ABSTRACT

VIRMANI, ADITYA. Position and Orientation Agnostic Gesture Recognition Using WiFi. (Under the direction of Muhammad Shahzad.)

WiFi based gesture recognition systems have recently proliferated due to the ubiquitous availability of WiFi in almost every modern building. The key limitation of existing WiFi based gesture recognition systems is that they require the user to be in the same configuration (*i.e.*, at the same position and in same orientation) when performing gestures at runtime as when providing training samples, which significantly restricts their practical usability. In this thesis, we propose a WiFi based gesture recognition system, namely WiAG, which recognizes the gestures of the user irrespective of his/her configuration. The key idea behind WiAG is that it first requests the user to provide training samples for all gestures in only one configuration and then automatically generates virtual samples for all gestures in all possible configurations by applying our novel translation function on the training samples. Next, for each configuration, it generates a classification model using virtual samples corresponding to that configuration. To recognize gestures of a user at runtime, as soon as the user performs a gesture, WiAG first automatically estimates the configuration of the user and then evaluates the gesture against the classification model corresponding to that estimated configuration. Our evaluation results show that when user's configuration is not the same at runtime as at the time of providing training samples, WiAG significantly improves the gesture recognition accuracy from just 51.4% to 91.4%.

Position and Orientation Agnostic Gesture Recognition Using WiFi

by
Aditya Virmani

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2017

APPROVED BY:

Harilaos Perros	Khaled Harfoush

Muhammad Shahzad
Chair of Advisory Committee

DEDICATION

To my lovely parents and sisters.

BIOGRAPHY

Aditya Virmani received his bachelor 's degree in Information Technology from Netaji Subhas Institute of Technology, affiliated to University of Delhi, India inMay 2012. He has been pursuing a master 's degree in Computer Science at North Carolina State University, Raleigh, NC since the fall of 2015.

ACKNOWLEDGEMENTS

I am indebted to my family for their understanding and support in all my endeavors. I sincerely appreciate the moral support offered by all my friends and colleagues whose constant reminders kept me moving in the direction of completing my masters program.

I would like to thank my mother, for teaching me the most valuable lesson - *You reap what you sow. Nothing is impossible, and one's achievements are 'directly proportional 'to what he does towards them*. I would especially, like to thank my father, for showing me that with self-belief-and-motivation, obstacles can be overcome. I would always be indebted to my sisters, for being the constant support and Shrutika, for being the source of hope, confidence and belief in my life.

I would like to express my sincere gratitude to Dr. Muhammad Shahzad for his constant guidance and support. The discussions with him over various research issues have been very knowledgeable and have opened a whole world of new ideas for me to think about. His inputs at the various stages of the thesis were extremely vital. I would like to acknowledge Dr. Harry Perros and Dr. Khaled Harfoush for agreeing to be on my thesis committee and for their valuable advice and recommendations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Motivation and Problem Statement

Background: As computing devices are becoming smaller, smarter, and more ubiquitous, computing has started to embed in our environments in various forms such as intelligent thermostats [Nes; Smab; Smac; Smaa], smart appliances [Whi; Lgs], and remotely controllable household equipment [Wif; Smad; Ins; Phi]. Consequently, we need new ways to seamlessly communicate and interact with such pervasive and always-available comput-

ing. A natural choice for such communication and interaction is human gestures because gestures are an integral part of the way humans communicate and interact with each other in their daily lives. Indeed, researchers have been developing various types of gesture recognition systems, which usually rely on cameras [AM11; Che00; Her00; Opr12; Moe06; Sho13], wearable sensors [YT12; Par11; Lem11; Ert11; Sin15], or wireless signals [Abd15; Pu13; Kel14]. Among these systems, WiFi based gesture recognition systems are receiving widespread interest because the cost and complexity of deploying a WiFi based gesture recognition system is potentially negligible. It needs as few as only two commodity WiFi devices, such as a laptop and an access point, which already exist in almost every modern building. The intuition behind WiFi based gesture recognition systems is that the wireless channel metrics, such as channel state information (CSI) and received signal strength (RSS), change when a user moves in a wireless environment. The patterns of change in these metrics are different for different gestures. WiFi based gesture recognition systems first learn these patterns of change using machine learning techniques for each predefined gesture and then recognize them as the user performs them at runtime.

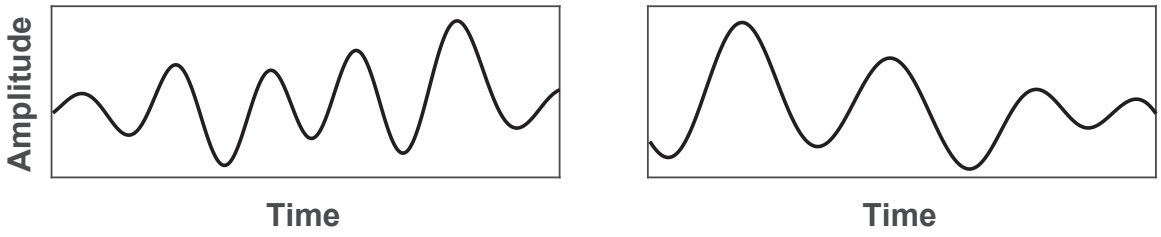
Limitations of Prior Art: While several WiFi based gesture recognition systems have been proposed within the last few years [Abd15; Pu13; Kel14; Adi14], and they work very well in the controlled environments they are designed for, their key limitation is that they recognize gestures with the reported accuracy only when the *position* and *orientation* of a user in the given environment does not change significantly compared to the position and orientation of the user at the time of providing training samples. Here, *position* is defined as the absolute location of the user in the given environment and *orientation* is defined as the direction in which the user is facing. Onward, we will use the term *configuration*

to refer to position and orientation collectively. Two configurations are equal only if their corresponding positions and corresponding orientations are the same.

The reason behind this limitation is that the patterns of change in wireless channel metrics due to a given gesture are different for different configurations of the user, and prior schemes do not take this into account. To illustrate this, consider a simple push gesture where a user moves her hand outward while facing towards the WiFi receiver. The amplitude of the signal reflected from the hand arriving at the WiFi receiver increases, as shown in Figure 1.1(a). Now consider the same push gesture by the user while facing away from the WiFi receiver. The amplitude of the signal decreases, as shown in Figure 1.1(b). This very simple illustration shows how same gesture results in different patterns of change in wireless channel metrics when performed in different configurations. Note that these two figures are obtained from real instances of the push gesture. We will provide our implementation details later in the project.

Unfortunately, this limitation significantly restricts the practical usability of existing WiFi based gesture recognition systems due to the inconvenience of requiring the user to always be in the same configuration when performing gestures at runtime as at the time of providing training samples. It is absolutely imperative to address and overcome this limitation in order to take WiFi based gesture recognition to its next level of evolution and to bring it a step closer to real world deployment.

Problem Statement: In this project, our goal is to design a WiFi based gesture recognition system that is agnostic to user's configuration, *i.e.*, it should recognize gestures of user irrespective of his/her configuration.



(a) Facing toward receiver (b) Facing away from receiver

Figure 1.1 Push gesture amplitude in two orientations

1.2 Proposed Approach

A seemingly obvious solution to making WiFi based gesture recognition agnostic to user's configuration is to first request the user to provide training samples for all gestures in all possible configurations in the given environment and then use these samples to learn patterns of change in wireless channel metrics for each gesture in all possible configurations. While this solution is theoretically plausible, it is impractical because the number of training samples to collect is prohibitively large and, therefore, almost impossible for a user to provide.

In this project, we present a WiFi based configuration agnostic gesture recognition system (WiAG), which recognizes gestures irrespective of user's configuration and at the same time, requires user to provide training samples in only one configuration. We have designed WiAG to work on commodity WiFi devices, such as laptops. To measure changes in the wireless channel due to gestures, WiAG uses channel state information (CSI), which is a well known wireless channel metric and has been used in several existing WiFi based gesture and activity recognition systems [Abd15; Han14; Wan14b; Wan15].

The key component of WiAG is our novel theoretically grounded *translation function* that can generate *virtual samples* of a given gesture in any desired configuration using a real sample of that gesture collected from the user in a known configuration. The key property of this translation function is that the virtual sample of a gesture that it generates for the desired configuration is identical to the real sample that would result from a real user performing that gesture in that configuration. The key idea behind WiAG is that instead of requesting the user to provide training samples in all possible configurations, it first requests the user to provide training samples for all gestures in only one configuration and then automatically generates virtual samples for all gestures in all possible configurations by applying our translation function on each training sample. Next, for each configuration, it generates a k-nearest neighbor (k-NN) based classification model using the virtual samples of the gestures in that configuration. To recognize gestures of a user at runtime, as soon as the user performs a gesture, WiAG first automatically estimates the configuration of the user and then evaluates the gesture against the classification model corresponding to that estimated configuration.

1.3 Technical Challenges

In designing WiAG, we faced several technical challenges, out of which, we describe three here. The first challenge is to develop a theoretical model to estimate changes in CSI measurements due to human gestures. This model is required in order to develop our translation function. We develop this model by first quantifying the changes in CSI measurements caused by a point object in linear motion. Next, we extend the model for the point object to a human arm (or any limb used during gesture) in linear motion. As human gestures often

comprise of non-linear motions of arm (*e.g.*, circle or wave gesture), we extend the model for arm in linear motion to arm in non-linear motion, and use this final model to develop our translation function.

The second challenge is to automatically estimate user's configuration when he/she performs a gesture. This estimate of user's configuration is required to select an appropriate classification model to evaluate the unknown gesture. To address this challenge, we made the formulation of our model for non-linear arm motion parametric in nature, where two of the parameters are the position and the orientation of user. To estimate user's configuration, we first ask the user to perform a preamble gesture and then solve our model to estimate the position and orientation using the CSI measurements observed during the preamble gesture.

