USING RADIO FREQUENCY AND MOTION SENSING TO IMPROVE CAMERA SENSOR SYSTEMS

Shiwei Fang

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2021

Approved by:

Shahriar Nirjon

Ron Alterovitz

Ketan Mayer-Patel

Jasleen Kaur

Sirajum Munir

ABSTRACT

Shiwei Fang: Using Radio Frequency and Motion Sensing to Improve Camera Sensor Systems
(Under the direction of Shahriar Nirjon)

Camera-based sensor systems have advanced significantly in recent years. This advancement is a combination of camera CMOS (complementary metal-oxide-semiconductor) hardware technology improvement and new computer vision (CV) algorithms that can better process the rich information captured. As the world becoming more connected and digitized through increased deployment of various sensors, cameras have become a cost-effective solution with the advantages of small sensor size, intuitive sensing results, rich visual information, and neural network-friendly. The increased deployment and advantages of camera-based sensor systems have fueled applications such as surveillance, object detection, person re-identification, scene reconstruction, visual tracking, pose estimation, and localization. However, camera-based sensor systems have fundamental limitations such as extreme power consumption, privacy-intrusive, and inability to see-through obstacles and other non-ideal visual conditions such as darkness, smoke, and fog.

In this dissertation, we aim to improve the capability and performance of camera-based sensor systems by utilizing additional sensing modalities such as commodity WiFi and mmWave (millimeter wave) radios, and ultra-low-power and low-cost sensors such as inertial measurement units (IMU). In particular, we set out to study three problems: (1) power and storage consumption of continuous-vision wearable cameras, (2) human presence detection, localization, and re-identification in both indoor and outdoor spaces, and (3) augmenting the sensing capability of camera-based systems in non-ideal situations. We propose to use an ultra-low-power, low-cost IMU sensor, along with readily available camera information, to solve the first problem. WiFi devices will be utilized in the second problem, where our goal is to reduce the hardware deployment cost and leverage existing WiFi infrastructure as much as possible. Finally, we will use a

low-cost, off-the-shelf mmWave radar to extend the sensing capability of a camera in non-ideal visual sensing situations.

To my family

ACKNOWLEDGEMENTS

The journey toward Ph.D. is not without ups and downs. The joy of accepted publications and the frustration of failed experiments are just a small glimpse into what makes graduate life memorable. While the Ph.D. degree is the reward, it's the people, events, laugh, and arguments most fulfilling, and I am lucky to have each and everyone along the way.

I am indebted to my supervisor Professor Shahriar Nirjon for his constant guidance throughout my Ph.D. His support has made this journey possible, and his encouragement to find and tackle new problems has trained me to be an independent researcher.

I am extremely grateful to all of my committee members: Prof. Ron Alterovitz, Prof. Ketan Mayer-Patel, Prof. Jasleen Kaur, and Dr. Sirajum Munir. Their feedback and suggestions have been very beneficial for my dissertation. I highly appreciate their guidance and encouragement on this journey.

I am also thankful to Dr. Sirajum Munir from Bosch Research and Technology Center for providing me the opportunity to conduct research in industry settings which had a profound impact on my future endeavors.

I would also like to thank all the present and past staff members of the Department of Computer Science at UNC. Their dedicated work and support are crucial to the success of my graduate life.

Moreover, I would like to thank my parents, Bilin Fang and Lin Tao, for all their sacrifices and hard work to help me in pursuing my dream. I am also thankful to all of my friends who encouraged me along the way, no matter where they are in the world.

Finally, I want to express my appreciation for my friends: Frances Hanlon, her daughter Carina, and her parents and relatives, who are like family to me in the United States. Their en-

couragement, hospitality, Christmas getaways, and fireside chats have guided me to stay true to myself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AM	Amplitude Modulation
AoA	Angle of Arrival
CABAC	Context-Adaptive Binary Arithmetic Code
CAVLC	Context-Adaptive Variable Length Code
CFR	Channel Frequency Response
CMOS	Complementary Metal Oxide Semiconductor
CTS	Clear to Send
CSI	Channel State Information
CW	Continuous Wave
DCT	Discrete Cosine Transform
FFT	Fast Fourier Transform
FM	Frequency Modulation
FMCW	Frequency Modulated Continuous Waveform
GSM	Global System for Mobile Communications
HEVC	High Efficiency Video Coding
IEC	International Electrotechnical Commission
IF	Intermediate-Frequency
IoT	Internet of Things
ISM	Industrial, Scientific, and Medical
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
JTC1	Joint Technical Committee
JVT	Joint Video Team

LTE	Long-Term Evolution
LTF	Long Training Symbol
LoRa	Long Range (a low-power Wide-area network modulation technique)
mmWave	Millimeter Wave
MIMO	Multiple Input Multiple Output
MPEG	Moving Picture Experts Group
MUSIC	Multiple Signal Classification
NFC	Near-Field Communication
OFDM	Orthogonal Frequency-Division Multiplexing
PD	Proportional–Derivative
PID	Proportional–Integral–Derivative
QP	Quantization Parameter
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SAR	Synthetic Aperture Radar
TDOA	Time Difference of Arrival
VCEG	Video Coding Experts Group

CHAPTER 1: INTRODUCTION

1.1 Overview

Accurate, low-power, low-cost, and unobstructed sensing systems are crucial in the push towards a connected and digitized world. We recognize that significant advancements in camera hardware and computer vision algorithms have accelerated the widespread deployment of cameras in numerous indoor and outdoor applications. With the improvement in the capability of the CMOS and reduction in cost, camera-based sensor systems have become ubiquitous in today's sensing systems. Cameras provide rich visual information, and the produced image is very similar to what humans see; thus, camera-produced images contain high-resolution information of the environment and are easy to interpret. These camera characteristics make them suitable for various tasks such as surveillance [79], object detection [148], person re-identification [74], scene reconstruction [112], scene understanding [10], visual tracking [111], pose estimation [49], and localization [83]. These capabilities put camera-based sensors in numerous systems such as body cameras, mobile robots, and human motion sensing input devices.

While cameras excel in many situations, they come with their limitations. For example, the power consumption of camera sensors is exceptionally high, and their performance is severely degraded in harsh conditions such as rain, fog, snow, and smoke. In situations like sensing and tracking objects behind walls and obstacles, cameras can not capture any information at all. As cameras provide rich visual data, they are privacy-invasive – which also limits their usage in many real-world environments. These limitations deteriorate the performance of camera-based sensor systems in the tasks mentioned previously. Literature has shown that by pushing the boundary of cameras hardware and algorithms, we can successfully reduce power [96, 28, 30], see around the corner [124, 65, 66], and see through fog [104]. However, while they push the

boundary of camera-based sensor systems, they introduce additional drawbacks as well. For example, the proposed systems have a higher computation load, require more expensive hardware, and their prerequisites are not suitable for practical scenarios.

To address these limitations, one can explore different sensors to complement the camera. For example, camera traps utilize low-power and low-cost motion sensors to trigger the camera for photos or video. This combination of sensors is very effective in the wild environment application. The motion sensor acts as a gate sensor to control the power-hungry camera's usage, thus significantly prolonging the whole system's operation time. However, the utilization of motion sensors does not work in applications requiring continuous vision, or the system is constantly moving. The motion sensor will always be triggered and no energy reduced. Therefore, the research community proposed designs to utilize different combinations of low-powered sensors (such as infrared sensors and low-resolution monochrome cameras) to control when it's worth the power consumption of the main camera sensor [88]. This approach mitigates the camera movement limitations but introduces high overhead in both computation and energy consumption. It also can not provide true continuous vision as it's only recording intermediately. As such, new methods and designs are required to enable continuous vision while minimizing overhead.

WiFi and WiFi-enabled smartphones are becoming ubiquitous in recent years as they are being adopted worldwide. With this rapid adoption and how the WiFi signal is being affected by the environment, literature has explored using WiFi signals to perform human sensing, such as presence detection [37] and indoor localization [23]. While camera-based systems generally achieve better performance in these tasks, they are also privacy-intrusive and require dense deployments, which can be difficult and costly. Moreover, for re-identification over a longer period of time, camera-based systems are becoming less useful as facial recognition is being banned in multiple cities and states. It is also expected to be banned in more places in the future. On the other hand, WiFi devices have no additional hardware deployment overhead with their ubiquitous presence. They are also privacy-aware with no human biometric information collected. New methods and algorithms are needed to exploit the advantages of the high accuracy of camera-based sensor

systems and ubiquity and privacy-aware of WiFi devices. We should also explore where we can provide additional features with already deployed WiFi systems.

Camera-based sensor systems are generally easy to work with and stable, but they can be interfered with environmental subjects, such as bad lighting, fog, and smoke. Radio frequency (RF) can penetrate and work where the camera fails. However, ubiquitous RF systems such as WiFi usually have a very low resolution due to their narrow bandwidth and lower frequency. As a result, they are unsuitable for many tasks. Higher frequency systems such as mmWave radar can provide higher resolution and penetrating capabilities, but they come at the cost of hardware complexity. The high hardware complexity is one of the reasons why they are not widely adopted, as higher complexity usually means higher cost. While cheap mmWave radars do exits, they are of lower resolution with limited application scenarios. To make robotic systems truly ubiquitous, we need to improve the performance of lower-cost mmWave radar systems and complement the usage of camera-based sensor systems. Thus, we need to explore methods that can increase the resolution while being computationally inexpensive.

In this dissertation thesis, we select a set of key problems surrounding camera sensing systems that will improve the efficiency as well as the capability of widely deployed camera sensors in various indoor and outdoor spaces. Specifically, we focus on three key research challenges. First, we propose a lightweight algorithm to extract features already embedded in the encoded video domain and from a low-power IMU to tackle the high power consumption problem in continuous vision systems such as a smart eyeglass or a body camera. Second, by exploiting the characteristics of WiFi signal propagation, we propose several methods to enable localization, tracking, re-identification, and detection with ubiquitous WiFi systems. These systems can be deployed with existing WiFi infrastructures and provide privacy to the end users. Third, we propose a two-stage algorithm to enhance the sensing capability of mmWave radars, which can be used jointly with camera-based sensor systems in non-ideal situations such as bad lighting conditions, smoke, and occlusion.

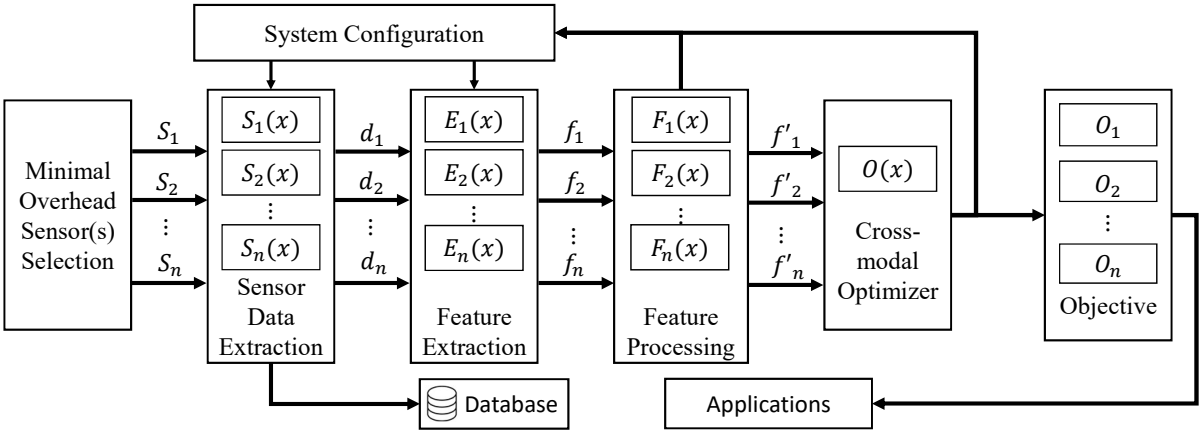


Figure 1.1: Proposed general framework to design a system that can achieve minimal overhead while improve performance.

1.2 General Framework

To efficiently design such systems, I propose a general framework as shown in Figure 1.1 to be the blueprint of my system designs. At the beginning of each system design, the system's objectives are identified: system properties – such as reduced power – or desired sensing results – such as location or high-resolution occupancy grid. Once the objectives are identified, the sensors with minimal overhead are chosen. This overhead is not only about the quantity of the sensors but also the added energy consumption, computational overhead, and added cost. The design of the system is intertwined with what types of features can be extracted from each sensor. These features are generally complemented to each other based on their strength and weakness. Once we defined the sensors and objectives, the following steps are to design the algorithms needed in each stage to perform *Sensor Data Extraction*, *Feature Extraction*, and *Feature Processing*. These algorithms can be independent and unique to different sensors. Still, the results of the *Feature Processing* should be more generalized and can perform *Cross-modal Optimization* if required. The results are the objectives identified at the beginning of the design and intended for the application. The objectives can also be used to control the system itself by providing information that can be used to calculate system control parameters.

1.3 Thesis Statement

"Through the addition of low-power IMU, ubiquitous WiFi devices, or low-cost mmWave radar, vision-based sensor systems can be more energy-efficient, have better identification accuracy, provide better tracking and localization, protect users' privacy, and enable extended sensing capabilities for richer information gathering in future sensing systems."

1.4 Contribution List

We present a brief list of contributions of my thesis:

- A method which exploits the underlying features in the encoded video domain that can be used to infer the dynamics of the scene. Combining these features with a low-power IMU sensor, we proposed a standalone movable camera system that enables continuous vision while saving power through a lightweight algorithm.
- A neural network model and multiple algorithms that can generate WiFi Angle of Arrival (AoA) swiftly and be used in cross-modal matching with cameras to provide high accuracy tracking, localization, and re-identification. This method is privacy-aware, low-cost, and ubiquitous. We have collected a large dataset in indoor environments.
- An insight and method on how to localize a user in a complex mobile system where the WiFi signal is severely distorted. We also collected a large unique dataset in the automotive setting.
- A method to explore a set of features that can be utilized to detect human presence in around the corner scenario, which is a non-line-of-sight situation and can not be achieved through a camera-based sensor system.
- A two-stage method to enhance the resolution of the signal from a mmWave radar. This two-stage method can benefit from a neural network model and a compressed sensing algorithm that mitigates their limitations.

1.5 Thesis Organization

The reminder of the thesis is organized as follows:

- In Chapter 2, we provide background on sensing modalities and techniques we exploited to provide better sensing and related works on how some of the modalities are utilized to perform targeted sensing tasks.
- In Chapter 3, we present ZenCam, a body camera system that is designed to have a continuous vision while reducing power and storage consumption. This system is implemented to show that we can exploit the already present underlying features in the encoded video domain and a low-power IMU sensor to analyze how the quality of the videos is being recorded. This quality analysis can be used to dynamically adjust the camera systems to produce videos that are consistent with human perception while reducing power and storage consumption. Furthermore, it can also result in smaller packaging and lower cost.
- In Chapter 4, we present our EyeFi system, which can be used for tracking, localization, and re-identification in indoor environments. This system demonstrates how the WiFi can be used in combination with camera-based sensor systems to provide advantages from both sensors while mitigating drawbacks from both. In addition, we proposed new algorithms to perform fast Angle of Arrival (AoA) estimation and matching between WiFi and camera data for higher accuracy.
- In Chapter 5, we present CarFi, a system that can localize a person who is holding a phone in an automotive environment. We provide insight into this unique setting and how some limitations and interference can be mitigated and utilized. This system is beneficial to automobiles with WiFi capability to localize where the passenger is and improve the experience of both the driver and passenger.
- In Chapter 6, we demonstrate a system that utilizes widespread WiFi access points to detect human presence in around-the-corner situations, which are occluded to the camera-based

systems. This system shows that by choosing the right features, WiFi can provide human presence data, and such data can be used to provide better indoor safety, especially for robots.

- In Chapter 7, we present SuperRF. This system utilizes mmWave signal and deep learning techniques to provide higher resolution radio reflections for 3D reconstruction and obstacle detection. This is a two-stage system that the deep learning results are further improved with compressed sensing techniques. Furthermore, the proposed mmWave system can be used in conjunction with cameras to provide sensing information during both ideal environment and non-ideal environment as mmWave are not being affected by lighting conditions and occlusions.
- In Chapter 8, we conclude the thesis and provide a summary and possible future directions based on this thesis.

CHAPTER 2: BACKGROUND

2.1 Introduction

In this chapter, we discuss the background of our dissertation work. This chapter starts with the background on video encoding and controllers. These are the foundations of the ZenCam work to reduce energy and storage consumption. Then we discuss the common properties of Radio Frequency (RF) sensing, which are applicable to both WiFi and mmWave technique we utilized in this dissertation. After the background on general RF sensing, individual properties of WiFi and mmWave radar are discussed. Finally, we discuss some techniques that can improve the sensing results from WiFi and mmWave.

2.2 Video Encoding

Video cameras produce videos by taking consecutive images (frames). However, if a video is composed of only images, the size of the videos can easily reach Gigabytes (GB) within a magnitude of several minutes. Video coding, sometimes referred to as video compression, reduces video data size by reducing spatial and temporal redundancies [78] [20]. Spatial redundancy reduction is for structural similarity within a single image, and temporal redundancy reduction is for similarity across different frames.

2.2.1 Video Compression Standards

While video codecs are software or hardware implementations that can encode and decode videos, video coding formats are the specifications on how the codec can be implemented. There are numerous video coding formats such as H.262 (MPEG-2 Part 2), MPEG-4 Part 2, H.264

(MPEG-4 Part 10), HEVC (H.265), Theora, RealVideo RV40, VP9, and AV1. These formats are standardized video compression algorithms, which are usually based on discrete cosine transform (DCT) [17] and motion compensation [90]. Video coding formats such as H.264 and HEVC are royalty-based, which can be expensive. The license fee is also one of the reasons that HEVC is slow to adopt. Some video coding formats such as VP9 and AV1 are royalty-free and open source, and especially the AV1 format is developed by the Alliance of Open Media (AOM). The list of companies in the alliance includes Amazon, Apple, ARM, Facebook, Google, Intel, Microsoft, Nvidia, etc. Such alliance indicates that the industry moves away from the royalty-based model and pushes for wider adoption of open source solutions.

2.2.2 Motion Compensation

For most videos, the difference between adjunct frames is minimum as the time interval between two consecutive frames is small (e.g., 24 frames per second results in 0.0417 seconds interval). This fact can be utilized to reduce the file size by reusing the information from adjacent frames. A common technique called motion-compensated prediction (MCP) [116] [92] is used in video codecs to reduce such temporal redundancy. Motion prediction generally works on macroblocks [29], in which the size is defined in different codecs. The motion vector (MV) [52] is used to identify the best match for such a macroblock in the current frame with another in the reference frame(s) [20]. The search of the motion vector is done through motion estimation (ME). The motion vector can be noisy and does not always show the true motion within the video as it tries to match the best macroblock. The best match is to ensure that the two macroblocks from the current frame and another from the reference frame generate the minimum residual. The residual is the difference between the predicted frame using only the motion vectors and the actual frame. Examples of the motion vector and residual are shown in Figure 2.1. As shown in the figure, motion vectors have a higher magnitude with a lower frame rate as the time interval between consecutive frames is higher; thus, the distance moved by the object is large

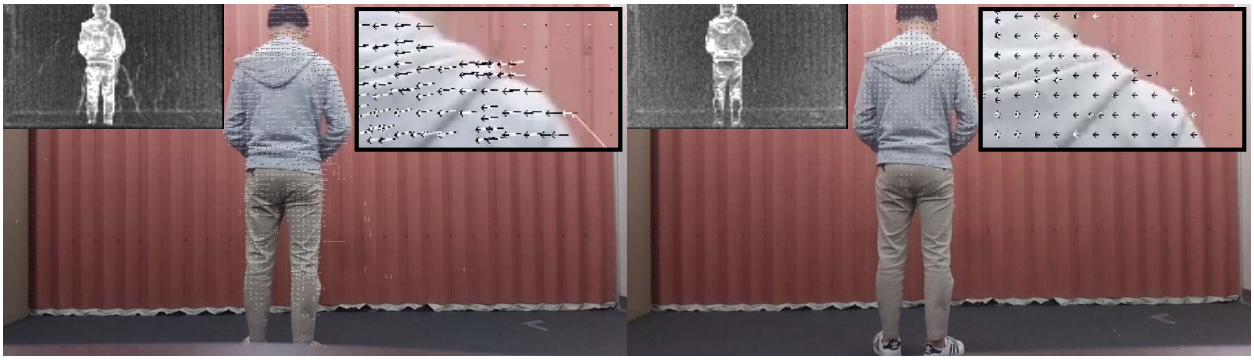


Figure 2.1: Motion vectors and residual of the same scene with different frame rate. The left is recorded at a lower frame rate than the right one.

