from the UWB transceiver, and we denote the measurement by $r_i(t)$ at time t.

Consider the user is pointing to a target IoT device whose index is denoted by i^*. Since pointing is a process to move the wrist closer to the target device, mathematically speaking, the distance between n_{i^*} and n_u should decrease the most comparing with any other n_i where $i \neq i^*$. By collecting the pairwise ranging measurements between the smartwatch and every smart device, we can find out i^* by solving the following equation:

$$\begin{aligned} i^* &= \underset{i}{\text{argmin}} \quad \big(\|p_i - p_u(t_f)\| - \|p_i - p_u(t_s)\|\big) \\ &= \underset{i}{\text{argmin}} \quad \big(r_i(t_f) - r_i(t_s)\big) \end{aligned} \tag{1}$$

where t_s and t_f are the start and finish time of the pointing gesture, respectively. In order to measure the distance between the user and any IoT device, we considered both *single-sided* and *double-sided* two-way ranging techniques, which are described in [23]. In the *single-sided* two-way ranging, two devices i and j take turns to send message to the other. The distance is derived by the round trip time of the message transfer. *Double-sided* two-way ranging can be seen as an extension of the *single-sided* technique, and the differenct is that the first device i sends the third message to j so that the round trip time can be also acquired from the second device. This approach usually gets a better ranging accuracy because the third message compensates for the clock drift.

However, both aforementioned ranging techniques have a range estimation error of around $10cm$ to $30cm$, which can impact the device selection accuracy. The possible reasons of the large ranging errors are the range noise, the pointing length, and the spatial diversity. In the following subsections, we will present what are the challenges to distinguish the target device from the incorrect ones, and then provide the features that have potential to identify the correct device.

4.1 Spatial Resolution and Gesture Length

When a user points to an IoT device, the length of that gesture inherently defines a signal-to-noise ratio (SNR). We denote this length ℓ, and we assume that the range error follows a non-zero mean Gaussian distribution $e \sim N(\mu_r, \sigma_r^2)$. This error is not necessarily be i.i.d, but for the sake of simplicity, we assume that the range error is independent across time. Ideally, the start wrist position $p_u(t_s)$,

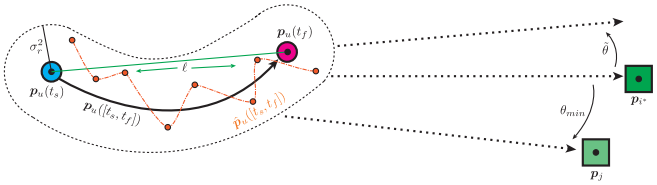

Figure 4: Ranging errors and angular (spatial) resolution in gesture-based IoT device selection.

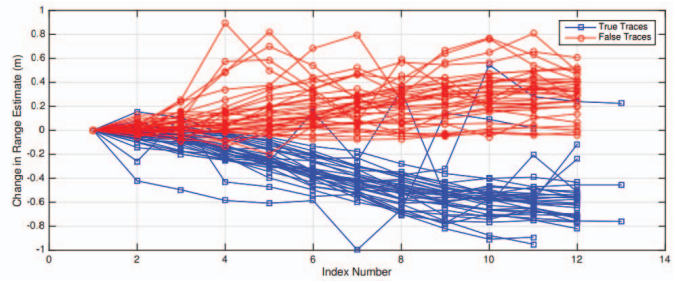

Figure 5: Example ranging traces during pointing. These examples are in an environment with high spatial diversity where $\triangle R$ is sufficient to identify the selected device

the end wrist position $p_u(t_f)$, and the selected device p_{i^*} should fall on a straight line. However, when a user points to a device, the eyes, the wrist, and the device are not co-linear, causing the pointing angular error $\tilde{\theta}$. Additionally, if there is a device close to the true device, our system may be mistaken and select the wrong one.

We define the angle formed by the true device n_{i^*}, the user n_u, and the cloest device n_j in terms of angle as θ_{min}, illustrated in Figure 4. In the case of high spatial diversity, i.e. when θ_{min} is quite large and when ℓ is large with respect to σ_r^2, it is relatively easy to distinguish the correct device n_{i^*} that a user is pointing to from any other devices n_j for $j \neq i^*$. Figure 5 shows the ranging difference measured from both target device n_{i^*} and other devices n_j over time in a high spatial diversity area. The range difference at k^{th} sample is defined as the delta of current range value and the start range value, or $\triangle R_i(t) = r_i(t) - r_i(t_s)$. Each trace in Figure 5 represents a pointing gesure measured by a device. The true traces which are plotted in blue correspond to the distance estimates between the user and n_{i^*}, and the false traces plotted in red are those corresponding to any other device. As we can see in the figure, the true traces show the range differences are negative, as the user moves her wrist closer to the target device. In contrast, most red traces measured from other devices are positive. These two kinds of traces can be easily separated.