The third challenge is to make WiAG resilient to static changes in the environment, such as adding an extra chair. Such static changes result in new multi-paths or change the lengths of existing multi-paths, which, in turn, modify the patterns of change in CSI measurements for the gestures. To address this challenge, we have developed our model for non-linear arm motion such that it characterizes the *change* in CSI measurements instead of the absolute CSI measurements. Consequently, the effects of the addition of new multi-paths or changes to the lengths of existing multi-paths as a result of static environmental changes get cancelled out in the model and the model only captures information about the multi-paths that are affected by moving objects, which in our case is user's arm.

1.4 Key Contributions

In this project, we make following three key contributions. First, we present a novel translation function that enables position and orientation agnostic gesture recognition without requiring the user to provide training samples in all possible configurations. Second, we present a novel configuration estimation scheme that automatically identifies the position and orientation of the user. To the best of our knowledge, there is no prior work that can estimate the orientation of user without requiring the user to hold a device in hand, such as an RFID tag [Sec13]. Third, we have implemented and extensively evaluated WiAG using commodity WiFi devices, which include a Thinkpad X200 laptop equipped with an Intel 5300 WiFi NIC and a TP-Link N750 access point. Our results from an extensive set of experiments show that when user's configuration is not the same at runtime as at the time of providing training samples, our translation function significantly improves the accuracy of gesture recognition from just 51.4% to 91.4%.

CHAPTER

2

RELATED WORK

In this chapter, we describe prior WiFi based gesture recognition systems and explain why each of them requires the user to always be in the same configuration at the time of performing gestures as at the time of providing training samples. In addition to gesture recognition systems, researchers have also proposed WiFi based activity recognition systems and WiFi based micro-movement sensing systems. Due to space limitation and rather less relevance to gesture recognition, we do not individually describe each activity and micro-movement recognition system. Instead, we only mention them and point the interested readers to

appropriate references.

Prior WiFi based systems can be divided into two broad categories: specialized-hardware (SH) based and commodity-devices (CD) based. The SH-based systems use software defined radios (SDRs), often along with specialized antennas or custom analog circuits, to capture changes in the wireless channel metrics due to human movements. The CD-based systems are implemented using commercially available devices, such as commodity laptops, and use the WiFi NICs in those devices to capture changes in the wireless channel metrics. The SH-based systems are usually slightly more accurate because SDRs can measure the wireless channel metrics more accurately compared to the WiFi NICs in commodity devices. Nonetheless, the CD-based systems have received a wider acceptance compared to the SH-based systems due to their potentially negligible deployment cost and complexity. Consequently, we have designed WiAG such that it can be implemented on commodity devices and does not require any specialized hardware. Next, we describe the existing work on CD-based systems followed by the SH-based systems.

2.1 CD-based Human Sensing using WiFi

To the best of our knowledge, Abdelnasser *et al.* proposed the only existing CD-based gesture recognition system, WiGest [Abd15]. WiGest uses RSS as the wireless channel metric. WiGest has two limitations. First, it works only when user's hand is in close proximity to the WiFi receiver because RSS values diminish rapidly with distance, making it difficult to observe any patterns in them as the distance increases. Second, it requires the user's hand to trace the same path with respect to the receiver that it did at the time of providing training samples because the patterns in RSS values depend on the relative location of hand with

respect to the receiver. In comparison, WiAG does not require user's hand to always trace the same path with respect to the receiver.

Researchers have also proposed other WiFi based human sensing systems that can be implemented on commodity devices, such as WiFall that detects a single human activity of falling [Han14], E-eyes [Wan14b] and CARM [Wan15] that recognize daily human activities, such as brushing teeth, taking shower, and doing dishes in fixed user configurations, WiDraw that enables in-air drawing [Sun15], WiHear that recognizes a predefined set of spoken words [Wan14a], WiKey that recognizes characters typed on keyboard [Ali15], WifiU that identifies people based on their gait [Wan16], FrogEye that counts the number of people in a crowd [Xi14], and indoor localization schemes [Sen13; Yan13].

2.2 SH-based Human Sensing using WiFi

Pu *et al.* proposed WiSee that uses an SDR to monitor micro-level doppler shifts in a carrier wave to recognize gestures [Pu13]. As doppler shifts depend on the direction of hand movement, WiSee cannot recognize gestures if user's orientation changes. Kellogg *et al.* proposed AllSee that uses an analog envelope-detection circuit to extract the amplitude of the received signal and learns the pattern of change in it to recognize gestures [Kel14]. As the magnitude of shift in amplitude at the receiver depends on the direction of movement of hand and the distance between the receiver and the hand, AllSee works only if the user is at a predefined distance and in a predefined orientation with respect to the receiver. Furthermore, AllSee works only when the user is within a short distance of less than 2.5 feet from the receiver. Due to these limitations, the authors of AllSee proposed to keep the receiver in user's backpack or pocket to keep the configuration of user with respect to the

receiver fixed. In comparison to both WiSee and AllSee, WiAG neither requires the user to be in a fixed configuration nor requires any specialized hardware.

Researchers have also proposed other WiFi based human sensing systems that utilize specialized hardware with objectives such as tracking humans [Adi14; Adi15a; AK13], measuring movement speeds of different parts of human body [VDG08], recognizing human gait [Lyo10], building images of nearby objects [Hua14], measuring breathing and heart rates [Adi15c], and localizing multiple users [Adi15b].

CHAPTER

3

TRANSLATION FUNCTION

In this chapter, we develop our translation function that can generate virtual samples of any given gesture in any desired configuration using a real sample of that gesture collected from the user in a known configuration. Note that any sample (either virtual or real) is essentially a time-series of the measurement values of one or more wireless channel metrics. Our translation function works on samples comprised of CSI measurements, which the commodity WiFi devices provide. Next, we first briefly describe what CSI measurements represent and then derive our translation function.

3.1 Channel State Information

Modern IEEE 802.11n/ac WiFi devices typically consist of multiple transmit and receive antennas and thus support MIMO. A MIMO channel between each transmit-receive (Tx-Rx) antenna pair comprises of 64 subcarriers (52 for data, 4 for pilot tones, and 8 for protective padding). WiFi devices continuously monitor the state of the wireless channel to effectively perform transmit power allocation and rate adaptation [Hal11]. For this monitoring, they utilize CSI measurements, which they calculate internally using preambles in the received signals. Let $X(f,t)$ and $Y(f,t)$ be the frequency domain representations of transmitted and received signals, respectively, on a subcarrier with carrier frequency f at time t between a given Tx-Rx pair. The two signals are related by the expression $Y(f,t) = H(f,t) \times X(f,t)$, where $H(f,t)$ represents the channel frequency response (CFR) of the wireless channel for the subcarrier of $X(f,t)$ at time t between the given Tx-Rx pair. A CSI measurement essentially consists of these CFR values, one for each subcarrier between each Tx-Rx pair. Let N_{Tx} and N_{Rx} represent the number of transmitting and receiving antennas, respectively, and let S represent the number of subcarriers between each Tx-Rx pair. Each CSI measurement contains S matrices (*i.e.*, one matrix per subcarrier) of dimensions $N_{Tx} \times N_{Rx}$ each. Each matrix contains CFR values for its associated subcarrier between all Tx-Rx pairs. As WiFi NICs generate CSI measurements repeatedly (*e.g.*, Intel 5300 WiFi NIC generates up to 2500 measurements/sec [Hal11]), we essentially obtain $S \times N_{Tx} \times N_{Rx}$ time-series of CFR values. Onward, we will call each time-series of CFR values a *CSI-stream*. As each CSI-stream is comprised of CFR values, next, we derive expressions that model the CFR in the absence and presence of gestures. These expressions will be used in deriving the

translation function.

3.2 CFR Modeling without Gestures

CFR of a wireless channel for a subcarrier quantifies the change in magnitude and phase, which that subcarrier experiences when passing through that wireless channel. Consider a wireless signal propagating on a subcarrier with frequency f. At time t, we can represent this propagating signal by a sinusoid $S(f,t) = A(t) \times e^{j2\pi f t}$, where $A(t)$ is the amplitude of the sinusoid and f is the subcarrier frequency. Assume that this signal is travelling through a time-invariant free-space, *i.e.*, the space contains no objects, obstacles, or humans. Let $X(f)$ represent this signal when it was initially transmitted at time $t = 0$. Thus, $X(f) = S(f,0) = A(0) \times e^{j\times 0}$. Let $Y(f)$ represent this signal when it arrives at the receiver at time T after traveling a distance D. Thus, $Y(f) = S(f,T) = A(T) \times e^{j2\pi f T}$. It is a well known fact that the amplitude of a wireless signal travelling through free-space is inversely proportional to the square of the distance it travels; thus $A(T) \propto \frac{A(0)}{(cT)^2} \Rightarrow A(T) = k\frac{A(0)}{D^2}$, where c is the propagation speed of the wireless signal and k is the proportionality constant, which caters for the effects of the relatively stationary characteristics of environment (such as its thermal properties) on wireless signal. If the wavelength of the sinusoid is λ, the phase of the sinusoid at the receiver will be $2\pi f T = 2\pi \frac{c}{\lambda} T = 2\pi \frac{D}{\lambda}$. Thus, $Y(f) = k\frac{A(0)}{D^2} \times e^{j2\pi \frac{D}{\lambda}} = (\frac{k}{D^2} \times e^{j2\pi \frac{D}{\lambda}}) \times X(f)$. As $Y(f) = H(f) \times X(f)$ for a time-invariant channel, we get

$$H(f) = k/D^2 \times e^{j2\pi \frac{D}{\lambda}} \tag{3.1}$$

The equation above quantifies the CFR of a time-invariant wireless channel in free-space for a subcarrier with wavelength λ (or frequency f, where $\lambda = \frac{c}{f}$).