Motion vectors can be non-integer values as the block being shifted with sub-pixel precision, and interpolation is utilized to generate in-between pixels. Half-pixel or quarter pixel precision is commonly used, such as in H.264 [131] and HEVC [114]. Sub-pixel precision requires much higher computational expense as the extra process required for interpolation, and the number of potential blocks needs to be evaluated. Although more bits are required to encode the sub-pixel precision (i.e., non-integer), the overall compression efficiency can sometimes be achieved through better prediction.

2.2.3 Transform Coding and Quantization

We can apply Discrete Cosine Transform (DCT) [17] [138] to convert the residual data into the frequency domain. DCT works by express a finite sequence of data points into a sum of cosine functions. Cosine functions are used instead of sine functions as fewer cosine functions are needed to approximate a typical signal. After the conversion, residuals can be further compressed through quantization, which leads to lossy compression. The quantization is controlled by the quantization parameter (QP) [35], which affects the bit rate during the encoding process. Higher QP results in higher compression but lower quality.

2.2.4 Entropy Coding

To further reduce the video file size, entropy coding [73] can be utilized to replace data symbols with context-adaptive variable length code (CAVLC) [86] or context-adaptive binary arithmetic code (CABAC) [115]. CAVLC is much less computationally expensive than CABAC. However, it does not compress the data as effectively as CABAC. CAVLC is supported on all profiles, while CABAC is only supported in the Main and higher profiles in the H.264 video coding format. CABAC is used in all HEVC profiles. The entropy coding is a lossless compression as it only replaces the representation of the data.

2.2.5 H.264/AVC

H.264 or MPEG-4 Part 10 was standardized by a partnership effort known as the Joint Video Team (JVT). It is the collaboration between the ITU-T (International Telecommunication Union Telecommunication Standardization Sector) Video Coding Experts Group (VCEG) and the ISO (International Organization for Standardization)/IEC (International Electrotechnical Commission) JTC1 (joint technical committee) Moving Picture Experts Group (MPEG). H.264 is by far the most commonly used video coding format. It is widely used in recording, compression, and distribution of video content. This format supports up to and including 8K UHD (8192×4320). However, H.264 is more commonly supported up to 4K (4096 x 2160) at 60 frames per second.

2.2.6 H.265/HEVC

High Efficiency Video Coding (HEVC), also known as H.265 and MPEG-H Part 2 is developed by the same organizations behind H.264 and is a successor to it. HEVC offers from 25% to 50% better data compression when compared to H.264. Different from H.264/AVC, HEVC uses DCT and DST [106] transforms with four transform units (TUs) of sizes 4 x 4, 8 x 8, 16 x 16, and 32 x 32. It supports resolution up to 8K (8192 x 4320) and utilized CABAC entropy cod-

ing. The macroblock is also superseded by coded tree blocks (CTB) [85]. More intra prediction modes are permitted in HEVC compared to H.264/AVC as well.

2.2.7 Software vs. Hardware, and Power

Video encoding and decoding generally require significant computing power. While software and hardware implementations have the same compression efficiency, which depends on video algorithmic implementations, utilize hardware implementation is more power-efficient. There are drawbacks of hardware implementation as well. For example, the supported video coding formats is limited and can not be upgraded at a later stage, and hardware implementation generally does not support concurrent encoding in multiple formats, multiple bit rates and resolutions, and additional features such as advanced advertising. The flexibility of software implementation also shines as newer video coding format can be supported in older platforms as a software update. In comparison, it will require new hardware to be installed for hardware implementations, which is impossible in some scenarios, such as mobile devices.

The power consumption of hardware implementation can be significantly less than the software implementation, especially with Application-Specific Integrated Circuit (ASIC). In addition to the power efficiency, which can be viewed as higher processing performance, hardware implementation also provides lower latency. To address the disadvantage of flexibility, Field-Programmable Gate Array (FPGA) has been used to provide some flexibility of being re-programmable. However, they are higher cost and consume more power than ASIC. A combination of software and hardware can also be utilized, enabling trade-off implementations suitable for the desired applications.

2.2.8 Compressed Domain Analysis

Motion vectors and residual information are generally available in the compressed videos. Although they are intended to reduce the size of video files, information is stored in these parameters. These parameters are required to be preprocessed due to their noisy nature in order

to perform analysis in the compressed domain. Once denoised, the motion vectors are usually used to compute the intended applications [21]. For example, motion vectors have been used to determine human action based on the changes in motion vectors [121] [25]. Video classification, indexing, and retrieval can also utilize motion vectors as extra features [24] [84]. Object detection, tracking, and segmentation can be loosely performed in the compressed domain as well [63] [119]. For compressed domain analysis, features such as transform coefficients, macroblock partitions, color, and more can also be utilized to achieve better results for the intended applications.

2.3 Feedback Control

A feedback controller is to observe the output signals of the unit under control and to compute and apply the right control input signal to the unit. A standard feedback controller like PID [47] controller requires an accurate model of the system, which is extremely difficult to acquire due to the presence of *disturbance* as the system operates in an open, real-world environment, especially for the ones that involve human. *Adaptive controllers* [71] are a special kind of feedback controllers that can update the model to better describe the system.

2.3.1 PD Control

The on-off control method can be used in a basic control system where the requirement is very low. However, the controller output is not associated with the control input. Proportional control is the control output proportional to the control input, which can be expressed as:

$$P\ controller\ output = K_p \times controller\ input \tag{2.1}$$

where K_p is the *gain* constant. Suppose we note the difference between the desired system value (e.g., the desired room temperature) and the current system value (e.g., current room tem-

perature) as an error. In that case, one can design a derivative control where the control output is proportional to the rate of change of the error. This can be expressed as:

$$D\ controller\ output = K_d \times rate\ of\ change\ of\ error \tag{2.2}$$

The constant K_d is commonly referred to as *derivative time* as it has a unit of time. Derivative control can be a rapid corrective response to error signals (e.g., the room temperature dropping too fast). However, as it does not respond to a constant error, which is zero in the rate of change, derivative control D is combined with proportional control P, expressed as:

$$PD\ control\ output = K_p \times error + K_d \times rate\ of\ change\ of\ error \tag{2.3}$$

PD control performs better than P control with fast process changes as the rate of change of error is taken into consideration.

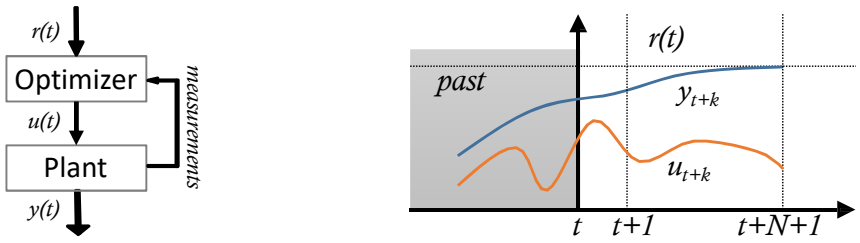
2.3.2 PID Control

Integral control is the control output is proportional to the integral of the error with respect to time, expressed as:

$$I\ control\ output = K_i \times integral\ of\ error\ with\ time \tag{2.4}$$

Integral control is generally not used alone, as it is more common in conjunction with proportional control. PI control has an advantage over P control is that the steady-state error can be eliminated. However, the integral process takes time, which can result in oscillations if the changes are relatively large.

Combining all three modes of control (proportional, integral, and derivative) can produce a system with no steady-state error and reduce the tendency for oscillations. PID (proportional-integral-derivative) controller is also called a three-model controller, and it can be expressed as:



(a) Model predictive controller overview.

(b) Model predictive controller input and output over time.

Figure 2.2: Model predictive controller.

$$PID\ control\ output = K_p \times error + K_i \times integral\ of\ error + K_d \times rate\ of\ change\ of\ error \tag{2.5}$$

While PID controllers can perform well in many applications, it does not provide optimal control. For more on PD and PID control, one can refer to [27].

2.3.3 Model Predictive Control

A *Model Predictive Controller* (MPC) [127] uses a dynamic model of the plant to predict its future outputs and optimizes the control signals by solving an optimal control problem over a finite future horizon. Figure 2.2 shows a model predictive controller and its inputs and outputs over time. The *plant* in the Figure 2.2a is the unit being controlled. Also shown in this figure that $y(t)$ is the output of the system, $r(t)$ is the reference output, and $u(t)$ is the control input. The shaded area on the left in Figure 2.2b denotes the past and the area on its right $(t, t+N)$ is the finite prediction horizon. At each time point t, predicted outputs and manipulated inputs are computed by the controller over the finite time horizon $t+0, \ldots, t+N$. However, only the first input $u(t)$ is applied to the system. Once reached time $t+1$, another set of N consecutive control inputs are being calculated and only the first control input is applied.

Model predictive controllers have several advantages, such as explicitly handle constraints on inputs and outputs, and the performance is optimized by solving an open-loop optimization problem.

Table 2.1: ITU designated radio frequency spectrum bands.

Frequency Range	Wavelength Range	ITU designation		IEEE bands
		Full Name	Abbr.	
Below 3 Hz	$>$ 105 km	Tremendously low frequency	TLF	N/A
3–30 Hz	105–104 km	Extremely low frequency	ELF	N/A
30–300 Hz	104–103 km	Super low frequency	SLF	N/A
300–3000 Hz	103–100 km	Ultra low frequency	ULF	N/A
3–30 kHz	100–10 km	Very low frequency	VLF	N/A
30–300 kHz	10–1 km	Low frequency	LF	N/A
300 kHz – 3 MHz	1 km – 100 m	Medium frequency	MF	N/A
3–30 MHz	100–10 m	High frequency	HF	HF
30–300 MHz	10–1 m	Very high frequency	VHF	VHF
300 MHz – 3 GHz	1 m – 10 cm	Ultra high frequency	UHF	UHF, L, S
3–30 GHz	10–1 cm	Super high frequency	SHF	S, C, X, Ku, K, Ka
30–300 GHz	1 cm – 1 mm	Extremely high frequency	EHF	Ka, V, W, mm
300 GHz – 3 THz	1 mm – 0.1 mm	Tremendously high frequency	THF	N/A

2.4 Radio Frequency (RF)

Radio frequency usually refers to the oscillation rate of the radio wave with a range lower than 300 GHz, which is roughly the lower limit of infrared frequencies. Infrared and radio waves are both electromagnetic radiation (so is visible light, ultraviolet, X-ray, and Gamma-ray). As they are the same, the radio wave also travels at the speed of light. Note that how the radio spectrum is being defined versus the range of infrared is a convention and arbitrary. The boundary (the boundary generally in the 300 GHz to 3 THz range) between these two can be different in different scientific fields.

2.4.1 Frequency

The radio spectrum of frequencies is divided into bands with conventional names assigned by the International Telecommunications Union (ITU), and the designation is shown in Table 2.1. The frequency above 1 GHz is conventionally called microwave, and 30 GHz - 300 GHz are designated millimeter wave. With a larger wavelength, radio waves are more widely used for

communications. This is due to the signal's propagation characteristics, as they can pass through the atmosphere, foliage, and most building materials. They also can be bend around obstructions and tend to be scattered rather than absorbed by objects larger than their wavelength. There are mainly three different propagation methods being used to communicate:

- **Ground Wave** When the radio frequency is lower than around 3 MHz, the radio wave travels effectively as ground wave, which allows them to follow the curvature of the Earth when they are parallel to and adjacent to the surface. This is due to their long wavelength, which is more strongly diffracted around obstacles. For example, AM radio stations can use Medium Frequency (MF) to broadcast, which can be received far away. Some other applications include over-the-horizon radar, maritime communications, and more.

- **Skywave** Radio waves with frequencies around the range of 3 MHz to 30 MHz can be reflected from the atmosphere's ionosphere layer. This property allows such radio waves to travel further and enable long-distance communications across large distances and mountainous terrains, which is not ideal for line-of-sight propagation. However, the stability of this propagation method can be greatly affected by the environment, for example, sunlight/darkness, season, solar activity, sunspot cycle, and more. Example applications in this range of frequencies include amateur radio and aviation communication.

- **Line-of-sight** For radio waves with a frequency higher than 30 MHz, it is usually transmitted through a straight line as the visible light does. Since higher frequency does not follow the curvature of the earth and is not reflected by the atmosphere's ionosphere, line-of-sight propagation normally has a theory limitation on the range of about 64 km (40 miles) as limited by the earth's shape. If the transmitter is placed high and the receiver is at sea level, the range can be extended as the earth's curvature allows it. Relays are required to enable long-distance communications such as cell phone towers and wired/wireless communication between those towers. These frequencies are used mostly for short-range communications

such as cell phones, WiFi, Bluetooth, and more. On the lower frequency of this range, the radio wave can pass through buildings, foliage, and other obstructions.

2.4.2 Propagation Loss

When radio waves are propagated through the environment, their power density will reduce even in free space. The propagation loss is the result of many factors not limited to:

- **Free Space Loss** When the signal travels through space, it will spread out. As a result, the power density of the signal will reduce.
- **Diffraction** Radio waves lose their power when they diffract around objects along their path. The losses are higher when diffraction happens around more rounded objects than sharp edges.
- **Multipath** There are usually multiple obstacles in the real environment. Thus, the signals will be reflected and then reach the receiver via several different paths. They may add or subtract each other based on their relative phases.
- **Absorption Losses** Radio waves pass through a different medium and will be absorbed. For example, furniture, buildings, and walls will attenuate the radio signal and more severe for higher frequencies. Atmospheric elements such as moisture will absorb the radio signals as well. Vegetation also introduces large attenuation, especially when they are wet.

To illustrate the radio wave attenuation due to the earth's atmosphere around ground level, Figure 2.3 are generated using Matlab with ITU atmospheric gas attenuation model for frequency from 1 GHz to 300 GHz. The condition simulated are with 15°C, $7.5g/m^3$ water vapor density (around 58% relative humidity), and an atmospheric pressure around $101.3kPa$. Note that the y-axis is in log scale with the unit of (dB/km). It is shown in this figure that the atmospheric loss is in the general trend of increase as frequency increases, especially in the more realistic wet condition ($7.5g/m^3$) versus the dry condition ($0g/m^3$). There is an increase in the atmospheric loss

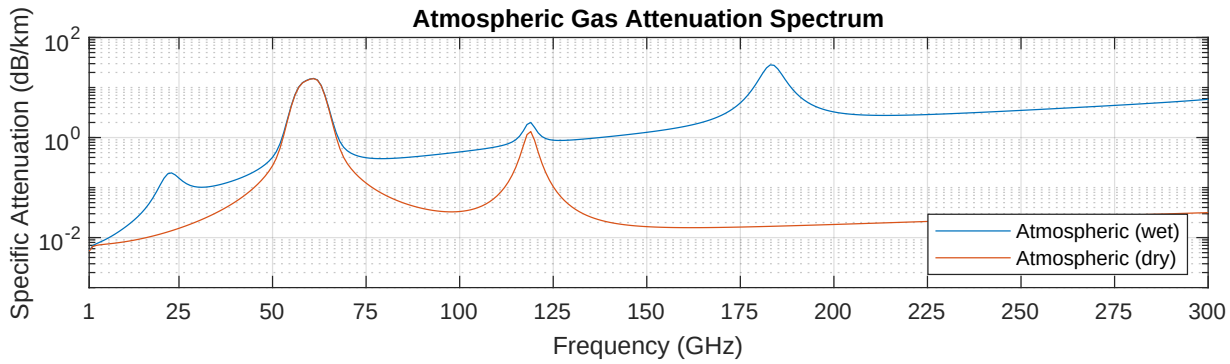
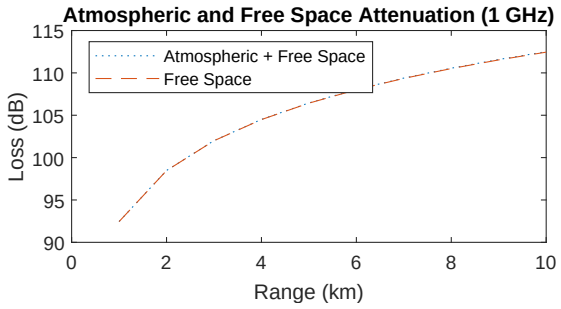


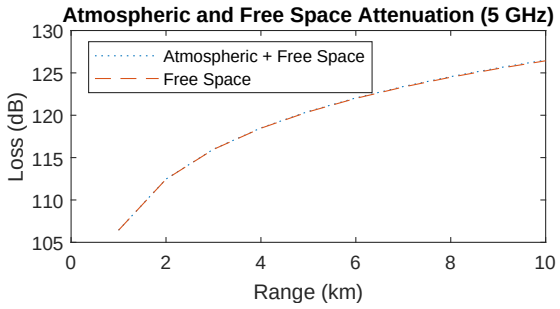
Figure 2.3: Atmospheric gas attenuation spectrum from 1 to 300 GHz.

around 50 GHz to 75 GHz in both dry and wet conditions, which is due to the oxygen absorption in those frequencies. A smaller increase of around 24 GHz is also observable due to the water vapor. As water vapor and oxygen presence decrease as altitude increases, such attenuation will reduce when the altitude is high.

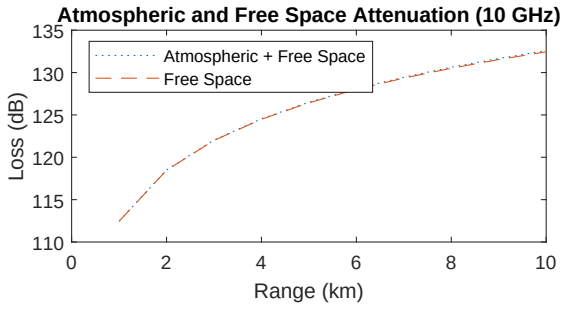
Some examples of atmospheric and free space attenuation at different frequencies versus distance are shown in Figure 2.4. From these figures, it is clearly shown that the signal loss increases as the frequency increases. The effects from the atmosphere are more severe with higher frequency as well. The dB loss is logarithm, and a 3dB loss means half of the power level, and 10 dB loss is a ten-fold decrease in signal level. As a result of the signal loss characteristics, lower frequencies are better to be used for communication transmission. However, higher frequency normally can provide larger bandwidth, which can significantly increase the transmission throughput. The choice of which frequency to use is not purely based on the signal loss as well. For example, the 60 GHz range, while suffering a huge loss due to oxygen absorption, can be used for high-speed communication with the availability of huge bandwidth, and regulations usually allow for high transmission power to offset the high power losses. As it is limited in the transmission range, it also opens up the possibility of reuse the same frequency without interference. With a working distance of 2 km for the 60 GHz frequency band, the distance between adjacent links can be just separated by around 4 km with minimal interference. In contrast, a 55 GHz with a working distance of around 5 km requires an 18 km distance between links to minimize the interference [82].