In the case of low spatial diversity, the simple metric $\triangle R$ no longer suffices as a reliable metric for discerning the desired IoT device from other IoT devices. For instance, Figure 6 shows estimated range differences for a low spatial diversity environment, which we deployed the IoT devices as illustrated in Figure 10. The result shows that true range measurement traces have similar pattern in high spatial diversity case (i.e., blue traces in Figure 5), but the false traces

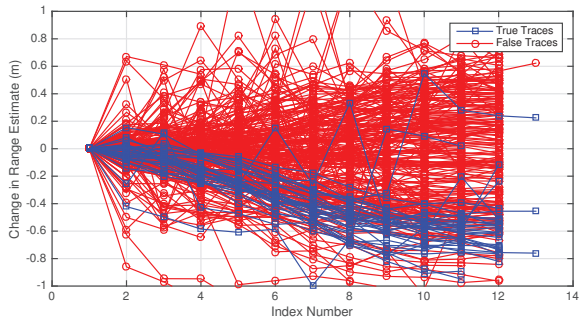

Figure 6: Example ranging traces during pointing. These examples are in an environment with low spatial diversity where $\triangle R$ is not sufficient to identify the selected device

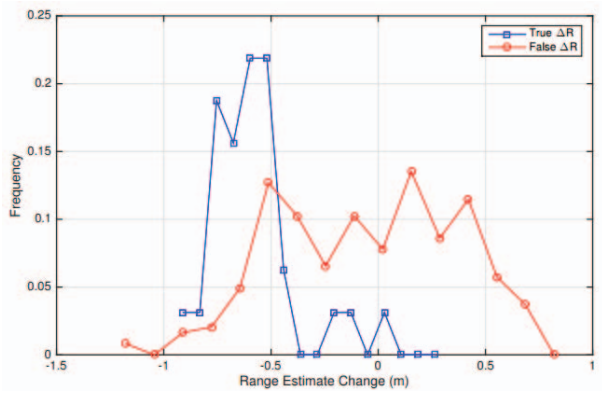

Figure 7: Range difference, $\triangle R$, for true and false (n_{i^*} and $n_{i \neq i^*}$) as a probability density function.

from some other devices overlap with the true traces. Figure 7 plots the distribution of ranging differences $\triangle R$. As we can see, there is an overlap between true and false range differences within $-0.5m$ to $-0.8m$. Thus, in order to provide high device selection accuracy, we need to explore other metrics. In the following subsection, we will explore a more principled approach to the device selection problem, keeping device power and computational overheads in mind.

4.2 Device Selection by Pattern Matching

We have demonstrated that the estimated distance change, $\triangle R$, is insufficient to reliably identify the target device to which the user is pointing. Additional features have to be explored to find correct devices. We first present features considered in our system, and explain how we determine the selected devices by the classification algorithm.

4.2.1 Relevant Ranging Features. SeleCon considers the following features:

Linear Fit: Although the wrist path of a point gesture is curved as illustrated in Figure 4, in reality the path is close to a straight line. Any divergence from this fit could indicate that the range estimated trace could belong to an undesired device. More specifically, if we estimate a linear fit $\hat{r}_i(k) = r_i(k) + v_k$ describing the range estimate between n_u and n_i for a given device i and noise v at sample index k, we describe the fitting error as the mean squared residual $MSR = \frac{1}{K}\sum_k v_k^2$, where K is the number of ranging samples when pointing. Intuitively, a low MSR indicates a good linear fit and consequently more likely being the true trace. We use this feature to enhance the pointing recognition process.

First Path Loss: Path loss is defined as power density reduction from a transmitter to a receiver. The accurate timing provided by UWB radios is enabled by measuring the energy in the radios accumulator corresponding to the communication along the first path. One consequence of this accurate timing is that the same energy can be used to estimate the power of the communication along the first path of communication typically the line-of-sight path. Since people rarely point to an object which is out of their sight, the chosen device is likely to report low first path loss, denoted by $fploss$. A general path loss formula is shown in equation 2.

$$PL(d) = PL_0 + 10 \times \gamma \times log(d/d_0) + S \tag{2}$$

PL_0 is the path loss at the point that is located at the reference distance d_0. γ is the slope of the average increase in path loss with dB distance. S is the variation which is zero-mean Gaussian random variable [18].

Range Data: If there are two devices which align on the direction that a user points towards, naturally we consider that the user is interacting with the closer one. Thus, the distance between a device and a user also plays an important role. To estimate this distance, we use the raw range result as a feature.

Angular Divergence: As a user points from $p_u(t_s)$ to $p_u(t_f)$, she is attempting to point as closely towards a device as possible. In an ideal case, the line between start and finish is perfectly co-linear with the coordinates of the IoT device itself. In practice, however, there is some angular divergence $\tilde{\theta}$ as shown in Figure 4. In order to calculate this angle, we must know the device position p_i as well as the start and stop position of the user's hand. This requires a collaborative position estimate among multiple IoT devices, which assumes a certain device density and additionally consumes more power due to added communication costs. When possible, however, $\tilde{\theta}$ can be used to increase the accuracy of gesture-based IoT device selection.

We combine the previous features into an aggregate feature vector, defined as

$$f = [\triangle R, MSR, fploss, \hat{r}(t_s), \tilde{\theta}]^T \tag{3}$$

For each pointing event, there is N different data points corresponding to the feature vectors computed from the ranging measurements between the user's smartwatch and each of the N surrounding devices. Only one among these data points corresponds to the true target device (n_{i^*}) while the remaining $N-1$ are not selected. Thus, we label a feature f_i as 1 for the true device (n_{i^*}) and 0 otherwise. Although we have included the expensive feature $\tilde{\theta}$ in our feature vector f definition, we report the system accuracy based on different experiments that uses a feature vector with and

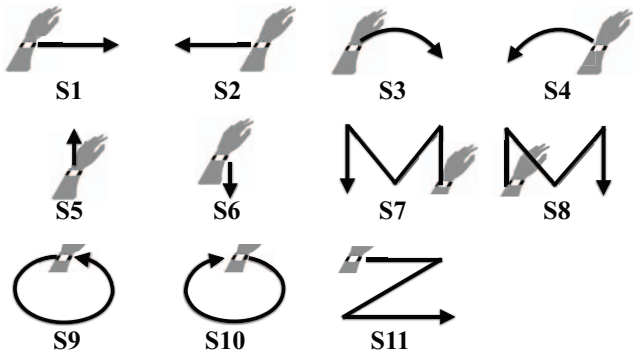

Figure 8: List gestures supported by SeleCon.