Note that Eq. (3.1) models CFR for the ideal setting where there are no surfaces in the space that reflect the signal and the signal travels from the transmitter to the receiver on a single line-of-sight path. In practice, however, the signal that arrives at the receiver is a linear combination of several signals traveling through different paths due to reflections from objects in the space. Let N represent the number of such paths and let D_i represent the total length of the i^{th} path. Assuming that the space is still time-invariant despite the presence of multiple objects (*i.e.*, all objects stay stationary), the CFR of the wireless channel in this space is quantified by the generalized version of Eq. (3.1) as below.

$$H(f) = \sum_{i=1}^{N} \frac{k}{D_i^2} \times e^{j2\pi \frac{D_i}{\lambda}} \tag{3.2}$$

3.3 CFR Modeling with Gestures

When a user is present in the environment, different paths that the signal traverses can be divided into two categories: non-user reflected paths and user reflected paths. Non-user reflected paths include both line-of-sight paths as well as the paths reflected from static objects. Such paths do not change due to human movements. We represent the aggregate CFR of all non-user reflected paths with a constant $H_s(f)$. The user reflected paths can be further divided into two subcategories. The first subcategory consists of the paths that go directly to the receiver after reflecting from the user, and the second subcategory consists of the paths that experience further reflections after reflecting from the user. Compared to the signals traveling on the paths belonging to the first subcategory, the signals traveling on the paths belonging to the second subcategory have relatively lower amplitudes when

they arrive at the receiver, and are thus approximated with 0. Lets assume for now that there is only a single path of length D that is reflected from the arm of the user as he/she performs the gesture. We will soon incorporate the more complex scenario where multiple paths reflect from the arm. Let $H_D(f)$ represent the aggregate CFR of the wireless channel containing a stationary human. Based on Eq. (3.2), $H_D(f)$ is calculated as below.

$$H_D(f) = H_s(f) + k/D^2 \times e^{j2\pi\frac{D}{\lambda}} \tag{3.3}$$

This equation still assumes that the channel is time-invariant despite user's presence and is valid only if the user stays stationary. Next, we first derive an expression to calculate CFR when the user is not stationary anymore, rather moves the arm in a linear motion, such as doing a push gesture. After that, we extend this expression to calculate CFR when the user moves the arm in arbitrary directions to perform any desired gestures.

3.3.1 Linear Gestures

Consider a point object, initially situated at a distance D_1 from the receiver. Suppose the object moves at an angle θ with speed v for time t and arrives at the point situated at a distance D_2, as shown in Figure 3.1(a). Let $d = v\,t$ represent the distance that the object moves in time t. Using basic trigonometric identities, we get

$$D_2 \sin(\phi) = D_1 \sin(\theta) \tag{3.4}$$

$$D_2 \cos(\phi) = D_1 \cos(\theta) - d \tag{3.5}$$

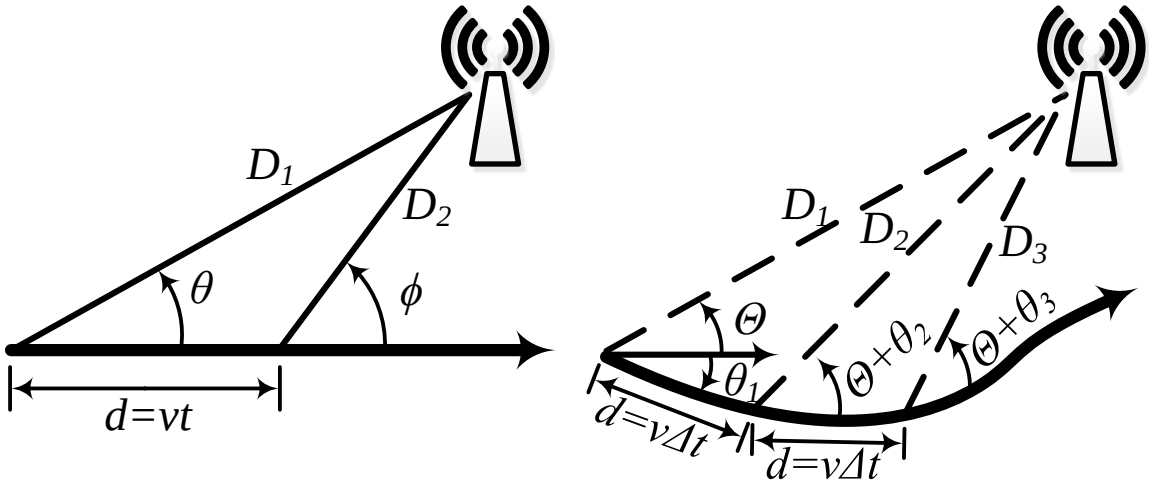
Dividing Eq. (3.5) by Eq. (3.4), squaring the resultant, and adding 1 yields the following on simplification.

$$(D_2)^2 = (D_1)^2\left\{1+\left(\frac{d}{D_1}\right)^2 - 2\frac{d}{D_1}\cos(\theta)\right\} \tag{3.6}$$

Replacing D in Eq. (3.3) with D_2 and substituting the value of $(D_2)^2$ from Eq. (3.6) into the denominator of the second term in Eq. (3.3), we get

$$H_{D_2}(f,t) = H_s(f) + \frac{k \times e^{j2\pi\frac{D_2}{\lambda}}}{(D_1)^2\left\{1+\left(\frac{d}{D_1}\right)^2 - 2\frac{d}{D_1}\cos(\theta)\right\}} \tag{3.7}$$

The equation above calculates CFR at time t when the point object has travelled a distance d and there is only one reflected path (*i.e.*, from the point object). Note that we have introduced t in the term $H_{D_2}(f,t)$ because $d = vt$, *i.e.*, the channel is no longer time-invariant due to change in the position of the point object with time.



(a) Linear Gestures (b) Non-linear Gestures

Figure 3.1 CFR modeling for gestures

When a human performs a linear gesture, such as a push or pull, the entire arm moves, and not just a point object. Thus, the number of paths reflected from the arm are a lot more

than just one. To calculate total CFR at time t, represented by $H_T(f,t)$, that incorporates reflections from the entire arm, we first divide the length of the portion of arm extended by time t into infinitesimal segments of size $\mathrm{d}d$ such that each segment offers a single reflected path (*i.e.*, acts like a point object). After that, we add the CFRs of paths reflected from each of these tiny segments by integrating the second term in Eq. (3.7) over d, which gives the following expression for the total CFR (after substituting $d = v\,t$).

$$H_T(f,t) = H_s(f) + \int_0^t \frac{k\,v \times e^{j2\pi\frac{D_2}{\lambda}}}{(D_1)^2\left\{1+\left(\frac{vt}{D_1}\right)^2 - 2\frac{vt}{D_1}\cos(\theta)\right\}}\mathrm{d}t$$

Unfortunately, it is not straightforward to obtain the value of the constant $H_s(f)$. To eliminate it, we differentiate the equation above with respect to t. This differentiation also eliminates the contribution of any static object in the environment to the CFR, and thus, makes WiAG resilient to static changes in the environment, such as adding an extra chair. which results in the following equation.

$$\mathrm{d}H_T(f,t) = \frac{k\,v \times e^{j2\pi\frac{D_2}{\lambda}}}{(D_1)^2\left\{1+\left(\frac{vt}{D_1}\right)^2 - 2\frac{vt}{D_1}\cos(\theta)\right\}} \tag{3.8}$$

The physical interpretation of this equation is that it calculates the change introduced in the total CFR for subcarrier with frequency f at time t as a result of the reflection introduced/removed by a small surface area as the arm extends/retracts by a small distance. Thus, if we know the initial distance D_1 and the orientation θ of the user, we can estimate the change in CFR values due to linear gestures using the parametric formulation in Eq. (3.8). In practice, we obtain the values of $\mathrm{d}H_T(f,t)$ for any desired subcarrier by taking the first order difference of the CSI-stream of that subcarrier.

A Practical Consideration: Eq. (3.8) describes both amplitude and phase of the first order derivative of the total CFR. Unfortunately, accurately capturing the CFR phase in commodity WiFi devices is difficult because it requires the sender and receiver to be perfectly synchronized, which is usually not possible due to the non-negligible carrier frequency offsets (CFO) in these devices [Gje14; Wan15]. The frequency drifts due to CFO lead to spurious changes in CFR phase calculated at the receiver and thus renders CFR phase infeasible for inferring human movements. More information on this topic is available in [Wan15].

Due to the non-negligible CFO, we do not use the phase information. Instead, we only utilize the amplitude information of the CFR. Even though the amplitude values of CFR are also affected by noise, the noise is additive in nature and can be filtered out. Thus, it does not significantly affect the change in CFR amplitude due to user's gestures. Taking only the amplitude into consideration, Eq. (3.8) reduces into the following.

$$\|\mathrm{d}H_T(f,t)\| = \frac{k\,v}{(D_1)^2\left\{1+\left(\frac{v\,t}{D_1}\right)^2 - 2\frac{v\,t}{D_1}\cos(\theta)\right\}} \tag{3.9}$$

This equation calculates the amplitude of the first order derivative of total CFR for linear gestures, such as push and pull. Next, we extend this model to gestures with arbitrary directions of motion. We will use the resulting extended model to derive our translation function.

3.3.2 Non-linear Gestures

Consider a point object initially situated at a distance D_1 from the receiver. Suppose the object moves along an arbitrary non-linear path at a constant speed v, starting at an angle θ_1 with respect to the user orientation, *i.e.*, the direction the user is facing, as shown in Figure 3.1(b). Let Θ represent the absolute user orientation. After a short duration Δt, the object reaches a position where its distance from the receiver is D_2. The duration Δt is small enough such that the path followed by the object during this time can be approximated by a straight line. Until this point, the situation is exactly the same as discussed for linear-gestures in Section 3.3.1. At this point, the object continues to move, without stopping, at the same speed v but at an angle θ_2 with respect to the user orientation. After another short Δt, it reaches a position where its distance from the receiver is D_3 as shown in the Figure 3.1(b) The object continues to move like this without stopping.