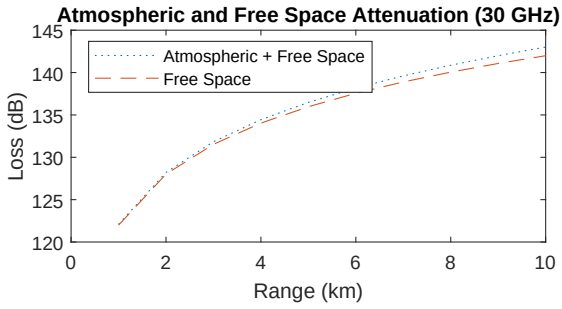
(a) 1 GHz



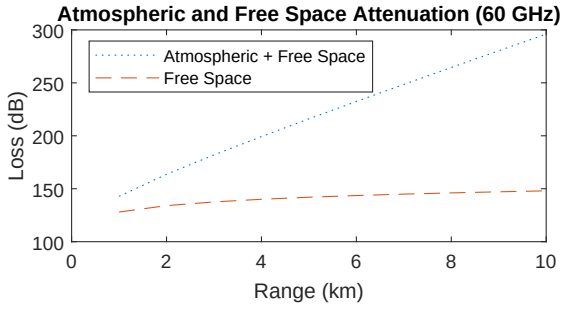
(b) 5 GHz.



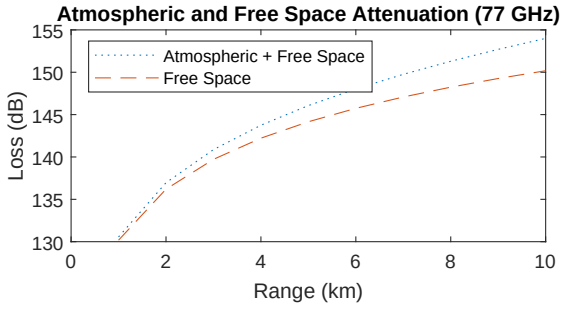
(c) 10 GHz



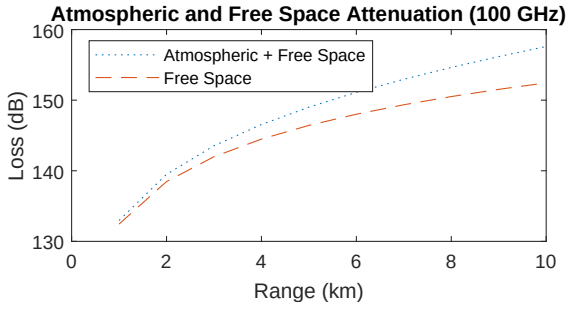
(d) 30 GHz.



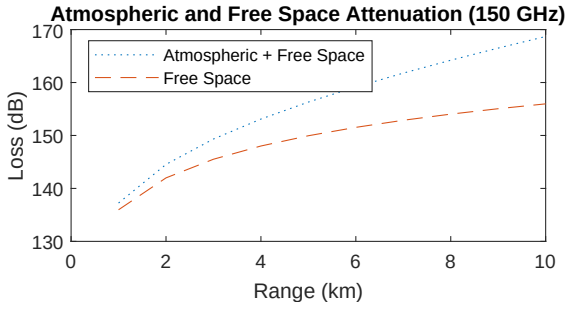
(e) 60 GHz



(f) 77 GHz.



(g) 100 GHz



(h) 150 GHz.

Figure 2.4: Atmospheric gas attenuation in atmosphere and free space for distance from 1 km to 10 km.

How the radio waves pass through obstacles are related to their frequency and the materials of the obstacle. This is related to the Skin depth of materials. This is usually calculated as:

$$\delta = \sqrt{\frac{\rho}{\pi f_0 \mu_r \mu_0}} \tag{2.6}$$

where δ is the skin depth, ρ is the resistivity, f_0 is the signal frequency, μ_r is the relative permeability, and μ_0 is the permeability of free space. From this equation, the skin depth is inversely related to the frequency, which means as frequency increases, the penetration capability decreases. In reality, the penetration capability is much complex and relates to other factors as well. For example, the wall's material and depth, a hollow wooden wall allows much better transmission than solid concrete ones. This is also the reason that 2.4 GHz WiFi has better indoor coverage and signal strength than 5 GHz, and 60 GHz bands are used for close-range high throughput scenarios such as virtual reality.

2.4.3 Communication Protocol and Regulations

To prevent interference with each other, radio frequencies are tightly governed by governments around the world. The radio spectrum allocations depend on numerous factors such as the characteristics of the frequencies, i.e., which frequency bands are suitable for the intended applications, priority of different applications, e.g., military applications have higher priority than consumer ones. As radio frequencies are limited, multiple technologies and protocols may share the same frequency. For example, both WiFi and Bluetooth utilizes the 2.4 GHz spectrum. In such cases, the power regulation, how to detect interference, and how to mitigate interference is part of the regulation and protocol design. Usually, communication-related signal power limitations are much smaller than those used in radar systems. Such limitation is to mitigate commercial radio frequency usage interference on radar systems such as weather, aviation, and defense. With these regulations, one can not use an arbitrary frequency but need to follow the rules closely.

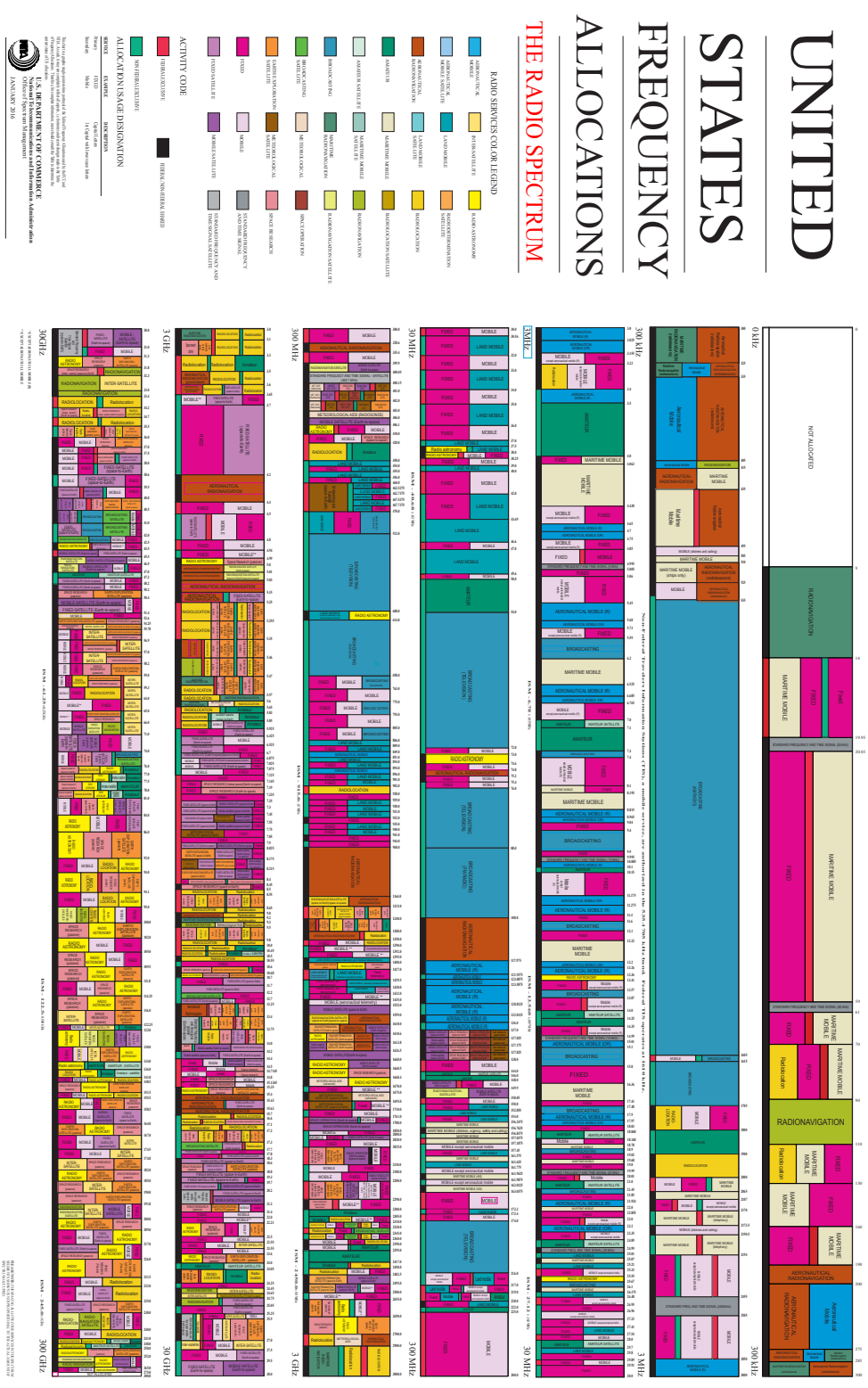


Figure 2.5: The 2016 radio spectrum chart of the United States frequency allocations. [118]

Besides regulated radio frequencies that would require a government license (for the transmitter) to operate in, e.g., GSM, LTE, TV broadcast, AM, FM, there are radio spectrum reserved for industrial, scientific, and medical (ISM) purposes other than telecommunications. An example application in these bands is the microwave oven. For wireless communication systems that utilize these frequencies, they need to tolerate the interference of other devices. As the available bands are limited and sparse, wireless communication systems such as WiFi, Bluetooth, cordless phones, near-field communication (NFC), garage door, remote control, and etc., are required to be low power and cooperate in these frequencies. With the limitation on transmission power, the range of such wireless systems is usually limited. For example, Bluetooth is around 10 m, WiFi is in the hundred-meter range, and lower frequency such as LoRa can go for several kilometers as the path loss is much smaller. How the radio spectrum is allocated in the US can be seen in Figure 2.5. The full-size detailed chart is available on their website. These allocations are ever-changing to meet the requirements of multiple entities.

2.4.4 Angle of Arrival (AoA)

When antennas receive an RF signal, the signal is coming from a certain direction. The angle of Arrival (AoA) is used to describe this direction. How the AoA is normally defined is shown in Figure 2.6. The circular dots represent the antennas. The dashed line is the coordinate system of the antenna array. The vertical dash line, which is perpendicular to the antenna array, is the boresight of this array. Boresight is the axis of maximum gain, and for most antennas, the boresight is also the axis of symmetry. Suppose the antennas shown in the figure are omnidirectional antennas. In that case, signals coming from either side and mirroring along the antenna array (i.e., top or bottom in the figure) will have the same AoA.

To calculate the AoA, the time difference of arrival (TDOA) is usually utilized. As the distance traveled by the signal is different for each antenna, as shown in the figure, the time will also be different. However, as RF signals traveling at the same speed as light, the time difference is

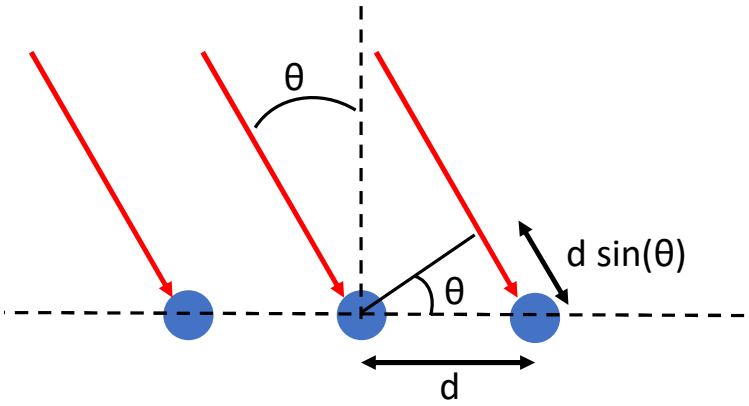


Figure 2.6: Angle of arrival of RF signals are measured as the angle between the signal and the boresight of the antenna array.

usually too small to be effectively measured by the hardware. As a result, the phase difference between the received signals is generally used to estimate the AoA.

2.4.5 Antenna Placement

Antennas are mostly arranged in a linear array with equal space between each adjacent antenna. The distance between the antennas (d) is determined by the frequency (f) of the system operates in. For a linear antenna array, as shown in Figure 2.6, the difference in distance traveled by the signal to two adjacent antennas is $dsin(\theta)$, with θ being the angle of arrival.

For the angle of arrival to be measured unambiguously, the distance d between antennas needs to be smaller than the wavelength λ of the signal. For example, if the distance $d = \lambda + \Delta l$, and the phase difference ω of the received signals meets:

$$0 < \frac{\lambda\omega}{2\pi} < \Delta l \tag{2.7}$$

it's unknown that if the phase difference is just ω, which is equal to the difference caused by $dsin(\theta)$ or the phase difference is actually $\lambda + \omega$, which is a different angle of arrival. Thus the distance between antennas needs to be smaller than the wavelength of the signal.

When the distance between antennas is smaller than the wavelength, the angle of arrival is usually estimated from phase difference (ω) between two received signals:

$$\omega = 2\pi \frac{dsin(\theta)}{\lambda} \tag{2.8}$$

$$\theta = sin^{-1}(\frac{\omega\lambda}{2\pi d}) \tag{2.9}$$

where λ is the wavelength of the signal. Because the phase difference ω can be uniquely estimated only from $-\pi$ to π, we can substitute the ω with π. Thus, the maximum unambiguous field of view (FOV) of the antenna array can be calculated with:

$$\theta_{FOV} = \pm sin^{-1}(\frac{\lambda}{2d}) \tag{2.10}$$

The maximum θ_{FOV} is achieved with $d = \lambda/2$, which results in $\theta_{FOV} = \pm 90°$. To achieve theoretical maximum field of view, antennas are commonly placed $\lambda/2$ apart with λ being the wavelength, and $\lambda = c/f$, where c is the speed of light and f is the frequency.

2.5 WiFi

WiFi is wireless communication technology. WiFi devices are defined as any "Wireless Local Area Network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards". This IEEE 802.11 is a set of medium access control (MAC) [53] and physical layer (PHY) [140] specifications for implementing Wireless Local Area Network (WLAN) communication. While the standards are universally adopted, the radio frequency spectrum may vary from country to country.

Table 2.2: IEEE 802.11 PHY Standards

Standard	Frequency Band	Bandwidth	Modulation	Advanced Antenna Technologies	Maximum Data Rate
802.11	2.4 GHz	20 MHz	DSSS, FHSS	N/A	2 Mbits/s
802.11b	2.4 GHz	20 MHz	DSSS	N/A	11 Mbits/s
802.11a	5 GHz	20 MHz	OFDM	N/A	54 Mbits/s
802.11g	2.4 GHz	20 MHz	DSSS, OFDM	N/A	542 Mbits/s
802.11n	2.4 GHz, 5 GHz	20 MHz, 40 MHz	OFDM	MIMO, up to 4 spatial streams	600 Mbits/s
802.11ac	5 GHz	20 MHz, 40 MHz, 80 MHz, 160 MHz	OFDM	MIMO, MU-MIMO, up to 8 spatial streams	6.93 Gbits/s

2.5.1 WiFi Standards

An overview of some of the IEEE 802.11 PHY standards is shown in Table 2.2. Earlier standards such as 802.11a/b/g are limited in terms of performance. They are only operated with limited channels and channel bandwidth. 802.11n improves dramatically over previous standards. It incorporates advanced signal processing and modulation techniques at the physical layer (PHY) to exploit multiple antennas and wider channels. At the media access control (MAC) layer, protocol extensions are more efficient. 802.11n also adds multiple-input and multiple-output (MIMO) and 40MHz channels. MIMO enables data can be simultaneously transmitted and received. Antenna configurations from 1x1 to 4x4 are defined. 802.11ac (also known as WiFi 5) builds on top of 802.11n with wider bandwidth (all 802.11ac devices are required to support 20, 40, and 80 MHz channels), more MIMO spatial streams (up to 8), multi-user MIMO, and higher density modulation.

The IEEE 802.11 standards cover the wireless networks' protocols and operations. These standards only control the two lowest layers, the physical layer and data link layer, of the Open Systems Interconnection Reference Model (OSI Model) [46]. The general theme is that the medium access control (MAC) or data link layer of all the 802.11 standards to be compatible while only the physical layer (PHY) differs.

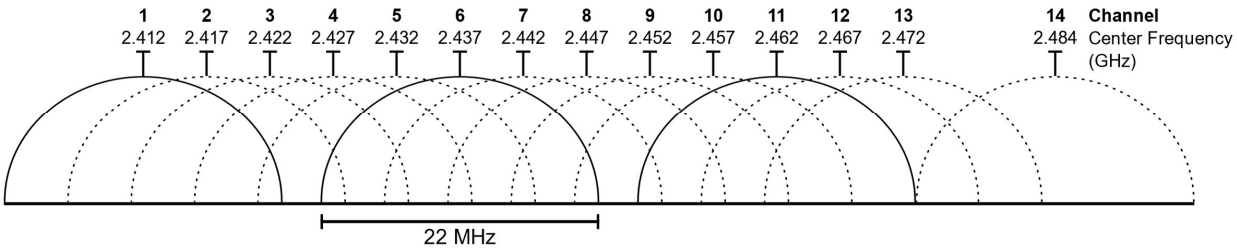


Figure 2.7: WiFi channels of the 2.4 GHz band. Image source: Wikipedia.

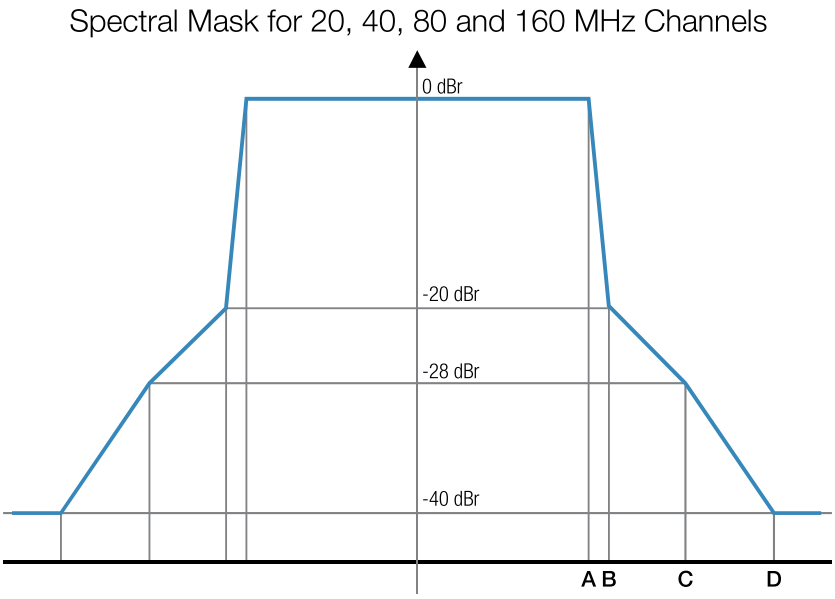
2.5.2 WiFi Channels

Radio frequency is utilized to perform wireless communication, and spectrum allocation is a vital part of the infrastructure. Several radio frequency ranges are used for WiFi communications and most notably the 2.4 GHz and 5 GHz frequency bands. Each spectrum is sub-divided into channels with a center frequency and bandwidth. Channels are numbered at 5 MHz spacing within a band (except 60 GHz band). Transmission is generally occupied at least 20 MHz, and newer standards allow channels to be bonded together to form a wider channel, which provides higher throughput. For example, WiFi channels and availability of 2.4 GHz band are shown in Figure 2.7 and Table 2.3 respectively. Although WiFi standards allow such channels to be used, the availability, maximum power levels, and etc., are subject to each country's regulations and changed throughout time.