Figure 9: Hardware prototype of UWB-equipped smartwatch

without this feature being included. We believe that these experiments mimic diverse settings and provide more realistic results for sparse deployments and restricted energy reserves.

4.2.2 Classification Methods. From each feature vector we can estimate whether a particular IoT device is selected or not, but this approach ignores any potential communication or collaboration between connected IoT devices. Another option in the classical classification methodology that a single centralized server classifies based on the feature vectors from IoT devices. However, this collaborative classification is not scalable and have a high communication cost. On the other hand, in non-collaborative classification, each IoT device operates independently, attempting to classify itself as selected or unselected. Other than the communication with n_u required for range estimates, no additional communication is performed. This saves power, but it comes at the cost of very high selection errors. We do not want multiple devices to be selected at the same time. Therefore, we choose to do collaborative classification; devices are allowed to communicate. Rather than sending entire feature vectors, however, each device will send only a summary of their classification results and indicate the certainty with which the classification was made. This allows the network to arrive at a consensus of maximum certainty of which device was selected, enabling collaboration without prohibitively high communication overheads.

5 HAND GESTURE RECOGNITION

The second half of a pointing session is a hand gesture to control the target device. Individual devices can be configured to execute different actions in response to the different gestures. Hand gesture is arguably one of the intuitive ways for describing actions and does not require moving closer to the target device. Currently, SeleCon supports 11 different gestures which can be assigned to up to 11 different operations for each individual device. Our system can be easily extended to support additional gestures, but we believe this number is satisfactory for the needs of most devices. The list of supported gestures is shown in Figure 8.

Our gesture recognition module relies on the measurements from the inertial sensors in our smartwatch. Namely, we use three axes of both accelerometer and gyroscope measurements. According to our language definition, a user should point to the device first then draw a gesture command in the air. Therefore, after the end of the pointing action, we collect 3 seconds of inertial measurements. From these samples, we compute a feature vector consisting of the following features along every axis of both accelerometer and gyroscope:

- The three quartiles (25%, 50%, 75%).
- Standard deviation σ
- Skewness:

$$skewness = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

- Kurtosis:

$$Kurtosis = \frac{E\left[(X-\mu)^4\right]}{\left(E\left[(X-\mu)^2\right]\right)^2}$$

As a result, in total we have 36 features. We use these features to train a machine learning classifier to recognize different gestures.

6 EVALUATION

We deployed a custom ultra-wideband RF testbed based on the DecaWave DW1000 IR-UWB radio [23] for evaluation of SeleCon. The whole experimental setup overview is shown in Figure 10. Our testbed consists of three main components:

Smart devices anchor nodes: Fixed anchor nodes powered by an ARM Cortex M4 processor with Power over Ethernet (PoE) and an expansion slot for a custom daughter board containing the DW1000, as shown in Figure 11. We deployed 8 UWB anchor nodes in various positions in a $10m$-by-$9m$ laboratory. Six nodes were mounted on the ceiling ($2.5m$ high) and two were placed on 1-meter high shelves. The heights of nodes are different to simulate IoT devices deployed in real homes. All the anchor nodes are connected to an Ethernet backbone both for power and for communication to the central server, and is fully controllable over TCP/IP from the central server.

UWB-equipped smartwatch: Figure 9 shows our prototype which is a UWB-equipped smartwatch powered by a cell button battery. The smartwatch sends the ranging communication messages via Bluetooth low energy (BLE) to a smartphone as a relay, which sends the data back to the central server over TCP/IP.

Collaborative Classifier	Single-sided ranging Accuracy (%)	Double-sided ranging Accuracy (%)
Voting on SVM (linear)	41.46	43.29
Voting on SVM (quadratic)	48.78	50.00
Voting on SVM (rbf)	50.60	55.48
Voting on RF	78.04	80.48

Table 1: Classification results for gesture-based IoT device selection, using collaborative technique. Angle is not part of the feature vector.

Collaborative Classifier	Single-sided ranging Accuracy (%)	Double-sided ranging Accuracy (%)
Voting on SVM (linear)	45.12	46.34
Voting on SVM (quadratic)	50.00	56.70
Voting on SVM (rbf)	64.63	66.46
Voting on RF	81.09	84.14

Table 2: Classification results for gesture-based IoT device selection, using collaborative techniques. Angle is part of the feature vector.

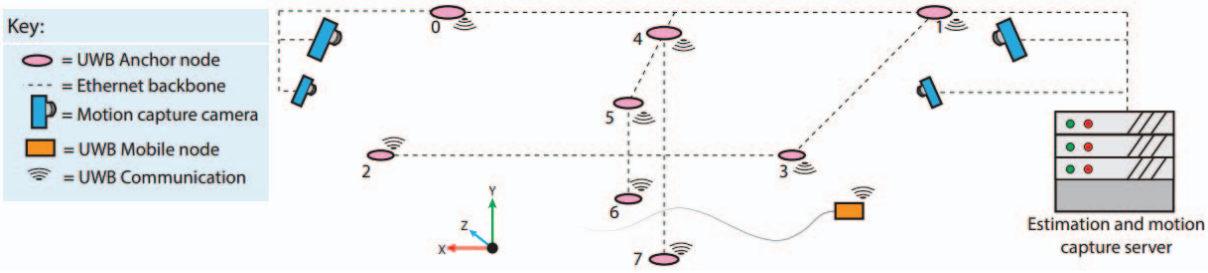

Figure 10: Experimental setup overview, including, UWB Anchor nodes, motion capture cameras, and a user wearing UWB-equipped smartwatch