As the object moves at a constant speed v, the distance d it covers in each Δt interval is constant, *i.e.*, $d = v \times \Delta t$. Following the same steps as we took to derive Eq. (3.6), we obtain the following equation.

$$(D_i)^2 = (D_{i-1})^2 \left\{ 1 + \left(\frac{d}{D_{i-1}} \right)^2 - 2 \frac{d}{D_{i-1}} \cos(\theta_i + \Theta) \right\} \tag{3.10}$$

Let t_i represent the time when the object has moved through i segments along its path, *i.e.*, $t_i = i \times \Delta t$. If we can automatically calculate the values of D_1, all θ_i, Θ, k, and v (we will describe in Section 3.4.1 how we automatically calculate these values), we can calculate values of D_i for all i. Following the same steps as for linear gestures in Section 3.3.1, we arrive at an equation similar to Eq. (3.9) that calculates the change in the amplitude of total

CFR during the i^{th} short time interval Δt as a result of a very small change in reflection due to a small change in the position of the arm as it moves by a small distance. This equation is given below.

$$\|\mathrm{d}H_T(f,t_i)\| = \frac{k\,v}{(D_i)^2\left\{1+\left(\frac{d}{D_i}\right)^2 - 2\frac{d}{D_i}\cos(\theta_i+\Theta)\right\}} \tag{3.11}$$

Eqs. (3.10) and (3.11) are the generalized versions of Eqs. (3.6) and (3.9), respectively, and are thus valid for both non-linear as well as linear gestures. As mentioned earlier, we obtain the values of $\mathrm{d}H_T(f,t_i)$ for any desired subcarrier by taking the first order difference of the CSI-stream of that subcarrier. Through simple polar maths, it is easy to see that during consecutive CSI measurements (10ms apart), the fastest moving point on a human arm moves by less than 2cm, which is small enough for our context. Thus, our assumption that the path followed by any point on the arm during Δt interval can be approximated by a straight line holds in practice.

3.4 Gesture Translation

Next, we use Eq. (3.11) to derive our translation function. Recall that $\|\,\mathrm{d}H_T(f,t_i)\,\|$ represents the difference between consecutive CFR values in the CSI-stream of subcarrier with frequency f at times t_i and t_{i-1}. Let $V(f,t_i,X)$ represent the value of $\|\mathrm{d}H_T(f,t_i)\|$ when user is in configuration X. Eq. (3.11) can be written as.

$$V(f,t_i,X)\times\frac{(D_i^X)^2}{k^X v}\left\{1+\left(\frac{d}{D_i^X}\right)^2 - 2\frac{d}{D_i^X}\cos(\theta_i^X+\Theta^X)\right\} = 1$$

If a user provides a training sample in configuration A and we desire to translate it to configuration B, as the right hand side of the equation above is a constant, translated values

of CFR, represented by $V(f, t_i, B)$, are given by the following equation.

$$V(f, t_i, B) = V(f, t_i, A) \times \frac{k^B (D_i^{\mathrm{A}})^2 \left\{1 + \left(\frac{d}{D_i^{\mathrm{A}}}\right)^2 - 2\frac{d}{D_i^{\mathrm{A}}}\cos(\theta_i^{\mathrm{A}} + \Theta^A)\right\}}{k^A (D_i^{\mathrm{B}})^2 \left\{1 + \left(\frac{d}{D_i^{\mathrm{B}}}\right)^2 - 2\frac{d}{D_i^{\mathrm{B}}}\cos(\theta_i^{\mathrm{B}} + \Theta^B)\right\}} \tag{3.12}$$

Eq. (3.12) is our gesture translation function. To apply this translation function, WiAG needs the values of the configuration parameters (D_1 and Θ), proportionality constant (k), shape parameters (all θ_i), and speed (v) at both configurations A (where the user provided training samples) and B (the desired configuration). Next, we describe how WiAG obtains the values of these parameters at both configurations.

3.4.1 Parameter Estimation for Configuration A

Config. Params. and Prop. Constant: We use our configuration estimation scheme to automatically calculate the values of D_1^{A}, Θ^{A}, and k^{A}. We will describe our configuration estimation scheme in Chapter 6.

Shape Parameters: To estimate the values of shape parameters θ_i^{A}, we request the user to hold a smart phone in hand for at least one sample per gesture when providing training data. The smart phone runs our custom application that applies standard algorithms on the measurements from the onboard inertial measurement unit (IMU) to calculate the direction and magnitude of displacement of hand along all three axes and obtains the values of all θ_i^{A}. *We emphasize here that we ask the user to hold a smart phone in hand* **only** *at the time of collecting training samples and* **never** *at the time of using the trained gesture recognition system at runtime.*

Speed: We again use our smart phone app to calculate the speed of the gesture. Our app divides the length of the path the hand follows during the gesture by the duration of the gesture to calculate speed. We again emphasize that the user holds a smart phone in hand only during training samples.

3.4.2 Parameter Estimation for Configuration B

Config. Params. and Prop. Constant: When generating classification models, the values of configuration parameters at configuration B are already known (we will see this shortly). When recognizing an unknown gesture from user at runtime, we use our configuration estimation scheme (Chapter 6) to calculate these values.

Shape Parameters: We use the same values for shape parameters at configuration B as at configuration A, *i.e.*, $\forall i$, $\theta_i^{\mathrm{B}} = \theta_i^{\mathrm{A}}$, because we want the shape of the gesture to stay the same in the virtual samples.

Speed: We use the same value for speed at configuration B as at configuration A because the speed of gesture should not change in the virtual sample.

3.4.3 Applying the Translation Function

To generate a virtual sample of a given gesture at a desired configuration B using a sample of that gesture collected at configuration A, WiAG first estimates the values of all parameters for both configurations as described above, and then $\forall i$, uses Eq. (3.12) to calculate $V(f, t_i, B)$ corresponding to each value $V(f, t_i, A)$ in the first-order difference of the filtered CSI-streams of that gesture. Filtering will be discussed shortly.

CHAPTER

4

WIAG – OVERVIEW

In this chapter, we provide an overview of WiAG and how it utilizes the translation function to perform position and orientation agnostic gesture recognition. To recognize gestures in any given environment, WiAG needs classification models for those gestures in all possible configurations in that environment. Theoretically, the number of possible configurations is infinite. However, practically, we observed that a change in position of up to 12 inches and a change in orientation of up to 45° does not have a significant impact on gesture recognition accuracy. Therefore, we recommend to generate classification models at all

positions corresponding to the intersection points of an imaginary grid, where the side of each square in the grid is at most 12 inches in length. At each position, we recommend to generate classification models for at least $360°/45° = 8$ orientations. Given the training samples collected in a single known configuration, WiAG builds classification models for each configuration in the following four steps.

1) CSI-Stream Conditioning: In this step, WiAG performs two tasks: noise removal and gesture extraction. CSI-streams contain a large amount of noise that occludes the variations caused by gestures in the channel frequency response. To remove this noise, WiAG first applies a principal component analysis (PCA) based de-noising technique followed by Butterworth filtering. It then calculates the first order difference of the filtered streams. We call the resulting streams "differentiated principal component" (dPC) streams. To extract a gesture from the dPC-streams, *i.e.*, to identify the start and end times of the gesture, WiAG uses a supervised thresholding scheme. It then uses the values in dPC-streams contained between the start and end times for further processing.

2) Configuration Estimation: Before collecting any training samples, WiAG requests the user to first perform a preamble gesture in the configuration where the user will provide the training samples. Using the dPC-streams from the preamble gesture, WiAG estimates the configuration parameters and the proportionality constant by solving the CFR model derived in Section 3.

3) Gesture Translation: In this step, using the parameter values obtained from the second step along with the values of the shape parameters and speed obtained from the IMU, WiAG applies the translation function on each sample of each gesture extracted from the dPC-streams during the first step to generate virtual samples of the gestures in all

configurations.

4) Classifier Training: In this step, for each configuration, WiAG takes the virtual samples of all gestures in that configuration from the third step and extracts appropriate features from them. It uses these features to perform k-NN based classification. Note that WiAG applies only the third and the fourth steps for each configuration in the given environment. The first and the second steps are applied only once on the training data.

Gesture Recognition: To recognize a gesture at runtime, WiAG requests the user to first perform a preamble gesture. The preamble gesture is not required if the time elapsed since the user performed the previous gesture is less than 10s. WiAG assumes that the user does not change configurations between consecutive gestures if they are separated in time by less than 10s. WiAG continuously applies CSI-stream Conditioning step on the incoming CSI-streams. As soon as it detects and extracts an unknown gesture, it checks whether more than 10s have elapsed since the previous gesture. If yes, WiAG applies the Configuration Estimation step, otherwise, it uses the values from previous most recently executed configuration estimation step. WiAG then takes the features (which it extracted during the fourth step) of the virtual samples corresponding to the estimated configuration and applies k-NN based classification to recognize the incoming gesture. Next, we describe CSI-Stream Conditioning, Configuration Estimation, and Classifier Training steps in detail. We do not discuss Gesture Translation step further because we have already described it in detail in Chapter 3.

CHAPTER

5

CSI-STREAM CONDITIONING

In this chapter, we first explain how WiAG removes noise from CSI-streams and then describe how it detects the start and end of a gesture.