5 GHz band are generally available from 5.250 to 5.350 GHz and 5.470 to 5.725 GHz. However, the regulations on the 5 GHz band vary a lot from country to country, and the numbering is less intuitive. More frequency bands are being used by different WiFi standards, such as the sub-gighertz band (900 MHz) for 802.11ah, which is designed for the Internet of Things (IoT) and support long-range and low-power communications but currently lack commercially available hardware. 60 GHz for 802.11ad/ay standards (also known as WiGig) provides fast internet speed but a limited transmission range, which is more targeted to applications such as virtual reality. There is only very limited hardware support to 802.11ad standard, and 802.11ay standard is still finalizing. Newer WiFi standards such as 802.11ax (WiFi 6E) can utilize some of the 6 GHz band. These regulations are either finalizing or expect to finalize in the near future.

Table 2.3: 2.4 GHz WiFi channel availability. Source: Wikipedia.

Channel	F_0 (MHz)	Frequency Range (MHz)	North America	Japan	Most of world
1	2412	2401–2423	Yes	Yes	Yes
2	2417	2406–2428	Yes	Yes	Yes
3	2422	2411–2433	Yes	Yes	Yes
4	2427	2416–2438	Yes	Yes	Yes
5	2432	2421–2443	Yes	Yes	Yes
6	2437	2426–2448	Yes	Yes	Yes
7	2442	2431–2453	Yes	Yes	Yes
8	2447	2436–2458	Yes	Yes	Yes
9	2452	2441–2463	Yes	Yes	Yes
10	2457	2446–2468	Yes	Yes	Yes
11	2462	2451–2473	Yes	Yes	Yes
12	2467	2456–2478	No except CAN	Yes	Yes
13	2472	2461–2483	No	Yes	Yes
14	2484	2473–2495	No	11b only	No



Channel Size	A	B	C	D
20 MHz	9 MHz	11 MHz	20 MHz	30 MHz
40 MHz	19 MHz	21 MHz	40 MHz	60 MHz
80 MHz	39 MHz	41 MHz	80 MHz	120 MHz
160 MHz	79 MHz	81 MHz	160 MHz	240 MHz

Figure 2.8: OFDM spectral mask used for 802.11a/g/n/ac. [117]

The 802.11 standards also specify the spectral mask, which defines how the power distribution across each channel is permitted. The mask for 802.11a/g/n/ac are shown in Figure 2.8. Note that while the channel size is 20 MHz, 40 MHz, 80 MHz, and 160 MHz, the highest power is permitted with a small bandwidth, and some power is still allowed outside of the band width. This means that some parts of the nearby channels' energy can still interfere with other channels, albeit with minimal interference if the standards are met.

2.5.3 Received Signal Strength Indicator (RSSI)

Received Signal Strength Indicator (RSSI) is a measurement of the power present in a received radio signal. RSSI is usually derived in the intermediate frequency (IF) stage before the IF amplifier and is a relative index. The 802.11 standards do not specify detailed rules regarding RSSI, and each manufacturer can define their own max value. There is no defined relationship between the RSSI value to the actual power level in milliwatts or decibels referenced on one milliwatt (dBm), and each vendor and manufacture can define their own accuracy, granularity, and range for the actual power. As a result, it is usually not accurate to compare RSSI between different chips, but a higher value with the same chip means a better signal. The RSSI value can also be used internally to determine when the network card is clear to send (CTS).

Different from RSSI, which measures the received power in preamble only, Received Channel Power Indicator (RCPI) is introduced to measure the received RF power in the selected channel for the received frame. It is also defined that the RCPI shall equal the received RF power within an accuracy of ± 5 dB (95% confidence interval) within the specified dynamic range of the receiver.

2.5.4 Channel State Information (CSI)

With newer IEEE 802.11 standards support Multiple-Input and Multiple-Output (MIMO) to increase the performance and stability of the WiFi systems, WiFi chipsets measure the channel at the Orthogonal Frequency-Division Multiplexing (OFDM) subcarrier level. The result is re-

Table 2.4: CSI carriers reported based on carrier grouping. [60]

BW	Grouping	Ns	Carriers for which matrices are sent
20 MHz	1	56	All data and pilot carriers: –28, –27,. . . –2, –1, 1, 2,. . . 27, 28
	2	30	–28,–26,–24,–22,–20,–18,–16,–14,–12,–10,–8,–6,–4,–2,–1, 1,3,5,7,9,11,13,15,17,19,21,23,25,27,28
	4	16	–28,–24,–20,–16,–12,–8,–4,–1,1,5,9,13,17,21,25,28
40 MHz	1	114	All data and pilot carriers: –58, –57, . . . , –3, –2, 2, 3,. . . , 57, 58
	2	58	–58,–56,–54,–52,–50,–48,–46,–44,–42,–40,–38,–36,–34,–32,–30, –28,–26,–24,–22,–20,–18,–16,–14,–12,–10,–8,–6,–4,–2, 2,4,6,8,10,12,14,16,18,20,22,24,26,28, 30,32,34,36,38,40,42,44,46,48,50,52,54,56,58
	4	30	–58,–54,–50,–46,–42,–38,–34,–30,–26,–22,–18,–14,–10,–6, –2, 2,6,10,14,18,22,26,30,34,38,42,46,50,54,58

ported in a standard Channel State Information (CSI) format, as shown in Table 2.4. In this table, grouping means the method used to reduce the size of the CSI report by reporting a single value for each group of N adjacent subcarriers. Ns is the number of subcarriers reported.

As CSI measures the subcarrier level, the information recorded is much granular than RSSI. The properties of such channels are a description of the combined effect of the scattering, fading, and power decay between the transmitter and receiver. Each CSI entry represents the Channel Frequency Response (CFR):

$$H(f;t) = \sum_{n}^{N} a_n(t) e^{-j2\pi f \tau_n(t)} \tag{2.11}$$

where $a_i(t)$ is the amplitude attenuation factor, $\tau_i(t)$ is the propagation delay, and f is the carrier frequency [122]. Since the signals between the transmitters and receivers are impacted by the environment and the displacement and movement of the antennas themselves, the CSI amplitude $|H|$ and phase $\angle H$ captures the wireless characteristics of the environment.

Before the data symbols are transmitted, the transmitter will send Long Training Symbols (LTFs), containing pre-defined symbols for each subcarrier. For each subcarrier, the signal is modeled as $y = Hx + n$, where y is the received signal, x is the transmitted signal, and n is the noise. H is the CSI matrix, as shown before. With the received LTFs and the known pre-defined

LTFs that are transmitted, the chipset can estimate the matrix H. As a result, the reported CSI values are not the most accurate as it is also affected by other factors within the chipset. In recent studies, researchers have found that CSI can be used for WiFi sensing, including but not limited to: angle of arrival estimation, human detection, breath detection, and etc.

2.5.5 MUSIC and Variants

For the angle of arrival (AoA) estimation, one classical method is the MUSIC algorithm [108]. If two antennas are separated d apart, the additional phase shift introduced due to the distance is $-2\pi \times d \times \sin(\theta) \times f/c$, where θ is the AoA, f is the signal frequency, and c is the speed of light. The MUSIC algorithm works by estimating the steering matrix A in: $X = AF$, where X is the measurement matrix of the received signal, and F is the matrix of complex attenuation. In a recent WiFi-based localization algorithm [69], the AoA of the direct path (which is relevant to the localization problem) is isolated by taking the eigenvector of the matrix, XX^H, for which, the eigenvalue is zero. The eigenvector goes through further processing to obtain the direct path.

2.6 mmWave Radar

In recent years, mmWave radars have become popular in the automotive industry, and currently, they are being used in applications such as advanced cruise control, driver monitoring, and autonomous driving. A mmWave radar is a detection and ranging system that operates in the frequency spectrum between 30 GHz and 300 GHz, corresponding to 10 mm to 1 mm in wavelength (e.g., a 77 GHz mmWave radar has a wavelength of 3.9 mm). Transmitted signals from these radars have forms including *Continuous Wave* (CW), *Frequency Modulated Continuous Waveform* (FMCW), and *pulsed*.

2.6.1 Frequency Modulation

A continuous wave (CW) radar is one of the simplest forms of radars. It continuously transmits radio signals at a constant high frequency. CW radar is usually used to detect speed and change in the environment based on the Doppler effect. A CW radar can not differentiate multiple reflecting objects as there is no information on the angle of the reflected signal. Some CW radar applications include traffic control radar (speed gauges), motion detection, and military uses. A simple continuous wave radar is cheap to make as the components needed are less, and the circuit is simple.

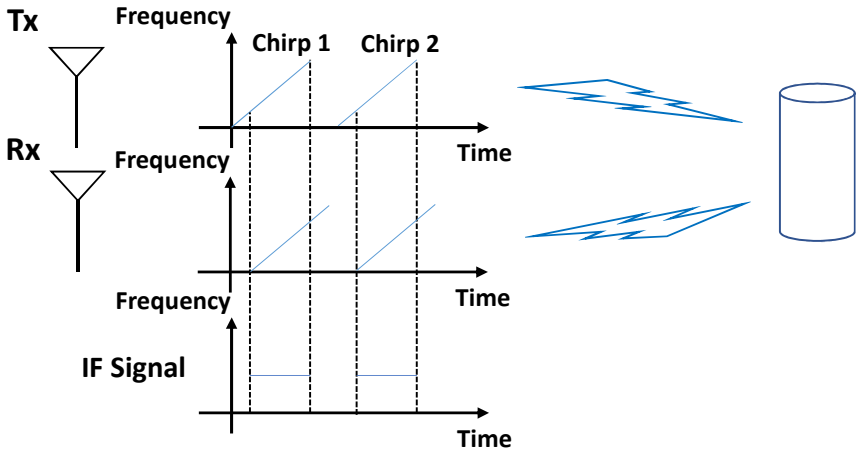


Figure 2.9: Operational principle of a mmWave radar.

Figure 2.9 illustrates a Frequency-Modulated Continuous Wave (FMCW) radar in action. Unlike a CW radar, FMCW transmits frequency modulated radio signals by changing its operating frequency during measurement, which the frequency can be increasing or decreasing periodically. The Tx antenna transmits a sequence of *chirps* – a frequency ramp over a short duration. When the signal hits an object, some of the reflected signals are captured by the Rx antenna, which has the same characteristics as the transmitted ones but is delayed in time. For example, a FMCW radar transmits a radio signal of an increasing frequency from f_0 to f_1 periodically in a static environment, and the received signal will also be a radio signal (less power as some being absorbed, pass through, scattered) with frequency from f_0 to f_1 periodically. However, the received signal will have a time delay Δt, which is equal to the time needed for the single to travel to the object

and back to the radar. As a result, the difference between the transmitted and the received signals of a FMCW radar is a constant frequency – namely the *Intermediate-Frequency* (IF) signal. The IF signal has a linear relation with the distance between the radar and the object, which can be used to detect the range.

2.6.2 Range Estimation

A FMCW radar can detect range by calculating the IF signal as described above. For a FMCW radar with a start frequency of f_0 and a bandwidth of B, if the radar can increasing the frequency from f_0 to $f_0 + B$ in a time duration of T_d, the slope of the radio signal (chirp) will be:

$$S = \frac{B}{T_d} \tag{2.12}$$

When the radar receives the reflected signal, the system has two signals, the one that the radar sends out (x_{radar}) and the one received (x_{object}):

$$x_{radar} = sin(\omega_1 t + \phi_1) \tag{2.13}$$

$$x_{object} = sin(\omega_2 t + \phi_2) \tag{2.14}$$

The IF signal is the difference between these two signals with instantaneous frequency, and phase is the instantaneous difference of these two:

$$x_{out} = sin[(\omega_1 - \omega_2)t + (\phi_1 - \phi_2)] \tag{2.15}$$

Since the transmitted and received signal is the same and delayed by a time (τ), the time is equal to the signal travel twice the distance d between the radar and the object:

$$\tau = \frac{2d}{c} \quad , \ c \text{ is the speed of light.} \tag{2.16}$$

As the instantaneous frequency of the IF signal is proportion to the time delay:

$$f_d = \frac{S2d}{c} \tag{2.17}$$

Thus we can derive the distance between the radar and object as:

$$d = \frac{f_d c}{2S} \tag{2.18}$$

When multiple objects are present in the scene, the IF signal is a combination of multiple frequencies. By applying Fourier transform, one can separate the multiple frequencies. For a given number of FFT bins, each bin corresponds to a distance, and the index of the bin is called the *range index*. Each bin contains information of all the measurements from that distance, which can be considered as an image of a slice in the 3D environment. Multiple such images can be combined to form a 3D representation of the scene, be discussed in Section 2.8. The higher the range resolution, the finer the 3D representation will be.

For the radar to distinguish two closely located objects, the objects need to be separated by an amount larger than the *range resolution*. Based on the Fourier transform theory, the observation window (T) can resolve frequency components that are separated by more than $1/T$ Hz. With Equation 2.17, the constraint can be expressed as:

$$\Delta f > \frac{1}{T_d} \tag{2.19}$$

$$\frac{S2\Delta d}{c} > \frac{1}{T_d} \tag{2.20}$$

$$\Delta d > \frac{c}{2ST_d} = \frac{c}{2B} \tag{2.21}$$

$$=> \quad d_{res} = \frac{c}{2B} \tag{2.22}$$

Where c is the speed of light, B is radar's bandwidth, T_d is the time duration needed to operate from f_0 to $f_0 + B$. For a mmWave radar that operates between 77GHz to 81GHz, the bandwidth is 4GHz, which results in a range resolution of around 3.75cm.

2.6.3 Angle Estimation

In order to localize or to image an object, a radar needs to calculate the angle of arrival (AoA) of the received signals. This is done by calculating the phase difference across different antennas caused by the difference in distance traveled from the object to the antennas. From Chapter 2.4.5, the angle can be estimated with phase difference ω as shown in Equation 2.9. The phase difference ω depends on $sin(\theta)$, which is a nonlinear dependency. $sin(\theta)$ is approximated with a linear function only when θ has a small value $sin(\theta) \sim \theta$. As a result, the estimation accuracy depends on the angle of arrival, and the smaller the angle, the better the accuracy.

For a linear antenna array with more than 2 antennas, for example, $N_{Rx} = 4$, each subsequent antenna will have an additional phase-shift of ω with respect to the proceeding antenna. Thus, for the four antenna array, the phase difference will be $[0\ \omega\ 2\omega\ 3\omega]$. This means the received signal has a spatial frequency:

$$\omega_1 = 2\pi \frac{dsin(\theta)}{\lambda} \tag{2.23}$$

for the phase difference across antennas. To separate multiple objects at that same range but at different angles, one can apply FFT to distinguish such frequencies. For an object with a $\theta + \Delta\theta$, is has a spatial frequency of:

$$\omega_2 = 2\pi \frac{dsin(\theta + \Delta\theta)}{\lambda} \tag{2.24}$$

Thus the difference between two spatial frequency of phase difference generated by these two objects will be:

$$\Delta\omega = \omega_2 - \omega_1 = \frac{2\pi d}{\lambda}(sin(\theta + \Delta\theta) - sin(\theta)) \tag{2.25}$$

Since the derivative of $sin(\theta)$ is $cos(\theta)$, Equation 2.25 can be approximated as:

$$\Delta\omega = \frac{2\pi d}{\lambda}(cos(\theta)\Delta\theta)) \tag{2.26}$$

For the N-point FFT to distinguish the spatial frequency differences, the FFT peaks need to be $2\pi/N$ away. The angle difference between two objects that can be distinguished is derived as:

$$\Delta\omega > \frac{2\pi}{N} \tag{2.27}$$

$$\frac{2\pi d}{\lambda}(cos(\theta)\Delta\theta)) > \frac{2\pi}{N} \tag{2.28}$$

$$\Delta\theta > \frac{\lambda}{Ndcos(\theta)} \tag{2.29}$$

From Chapter 2.4.5, the maximum field of view is achieved with the antennas are separated by $2/\lambda$, with antenna angular resolution is usually calculated at the bore-sight view (angle of arrival $\theta = 0$), the angular resolution is calculated as:

$$\theta_{res} = \frac{2}{N} \tag{2.30}$$

N here is the number of antennas. Note that the number of FFT bins can be larger than the number of antennas available through zero-padding, but this does not increase the resolution (angle between two objects that can be distinguished). Zero-padding FFT increases the resolution of each FFT bin, and you can get a smoother and sharper peak (i.e., each FFT bin only represents a smaller angle).

2.6.4 Velocity Estimation

From Equation 2.15, we know that the phase difference is the instantaneous phase difference between the transmitted and received signal. This difference is caused by small distances. The phase difference between multiple chirps can also be obtained, and such a difference can be used to measure the speed of the object. For the velocity of the object being v, we can get the phase difference as:

$$\Delta\phi = 2\pi\frac{2\Delta d}{\lambda} = \frac{4\pi\Delta d}{\lambda} \tag{2.31}$$

$$\Delta d = vT_d \tag{2.32}$$

$$=> \Delta\phi = \frac{4\pi vT_d}{\lambda} \tag{2.33}$$

The Δd is the distance difference of the distance between the object and radar T_d apart, and T_d is the time duration of a single chirp f_0 to $f_0 + B$ as before (we assume 0 time between two chirps). Thus, the speed of the object can be calculated as:

$$v = \frac{\lambda\Delta\phi}{4\pi T_c} \tag{2.34}$$

The measurement is unambiguous if $|\Delta\phi| < \pi$, combined with the equation above, the maximum relative speed (relative to the radar) can be measure by two chirps is:

$$v_{max} = \frac{\lambda}{4T_d} \tag{2.35}$$

When multiple chirps are transmitted in sequence, we can distinguish multiple objects with different speeds by obtaining the different phase difference using FFT. Similar to the angular resolution, the peak in the FFT bins needs to be $2pi/N$ away. With Equation 2.33, the resolution of the velocity can be derived as:

$$\Delta\omega = \omega_2 - \omega_1 > \frac{2\pi}{N} \tag{2.36}$$

$$\frac{4\pi v T_d}{\lambda} > \frac{2\pi}{N} \tag{2.37}$$

$$v > \frac{\lambda}{2NT_d} \tag{2.38}$$

The time period NT_d is defined as frame time (T_f), which is the time needed to transmit N consecutive chirps. The velocity resolution can be determined as:

$$v_{res} = \frac{\lambda}{2T_f} \tag{2.39}$$

2.6.5 MIMO Radar

To increase angular resolution, the most effective way is to increase the number of receiving antennas. However, instead of only increasing the number of receiving antennas, one can also add additional transmitting antennas to create a *multiple input multiple output* (MIMO) radar [26], one such configuration is shown in Figure 2.10. The top row shows an antenna configuration where two transmitting antennas are separated by 2λ and four receiving antennas are separated by $\lambda/2$. This configuration creates a virtual antenna array, shown in the bottom row of Figure 2.10, where the number of receiving antennas becomes $2 \times 4 = 8$ — which effectively doubles the angular resolution.