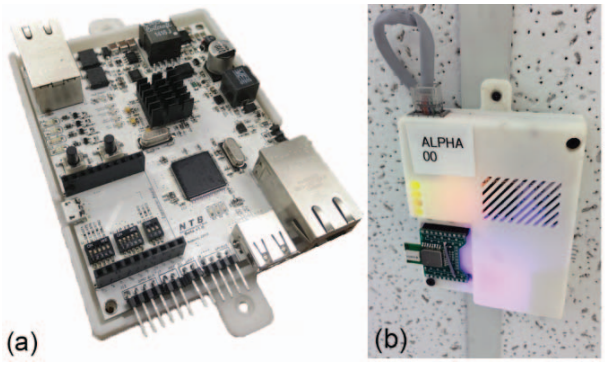

Figure 11: (a) Custom ranging anchor circuit board, and (b) ceiling mounted anchor node

Motion Capture System: In order to accurately capture the path of pointing gestures, we use the OptiTrack motion capture system [5] to measure the smartwatch position over time. Our motion capture system consists of 8 cameras placed along the perimeter of the experimental area. We attach the IR reflectors on the smartwatch so that the OptiTrack system can track the object. The IR reflectors have to be captured by at least 4 cameras to determine the location. The localization error of OptiTrack system is within $\pm 0.5mm$, which is sufficiently accurate as the ground truth.

6.1 Pointing Event Detection

To evaluate the accuracy of pointing event detection, we conduct the following two experiments. The first experiment evaluates how well our system can capture pointing events. We ask three participants to wear our smartwatch and perform at least 50 pointing events. The participants are instructed to use their natural way to point to any device (e.g., an anchor node) without restrictions. During the data collection sessions, participants could move freely within the testbed area. We used the OptiTrack motion capture system [5] to collect smartwatch location traces and post-processed the pointing events. The detection rate is 91.9%.

With such a promising detection rate, one might naturally ask the false alarm rate. This has to be broken down to two further questions: (1) How frequent the UWB radio has to be active? (2) What is the false alarm rate of our system reports pointing events? To answer these questions, we conduct a second experiment to collect non-pointing inertial data from five students. Participants

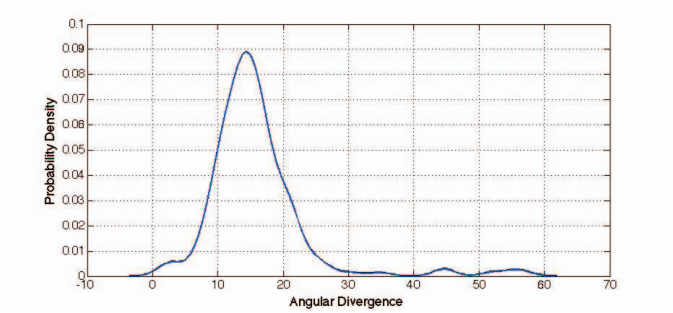

Figure 12: Probability density function of the angular divergence of the pointing events.

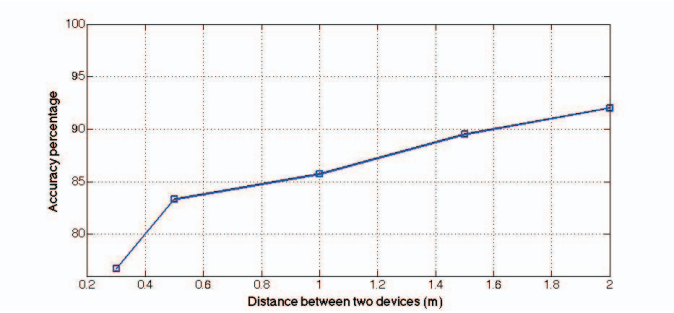

Figure 13: The effect of distance between two devices on SeleCon accuracy

are allowed to do any acticity they want as long as no pointing gestures are involved. We collect 11.6 hours of data in total. Our results show that SeleCon only has to enable the UWB ranging for only 8.0% of the time. SeleCon reports 73 false events in total, with an average of 6.29 false events per hour.

To further reduce the false event rate, SeleCon considers the result from the gesture recognition module. Since a valid command should include a gesture, if the gesture recognition module returns none of these 11 predefined gestures, we discard the pointing event. This further step decreases the false alarm rate to 2.5 false events per hour.

6.2 Device Selection

In order to evaluate the pointing-based IoT selection system, we perform a series of pointing events using our prototype of the UWB-enabled smartwatch. For groundtruth collection of the users' wrist motion, we attach a set of infrared reflectors to the smartwatch prototype. We use the OptiTrack [5] motion capture system to track the realtime positions. We collected totally 200 pointing events to various devices performed by various users. We evaluated the accuracy of different classification techniques for device selection. The results of collaborative classification schemes using support vector machine (SVM) and random forest (RF) are shown in Table 1 where the angle $\tilde{\theta}$ is not part of the feature vector. Table 2 shows the result of different classification schemes under collaborative schemes in which the angle $\tilde{\theta}$ is part of the feature vector. The used angular divergence in pointing has the shown probability density function in Figure 12. The ground truth angular divergence is calculated by processing the motion capture logs while pointing.

Collaborative classification achieves a good accuracy. This is due to the voting scheme among all the N IoT devices, which follows non-collaborative classification. We now communicate classification certainties between all devices, arriving at a maximally certain positive label. In particular, in the case of SVM we communicate the margin between a datapoint and its n-dimensional polytope. In the case of RF, we communicate the numerical average of the ensemble prediction. We report the results with and without the expensive feature $\tilde{\theta}$ in order to provide more realistic results for sparse deployments and restricted energy reserves. We will show also the results for both single-sided and double-sided ranging.