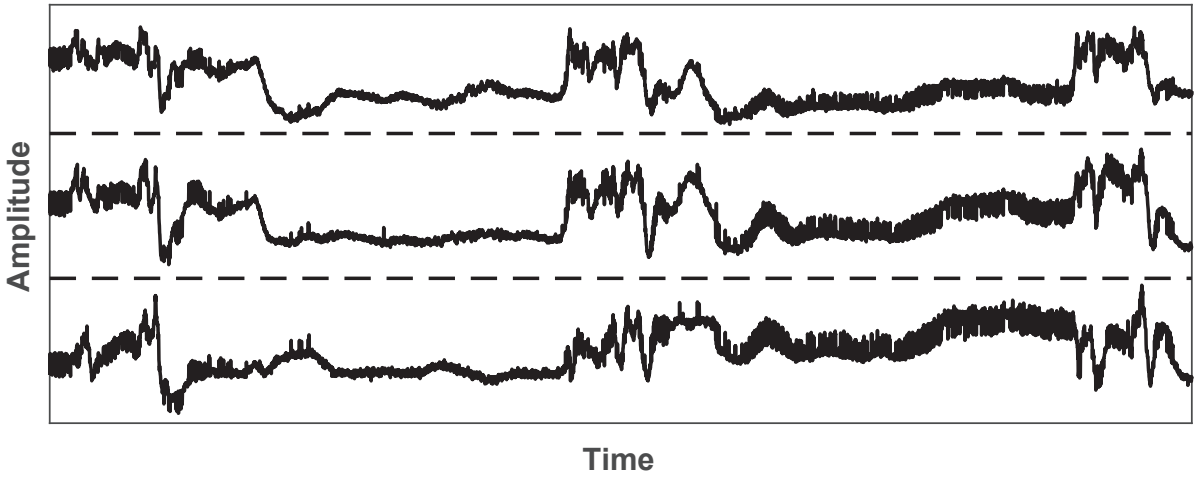
5.1 Noise Removal

The CSI-streams provided by WiFi NICs are extremely noisy. This noise occludes the variations caused by gestures, which makes it difficult to recognize them. The major source of noise in CSI-streams is the internal state transitions in sender and receiver WiFi NICs, such

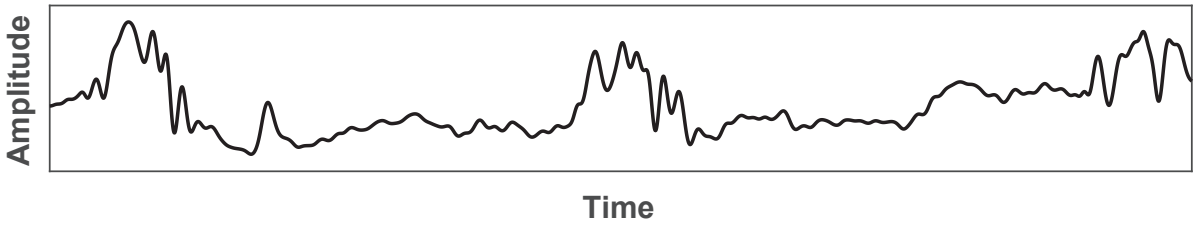
as transmission power changes and transmission rate adaptations. These state transitions manifest as high amplitude impulse and burst noises in CSI-streams. Before we describe how WiAG removes this noise, recall from Section 3.1 that there are 64 subcarriers between each Tx-Rx antenna pair. However, the driver of our Intel 5300 WiFi NIC reports CSI measurements for only 30 out of the 64 subcarriers. Therefore, we get only 30 CSI-streams for each Tx-Rx antenna pair.

To remove the noise, WiAG uses the de-noising scheme of CARM, proposed in [Wan15]. We briefly summarize this scheme here and refer interested readers to [Wan15] for details. CARM's de-noising scheme leverages the observation that human movements cause correlated changes in all CSI-streams. Principal component analysis (PCA) is a natural choice to capture these correlated changes. For each Tx-Rx pair, we first apply PCA on the 30 CSI-streams to capture all correlated human movements and then pass each of the resulting streams through a butterworth filter to remove any noise in the captured human movements. Consequently, we get 30 new streams per Tx-Rx pair, ordered based on the amount of information each stream contains. We call these new streams principle component (PC) streams. CARM recommends to use the second and third PC-streams for further processing because they contain little to no noise while at the same time clearly capture human movements. CARM does not use the first PC-stream because it captures the leftover correlated noise that the butterworth filter could not remove. We, however, chose to proceed only with the third component because we observed that the second component also captured some noise. Consequently, we get $N_{Tx} \times N_{Rx}$ PC-streams, one per Tx-Rx pair. Figure 5.1(a) shows three randomly chosen raw CSI-streams out of 30 CSI-streams between a Tx-Rx pair. We observe from this figure that these streams are indeed very noisy. Figure

5.1(b) shows the third PC-stream corresponding to the same duration as the CSI-streams in Figure 5.1(a). We observe that this PC-stream contains no visible noise while at the same time captures human gestures clearly.



(a) Raw CSI-Streams

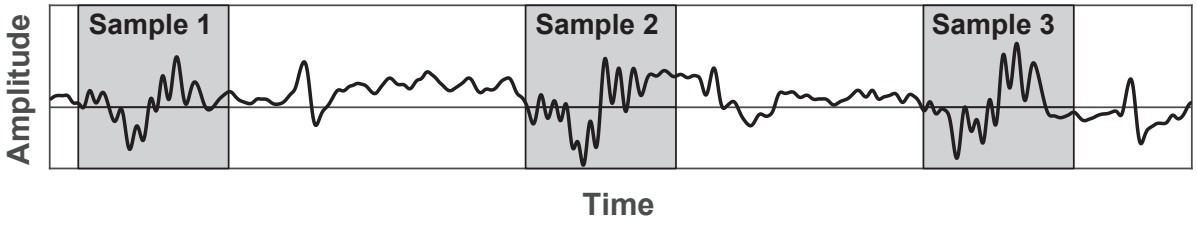


(b) Third PC-stream

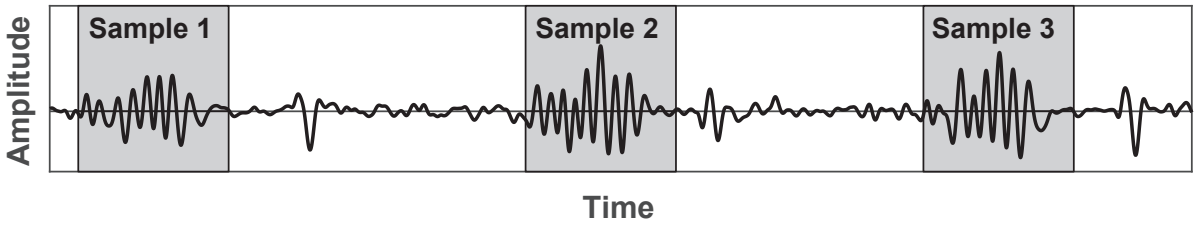
Figure 5.1 CSI- and PC-streams

5.2 Gesture Detection

To detect the start and end of a gesture, like most gesture recognition systems (including non-WiFi based), we request the user to take brief pauses before and after the gesture. Figure 5.2(a) plots a PC-stream when the user performs a push gesture three times, with brief pauses before and after each gesture. We observe from this figure that there is an absolute increase or decrease in the stream values at the start or end of gestures. This happens due to change in the position of arm at the start and end of a gesture. This observation makes it challenging to develop a threshold based gesture detection scheme because one never knows where the arm will be at the start and end of an arbitrary gesture.



(a) Third PC-Stream



(b) Third dPC-stream

Figure 5.2 Effect of differentiation on PC-stream

We also observe from Figure 5.2(a) that during the pauses, the values in the PC-stream

experience only small changes, whereas, during the gestures, they experience larger changes. WiAG leverages this observation to detect the start and end of gestures. More specifically, it takes the first order difference of the PC-stream by subtracting each value in the PC-stream from the next value. We represent this differentiated PC-stream with dPC-stream. Figure 5.2(b) plots the dPC-stream corresponding to the PC-stream in Figure 5.2(a). From this figure, we make two key observations. First, during the gestures, the amplitudes of variations in the dPC-stream are very high, whereas during the pauses, the amplitudes are very low. Second, the dPC-stream is centered around 0, especially during the pauses, regardless of what the position of the arm is at the start and end of gestures. These two observations enable us to set a threshold to identify the start and end of any gesture.

To identify the start and end of a gesture, WiAG monitors the time-separation between peaks in the dPC-stream of any one Tx-Rx pair. But first, it has to distinguish between peaks generated during gestures from the peaks generated during pauses. WiAG identifies peaks in the dPC-stream by comparing each value in the stream with the value before and the value after it. If any value is less than both values before and after it, that value is a local minima. Similarly, if any value is greater than the values before and after it, it is a local maxima. From the empirical study of our data set, we observed that the absolute amplitudes of almost 100% of the peaks during pauses is less than a threshold $T = 0.8$, whereas the absolute amplitudes of 98% of the peaks during gestures is greater than T. WiAG uses this threshold T to distinguish between peaks belonging to gestures from the peaks belonging to pauses.

While $T = 0.8$ in our data set, WiAG actually automatically calculates this threshold for any given environment by asking a user to first stay stationary and then move the arm

randomly. WiAG takes the time when the user started moving the arm as input. Using one of the resulting dPC-streams, it calculates average μ_S of absolute amplitudes of peaks when the user was stationary and average μ_G of absolute amplitudes of peaks when the user was moving the arm, and sets the threshold as $T = (\mu_S + \mu_G)/2$.

After identifying all peaks with absolute amplitudes $> T$, WiAG makes groups of peaks that are closely spaced in time. More specifically, it groups together all peaks for which the largest time separation between any pair of consecutive peaks is no more than 300ms. Each group represents a gesture, where the start and end times of the gesture are the times at which the first and last peaks in the group are located, respectively. With this method, WiAG successfully identified the start and end times of 96% of gestures in our data set.

CHAPTER

6

CONFIGURATION ESTIMATION

In this chapter, we explain how WiAG estimates the configuration parameters (D_1 and Θ) and the proportionality constant (k). For this purpose, WiAG requests the user to perform a preamble gesture, which in our implementation is a push gesture. We will later show that push is also one of the six gestures on which we evaluated WiAG. If a user wants to perform a push gesture, but is required to perform the preamble gesture first because the user has not performed any gesture in last 10 seconds, the user simply performs the push gesture twice, with a brief pause in between. WiAG treats the first gesture as the preamble

gesture and the second as the regular gesture. As push is a linear gesture, our configuration estimation scheme is based on Eq. (3.9).