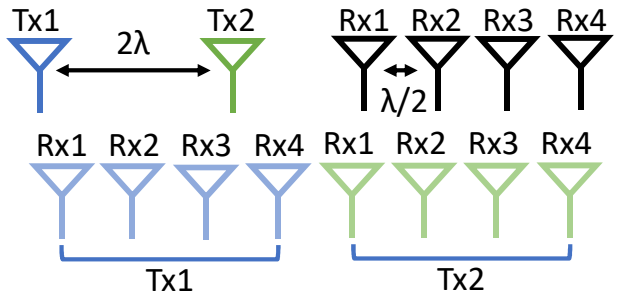


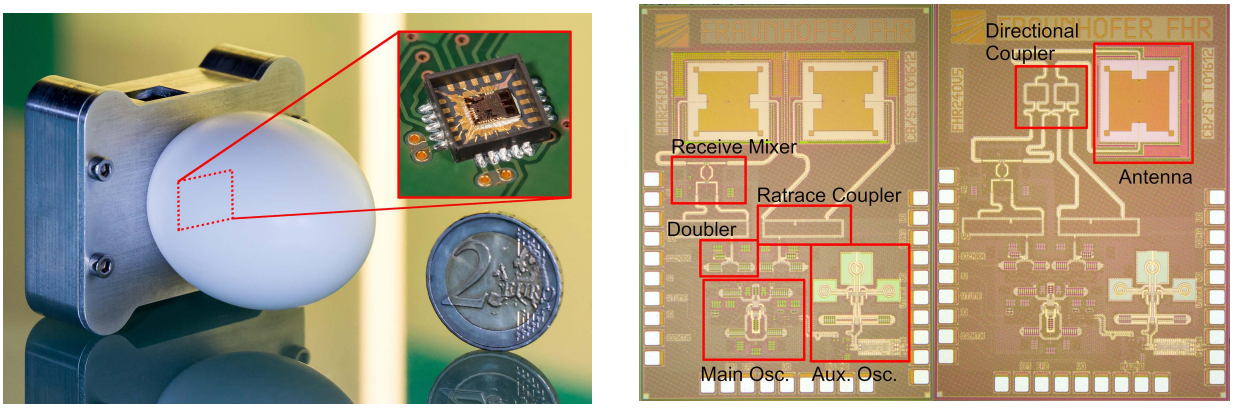
Figure 2.10: Illustration of a MIMO radar setup.

This MIMO configuration works because the distance traveled by the signal from Tx1 to receiving antennas and signals from Tx2 to receiving antennas are separated by $2\lambda sin(\theta)$ which is $4dsin(\theta)$, (since $d = \lambda/2$). The phase difference between the receiving antennas with the signal from the Tx1 is $[0\ \omega\ 2\omega\ 3\omega]$ and the phase difference between the Tx2 and receiving antennas with respect to Tx1-Rx1 pair is $[4\omega\ 5\omega\ 6\omega\ 7\omega]$. Thus one can apply the Fourier transform on the combined signal [Tx1-Rx1, Tx1-Rx2, Tx1-Rx3, Tx1-Rx4, Tx2-Rx1, Tx2-Rx2, Tx2-Rx3, Tx2-Rx4] to obtain the angles. As the angular resolution is determined by $2/N$, the antenna array with a virtual 8 antennas has twice the resolution of an antenna with 4 antennas.

Although increasing the number of antennas increases resolution, such a method requires the radar system to have a complex data processing system to handle a large amount of data. Furthermore, an increased number of transmit antennas means the system requires more time for each transmission antenna to complete the signal generation and transmit sequentially. Hence, large antenna arrays are infeasible for systems that require fast sensing or have a limited computational capability.

2.6.6 Hardware and Cost

High-performance radar systems are usually high cost, which is associated with multiple factors of such device. For example, the number of antennas determines the amount of information that needs to be processed. Also, a higher number of antennas require more complex circuit design and higher-end components to meet transmission needs and power delivery in a short amount of time. The frequency of the radar operates in is also contributes to the cost of the system. While automotive radar is around 77 GHz, which is the frequency band allocated to such applications, a higher frequency band around 240 GHz has also been studied for high-resolution imaging. A lower frequency (77 GHz) radar is easier to manufacture as the relative size of the antenna is larger than those of the high frequency (240 GHz). An example of the 240 GHz radar is shown in Figure 2.11. From this figure, the size of such radar is shown. As the size is small, the challenge to design and manufacture such a radar for a real-world environment (with multiple



(a) The radar system.

(b) Bistatic and monostatic SiGe transceiver MMIC.

Figure 2.11: A 240 GHz FMCW radar system [120].

antennas) is high, especially a large signal power is required at such a high frequency on a small system-on-chip package.

2.7 SAR Imaging

Instead of using a large physical radar, one can apply *Synthetic Aperture Radar* (SAR) [33] [87] principle to create a virtual, large-aperture radar. SAR works by physically and linearly moving a small-aperture radar. FFT is applied to the received signals across synthesized antennas to determine small differences in the distance where the reflected signal traveled back to the receiving antennas.

In the case of a mmWave radar, such as the one we are using in this paper, we can move the radar in a vertical direction to perform SAR operation. By moving the radar in the vertical direction and stopping at 10 positions where consecutive stops are separated by $\lambda/2$, we form a synthetic aperture radar having 10 by 8 antennas. We first apply FFT on each column of the measurements from the synthetic antennas, and then apply FFT on each row of the result from the previous FFT step. By applying the 2D FFT on the measurements, we obtain a 2D matrix where each element corresponds to the signal intensity received by the radar from a specific vertical and horizontal angle. Thus, we obtain the intensity of the corresponding location in the 3D space.

2.8 RF 3D Representation

As each range bin produces an intensity image, a 3D intensity map is produced by combining all the intensity images sequentially. For the intensity image, each axis is the angle defined by the filed of view and FFT bins, for example, the azimuth field of view is $120°$, and with 64 FFT bins, each intensity image pixel in the x-axis represents $1.875°$. Note that increasing the number of FFT bins does not increase the actual angular resolution of the radar as it is limited by the number of physical antennas. The intensity in the 3D space is calculated by applying trigonometry using the distance (derived from the range bin) and the angle (represented by pixels in the intensity image). An illustration of the 3D representation is shown in Figure 2.12.

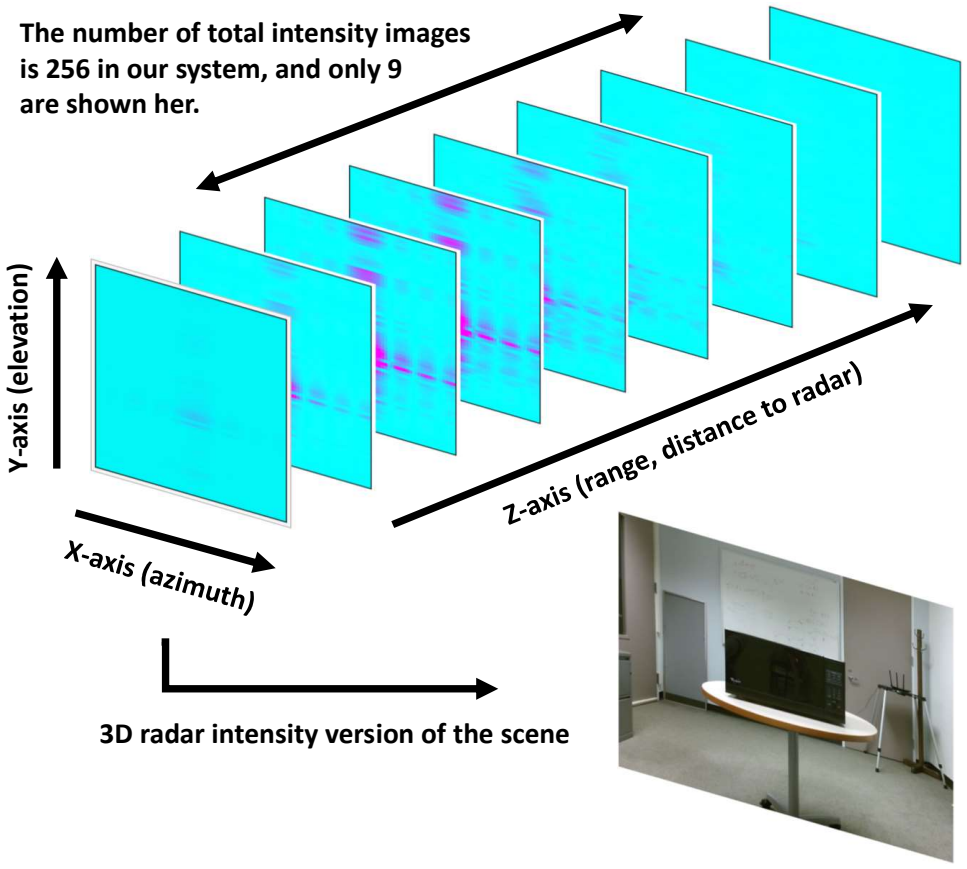


Figure 2.12: Illustration of 3D representation from mmWave radar intensity images. The intensity images are arranged in order from near to far from the object (microwave). By combining the intensities from all intensity images, a 3D intensity map containing the object (microwave) is constructed.

2.9 Compressed Sensing

In digital signal processing, the Nyquist-Shannon sampling theorem dictates that in order to reconstruct a continuous-time signal from its sampled discrete values, the sampling rate has to be at least twice the bandwidth of the signal. However, when we have prior knowledge about the signal, a sampling rate lower than the Nyquist rate can be used to reconstruct the signal. In compressed sensing, the sparsity of a signal is exploited to reconstruct the signal, $\vec{x}$, from a very small number of samples, $\vec{y}$ by solving the under-determined system of linear equations, $\vec{y} = D\vec{x}$, where the L^1-norm of the signal, $\vec{x}$ is minimized to impose the sparsity constraint.

2.9.1 Nyquist-Shannon Sampling Theorem

In digital signal processing, real-world continuous-time signals require sampling to be converted to discrete-time signals to be used in computing devices. The Nyquist-Shannon sampling theorem specifics the upper bound on the sampled frequency of a continuous-time signal based on the sampling interval. When the sampling rate is at least twice the maximum frequency of the signal, the resulted measurement can be perfectly reconstructed. Note that this theorem only applies to signals which have Fourier transform with zero outside of a finite region of frequencies, and the sampling is performed uniformly. Express this theorem in mathematics term is a function $x(t)$ contains no frequency higher than B hertz, it can be completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart [110].

2.10 Smoothing

For data with noise, the most intuitive action is to filter out the noise data and use the ones that are good. However, such operation may reduce the number of data available which could negatively impact the intended application. For replacing the noise data to capture the more important patterns, one can use smoothing algorithms to achieve such a result. One of the most straight forward algorithms is moving average, which smooths the data by replacing each data

point with the average of the data defined within the neighboring span. The span size is the smoothing windows size. There are numerous other smoothing algorithms such as Savitzky-Golay Filtering [99], Local Regression Smoothing [31], and low-pass filter. These algorithms mostly assume that the noise has a special distribution or the true data lies in the average of the noisy data.

2.11 Trajectory Matching

Trajectory matching can be as simple as Euclidean distance which calculates the distance between each corresponding point and determines the matched trajectory by the shortest distance between the two. To account that a single point from one trajectory can be corresponding to multiple points from another trajectory, such as two trajectories can be varying in speed, dynamic time warping can be used to measure the similarity between two-time series data. In some cases where non-parametric methods may be required to work with outliers with non-linear trends, such measures can be Spearman [137] and Kendall [12] rank-based correlation.

CHAPTER 3: CONTEXT DRIVEN ADAPTIVE CAMERA SYSTEMS

3.1 Introduction

In recent years, the use of body-worn cameras has increased exponentially—from law enforcement officers, to first responders, to the military. The need for having a body camera to carry out day-to-day jobs with a higher level of competency has brought many commercial products into the consumer market. Typically, body cameras are used as a recording device which stores multimedia content inside the device. These video feeds are later downloaded and analyzed off-line. Besides satisfying application-specific video-quality requirements, the two most desirable properties of body cameras are *extended battery life* and *efficient storage*. However, most body cameras of today have a short battery life which limits our ability to capture hours-long events. Storage space is also limited in these portable systems. Although the cost of storage has become cheap nowadays, efficiency is always desirable, and in some cases, it is necessary as there is monetary cost associated with archiving a large amount of data, and for this reason, often security videos are archived for a limited period.

Unfortunately, today's body cameras are developed somewhat in an ad hoc manner by gluing together different sensing, computational, and communication modules with limited or guaranteeless optimization in their designs. When such systems are deployed in the real world, they fail to provide a promised quality of service and an extended battery life. We argue that in order to develop an efficient body camera that provides an *extended battery-life*, *efficient storage*, and *satisfactory video quality*, instead of an ad hoc combination of independently developed modules, we need to apply context-aware control-theoretic principles and engineer a body-worn camera that is dependable, efficient, and robust.

We envision that as the technology behind video capture and processing matures, we will see increased use of body cameras in the days to come. Of particular interest to us in this paper are the body cameras worn by the law enforcement officers, because of their direct impact on our society. Several studies [64, 32, 103] have shown that the use of body cameras among the officers increases transparency and reduces the abuse of power. For instance, a Phoenix-based study [64] in 2014 showed that officers who kept their body cameras on made 17% more arrests and received 23% fewer complaints than the officers who did not. The reason behind this is psychological for the most part, as the officer is aware that his actions are being recorded. This develops an extra sense of dutifulness among the law enforcement officers, which is likely to reduce the number of disturbing events like the ones we have witnessed in recent years all over the country.

Existing implementations of body cameras typically consist of an on/off switch that is controlled by its wearer. While this design suits the purpose in an ideal scenario, in many situations, the wearer (e.g., a law-enforcement officer) may not be able to predict the right moment to turn the camera on or may completely forget to do so in the midst of an action. There are some cameras which automatically turns on at an event (e.g., when a gun is pulled), but they miss the "back-story," i.e., how the situation had developed.

We advocate that body cameras should be *always on* so that they are able to continuously capture the scene for an extended period. However, cameras being one of the most power-hungry sensors, the lifetime of a continuous vision camera is limited to tens of minutes to few hours depending on the size of the battery. A commonsense approach to extend the battery-life of a continuous vision camera is to analyze the scene and record it at high resolution and/or at high frame rate only when the scene dynamics is high and vice versa. However, on-device scene dynamics analysis is extremely costly and the cost generally outweighs the benefit. Recent works [57, 88] therefore employ secondary low-power cameras (e.g., a thermal imaging sensor) and FPGA-based scene analysis to wake-up the primary camera when an interesting event is detected at a lower energy cost. These systems, however, have the same limitation of missing the back-story,

and in general, they are bulkier than a single camera-based system due to the additional camera sensor.

In this chapter, we present *ZenCam* [41] [42], which is an *always on* and *continuously recording* body camera that ensures the *desired battery-life* and a *near-optimal*[1] *video quality*. The camera analyzes the *dynamics of the scene* as well as the *activity level* of the wearer in real-time, and controls the parameters of the body camera (e.g., frame rate and resolution) in order to satisfy the battery-life and the video quality requirements. The novel technical aspects of ZenCam are two-fold:

- First, we develop a light-weight video analytics algorithm that operates entirely on the encoded video stream obtained directly from the camera firmware for fast and low-power scene dynamics classification. This is different from existing algorithms that decompress and analyze video frames in the pixel domain.
- Second, a novel control-theoretic approach where we employ a *model predictive controller* [127] to dynamically control the frame rate and resolution of the camera sensor to achieve the desired battery life and a near-optimal video quality.

We develop a prototype of ZenCam using low-cost, off-the-shelf sensors, and lightweight, open source software modules. We identify the system parameters that affect the output power and the video quality, and empirically model their relationship. We implement a scene dynamics analysis algorithm which uses natively available motion vectors from the camera and implement a light-weight activity-level classifier that uses an on-board accelerometer to determine the activity-level of the wearer. The control algorithm uses both types of contextual information to dynamically adapt the camera parameters.

We deploy ZenCam in multiple real-world environments, such as an office building, a street, and inside a car, and evaluate its performance. We demonstrate that ZenCam achieves a 29.8-35% reduction in energy consumption and a 48.1-49.5% reduction in storage when compared

[1] Around 2% decrease in video quality compared to baseline model.

to a fixed-configuration body camera, without losing the video quality. Compared to an Oracle system, ZenCam's computational overhead is 10-17% with unoptimized hardware and software implementation. To the best of our knowledge, ZenCam is the first of its kind to achieve such energy and storage savings via an on-device, single camera-based, light-weight, low-power scene dynamics analysis algorithm, and a minimal hardware addition (i.e., an IMU), without sacrificing the video quality as the scene and user dynamics change at run-time.

3.2 Usage Scenario

We describe a usage scenario of ZenCam having a law enforcement officer as its intended user. An officer wears the body camera at the beginning of their shift. When they patrol an area in their car, ZenCam determines that there is a medium level dynamics in the surrounding scene and starts recording at a medium frame rate, e.g., 15 frames per second, which is high enough to capture the information. The officer parks their car and waits for traffic violations. As they wait, the ZenCam determines that the officer is in a low activity mode and the scene mostly has low to medium dynamics. ZenCam, in this case, changes its configuration to a low frame-rate recording setting to save system resources. At some point, the officer receives a dispatch command. They quickly arrive at the chaotic area. The motion of the camera, along with the high dynamics of the scene, triggers ZenCam's high-quality mode to capture the scene at the best video quality. Once the officer returns to their office, ZenCam detects inactivity and low scene dynamics and switches back to a low system resource consumption setting. This whole process is autonomous. ZenCam does not require the officer's engagement to turn on/off or configure the camera, and all information is preserved at the desired quality while satisfying the day-long battery life. The most important part is that the dynamic adjustments performed by ZenCam have minimal impact on the quality of the video recorded. ZenCam records continuously and smoothly so the human perception of the recordings is near identical to a fixed camera setting, thus all important information is recorded.

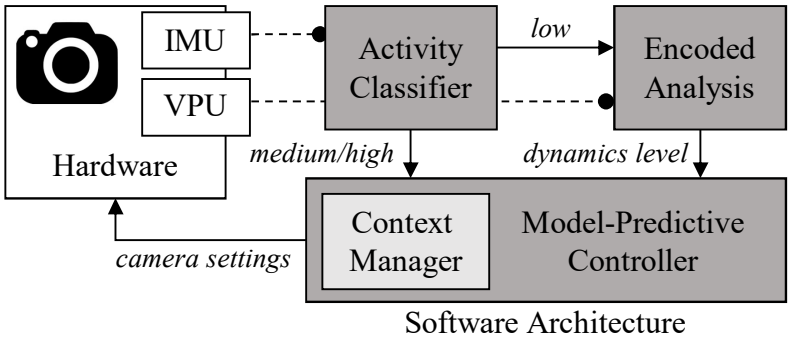


Figure 3.1: ZenCam system architecture.

3.3 ZenCam System Design

ZenCam is an autonomous, always-on, continuous-vision body camera that dynamically adjusts its configurations based on both the degree of body movement of the wearer and the dynamics of the scene. As a result, ZenCam achieves a desired battery life and ensures an optimal video quality of the recorded scenes. The system architecture of ZenCam is depicted in Figure 4.1.

3.3.1 Hardware Components

ZenCam's hardware consists primarily of a camera sensor, an onboard video processing unit (VPU), and an inertial measurement unit (IMU). Each of these components is independently adjustable to suit application-specific requirements. The video processing unit (VPU) is a chip (such as graphics processing unit (GPU), application-specific integrated circuit (ASIC), or field-programmable gate array (FPGA)) that provides hardware-processed video frames along with the meta-data such as the motion vectors. ZenCam exploits these readily available meta-data to classify the scene dynamics efficiently at run-time. The inertial measurement unit (IMU) is a low-cost and low-power sensor that provides raw accelerometer data, which is used by ZenCam to estimate the activity level of the user in real-time.

3.3.2 Software Components

ZenCam's software architecture consists of three major components: `Encoded Analysis`, `Activity Classifier`, and `Model Predictive Controller`. The `Activity Classifier` detects and classifies the user's body activity level into low, medium, and high levels. If the detected activity level is *high* or *medium*, which implies that the user is in motion, the result is directly used by the `Context Manager` of the `Model Predictive Controller` to determine the corresponding camera settings. In this case, the `Encoded Analysis` component remains dormant, which minimizes the energy and computational overhead of the system.