In doing so, we achieve for single-sided ranging without the feature $\tilde{\theta}$ an accuracy of 50.6% for collaborative RBF SVM, and 78.04% for collaborative RF. Collaborative linear and quadratic SVM show 41.46% and 48.78%, respectively. On the other hand, using double-sided ranging messages achieves 80.48 for collaborative RF. Other classifiers results are shown in Table 1. Introducing the feature $\tilde{\theta}$ enhanced the accuracy. Collaborative RF achieves the best accuracy of 84.14% and 81.09% for double-sided ranging and single-sided ranging, respectively. Collaborative RBF stands in rank two position with accuracy 64.63% and 66.46% for single-sided ranging and double-sided ranging, respectively. We report only accuracy, because at any given pointing event we know that only a single IoT device out of N possible is selected. The accuracy is the percentage of times we choose the correct device. Recall that our testbed size is $9m$-by-$10m$, which on average every deployed device is $3.5m$ apart. We should mention that the reported accuracy is the average accuracy when standing in different positions in the room. This accuracy depends on the co-linearity between the devices with respect to the user's position. We will analyze the effect of co-linearity between devices and the distance between the them on the accuracy at the next subsections.

6.2.1 Distance Between two Devices Analysis. We study the effect of the distance between two devices on the accuracy of selecting one of them. We conducted an experiment where the user points to one of two devices. We change the distance between the two devices from $30cm$ to $2m$ and compute the accuracy of device selection at each distance. Figure 13 shows how the selection accuracy changes with the change of the distance between devices. We should emphasis that the reported accuracy in Tables 1 and 2 is the average accuracy across different positions. On the other hand, the reported accuracy in Figure 13 is the result of standing in one position in the middle of the testing environment. Therefore, the reported accuracy in Figure 13 is better than the previous results.

6.2.2 Co-linearity Analysis. We also study the effect of co-linearity between different devices and the user's position on the reported accuracy. The co-linearity is defined in terms of the angle α in the x-z plane as shown in Figure 14. Two devices are co-linear if $\alpha = 180$. In order to analyze the co-linearity effect on the reported accuracy, we conducted two sets of experiments. Two devices are

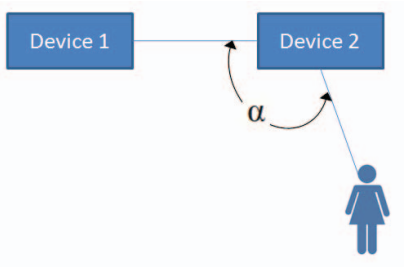

Figure 14: Co-linearity effect.

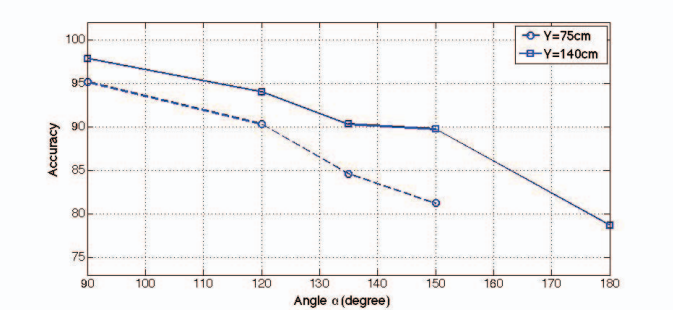

Figure 15: Co-linearity effect while pointing from height 75 cm and 140 cm at devices at 75cm in the y direction.

places at height of 75cm, i.e, their position is 75cm in the y direction. In the first set of experiment a user is asked to keep pointing to the two devices while standing, i.e, from height 140cm in the y direction. On the other hand, the second set of experiments is conducted while sitting, i.e, from height 75cm in the y direction. Figure 15 shows the co-linearity effect while pointing from height 75cm and 140cm on devices at 75cm in the y direction. Again, we should emphasis that the conducted experiments were done while standing roughly in the middle of the $9 * 10m^2$ room. Therefore, the reported accuracy is much higher than the average accuracy in Tables 1 and 2.

6.2.3 *Power Analysis*

. This device selection technique targets IoT devices that are potentially battery powered. Because of this, care has to be taken to ensure that power consumption is minimized. Here we briefly analyze power consumption for just the mobile (e.g. smartwatch) device that the user wears. Note that in practice, some of the stationary (anchor) IoT devices may be battery powered as well, and in these cases a low power *sniffing* strategy should be employed for listening to messages from the mobile device. For the mobile case, we will ignore the energy cost of computation for classification, as this is dwarfed by the energy required to transmit and receive using the DW1000 UWB transceivers.

In the idle case (when the smart watch is not ranging and is idle, listening for a pointing gesture), $6\mu W$ of power are consumed. When transmitting or receiving UWB frames, there is a $5ms$ wakeup period during which $3mW$ of power are consumed, followed by a $260mW$ for $200\mu s$ for TX and $370mW$ for RX. We will ignore the sleep power consumption ($6\ \mu W$) for now, as it will be shadowed by the processing consumption and analog conversions for gesture detection on the smart watch. These power numbers are derived from [23]. For N IoT devices, energy consumption during each "pointing" session is calculated as follows:

$$E = K \cdot N^*(E_{wake} + 2 \cdot (E_{TX} + T_{RX}P_{RX})) \quad (4)$$
$$= K \cdot N^*(15\mu J + 2 \cdot (52\mu J + 370\mu J))$$

with a listen period of $T_{RX} = 1ms$ following each transmit (and expected response). Here N^* represents all nodes within communication range, rather than the full number of networked devices, and K is the number of range measurements calculated per node—roughly 20, in the above experiments. For $N = 8$, as is the case in the results shown here, we have $E = 137.4mJ$ per point. For today's smart wearables, a battery with between 750 and 1400 mWh is common—this gives between $2.7kJ$ and $5kJ$ in each battery, meaning that if, for example, 100 pointing gestures are made each day, there is a 0.27% to 0.5% overhead in battery life. This is a very reasonable price to pay for the added convenience of high fidelity gesture-based IoT device selection. With improvements in commodity UWB hardware, however, it is likely that this overhead will decrease further.