The terms $(\frac{v\,t}{D_1})^2 - 2\frac{v\,t}{D_1}\cos(\theta)$ in the denominator of Eq. (3.9) almost always evaluate to less than 1 because $v\,t$ is the distance the hand travels, whose maximum value for a push gesture is approximately equal to the length of the arm. Furthermore, as D_1 is the distance of the user from the receiver, it is almost always greater than the length of the arm. Consequently, we can do binomial expansion of $\| \mathrm{d}H_T(f,t) \|$ in Eq. (3.9), which gives us

$$\| \mathrm{d}H_T(f,t) \| \approx \frac{k\,v}{(D_1)^2}\Big\{1 + \Big(2\frac{v}{D_1}\cos(\Theta)\Big) \times t - \Big(\frac{v^2}{D_1^2}(1 - 4\cos^2(\Theta))\Big) \times t^2\Big\}$$

Note the that we have replaced θ with Θ in the equation above because in our formulation for linear gestures, the orientation Θ of the user was represented by θ (see Figure 3.1(a)). The binomial expansion above expresses $\| \mathrm{d}H_T(f,t) \|$ as a polynomial in t. To estimate the values of D_1, Θ, and k, we fit a polynomial of degree n in least squares sense on an observed dPC-stream of the preamble gesture, which gives us $n+1$ polynomial coefficients. In our implementation, we used $n = 8$ as it provided the best fit. Let a_i represent the i^{th} coefficient from the polynomial fit. By comparing the estimated values of the coefficients a_0, a_1, and a_2 from the polynomial fit with the coefficients of t^0, t^1, and t^2, we get

$$a_0 = \frac{k\,v}{(D_1)^2}, \qquad a_1 = \frac{k\,v}{(D_1)^2}\Big(2\frac{v}{D_1}\,cos(\Theta)\Big)$$
$$a_2 = -\frac{k\,v}{(D_1)^2}\Big(\frac{v^2}{D_1^2}(1 - 4\cos^2(\Theta))\Big)$$

Recall that we already know the value of v from our IMU based technique. Thus, we have

three unknowns D_1, Θ, and k and three equations, which we solve simultaneously to obtain the values of these unknowns. Also recall that WiAG obtains $N_{Tx} \times N_{Rx}$ dPC-streams per gesture. Thus, it actually first estimates the coefficients a_0, a_1, and a_2 from each of the $N_{Tx} \times N_{Rx}$ dPC-streams and then uses their average values in the three equations above to estimate the values of D_1, Θ, and k.

CHAPTER

7

CLASSIFIER TRAINING

WiAG builds a classification model for gestures in each configuration. As WiAG uses k-NN based classifier, the process of building the classification model in any given configuration essentially involves only extracting features form the virtual samples in that configuration. Recall from Chapter 4 that WiAG obtains virtual samples in any desired configuration by applying the three steps of CSI-Stream Conditioning, Configuration Estimation, and Gesture Translation. Next, we first describe how WiAG extracts features from virtual samples and then explain how it evaluates an unknown gesture.

7.1 Feature Extraction

WiAG uses discrete wavelet transform (DWT) to extract features from virtual samples. We chose DWT due to its inherent ability to extract features with high classification potential when time-series for different samples belonging to the same class have similar shapes while the time-series of different samples belonging to different classes have different shapes. DWT is a popular tool to extract features and has extensively been used in both WiFi based [Wan15; Abd15; Ali15] as well as non-WiFi based human sensing systems [Tza01; Oca09]. One of the reasons behind its popularity is that it provides a good resolution in both time and frequency and enables measurements of both fast and slow gestures. WiAG extracts features using DWT in the following four steps.

1) Aggregation: Recall that each virtual sample consists of $N_{Tx} \times N_{Rx}$ dPC-streams and that the CFR values across all Tx-Rx pairs are correlated. Consequently, all $N_{Tx} \times N_{Rx}$ dPC-streams are also correlated. We leverage this observation to combine these $N_{Tx} \times N_{Rx}$ dPC-streams into a single stream by first applying PCA on these streams and then picking the resulting PCA component with the largest Eigen value. We name this component dPC-component. The motivation behind combining these streams into a single stream is to reduce the computational cost of the k-NN based classifier when recognizing gestures at runtime.

2) Extrapolation: Due to human imprecisions, users almost always take different amounts of time to perform two samples of even the same gesture. Consequently, the number of points in the dPC-component of each virtual sample is also almost always different. To apply DWT, we need the number of points in dPC-component of every sample to be the

same. From the exploratory study of our data set, we observed that our volunteers always took less than 10s to perform any gesture. As we will describe later, we measure CFR values from our WiFi NIC at a rate of 100 samples/sec. Consequently, each dPC-component always has less than 1000 points. Thus, we chose a number 1024 (>1000 and an exact power of 2, which makes it easy to apply DWT), and used smoothing splines [Smo] to extrapolate each dPC-component to 1024 points.

3) DWT: DWT is a hierarchical transform that gives *detail coefficients* at multiple *levels*, where the frequency span at any given level is half of the span at the level before it. When we apply DWT to a dPC-component, the resulting detail coefficients at any given DWT level represent the correlation between the dPC-component and the wavelet function at the frequency corresponding to that DWT level. The detail coefficients are also ordered according to their occurrence in time. In our implementation, we used Daubechies D4 wavelet because it is the most commonly used wavelet and also gave the highest accuracy in our experiments. Similarly, we used the detail coefficients corresponding to DWT level 3 due to its high accuracy. At level 3, we get $1024/2^3 = 128$ detail coefficients for each virtual sample. Due to space limitation, we do not give the theory and other details of DWT here, and refer interested readers to [**burrus1997introduction**; Lan96].

4) Energy Calculation: While the detail coefficients contain distinct patterns for virtual samples of different gestures, we observed that these patterns also had slight shifts across the virtual samples of the same gesture due to human imprecisions. Thus, using these coefficients directly for classification results in relatively low accuracy. To mitigate this, WiAG distributes these 128 detail coefficients almost equally into 10 bins. More specifically, it puts the first 8 sets of 13 consecutive coefficients in first 8 bins and the remaining 2 sets of

12 consecutive coefficients in the remaining two bins. It then calculates the energy of each bin and uses it as a feature. The energy of a bin is equal to the sum of square of all values in it. Thus, WiAG extracts 10 features per virtual sample, which it uses for classification.

7.2 Recognizing Gestures at Runtime

To recognize a gesture at runtime, WiAG follows the steps described under "Gesture Recognition" in Chapter 4. Here we only describe how it applies the k-NN based classification step at the estimated user configuration. To classify an unknown gesture, WiAG first extracts the 10 features from its $N_{Tx} \times N_{Rx}$ dPC-streams. After that, it applies the k-NN classifier, which essentially looks at the k nearest neighbors of this unknown gesture in the 10 dimensional space containing all virtual samples of all gesture at the estimated configuration, and declares the unknown gesture to be the one whose virtual samples are most frequent among its k nearest neighbors. In our implementation, $k = 20$.

CHAPTER

8

IMPLEMENTATION & EVALUATION

We implemented WiAG on a Thinkpad X200 laptop equipped with an Intel 5300 WiFi NIC and installed the tool developed by Halperin *et al.* [Hal11] on the laptop to obtain CSI measurements in the 2.4GHz WiFi frequency band with 20MHz bandwidth subcarriers. The laptop communicates with a TP-Link N750 access point (AP). In our implementation, the laptop and AP contain two and three antennas, respectively, *i.e.*, $N_{Tx} = 2$ and $N_{Rx} = 3$. To collect CSI measurements, we probed the AP using ping command every 10ms and achieved a sampling rate of 100 samples/sec. In comparison, existing schemes, such as CARM, ping

the AP at very high rates of up to 2500 pings per second, which consumes a significant portion of the bandwidth, leaving little bandwidth for transferring actual data. Next, we first describe our evaluation setup along with information about the data we collected. After that, we evaluate the configuration estimation and gesture recognition accuracies of WiAG. Last, we evaluate the performance of WiAG under changing environmental conditions. All evaluations are done using using real world data.

8.1 Evaluation Setup

We collected 1427 samples from 10 volunteers for 6 gestures at 5 different positions on 8 different days. The names of the six gestures and the number of samples per gesture are listed in Table 8.1. Note that WiAG does not require the training samples to come from the same user whose gestures it has to recognize. Consequently, the number of volunteers in the data set is irrelevant. Nonetheless, we still collected samples from 10 volunteers. We collected these samples in a 25ft×16ft room that contains usual furniture including 7 chairs and 3 tables. Figure 8.1 shows the layout of the room along with the locations of access point (Tx), laptop (Rx), and the 5 positions (represented by triangles) where we collected samples. The ordered-pair under each triangle gives volunteers' absolute position in inches (with Tx being at origin) and the value above gives volunteers' orientation with respect to the receiver at that position. We collected these samples after obtaining IRB approval. To incorporate the effects of environmental changes, we randomly moved furniture in the room each day before collecting samples on that day.