On the other hand, when the `Activity Classifier` determines that the user's activity level is *low*, it notifies the `Encoded Analysis` component to analyze the encoded video data along with the IMU data to determine the scene dynamics. The rationale behind this is that even if a user is inactive, the scene may have a different level of dynamics (e.g., a mostly static environment vs. an ongoing riot), which would require a different camera settings to capture the scene. With direct access to the motion vectors and residual values, ZenCam does not have to decode the video. The `Encoded Analysis` component works directly on the encoded video stream to determine the scene dynamics and passes the result to the `Context Manager`. Based on the context of the user as well as the scene, the `Model Predictive Controller` determines and sets the optimal values of the camera parameters.

3.3.3 The Novelty of ZenCam Design

We highlight novel aspects of ZenCam that differentiates it from existing continuous vision camera systems.

Uses Readily Available Information – Understanding the scene dynamics is important to control the camera parameters optimally. However, processing video frames in real-time is not practical in CPU and energy-constrained embedded systems. Hence, ZenCam relies on readily available information in the video meta-data, such as the motion vectors and residual data, to infer the scene dynamics. This information is available in every camera chip and comes at no

extra cost of computation. As a result, unlike existing smart cameras, ZenCam does not have to decode, then process, and then again encode the video.

No Pixel-Domain Analysis – ZenCam does not analyze videos in the pixel domain, which is an extremely CPU and energy demanding task. Instead, it performs simple operations to filter unwanted motion vectors and to compute the entropy of a frame using these vectors – which accurately determines the dynamics of the scene at several order of magnitude less cost than pixel domain analysis of video frames.

Based on Control Theoretic Principles – In ZenCam, a model predictive controller is employed that dynamically adjusts the camera settings in order to achieve both reduced energy consumption and storage requirement while maintaining the desired video quality for a given user and scene context. Unlike existing continuous vision cameras that rely on heuristics to balance battery life and image quality, ZenCam provides a provable near-optimal performance.

Lightweight Algorithms – ZenCam employs lightweight algorithms in all of its software components. For instance, the scene analysis algorithm uses simple arithmetic operations to obtain the scene dynamics in linear time. The activity classifier computes lightweight features from IMU data in $O(1)$ and uses a pre-trained classifier. The controller's computational complexity at runtime is also optimized to simple table look-ups and constant time operations. Hence, the overhead of the system is negligible and its benefit outweighs the cost.

3.4 Encoded Video Analytics

When a video is recorded, the encoder calculates *motion vectors* and *residual values* [101]. Motion vectors represent the translation of pair-wise most similar macro-blocks (e.g., 8x8 pixels) across frames. Residual value is calculated as the difference between the macro-block in the current frame and the previous frame's matching one. In the final video, the current frame is reconstructed from macro-blocks of the previous frame and the differences. ZenCam exploits these two freely available information as the foundation of its lightweight scene dynamics classification algorithm.

3.4.1 Preprocessing Motion Vectors

Motion vectors are computed by video coding algorithms which matches similar macro-blocks across different frames in order to minimize the residual value. Motion vectors produced in this manner sometimes do not represent the true motion of the objects in the scene as these greedy algorithms often match *similar* macro-blocks that minimize residuals. While such motion vectors, combined with residuals, are perfectly okay for video coding, they are problematic when determining the true movements of objects in a scene.

ZenCam filters out these erroneous vectors in a preprocessing step. From the latest frame in a time period T, we calculate the angle, $d_i = \arctan \frac{\hat{v}_y}{\hat{v}_x}$ of a motion vector $\hat{v} = (\hat{v}_x, \hat{v}_y)$. To identify erroneous cases, we create a histogram of $\{d_i\}$ and discard the motion vectors whose angles fall into bins that has a very small number of elements.

There are two intuitions behind this step. First, the number of erroneous motion vectors tends to be small within a certain angular range. Second, even if the discarded motion vectors correspond to correct motion of an object in the scene, we still should discard those as the object occupies a minimal area where a large macro-block is 16x16 pixels. This means that the object is either far away or tiny. In this situation, the motion of the object typically does not generate a fast motion across the frame. Thus, such scenes are not to be recorded at high frame rate.

After discarding erroneous motion vectors, we normalize the vectors to compensate for the effect of frame rates.

$$\textit{Scaled } v_i = v_i \times \frac{\textit{current frame rate}}{\textit{reference frame rate}} \tag{3.1}$$

The scaling is necessary as different frame rates produce motion vectors of different magnitudes even when an object is moving at the same speed. This is shown in Figure 3.2a. The motion of the human subject is identical in both frames as they are recorded simultaneously. The top one has a rate of 10 fps, and the bottom one is 30 fps. The zoomed-in view shows that the magnitude of the motion vector is larger at the lower frame rate. By scaling the motion vectors, we

normalize the motion of an object across different frame rates. After scaling, we filter out motion vectors whose magnitude, $m_i = \sqrt{\hat{x}^2 + \hat{y}^2}$ is below a small threshold to eliminate random noise or movements corresponding to low scene dynamics.

Finally, we compensate the effect of camera movements on motion vectors. We calculate ZenCam's orientation using the gyroscope data and estimate the direction of the frame's movement by projecting the orientation on a plane parallel to the camera. We discard the motion vectors that are in the opposite direction of the camera's movement. This is because, when the camera moves, the frame tends to move in the opposite direction. As a result, we discard the motion vectors that are in the opposite direction of the camera's movement.

3.4.2 Scene Dynamics Estimation

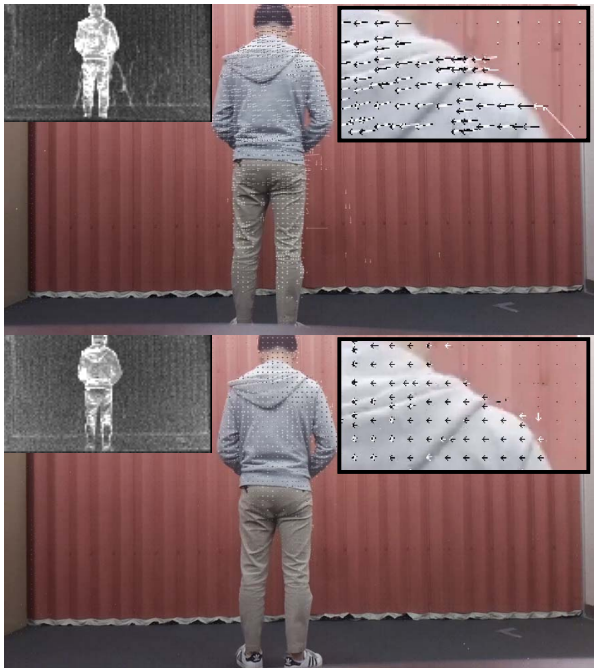
A high residual value indicates that the macro-block does not match well across frames. In such cases, we consider that the probability of movement is high. Within a frame, we convert the residual values into a distribution, p_i by normalizing against the sum of residuals of a macro-block. We apply a *softplus* function to convert motion vectors to intensity:

$$I(\hat{v_i}) = \ln(1 + e^{m_i}) \tag{3.2}$$

We calculate the entropy of a frame as follows:

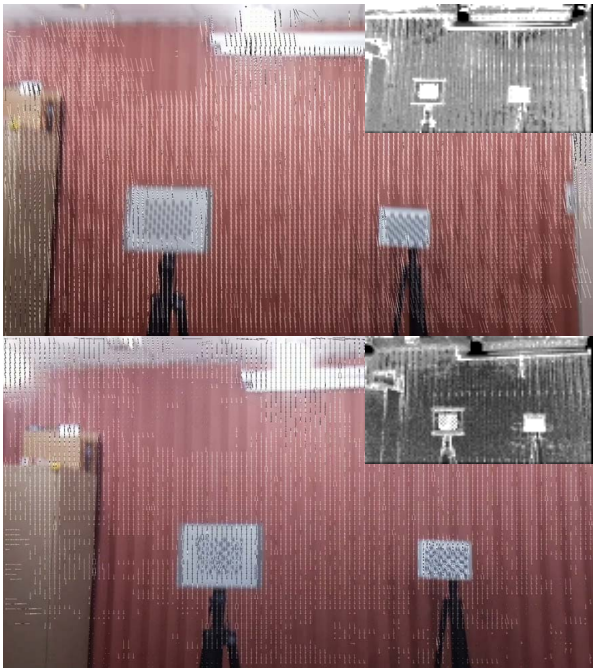
$$E = \sum_{i=0}^{N} p_i I(\hat{v_i}) \tag{3.3}$$

where, N is the total number of motion vectors after the preprocessing step. We divide E by N to get a score S for the current frame, which classifies the scene dynamics. We accumulate consecutive η classification results to perform a majority voting. The rationale behind this algorithm is that when there are no moving objects in a scene, motion vectors caused by small camera movements will be removed and the score will be low. The score of a scene having moving ob-



(a) Standing with minimal movements

(b) Running in front of static camera



(c) Camera movements in static scene

(d) Running in front of moving camera

Figure 3.2: Motion vectors overlaid on video frames. The top frames are recorded at 10 fps while the bottom frames are at 30 fps. Black and white pictures are the magnitudes of the residual values. a also shows enlarged motion vectors near the subject's shoulder.

jects in it depends on the speed of motion and the percentage of the area occupied by the object in the frame. The larger the score, the more dynamic a scene is.

The proposed encoded-domain scene analysis technique uses readily available features and requires neither decoding nor re-encoding the frames. The algorithm is lightweight and parallelizable. The implication is that it can be implemented in hardware (e.g., an FPGA or even ASIC) to gain substantial energy savings and speedup.

3.5 Activity Level Classification

Activity level classification is a crucial component of our system. We use an IMU sensor to perform human activity level classification for three different levels: *high* (i.e. running), *medium* (i.e. walking), and *low* (i.e. standing or sitting). The detected activity level is used for triggering compressed domain analysis when the body activity is low or directly used to generate reference value when the activity is either *high* or *medium*.

Similar to [93], provided the raw 3-axis IMU data, we extract the raw accelerometer data and calculate the linear magnitude of the acceleration: $\alpha = \sqrt{a_x^2 + a_y^2 + a_z^2}$. The standard deviation of the raw data over 0.5 seconds is then used to train a support vector machine (SVM) classifier which uses radial basis function (RBF) kernel. With a trained SVM classifier, we extract the data and perform classification on data points from past 0.5 seconds every 0.1 seconds (i.e. 10 Hz). We first collect a set of classified values, $\{x_i\}$, over τ seconds. Then using majority voting to determine activity level of the highest probability. In our experiments, we empirically set $\tau = 5$ seconds which a total of 50 values are used for majority vote. If the classification result is either *high* or *medium*, the result is directly send to the controller and compressed domain scene analysis will not be performed which reduces energy consumption. When the activity level is *low*, the classification result and raw data will be send to scene dynamics analysis component and trigger the operation.

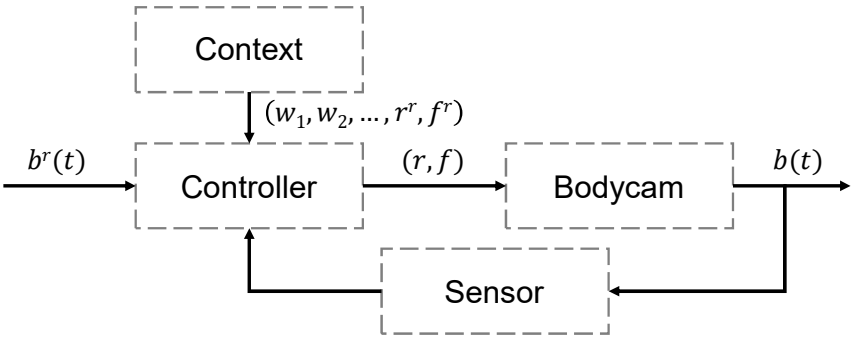


Figure 3.3: Controller design for the body camera

3.6 Controller Design

3.6.1 Camera Controller

We design the body camera control system as shown in Figure 3.3. The controller takes input from the sensor which measures the system's battery level $b(t)$. The context look-up table provides the weights $(w_1, w_2, \dots)$ and the reference values of resolution r^r and frame rate f^r for the current context. The controller computes the control input (r, f) based on these parameters and the reference value of the desired battery level $b^r(t)$.

We define the objective function for the controller to match the actual power consumption to a reference value (e.g., set by the application-specific policy) while maintaining a certain video quality. For instance, assuming a fixed camera setting and linear battery drain, if the target battery life is 8 hours, the system should have 50% battery remaining after 4 hours. In this case, the linearly dropping battery level is the reference at any point in time. The cost function is defined as:

$$J = (b - b^r)^2 + \sum_{i=0}^{N} w_i S(x_i - x_i^r) \tag{3.4}$$

$$\text{where, } S(x) = \frac{1}{1 + e^{-a \cdot x}} - 0.5 \tag{3.5}$$

In (3.4), b is the future battery level *after* applying the camera setting for the time step, b^r is the reference battery level, x_i is the control input to the system (i.e., (r, f) in Figure 3.3), and

x_i^r is the reference values (i.e., (r^r, f^r) in Figure 3.3), w_i are the weights of these variables, S is the sigmoid function where a determines the range of configurations that we want to focus and apply aggressive control. Each context has its own reference values for control variables and weights, and the sigmoid terms reward or penalize certain behavior in each context. We use sigmoid functions because of their decreased rate of change when the actual value moves away from the reference and vice versa – which enables the system to take more aggressive action when necessary.

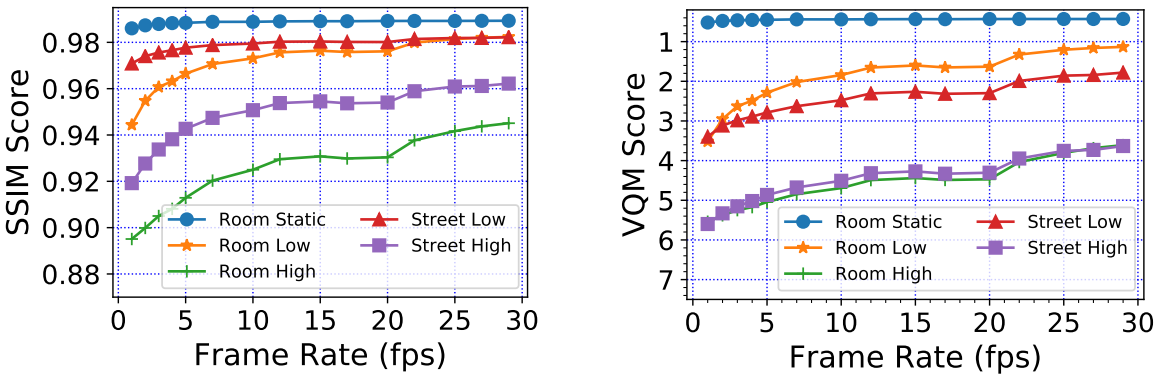
The model predictive controller optimizes its objective over a finite time horizon. Given the current context, we optimize (3.4) over the next time step considering the most likely sequence of contexts and their average duration. The sequence of contexts and their duration are statistically estimated over time. The power consumption for each camera configuration is updated to minimize the impact of inaccuracies in system modeling and changes in hardware performance such as battery degradation.

3.6.2 Determining Reference Values

There are studies that model user perception of video quality at varying frame rates and resolution [94, 145]. Although these data can not be directly used in ZenCam due to the differences in the system architecture and goals, we use the technique from [145] to calculate the quality scores for different camera settings. We use two metrics: *Structural Similarity Index Metric (SSIM)* [130] and *Video Quality Metric (VQM)* [98]. These two objective metrics correlates with human perceptions.

Location	**Scene**	**Activity Level**	**Length (s)**
Room	Static	None	180
Room	Small Movement	Low	180
Room	Running	High	180
Street	Low Car Volume	Medium	289
Street	High Car Volume	High	360

Table 3.1: Scenes used in different videos.



(a) SSIM Score

(b) VQM Score

Figure 3.4: SSIM and VQM Scores at different frame rates.

We record five types of videos at different indoor and outdoor scenarios at 1920x1080 resolution and 30 fps. Location, scene, activity level, and duration of these videos are listed in Table 3.1. Frame rates of these videos are altered by dropping frames to simulate different camera settings. Finally, all videos are converted to the same frame rate using the same encoder for comparison. We increase the frame rate of low-frame-rate videos by duplicating frames. We use MSU Video Quality Measurement Tool [50] to measure the SSIM and VQM scores for each video. The SSIM scores (higher is better) are shown in Figure 3.4a and the VQM scores (lower is better) are shown in Figure 3.4b.

We observe in both figures that as the frame rate reduces, the quality decreases and the rate of change in quality sharply increases once the frame rate drops below a certain threshold. Also, the change in quality is smaller at low scene dynamics. This is due to the fact that for static scene, a single image can be perceived as a video. Based on these observations, we design ZenCam's reference settings in a manner that the frame rate of the video should increase as the video frame dynamics increases whether it is the result of camera movement or scene dynamics. We determine that the resolution should increase when the frame rate increases as the added video quality will be beneficial for either human analysis or vision algorithms. We choose the reference values such that they result in about 2% decrease from the highest SSIM score. This gives us the minimal decrease in video quality and low frame rate that save system resources.

We note that the reference value can be chosen based on the application where different application scenarios require different video quality. The ZenCam controller is designed in a manner that the reference can be easily configured to meet an application's requirement. This feature allows users without any technical knowledge to manage and fine-tune the system for their specific needs. For the ZenCam system, each camera state can be associated with a different reference value as well as different camera states can share the same reference value. For example, for two scenarios where the IMU reports *Low* in both cases while the scene analysis reports *High* and *Low* respectively, one can choose a higher frame rate for the first scenario while choosing a lower frame rate for the later one. For two scenarios where the IMU reports *High* activity for the first scenario, and for the second scenario, IMU reports *Low* and the scene analysis reports *High*, one can choose the same high frame rate for both cases as it would be better for viewing.

3.7 System Implementation

3.7.1 Hardware Development

We implement a prototype of ZenCam which is shown in Figure 3.5. We use a Raspberry Pi Zero W and a Pi Camera Module V2 as the main components. We connect an LSM9DS0 IMU to the Raspberry Pi via I^2C. A Raspberry Pi Zero is used to demonstrate that our system works with off-the-shelf parts and runs on resource-constrained platforms. Pi Camera also provides us with an easy access to motion vectors and residual values without decoding the frames. This information is available in nearly every camera chip but requires a slight software modifications to expose it. For example, [78] made changes to the OS to expose the motion vectors on an Android device.

A PowerBoost 1000C power supply is used to charge the battery and to supply 5V to the Raspberry Pi. We use an MCP3008 ADC chip to measure the battery voltage and convert that to a percentage level used by the controller. We measure the voltage over time while draining the

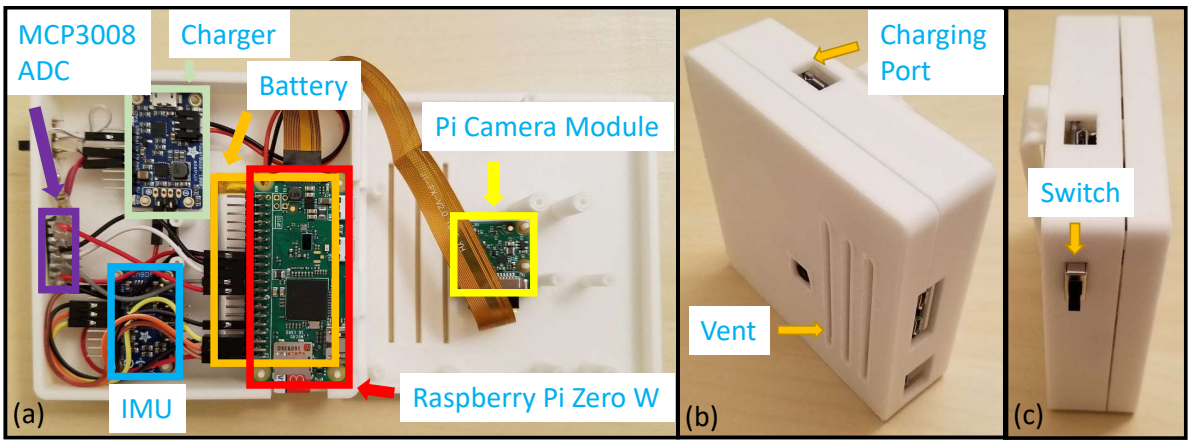


Figure 3.5: ZenCam Prototype: (a) Internal components are labeled. (b) The right-front view. The camera is near the vent. (c) The left view, switch, charging port, and clip.