6.3 Gesture Recognition

To evaluate our hand gesture recognition module, we collected 1290 gestures samples from volunteers who were given the freedom to wear the watch on their left hand or right hand. On average, we collected 117 samples for each gesture from the supported gestures shown in Table 8. We used the collected measurements to evaluate the performance of different classification algorithms at the *gesture recognition* module. Table 3 shows the accuracy of the Gesture Recognizer module. Linear SVM achieves about 97%. Quadratic SVM achieves an accuracy of 96.5%. Then, Random Forest comes the third with 95.48% accuracy. Finally, SVM with RBF kernel achieves 93.12%. Figure 16 shows the confusion matrix of the gesture recognition. In summary, SeleCon has a robust gesture recognizer module with 11 gestures which should be sufficient to control any IoT device.

Classifier	Accuracy
SVM (Linear)	97.03%
SVM (quadratic)	96.50%
SVM (RBF)	93.12%
RF	95.48%

Table 3: The Accuracy of Different Classifier for Gesture Recognition

7 RELATED WORK

To examine various interaction modalities that are used to communicate with IoT devices, including device selection and device control, we broadly partition prior work into three groups and compare the most related work in Table 4:

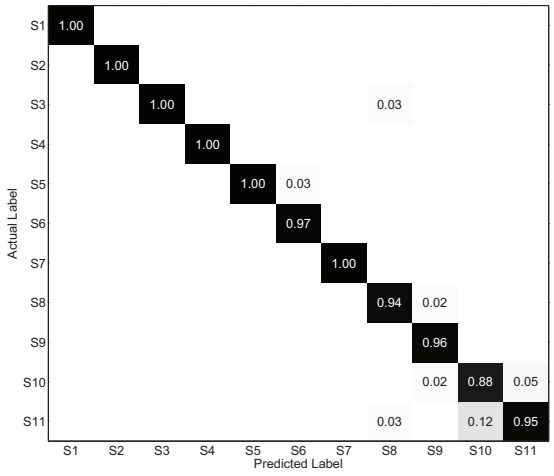

Figure 16: Confusion Matrix of the Gesture Recognition Classifier

(i) **Wireless-based interaction.** Prior work attempts to use RFID [12, 14] and audio [19] for gesture recognition. However, they are not feasible because an RFID reader typically has a coverage limit of $10m^2$. Also, ambient sound may add noises and decrease the accuracy. Perhaps WiFi is a better medium to "observe" interactions between humans and devices as several works have demonstrated WiFi can mimic human sensations as in WiSee [39], WiHear [44], and WiFinger [30]. One popular technique is leveraging the Doppler effect to detect moving objects [7] which allows the system to "see through the wall" [6]; Kim et al. also exploit this phenomenon to monitor human activities [27]. Extracting information Channel State Information (CSI) from the Network Interface Card (NIC) is another technique to sense the environment, such as occupancy detection [47], typing [9], falling detection [20], and activities involving body displacements in general [35, 45]. Though gesture recognition via WiFi has been proved possible, none of this previous work is capable of identifying which device is pointed at by a user. This is because WiFi cannot give good range resolution. In contrast, UWB can provide high resolution of ranging measurement, which makes it a promising technology for indoor localization [17]. In SeleCon, we choose UWB for device selection because pointing is instant and the distance between wrist start and stop positions is short.

(ii) **Inertial-based interaction.** Inertial sensors are good at capturing local movements. Previous research has demonstrated using inertial sensors for full body posture [16]. As wristbands become more and more popular, a body of literature explores the sensing boundary in this form factor. Xu et al. [48] point out that it is possible to track arm-level [25], hand-level [15, 26], and finger-level gestures [41, 46]. Shen et al. also show that since wrist position is determined by both the shoulder and the elbow motions [42], the full arm posture can be sensed even by a wristband. Based on this work, several interesting sensing applications such as driving [13, 37], whiteboard writing [11], gaming control [4, 49], and writing or drawing in the air [8, 38, 48] have been developed. Researchers found inertial sensors are capable of capturing subtle hand movements, and sensitive information can be leaked when a user types [21, 32, 36]. Interestingly, WristQue [33] combines environmental and inertial sensing with precise indoor localization, using UWB for pointing and gestures recognition. However, WristQue needs magnetic field pre-calibration and full localization information, which are critical limitations in that work. SeleCon aligns with all these works and employs inertial sensors for two use cases: Accurate pointing action detection to save energy and hand gesture recognition for controlling devices.

(iii) **Vision-based interaction.** Commercial products such as Microsoft Kinect [3] employ both an RGB and a depth camera to track the human skeleton. Sharing the same idea, Digiteyes [40] reports 27 degrees of freedom (DOF) hand model at a speed of 10 Hz. This technique has also been applied on detecting human gestures [34] and sign language [43]. HeatWave [28] and HeatProbe [22] use thermal cameras to detect and track how users interact with appliances. Jing et al. [24] recognize pointing events using Kinect. Perhaps intuitive to humans, vision-based approaches, however, suffer from requiring a line of sight and good lighting conditions, and also impose severe privacy invasion.