Table 8.1 Summary of gesture data set

Gesture	**Samples**
Push	207
Pull	235
Flick	204
Circle	324
Throw	152
Dodge	305

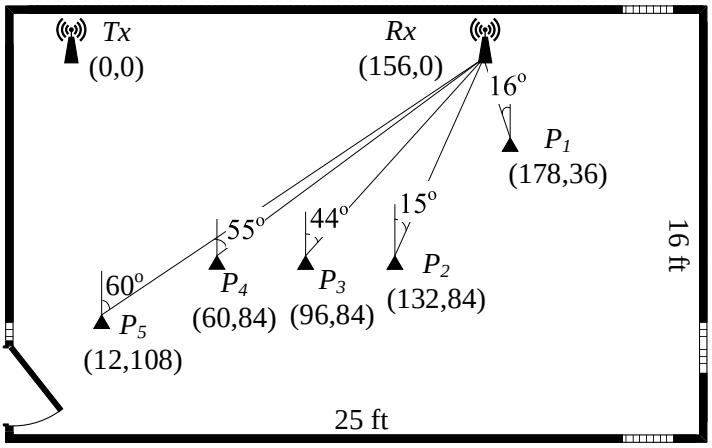


Figure 8.1 Layout of data collection environment

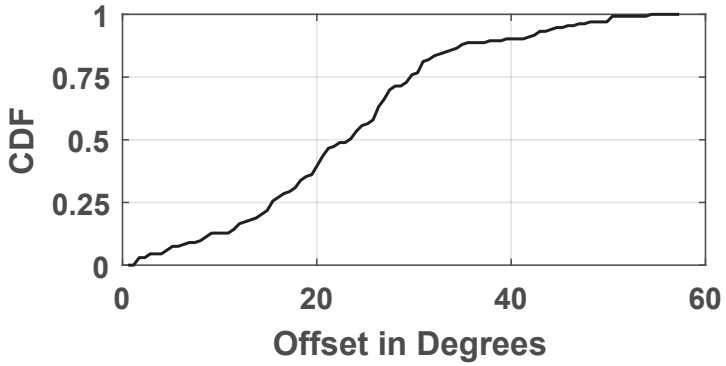


Figure 8.2 CDF of absolute error in WiAG's orientation estimation

8.1.1 Orientation Estimation Accuracy

To evaluate WiAG's configuration estimation scheme, we show the performance of WiAG in estimating orientations of users. For this, we took all samples of the push gesture (recall that WiAG uses push gesture as preamble) from our data set and estimated user orientation from each sample. We report the performance of WiAG's orientation estimation in terms of *absolute error*, which is the absolute value of the difference between the estimated orientation and the actual orientation, measured in degrees. We observe from our experiments that WiAG achieves average absolute error of less than 23° in estimating user orientation. Figure 8.2 plots the CDF of absolute errors across all samples of push gesture collected at all positions. Recall that we recommended to generate classification models for 8 orientations per position, which corresponds to 45° per orientation. As WiAG achieves an average error of almost half of 45°, it can correctly identify the classification model that it should use when evaluating an unknown gesture. We also observed from our experiments that the smallest absolute error that WiAG achieved is just 1.3°.

8.2 Gesture Recognition Accuracy

Next, we first show the performance of WiAG in recognizing gestures when only the orientation of user changes compared to the orientation of user at the time of providing training samples. For this set of experiments, the position of user stays the same. After that, we show the performance of WiAG when both orientation and position of user change. We report the gesture recognition performance of WiAG in terms of *accuracy*, which is defined as the percentage of samples correctly recognized.

Accuracy with Change in Orientation: For this set of experiments, in addition to collecting samples in the 44° orientation at position P_3 (see Figure 8.1), we also collected samples from our volunteers for all 6 gestures in two randomly chosen orientations of 134° and 314° at P_3. Next, we first obtained virtual samples corresponding to the 134° and 314° orientations using real samples collected in the 44° orientation, and then evaluated the real samples collected in the 134° and 314° orientations using the corresponding virtual samples. Figure 8.3(a) plots WiAG's aggregate accuracy across all gestures for each test orientation using a black bar and accuracy for individual gestures using patterned and grey-scale bars. We observe from this figure that WiAG achieves an aggregate accuracy of at least 89.9% across six gestures when trained and tested in different orientations. Figure 8.3(b) plots the accuracies when we repeat the same set of experiments, except that we use the samples in the 44° orientation directly without applying our translation function. We observe from this figure that the accuracy drops significantly to 47.62% and 53.26% for the two test orientations. This result has two implications: 1) change in orientation indeed severely deteriorates the accuracy of WiFi based gesture recognition, and 2) our translation function is very effective

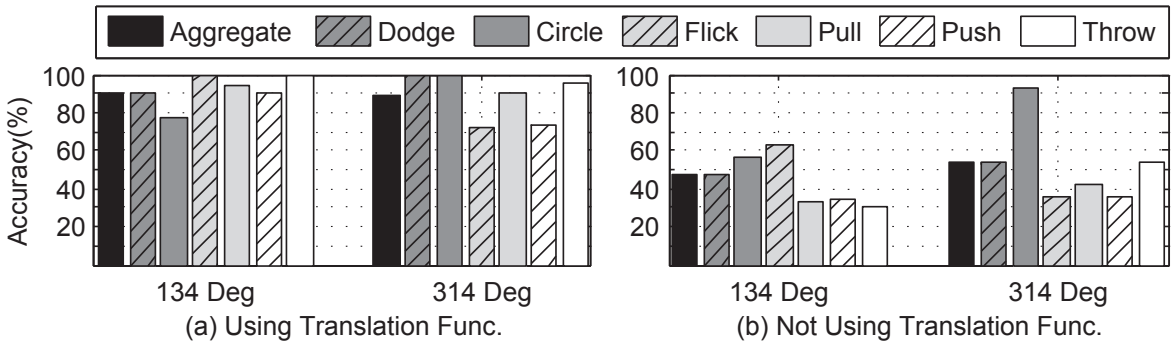


Figure 8.3 Accuracy with change in orientation

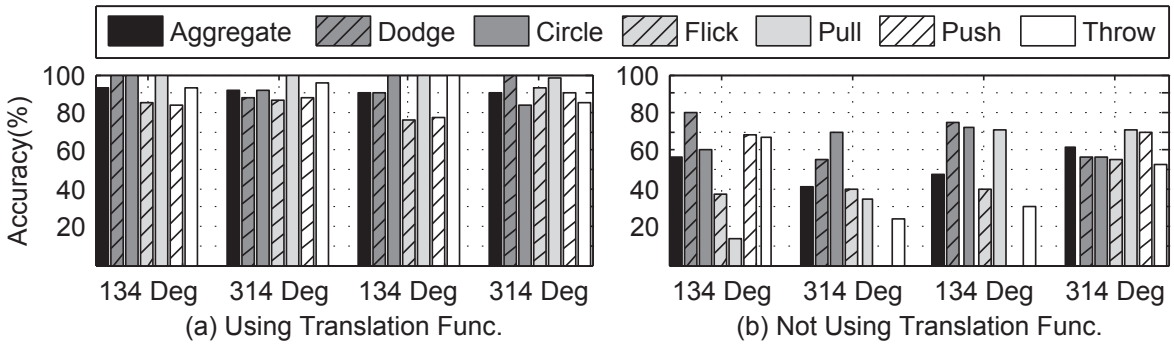


Figure 8.4 Accuracy with change in orientation & position

in making WiFi based gesture recognition agnostic to orientation.

Accuracy with Change in Orientation and Position: For this set of experiments, we used the samples collected at position P_3 in 44° orientation and generated virtual samples corresponding to configurations at positions P_1, P_2, P_4 and P_5. We then evaluated the real samples collected at each of these four positions using the virtual samples generated for the corresponding configurations and applying the k-NN classifier. Our experimental results show that WiAG achieves an average accuracy of 91.4% in recognizing the six gestures in all 4 different configurations. Note that this accuracy is comparable to that reported by prior

(a) Using translation func.

(b) Not using translation func.

Figure 8.5 Confusion matrix across all configurations

WiFi based gesture and activity recognition systems [Abd15; Pu13; Kel14; Han14; Wan14b; Wan15; Adi15b; Adi14; Adi15a; AK13; VDG08], which train and test in same configurations. Figure 8.4(a) plots WiAG's aggregate accuracy across all gestures in each configuration using a black bar and accuracy for individual gestures using patterned and grey-scale bars. We see in this figure that WiAG achieves an aggregate accuracy of no less than 90.6% across the six gestures. Figure 8.5(a) shows the confusion matrix for the six gestures across all four configurations. We see that WiAG achieves the highest average accuracy of 96% for the Push gesture.

Figures 8.4(b) and 8.5(b) show results from the same set of experiments as conducted to generate Figures 8.4(a) and 8.5(a), respectively, except that we used the samples at P_3 in 44° orientation directly without applying our translation function. From Figure 8.4(b), we observe that even the highest aggregate accuracy among the 4 configurations is just 61.24%, which is significantly less than the minimum aggregate accuracy of 90.6% when using the translation function. We observe similar trends on comparing Figures 8.5(a) and 8.5(b). From Figure 8.5(b), we also observe that 43% of push gestures are recognized as pull because without taking orientation into account, push gesture in one orientation and pull

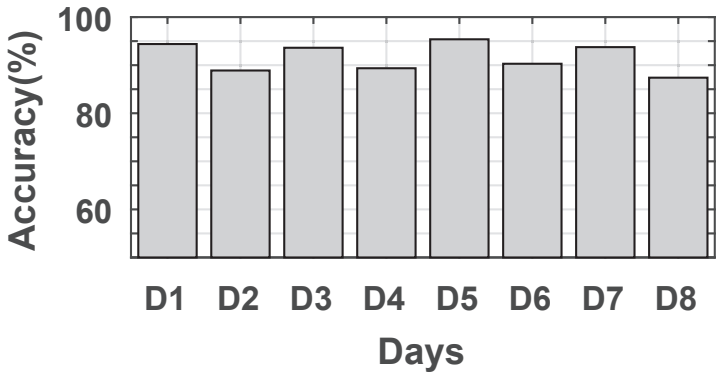


Figure 8.6 Effect of environmental changes on accuracy

gesture in another can appear very similar to the receiver. These observations show that our translation function indeed makes WiAG agnostic to both position and orientation.