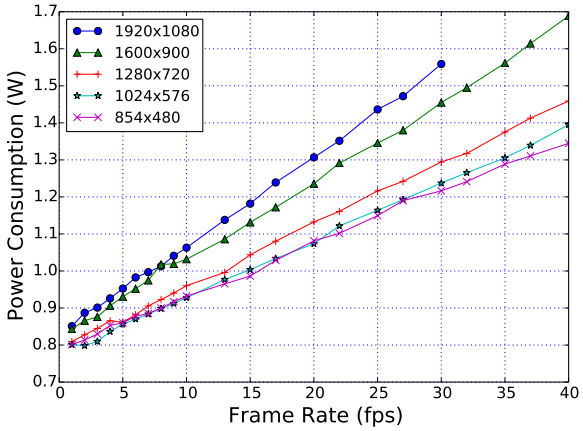
battery with a constant current. The measurements from the battery are used to fit a polynomial regression model. The model is later used to determine the remaining battery percentage.

3.7.2 Software Development

We use the `picamera` Python library to control the camera configuration and to retrieve the encoded domain data. The `scikit-learn` Python library is used for SVM classification. We collect IMU data at 10Hz which is sufficient for user activity detection at a minimal CPU utilization. The battery voltage is queried every second and the percentage is calculated from the average voltage over 30s. Video scene analysis is performed every 250ms when the camera activity is low, which results in a high accuracy at a minimal overhead.

To classify the activity level of a user, we implement a lightweight algorithm similar to [93, 40]. From the raw accelerometer data, we calculate the magnitude of the acceleration: $\alpha = \sqrt{a_x^2 + a_y^2 + a_z^2}$. The standard deviation of $\{\alpha_i\}$ over 500ms is used to train a support vector machine (SVM) with a radial basis function (RBF) kernel. With a trained SVM classifier, we extract the data and perform classification on data points from past 500ms every 100ms (i.e., 10 Hz). We collect a series of classification results over 5s and apply majority voting to determine the most likely activity level of the user.

(a) Field of view for each sensor setting



(b) Power consumption with sensor mode 4

Figure 3.6: Sensor modes and power consumption for sensor mode 4

3.8 Energy Overhead Measurement

We use Pi Camera sensor mode 4 (out of the 7 modes available), shown in Figure 3.6a, as it utilizes the full sensor area and produces frame rates from 0.1 fps to 40 fps. The power consumption of the system for different combinations of resolution and frame rates are shown in Figure 3.6b. Based on these measurements, we select our resolution settings from a pool of three choices: `1920x1080`, `1600x900`, and `1280x720`, which are commonly used in real-world applications. Our measurements reveal that they have different power consumption for energy saving purpose. A linear proportional relationship between the CPU load and power consumption is observed.

To understand the overhead of ZenCam algorithms, we measure its power consumption for the same camera setting with and without the algorithms running. The results are shown in Table 3.2. The *baseline* refers to the same camera settings as ZenCam but without the algorithms running. We see that the overhead is between 3.7%–17.6%, and it's the smallest when the system uses the highest frame rate. The overhead of ZenCam when the camera is steady decreases as the scene dynamics decreases because there are fewer motion vectors to process when the camera records at a lower resolution.

IMU	Scene Level	Resolution	Frame Rate (fps)	Baseline Power (W)	ZenCam Power (W)
High	N/A	1600x900	40	1.88	1.95
Medium	N/A	1920x1080	26	1.53	1.62
	High	1920x1080	25	1.47	1.73
Low	Medium	1600x900	15	1.14	1.33
	Low	1280x720	4	0.86	0.99

Table 3.2: Power consumption at different contexts and settings.

IMU	Low			Mid	High	Remaining Battery (%)			File Size (GB)	
Scene Level	Low	Mid	High	NA	NA	Baseline	ZenCam	Oracle	Baseline	ZenCam
Time #1 (min)	30	33	34	22	11	29.9	50.8	55.3	5.4	2.8
Time #2 (min)	48	43	19	20	14	36.6	58.8	64.8	9.1	4.6

Table 3.3: Time duration for each classified activity in minutes, battery remaining as end of system, and size of video files generated.

When ZenCam is in *high* or *medium* activity level, the energy overhead to drive the IMU sensor and its processing algorithm is about 80mW. This overhead can be reduced further to as little as 5mW by using an ultra-low-power microcontroller instead of a Raspberry Pi [88]. The energy overhead of scene dynamics analysis can be reduced to 50mW with an FPGA-based implementation [88].

3.9 Real-World Deployment

3.9.1 Deployment Setup

Two body cameras having identical hardware are deployed in multiple real-world scenarios (indoors and outdoors) in two sessions for a total of 271 minutes. In each session, both cameras start and finish recording at the same time. The first camera runs ZenCam algorithms and the other one uses a fixed settings of 1600x900 resolution and 40fps to record high-quality ground-

(a) Low (b) Medium (c) High

Figure 3.7: Classification of scene dynamics.

truth video of the scene. This second video footage is later used in our lab to produce two different solutions to compare with the ZenCam. We call these two: *Baseline* and *Oracle*, respectively.

The *Baseline* is obtain by re-recording the same high-quality ground-truth video footage (1600x900 @40fps) obtained from the field experiments at a fixed settings of 1920x1080 @30fps – which is a common settings used in most video recorders. Re-recording is performed by playing the footage on a computer screen and then capturing the scene with the same camera used as ZenCam. We use this Baseline solution to compare ZenCam's performance against a typical fixed-configuration camera.

Likewise, the *Oracle* is also obtained by re-recording the high-quality footage as in Baseline, but this time the camera dynamically adapts to different camera settings based on its prior knowledge of the user's context and the scene dynamics obtained from ZenCam's classification results. We use this Oracle solution to quantify ZenCam's computational overhead as these two are identical in their hardware and software, except that the context detection algorithms do not run in the Oracle.

The duration of different activity levels and scene dynamics for both sessions are shown in Table 3.3. The first session lasts for 130 minutes. It includes user's activity such as sitting in front of a desk, walking on corridors, and running outside. The second session lasts for 141 minutes which includes activities such as running and walking outdoors, driving, and sitting in front of a desk. These activity sequences, although may not be overly extensive, are chosen to mimic daily activities of a law-enforcement or a patrol officer, and captures all possible activity and scene types of ZenCam.

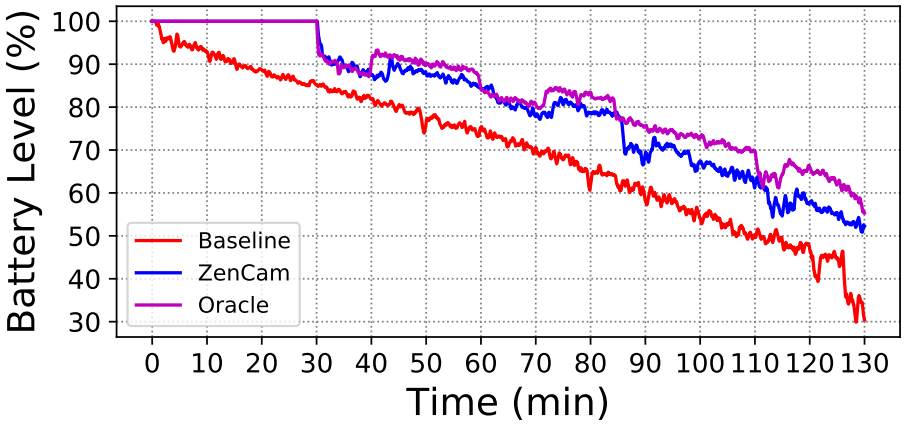
We manually inspect the encoded video scene analysis results of ZenCam to verify that they match our expectation. Three sample frames are shown in Figure 3.7. Figure 3.7a is classified as low dynamics as the street is almost empty. Figure 3.7b is classified as medium dynamics where there is only one car. Figure 3.7c has multiple cars on the street which the system classifies as a scene with high dynamics.

3.9.2 Extended Battery Life

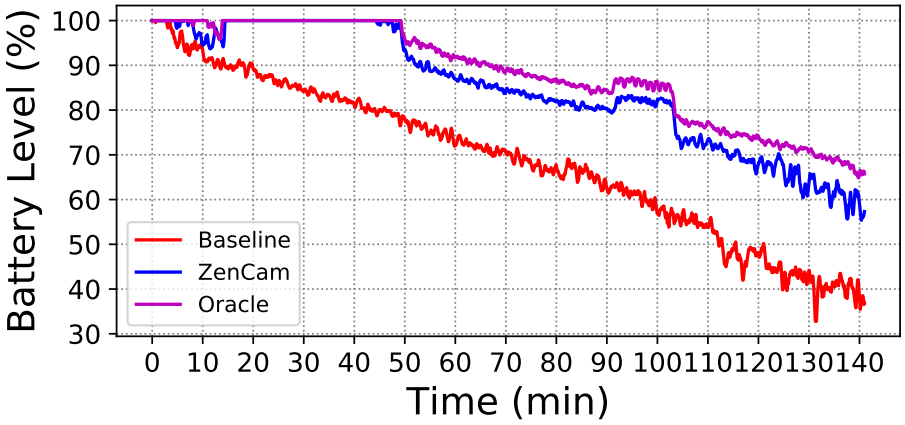
In Figure 3.8, we plot the remaining battery life of the three camera systems: ZenCam, Baseline (1920x1080 at 30 fps.), and the Oracle. At the end of the two sessions, energy savings with ZenCam is 29.8% and 35%, respectively, when compared to the baseline. ZenCam's energy overhead is 10% and 17%, respectively, when compared to the Oracle. The performance difference in the two sessions is expected as the first session contains more high and medium level activities that consume more energy but incurs less overhead due to lower amount of scene dynamics analyses. The second session contains more low level activities which results in a lower energy consumption but results in a higher overhead due to more frequent scene dynamics computation.

3.9.3 Storage Efficiency

To compare the storage efficiency, we examine the sizes of the generated video files. All recordings use the same quantization parameter to prevent the system from automatically changing compression ratio which affects the file size. During the first session, the baseline model produces a total of 5.4 GB video files while ZenCam produces 2.8 GB (48.1% reduction). The baseline model generates 9.1 GB file in the second session while ZenCam generates 4.6 GB (49.5% reduction). The difference is due to the second session having more outdoor activities where the spatial complexity is more than the first session. Further discussion on this is in Section 3.10. This result demonstrates that our system reduces the storage requirement significantly, and thus reduces the cost of archiving data.



(a) Session #1

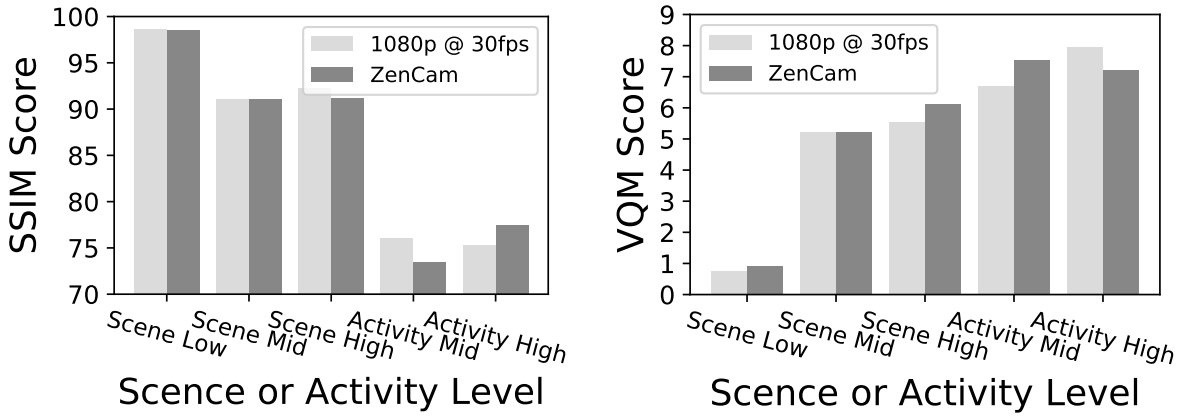


(b) Session #2

Figure 3.8: Remaining battery life (%) over the duration of experiments.

3.9.4 Video Quality Analysis

In this experiment, we investigate how well ZenCam preserves the video quality by using the method metrics used in Section 3.6. Figure 3.9a and Figure 3.9b compares video quality from ZenCam and the Baseline (1080p @30fps) in terms of SSIM and VQM, respectively. Recall that lower VQM scores implies better video quality. We observe that for low and medium scene dynamics, there is no significant difference in the video quality since the frame rate does not affect the quality of the video significantly in a mostly static scene. For high dynamic scenes and medium activity levels, ZenCam produces slightly lower quality videos than the baseline but the difference is small. During high activity level, ZenCam produces better quality videos as the



(a) SSIM Score vs. Environment

(b) VQM Score vs. Environment

Figure 3.9: Video quality analysis for different scenes

frame rate goes higher than 30 fps. This is expected as high frame rates improve the video quality in a highly dynamic environment.

3.9.5 Summary

Through a series of empirical evaluation, we demonstrated that ZenCam significantly reduces energy consumption and storage space requirement while maintaining a competitive video quality and outperforms typical fixed configuration cameras. ZenCam reduces energy consumption by trading in minimal video quality. Our evaluation also shows that ZenCam has a lower overhead when the system is at the highest demand – which is a property that no state-of-the-art system has achieved till date. With a more efficient hardware implementation, ZenCam can achieve even higher efficiency and resource savings.

3.10 Discussion

Why the video features are free? The video features we use (i.e., motion vectors and residuals) are generated by the video codec during the video encoding process. Since these two are already computed inside the camera firmware, all we require is to expose them to the application layer and use them at run-time in the proposed scene dynamics analyzer. Since there is no over-

head of frame decoding, feature computation, and frame re-encoding, we are relieved from an expensive step of feature computation at zero cost.

Why not use machine vision techniques? We consider two aspects of machine learning and computer vision: energy consumption and accuracy. Modern computer vision techniques that are based on Deep Neural Networks (DNNs) require cloud and/or GPU machines as opposed to low-power embedded processors [88, 58]. Even specialized vision processing units consume more power than a Raspberry Pi (around 1W) [5]. In contrast, our unoptimized ZenCam implementation having no special hardware adds less than 400mW overhead. Furthermore, current commercial products with machine learning techniques for video capturing such as Google Clips [3] are not on par with average user's expectation [4].

Can existing image processing algorithms be used on videos recorded by ZenCam? Since ZenCam generates standard video files, which we didn't modify the standard video compression algorithm that's been used, existing computer vision and image processing techniques are directly applicable to its output. Unlike [88], ZenCam preserves all information throughout its lifetime.

Would the use of CPU consumes more power than just directly store the video? The Raspberry Pi used in this work is to demonstrate the low computation requirement for our ZenCam framework. By implementation the framework on a much lower energy consumption platform such as FPGA, the power consumption overhead can be much lower. Moreover, newer body cameras that are being deployed in the field have added smart features such as LTE connection, GPS, microphone, touch screen operation, smartphone remote control, and etc. These features require on-board computer which can be used with a more efficient implementation of our framework. Thus, ZenCam framework can be just an add-on feature that improves performance with minimal overhead.

Does reducing frame rate help saving storage space? We record multiple 3-minute videos varying the frame rate for three different contents displayed on a computer monitor. The result is shown in Figure 3.10. `Static White` is a purely white picture, `Static Street` is a picture of a street, `Dynamic Backyard` is a short video clip of playing in the backyard. The file size of the

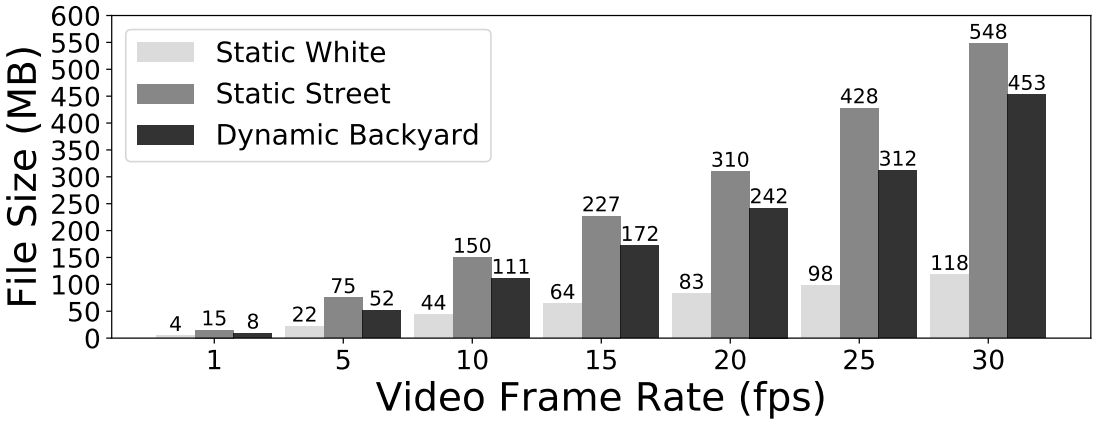


Figure 3.10: File size vs. frame rate

`Static Street` is larger than the `Dynamic Backyard` as the street scene has higher spatial complexity[2]. Overall, we see a decrease in the file size as the frame rate decreases. ZenCam saves a significant amount of storage space by reducing the frame rate significantly when the user is static facing a high complexity static scene.

Does saving storage space matters in nowadays? One may point out that the storage space on a modern camera is cheap and can be more than enough for recording purposes, however, reduction in storage space consumption can still be beneficial on multiple aspects. The direct cost associated with archiving videos can be significantly reduced as companies charge users based on storage space used for online archive. The uploading of videos to cloud either through wireless or wired connection can be more streamlined through reduced time needed and bandwidth. Even though the same algorithm can be used to reduce the video size by analyze and re-encode afterwards, the cost of re-encoding is high and will be both time consuming and power hungry. For a large number of body cameras uploading videos daily, a data center may needed to just re-encoding the videos for reduced space consumption purpose which is not efficient. With our on-board ZenCam framework, the reduction in storage consumption has no negative effects.

Can this technique be used in different applications? The ZenCam framework we developed is modular. Components of the system can be used in other applications. For example, the en-

[2]Video encoding happens in both spatial and temporal domain. Our system uses information from the temporal encoding only.

coded scene analytics technique can benefit surveillance cameras, such as the ones installed in front of a building, by increasing their capacity to store information efficiently in their limited storage. Currently, some of these cameras allow a fixed lower frame rate to save storage space but the resulting footage generally results in degraded image quality. With the encoded scene analytics, the frame rate can be dynamically adjusted based on the scene, which can preserve the quality and reduce the required storage at the same time.

3.11 Summary

In this chapter, we design, implement and evaluate ZenCam, an always-on body camera that saves system resources via scene dynamics analysis in the encoded video domain and human activity classification using IMU-based sensing. ZenCam uses this information to control camera parameters for energy and storage consumption reduction. Our evaluation shows that ZenCam achieves 29.8-35% energy savings and 48.1-49.5% storage space reduction, while maintaining a competitive or better video quality, when compared to a baseline system having a fixed configuration of 1920x1080 at 30fps.