8 LIMITATIONS AND FUTURE WORK

While we are very positive about SeleCon current capabilities, we admit the following limitations in our architecture:

- SeleCon requires the user to wear a custom smartwatch equipped with both an inertial measurement unit (IMU) and an ultra-wideband (UWB) radio.
- Currently, SeleCon cannot be used by more than one user simultaneously.
- We assume that smart devices around the user are also equipped with UWB radio. Although one might argue that these assumptions are too strict, we anticipate that UWB radios will permeate the IoT scene in the next few years, given their success and growing adoption rate.

We believe that the next step is to add SeleCon in the real-life deployment of many smart homes, collect users reviews and enhance the system architecture. Also, enhancing the pointing recognition accuracy by considering more features is another feasible future work. Supporting multiple users at the same time is a key challenge in current SeleCon implementation.

9 CONCLUSION

In this paper, we have described and evaluated SeleCon which is a novel system for IoT device selection and control. SeleCon provides a simple and intuitive interface to interact with a myriad of smart devices through pointing actions and hand gestures. All smart devices have to be UWB enabled, and users only need to wear a custom smartwatch equipped with inertial sensors and UWB transceiver. We have designed and implemented hardware prototypes of both the custom smartwatch and the smart devices anchor nodes. SeleCon employs different machine learning classifiers to accurately identify the selected target device from the UWB ranging measurements. In addition, SeleCon supports a language of 11 different gestures to provide control of the selected device. We also

Research	Pointing Detection	Gesture recognition	Requirements
SeleCon	✓	✓	Smart watch.
WristQue [33]	✓	✓	Pre-calibration of magnetic fields, full localization using UWB.
WiTrack [6]	✓	✗	Does not require the user to carry any device.
Inertial Gestures [38]	✗	✓	Smart watch.
Kinect Pointing [24]	✓	✗	Kinect within 3.5m.

Table 4: Summary of related work.

presented an energy saving approach which uses the low-power inertial sensors to trigger UWB such that UWB can be in sleep mode in 92% of the time. Our experimental results demonstrate that SeleCon achieves 84% accuracy for device selection even with a high device deployment density, and our system achieves 97% accuracy for hand gestures recognition.

ACKNOWLEDGMENTS

This research is funded in part by the National Science Foundation under awards # CNS-1329755 and ACI-1640813, and by the NIH Center of Excellence for Mobile Sensor Data to Knowledge under award # 1U54EB020404-01. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, NIH, or the U.S. Government.

REFERENCES

[1] Amazon Echo. http://amazon.com/echo.
[2] Google Home. https://madeby.google.com/home/.
[3] Microsoft Kinect. https://developer.microsoft.com/en-us/windows/kinect.
[4] Nintendo Wii. http://www.nintendo.com/wii.
[5] OptiTrack Motion Capture System. http://www.optitrack.com.
[6] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D Tracking via Body Radio Reflections. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*. USENIX Association, Berkeley, CA, USA, 317–329. http://dl.acm.org/citation.cfm?id=2616448.2616478
[7] Fadel Adib and Dina Katabi. 2013. See Through Walls with WiFi!. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 75–86. DOI: http://dx.doi.org/10.1145/2486001.2486039
[8] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. 2011. Using Mobile Phones to Write in Air. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*. ACM, New York, NY, USA, 15–28. DOI: http://dx.doi.org/10.1145/1999995.1999998
[9] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke Recognition Using WiFi Signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, New York, NY, USA, 90–102. DOI: http://dx.doi.org/10.1145/2789168.2790109
[10] Fernando Alonso-Martín, Álvaro Castro-González, Francisco Javier Fernandez de Gorostiza Luengo, and Miguel Ángel Salichs. 2015. Augmented Robotics Dialog System for Enhancing Human–Robot Interaction. *Sensors* 15, 7 (2015), 15799–15829.
[11] L. Ardser, P. Bissig, P. Brandes, and R. Wattenhofer. 2016. Recognizing text using motion data from a smartwatch. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 1–6. DOI: http://dx.doi.org/10.1109/PERCOMW.2016.7457172
[12] Parvin Asadzadeh, Lars Kulik, and Egemen Tanin. 2012. Gesture recognition using RFID technology. *Personal and Ubiquitous Computing* 16, 3 (2012), 225–234.
[13] Cheng Bo, Xuesi Jian, Xiang-Yang Li, Xufei Mao, Yu Wang, and Fan Li. 2013. You're driving and texting: detecting drivers using personal smart phones by leveraging inertial sensors. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 199–202.
[14] Kevin Bouchard, Abdenour Bouzouane, and Bruno Bouchard. 2014. Gesture Recognition in Smart Home Using Passive RFID Technology. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '14)*. ACM, New York, NY, USA, Article 12, 8 pages. DOI: http://dx.doi.org/10.1145/2674396.2674405
[15] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. 2003. Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and vision computing* 21, 8 (2003), 745–758.
[16] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. Humantenna: using the body as an antenna for real-time whole-body interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1901–1910.
[17] Sinan Gezici, Zhi Tian, Georgios B Giannakis, Hisashi Kobayashi, Andreas F Molisch, H Vincent Poor, and Zafer Sahinoglu. 2005. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE signal processing magazine* 22, 4 (2005), 70–84.
[18] SS Ghassemzadeh, LJ Greenstein, A Kavcic, T Sveinsson, and V Tarokh. 2003. UWB indoor path loss model for residential and commercial buildings. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, Vol. 5. IEEE, 3115–3119.
[19] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. SoundWave: Using the Doppler Effect to Sense Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1911–1914. DOI: http://dx.doi.org/10.1145/2207676.2208331
[20] Chunmei Han, Kaishun Wu, Yuxi Wang, and Lionel M Ni. 2014. WiFall: Device-free fall detection by wireless networks. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 271–279.
[21] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. 2012. Accomplice: Location inference using accelerometers on smartphones. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*. IEEE, 1–9.
[22] Bo-Jhang Ho, Hsin-Liu Cindy Kao, Nan-Chen Chen, Chuang-Wen You, Hao-Hua Chu, and Ming-Syan Chen. 2011. HeatProbe: a thermal-based power meter for accounting disaggregated electricity usage. In *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 55–64.
[23] DecaWave DW1000 IR-UWB. http://www.decawave.com/products/dw1000.
[24] Pan Jing and Guan Ye-peng. 2013. Human-computer interaction using pointing gesture based on an adaptive virtual touch screen. *International Journal of Signal Processing, Image Processing and Pattern Recognition* 6, 4 (2013), 81–92.
[25] Holger Junker, Paul Lukowicz, and Gerhard Tröster. 2004. Continuous Recognition of Arm Activities With Body-Worn Inertial Sensors.. In *ISWC*. 188–189.
[26] Hamed Ketabdar, Peyman Moghadam, Babak Naderi, and Mehran Roshandel. 2012. Magnetic Signatures in Air for Mobile Devices. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services Companion (MobileHCI '12)*. ACM, New York, NY, USA, 185–188. DOI: http://dx.doi.org/10.1145/2371664.2371705
[27] Youngwook Kim and Hao Ling. 2009. Human activity classification based on micro-Doppler signatures using a support vector machine. *IEEE Transactions on Geoscience and Remote Sensing* 47, 5 (2009), 1328–1337.
[28] Eric Larson, Gabe Cohn, Sidhant Gupta, Xiaofeng Ren, Beverly Harrison, Dieter Fox, and Shwetak N. Patel. HeatWave: thermal imaging for surface user interaction. In *In Proc of CHI 2011*. 2565–2574.
[29] Won Oh Lee, Yeong Gon Kim, Hyung Gil Hong, and Kang Ryoung Park. 2014. Face recognition system for set-top box-based intelligent TV. *Sensors* 14, 11 (2014), 21726–21749.
[30] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. 2016. WiFinger: Talk to Your Smart Devices with Finger-grained Gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*