8.3 Effect of Environmental Changes

In this section, we evaluate the performance of WiAG when the number and position of stationary objects in an environment change across samples, such as moving or adding a chair. Recall from Chapter 8.1 that when collecting samples, we randomly moved furniture each day before collecting samples on that day. For this set of experiments, for each day in our data set, we evaluated the samples collected on that day using the samples collected on all other days and applying the k-NN classifier. Figure 8.6 shows the resulting average accuracy of WiAG across all gestures for each day in our data set. We observe from this figure that the accuracy of WiAG does not change significantly across days, which shows that the static environmental changes do not affect WiAG. WiAG achieves this property due to the differentiation step of Eq. (3.8).

CHAPTER

9

CONCLUSION & FUTURE WORK

In this thesis, we proposed WiAG, a theoretically grounded position and orientation agnostic gesture recognition system. The key novelty of WiAG lies in its *translation function* that can generate virtual samples of any gesture in any desired configuration, and in its *configuration estimation scheme* that can estimate the position and orientation of user with respect to the receiver. The key technical depth of WiAG lies in its parametric formulation of the model for calculating changes in CFR due to linear and non-linear human gestures. This parametric formulation lies at the the heart of WiAG's translation function and configuration estimation

scheme. We implemented WiAG using cheap, commercially available, commodity WiFi devices and demonstrated that when user's configuration is not the same as at the time of collecting training samples, using our translation function, WiAG significantly improves the gesture recognition accuracy from just 51.4% to 91.4%.

9.1 Future Work

The WiFi based Gesture Recognition system needs to be optimized further. The proposed scheme works in a rather quiet environment, when there are only minor variations other than the ones caused by human gestures, and it is able to recognize gestures when only one person is performing the gesture. When two or more users perform the gestures simultaneously, CSI values recorded by the receiver show only the net change, resulting from all the gestures. It, thus, becomes necessary to propose a mechanism which can realize the contribution of each user in order to make such a system viable in home-setups.

While various schemes are available to quantitatively separate mixed signals, the problem is not trivial, primarily because there is no underlying property (for example, frequency), that remains constant for each person. Although the gesture-streams are unique for each of them, they are entirely dependent on the trajectory of gestures, and the speed with which they are performed, and does not depend on the person performing the gesture. This implies that there is no underlying constant factor that could separate the streams when two person perform the gestures simultaneously.

Interestingly, there are schemes to recognize the signals arriving from different directions, generated at different sources. These schemes, though, rely on signals arriving to be uncorrelated. Since each signal traverse a different path, which changes as the user

performs a gesture, it becomes possible to separate the variations resulting from different configurations, and consequently, identify the individual gestures. We envision that WiAG will help the research on WiFi based gesture recognition to step into its next phase of evolution where it will become more ubiquitous, pervasive, and practical.

BIBLIOGRAPHY

[Abd15] Abdelnasser, H. et al. *WiGest: A Ubiquitous WiFi-based Gesture Recognition System*. 2015. arXiv: 1501.04301.

[AK13] Adib, F. & Katabi, D. "See Through Walls with WiFi!" *SIGCOMM Comput. Commun. Rev.* SIGCOMM '13 **43**.4 (2013), pp. 75–86.

[Adi14] Adib, F. et al. "3D Tracking via Body Radio Reflections". *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*. NSDI'14. Seattle, WA: USENIX Association, 2014, pp. 317–329.

[Adi15a] Adib, F. et al. "Capturing the Human Figure Through a Wall". *ACM Trans. Graph.* **34**.6 (2015).

[Adi15b] Adib, F. et al. "Multi-person Localization via RF Body Reflections". *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*. NSDI'15. Oakland, CA: USENIX Association, 2015, pp. 279–292.

[Adi15c] Adib, F. et al. "Smart Homes That Monitor Breathing and Heart Rate". *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 837–846.

[AM11] Aggarwal, J. K. & Michael, S. R. "Human activity analysis: A review." *ACM Computing Surveys* **43**.3 (2011).

[Ali15] Ali, K. et al. "Keystroke Recognition Using WiFi Signals". *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. MobiCom '15. Paris, France: ACM, 2015, pp. 90–102.

[Smaa] *Allure Smart Thermostat*. https://www.allure-energy.com/.

[Che00] Cheung, G. K. et al. "A real time system for robust 3D voxel reconstruction of human motions". *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 714–720.

[Ert11] Ertin, E. et al. "AutoSense: unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field". *Proceedings of ACM Sensys*. 2011.

[Gje14] Gjengset, J. et al. "Phaser: Enabling Phased Array Signal Processing on Commodity WiFi Access Points". *Proceedings of the 20th Annual International Con-*

ference on Mobile Computing and Networking. MobiCom '14. Maui, Hawaii, USA: ACM, 2014, pp. 153–164.

[Hal11] Halperin, D. et al. "Tool Release: Gathering 802.11n Traces with Channel State Information". *ACM SIGCOMM CCR* **41**.1 (2011), p. 53.

[Han14] Han, C. et al. "WiFall: Device-free fall detection by wireless networks". *Proceedings of IEEE INFOCOM*. 2014, pp. 271–279.

[Her00] Herda, L. et al. "Skeleton-based motion capture for robust reconstruction of human motion". *Computer Animation 2000. Proceedings*. IEEE. 2000, pp. 77–83.

[Smab] *Honeywell Lyric Thermostat*. `http://wifithermostat.com/Products/Lyric/`.

[Smac] *Honeywell Smart Thermostat*. `http://wifithermostat.com/Products/WiFiSmartThermostat/`.

[Hua14] Huang, D. et al. "Feasibility and limits of wi-fi imaging". *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM. 2014, pp. 266–279.

[Ins] *Insteon LED Bulbs*. `http://www.insteon.com/led-bulbs/`.

[Kel14] Kellogg, B. et al. "Bringing Gesture Recognition to All Devices". *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*. NSDI'14. Seattle, WA: USENIX Association, 2014, pp. 303–316.

[Lan96] Lang, M. et al. "Noise reduction using an undecimated discrete wavelet transform". *IEEE Signal Processing Letters* **3**.1 (1996), pp. 10–12.

[Lem11] Lemmey, T. et al. *System architecture and methods for distributed multi-sensor gesture processing*. US Patent App. 13/210,370. 2011.

[Lgs] *LG Smart Appliances*. `http://www.lg.com/us/discover/smartthinq/thinq`.

[Lyo10] Lyonnet, B. et al. "Human gait classification using microdoppler time-frequency signal representations". *2010 IEEE Radar Conference*. IEEE. 2010, pp. 915–919.

[Moe06] Moeslund, T. B. et al. “A survey of advances in vision-based human motion capture and analysis”. *Computer vision and image understanding* **104**.2 (2006), pp. 90–126.

[Nes] *Nest Thermostat*. `https://nest.com/thermostat/meet-nest-thermostat/`.

[Oca09] Ocak, H. “Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy”. *Expert Systems with Applications* **36**.2 (2009), pp. 2027–2036.

[Opr12] Oprisescu, S. et al. “Automatic static hand gesture recognition using tof cameras”. *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE. 2012, pp. 2748–2751.

[Par11] Park, T. et al. “E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices”. *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2011, pp. 260–273.

[Phi] *Philips Hue*. `http://www2.meethue.com/en-us/`.

[Pu13] Pu, Q. et al. “Whole-home Gesture Recognition Using Wireless Signals”. *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*. MobiCom ’13. Miami, Florida, USA: ACM, 2013, pp. 27–38.

[Sec13] Seco, F. et al. “Joint estimation of indoor position and orientation from RF signal strength measurements”. *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*. IEEE. 2013, pp. 1–8.

[Sen13] Sen, S. et al. “Avoiding multipath to revive inbuilding WiFi localization”. *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM. 2013, pp. 249–262.

[Sho13] Shotton, J. et al. “Real-time human pose recognition in parts from single depth images”. *Communications of the ACM* **56**.1 (2013), pp. 116–124.

[Sin15] Singh, G. et al. “Inviz: Low-power personalized gesture recognition using wearable textile capacitive sensor arrays”. *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*. IEEE. 2015, pp. 198–206.

[Smad] *Smart Home*. http://www.smarthome.com/.

[Smo] *Smoothing Spline Matlab*. https://www.mathworks.com/help/curvefit/smoothing-splines.html.

[Sun15] Sun, L. et al. "WiDraw: Enabling Hands-free Drawing in the Air on Commodity WiFi Devices". *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. MobiCom '15. Paris, France: ACM, 2015, pp. 77–89.

[Tza01] Tzanetakis, G. et al. "Audio analysis using the discrete wavelet transform". *Proc. Conf. in Acoustics and Music Theory Applications*. 2001.

[VDG08] Van Dorp, P. & Groen, F. "Feature-based human motion parameter estimation with radar". *IET Radar, Sonar & Navigation* **2**.2 (2008), pp. 135–145.

[Wan14a] Wang, G. et al. "We Can Hear You with Wi-Fi!" *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: ACM, 2014, pp. 593–604.

[Wan15] Wang, W. et al. "Understanding and Modeling of WiFi Signal Based Human Activity Recognition". *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM. 2015, pp. 65–76.

[Wan16] Wang, W. et al. "Gait recognition using wifi signals". *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM. 2016, pp. 363–373.

[Wan14b] Wang, Y. et al. "E-eyes: Device-free Location-oriented Activity Identification Using Fine-grained WiFi Signatures". *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: ACM, 2014, pp. 617–628.

[Whi] *Whirlpool Smart Appliances*. http://www.whirlpool.com/smart-appliances/.

[Wif] *WiFi Plug*. http://www.wifiplug.co.uk/.

[Xi14] Xi, W. et al. "Electronic Frog Eye: Counting Crowd Using WiFi". *Proceedings of IEEE INFOCOM*. 2014.

[Yan13] Yang, Z. et al. “From RSSI to CSI: Indoor localization via channel response”. *ACM Computing Surveys (CSUR)* **46**.2 (2013), p. 25.

[YT12] Yatani, K. & Truong, K. N. “Bodyscope: a wearable acoustic sensor for activity recognition”. *Proceedings of ACM UbiComp*. 2012, pp. 341–350.