(UbiComp '16). ACM, New York, NY, USA, 250–261. DOI: http://dx.doi.org/10.1145/2971648.2971738

[31] Asier Lopez-Basterretxea, Amaia Mendez-Zorrilla, and Begoña Garcia-Zapirain. 2015. Eye/head tracking technology to improve HCI with iPad applications. *Sensors* 15, 2 (2015), 2244–2264.

[32] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 551–562.

[33] Brian D Mayton, Nan Zhao, Matt Aldrich, Nicholas Gillian, and Joseph A Paradiso. 2013. WristQue: A personal sensor wristband. In *2013 IEEE International Conference on Body Sensor Networks*. IEEE, 1–6.

[34] Brice Michoud, Erwan Guillou, and Saïda Bouakaz. 2007. Real-time and markerless 3D human motion capture using multiple views. In *Human Motion–Understanding, Modeling, Capture and Animation*. Springer, 88–103.

[35] Rajalakshmi Nandakumar, Bryce Kellogg, and Shyamnath Gollakota. 2014. Wi-fi gesture recognition on existing devices. *arXiv preprint arXiv:1411.5394* (2014).

[36] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 9.

[37] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. 2011. E-Gesture: A Collaborative Architecture for Energy-efficient Gesture Recognition with Hand-worn Sensor and Mobile Devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11)*. ACM, New York, NY, USA, 260–273. DOI: http://dx.doi.org/10.1145/2070942.2070969

[38] 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*. ACM, 19–24.

[39] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 27–38.

[40] James M. Rehg and Takeo Kanade. 1994. Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking. In *Proceedings of the Third European Conference-Volume II on Computer Vision - Volume II (ECCV '94)*. Springer-Verlag, London, UK, UK, 35–46. http://dl.acm.org/citation.cfm?id=645308.649148

[41] Jun Rekimoto. 2001. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*. IEEE, 21–27.

[42] 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 (MobiSys '16)*. ACM, New York, NY, USA, 85–96. DOI: http://dx.doi.org/10.1145/2906388.2906407

[43] Thad Starner and Alex Pentland. 1997. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*. Springer, 227–243.

[44] Guanhua Wang, Yongpan Zou, Zimu Zhou, Kaishun Wu, and Lionel M Ni. 2014. We can hear you with wi-fi!. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 593–604.

[45] Yan Wang, Jian Liu, Yingying Chen, Marco Gruteser, Jie Yang, and Hongbo Liu. 2014. E-eyes: Device-free Location-oriented Activity Identification Using Fine-grained WiFi Signatures. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom '14)*. ACM, New York, NY, USA, 617–628. DOI: http://dx.doi.org/10.1145/2639108.2639143

[46] 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 (CHI '16)*. ACM, New York, NY, USA, 3847–3851. DOI: http://dx.doi.org/10.1145/2858036.2858466

[47] Wei Xi, Jizhong Zhao, Xiang-Yang Li, Kun Zhao, Shaojie Tang, Xue Liu, and Zhiping Jiang. 2014. Electronic frog eye: Counting crowd using wifi. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 361–369.

[48] 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*. ACM, 9–14.

[49] Xu Zhang, Xiang Chen, Wen-hui Wang, Ji-hai Yang, Vuokko Lantz, and Kong-qiao Wang. 2009. Hand Gesture Recognition and Virtual Game Control Based on 3D Accelerometer and EMG Sensors. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI '09)*. ACM, New York, NY, USA, 401–406. DOI: http://dx.doi.org/10.1145/1502650.1502708