CHAPTER 4: WIFI ENHANCED VISION SYSTEM FOR TRACKING AND RE-IDENTIFICATION

4.1 Introduction

Human sensing, motion trajectory estimation, and identification have a wide range of applications, including in retail stores, surveillance, public safety, public address, and in access control. For example, in retail stores, it is useful to capture customer behavior to determine an optimal layout for product placement, detecting re-appearing shoppers after weeks to track shopper retention rate, separating employees from shoppers to generate an accurate heatmap of motion pattern of (only) shoppers. For surveillance, it is useful to identify and track a limited set of people from a crowd, e.g., tracking undercover police agents from a group of people to ensure their safety. Once a class of people is identified, that can be leveraged for public address based on additional contexts. For example, when an active shooter in a building has been identified through a security camera, targeted and customized messages can be sent to different groups of people in different parts of the building through accurate identification to help to find safe escape routes – instead of sending a generic SMS to everyone, possibly including the shooter.

A wide variety of sensing technologies exist for human sensing, motion trajectory estimation, and identification that uses cameras, WiFi, Bluetooth, and ultrasonic sensors. However, there are shortcomings of each of these sensing modalities. For example, ultrasonic sensor-based identification [67] does not scale to thousands of reappearing shoppers in retail stores. Camera-based solutions suffer from illumination, occlusion, background cluttering, and change of perspective and fail to support long term identification, e.g., detecting a shopper after two weeks when they show up in a different colored dress in a retail store when body appearances based identification is applied [56]. Facial recognition can be potentially used for person identification at scale.

However, facial recognition is banned in many places, e.g., San Francisco [6], and it is difficult to employ in some settings such as in retail stores where typically panoramic cameras mounted on ceilings can hardly see faces. Sniffing WiFi MAC addresses provide coarse-grained location information, e.g., a shopper is within 30 meter radius of a WiFi access point without providing location insights to infer customer-product interaction. To achieve precise localization using WiFi, multiple WiFi beacons or receiving units need to be set up, maintained, and coordinated, which can be very expensive [69].

In this chapter, we propose to fuse two powerful sensing modalities – WiFi and camera – in order to overcome the aforementioned limitations of the state-of-the-art solutions. We call our proposed solution EyeFi [39]. EyeFi does not require facial recognition, provides long-term re-identification, does not require deployment and maintenance of multiple WiFi units, and has the potential to provide such intelligent capabilities on a standalone device. To this end, EyeFi integrates a WiFi chipset (with multiple antennas) to a camera. As a result, a single EyeFi unit can detect, track, and re-identify people as far as the camera can see. Our current implementation of EyeFi uses a panoramic camera mounted on a ceiling. However, other types of cameras such as a bullet camera will also work.

EyeFi uses the on-board camera to detect, track, and estimate the motion trajectories of the people in its field of view. Simultaneously, using the on-board WiFi chipset, EyeFi overhears WiFi packets from nearby smartphones and extracts the Channel State Information (CSI) data from the WiFi packets. The CSI information is used to estimate the Angle of Arrival (AoA) of the smartphone from the EyeFi unit. Compared to existing WiFi-based AoA estimation techniques [69], EyeFi uses a smartphone in motion as a transmitter (not a stationary desktop computer), uses a low sampling rate (around 23 packets per second), and uses a novel teacher-student based visually guided neural network to speed up the AoA estimation by over 3,800 times. For each person (i.e., smartphone) generating the WiFi traffic, a sequence of AoAs is estimated to capture the motion trajectory of the individual. Then EyeFi performs cross-modal trajectory matching to determine the identity of the individuals. It is based on the assumption that most

people use smartphones and the same smartphone is usually used for an extended period of time. Also, as smart watches are becoming popular and getting equipped with WiFi chipsets (e.g., Samsung Gear S3 and Apple Watch 4), EyeFi can leverage wireless devices beyond smartphones. Note that the MAC addresses can be hashed to safeguard the privacy of the individuals, but it is still useful for long-term re-identification and behavior analysis.

This work has the following contributions:

• First, we design and implement a novel multi-modal sensing system called EyeFi, which is the first system that fuses WiFi CSI with camera for human sensing, motion trajectory estimation, and long-term identification and has the potential to offer such analytics on a standalone device. EyeFi overcomes several limitations of the state-of-the-art solutions as it does not require the use of facial recognition and the cost of deployment and installation of multiple WiFi units.

• Second, since no such system and datasets are available, we collect over 74 GB of data containing videos and WiFi CSI values of over one million WiFi packets with over 15 volunteers from two different environments to develop and test our solution. We annotate a major portion of the dataset[1].

• Third, we develop a novel student-teacher based neural network to estimate AoA from CSI values. Instead of just using camera-based motion trajectory as the ground truth, we force the network to regress the AoA of state-of-the-art SpotFi algorithm [69], and thereby, forcing the network to learn multipaths and estimate AoA more accurately. We also propose novel techniques to smooth the WiFi-based trajectory for cross-modal matching.

• Finally, based on extensive evaluation using real-world data, we find that EyeFi improves WiFi CSI-based AoA estimation accuracy by more than 30% and offers 3,800 times computational speed up over the state-of-the-art solution, enabling EyeFi a real-time system. We observe that the average accuracy of EyeFi for person identification is 75% when the number of people varies from 2 to 10.

[1]More information on EyeFi data and deployment can be found at the project page `https://github.com/munir01/EyeFi`

4.2 Usage Scenario

We describe two real-world usage scenarios of EyeFi.

• *Customer Behavior Analysis.* Once a customer arrives at a store, his smartphone generates WiFi traffic to discover local access points. After he connects to the local WiFi access point, his checking of notifications, viewing of websites for better prices or deals of similar items, listening of SpotiFi music, or messaging of friends generates more WiFi traffic. All of these WiFi traffic is overheard by the WiFi chipset of EyeFi system. EyeFi extracts the MAC address and CSI values from the WiFi packets, timestamps each value, and records them. Using our proposed algorithm, EyeFi performs AoA estimation and matches the AoA sequence with one of the trajectories observed from the camera. Due to the use of the MAC address, the customer can be identified over a long period and even at a different store. EyeFi hashes the MAC address to anonymize the customers, but can still generate high-level analytics of aggregated customer behavior.

• *Emergency Situation.* During emergency situations, EyeFi can send location and person-specific targeted messages to guide people to safety. For example, in a retail environment, EyeFi can send different messages to employees who know the store area, law enforcement officers that are armed, and customers who need help. In case of an emergency, such as the presence of an active shooter, the law enforcement officers can be notified of the shooter's exact location (determined using cameras) so that they can take proper actions, employees can be instructed to assist customers and to commence emergency protocol, and customers can be given specific instructions on their phones based on their location (e.g., the nearest and safe escape route or a safe hiding place).

4.3 EyeFi System Design

4.3.1 Overview

EyeFi is a framework that fuses information from visual domain captured from camera and CSI measurements of WiFi packets to jointly track human motion trajectories and identify them

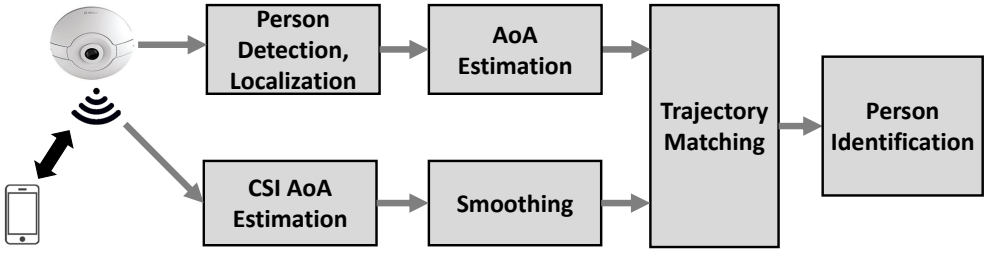


Figure 4.1: An overview of EyeFi system architecture.

for many applications. Using on board computer vision algorithm, the camera detects people, estimates their location and motion trajectories, but unable to identify and re-identify people across time and/or multiple cameras without a shared field of view across cameras. However, WiFi provides a way of identification through user-specific information, i.e., the MAC address of the user's smartphone, but the derived motion trajectory is coarse grained and inaccurate. EyeFi exploits the properties of these two sensing modalities to fuse the trajectories obtained from both for fast and accurate person identification across time and space. The system primarily consists of a camera and a WiFi sniffer. EyeFi does not require installation of any apps or beacons on the users' smartphone and does not add additional overhead to the phone.

A high-level architecture of EyeFi is shown in Figure 4.1. A surveillance camera with a WiFi chipset is installed at the desired location and it overhears WiFi packets of intended subjects like shoppers. Smartphones generate WiFi packets after connecting to the local WiFi access point. When a smartphone is not connected to an access point, it still generates WiFi packets to discover nearby access points. These packets are captured by EyeFi along with CSI information, which is used to estimate AoA of the WiFi source. However, since the estimated AoAs are very noisy, they are further processed to smooth the motion trajectory. Meanwhile, the camera reports the locations of detected human subjects (which may contain more/less people than the number of smartphones that the WiFi unit has detected) and their motion trajectories. Both the trajectories from the camera and the WiFi are sent to the trajectory matching module that identifies people using WiFi MAC addresses by performing cross modal trajectory matching.

4.3.2 Motivational Experiments

EyeFi is motivated by the poor performance of existing person identification solutions. For example, a possible alternative to EyeFi is to use a camera-based solution that uses facial recognition to track people across time and locations. However, cameras installed in public places like a retail store can barely see the faces of the customers. In order to understand the performance of a camera-based system, we apply a facial recognition algorithm on a video feed that we collected during our empirical study.

Figure 4.2 shows a frame from our video feed which contains eight human subjects highlighted using red rectangular boxes. We use a Python-based facial recognition software [2] to detect faces in this frame. The result is catastrophic. The software detects 0 faces after running the algorithm on the entire video. This is because as we can see in the example video frame, a human subject can be facing away from the camera and his dress and floor color can be very similar – which poses an additional challenges to object recognition and matching in a purely vision based domain. Also, existing pre-trained vision-based models do not work well with panoramic images.

To complement the vision-based system, one can add WiFi-based localization to the system by running the SpotFi [69] algorithm on the collected CSI data. However, based on our experiments, the Matlab implementation [7] of the SpotFi algorithm (provided by the authors) requires around 1.5 - 2 seconds to generate AoA estimation for a single WiFi packet. For 8 hours of continuous WiFi stream at the data rate of 20 packets per second, the total number of WiFi packets is $576,000$. With 1.5 seconds computation time for each packet, the AoA estimation requires 240 hours. Even though the computation time can be reduced by using a more efficient implementation, the computation time will still be too long to be viable for processing a large number of WiFi data points for the intended application. Based on these initial experiments, we develop EyeFi to overcome these limitations and to achieve a faster, accurate, and practical solution that works in real-life scenarios.

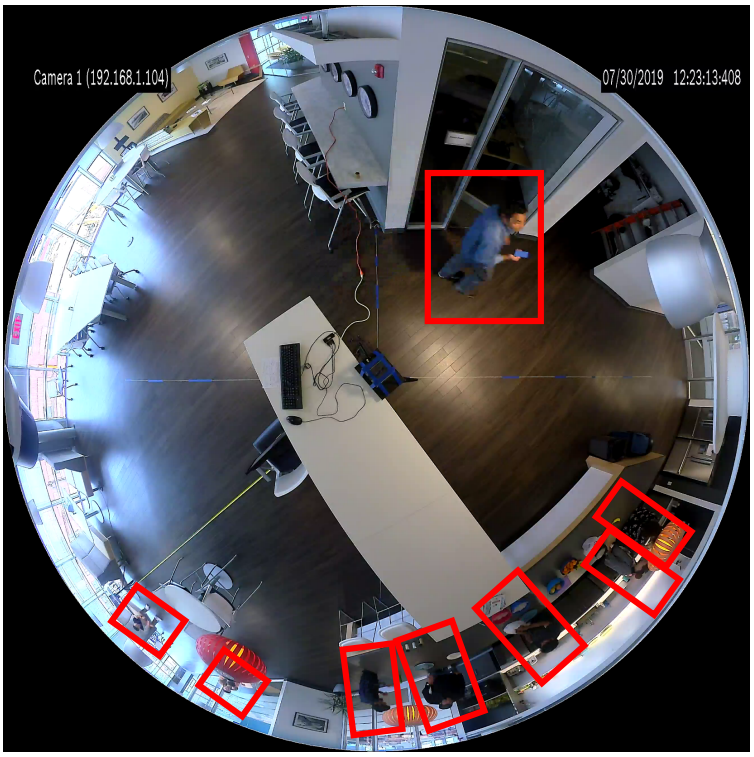


Figure 4.2: Facial recognition software can not recognize any of the 8 human subjects present in the view. All figures best viewed in color.

4.4 Algorithm

EyeFi is a modular system that combines information from visual domain with RF domain. For camera based person detection, tracking, and trajectory estimation, EyeFi uses the proprietary software that comes with Bosch Flexidome IP Panoramic 7000 camera. Our evaluation shows that the existing firmware of the camera can estimate AoA of individuals with an average of 1.03 degree error. EyeFi is agnostic of underlying computer vision technique of person detection and tracking as long as the accuracy is similar. So, we focus on WiFi based AoA estimation, trajectory smoothing, and cross modal trajectory matching. However, we use the camera based location information to improve accuracy and execution speed of WiFi based AoA estimation. In this chapter, we describe each of these components in detail.

4.4.1 Camera Assisted WiFi Based AoA Estimation

WiFi communication produces Channel State Information (CSI) which can be used to estimate the Angle of Arrival (AoA) of the incoming WiFi signals. Methods like SpotFi [69] extend computationally expensive MUSIC algorithm which uses linear algebra to decompose and estimate AoA. However, from our experiments, the SpotFi algorithm is evidently slow and does not work well in large AoA scenarios (e.g., 60°- 90°). Examples are given in Chapter 4.6.2.

To reduce the computation time and to improve overall AoA estimation performance in order to be viable for EyeFi, we seek a data-driven machine learning-based approach to estimate AoA from CSI data. To that end, we try different neural network architectures and after observing similar performance of multiple complex networks, we choose a fully connected neural network that takes CSI data as input and regresses the AoA values (shown in Figure 4.3) as it performs equally well. For the CSI data, there are 90 complex numbers for each packet, which correspond to 30 subcarriers from 3 antennas. We format these 90 complex numbers into a vector of 180 numbers which represent the real and imaginary parts. The neural network has 6 hidden layers in addition to the input and output layer. For the activation function, we use Leaky ReLU [81] for improved performance [134] and to accommodate the negative value of the imaginary part:

$$y = \begin{cases} x, & \text{if } x \geq 0 \\ \text{slope} \times x, & \text{otherwise} \end{cases} \tag{4.1}$$

We also use dropout [113] to reduce overfitting. We use L1 loss function and Adam optimizer for training.

For training the neural network, we take inspiration from knowledge distillation, more specifically, the teacher-student model where the student network learns from the soft labels of a teacher model [**?**]. We treat the SpotFi [69] algorithm as the teacher model for the training purpose. Even though the goal of our neural network is to regress AoA using CSI and we provide hard label of AoA from camera as ground truth, forcing the network to regress the AoAs of the SpotFi, which are the soft labels in our case, in addition helps to ensure the network is forced to learn

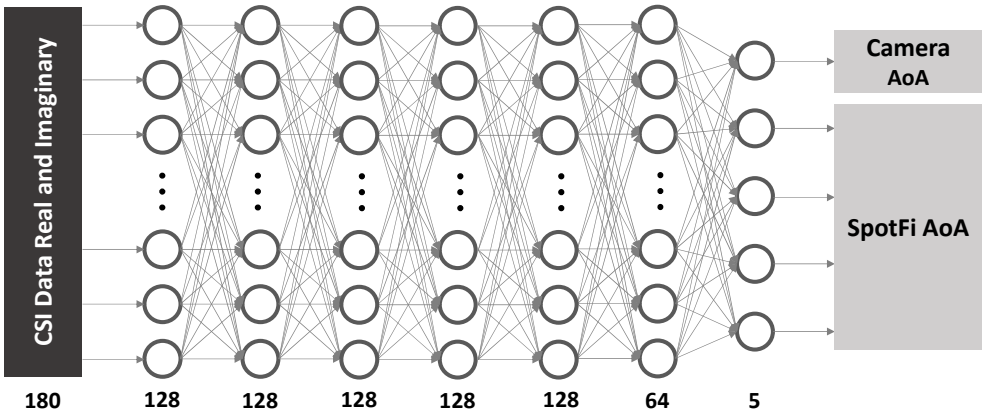


Figure 4.3: Neural network model used in the AoA estimation. Training data includes the ground truth AoA from camera and SpotFi generated AoA data. There are 6 hidden layers and the number of neurons for each layer is listed underneath.

multipaths as accurately as the teacher model (SpotFi) and hence has a better chance of generalizing to a different environment. As shown in Figure 4.3, the output of the network is a vector of size five as the network regresses to one AoA from camera and four multipaths from SpotFi. We test our hypothesis regarding the need for a teacher-student model and find that such a model helps to improve AoA estimation accuracy and helps with generalization in a different environment as described in Chapter 4.6.2.

4.4.2 WiFi based Trajectory Smoothing

The estimated AoA from CSI data is noisy – which we can see from Figure 4.4, where both the estimations from SpotFi and our neural network generated ones show similar characteristics. Such noisy characteristic can be caused by sensor measurement noises, body-shadows, and multipath effects. Even in a controlled environment, the noisy situation improves but is not eliminated. To address this issue, we smooth the data to better align with the ground truth.

Typically smoothing is based on the idea that noise has a certain distribution and added to the underlying real data. For such data, employing moving average usually achieves a good result. However, the data from the AoA estimation does not show such a distribution. Applying moving average in our time series data causes the result to bias towards the noisy direction. An example

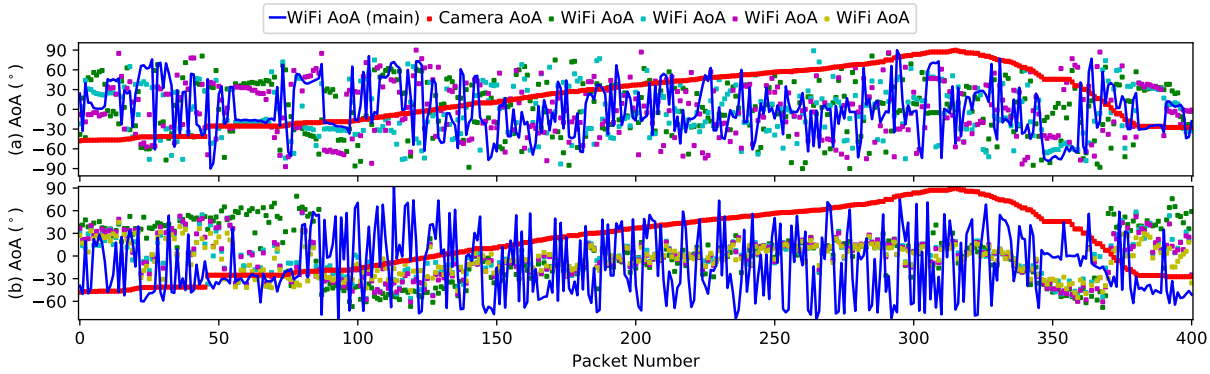


Figure 4.4: Both outputs from (a) SpotFi and (b) neural network exhibit noisy characteristics. Red represents the ground truth from camera and blue represents the first multipath from SpotFi (in (a)) and neural network (in (b)) generated AoA, respectively.

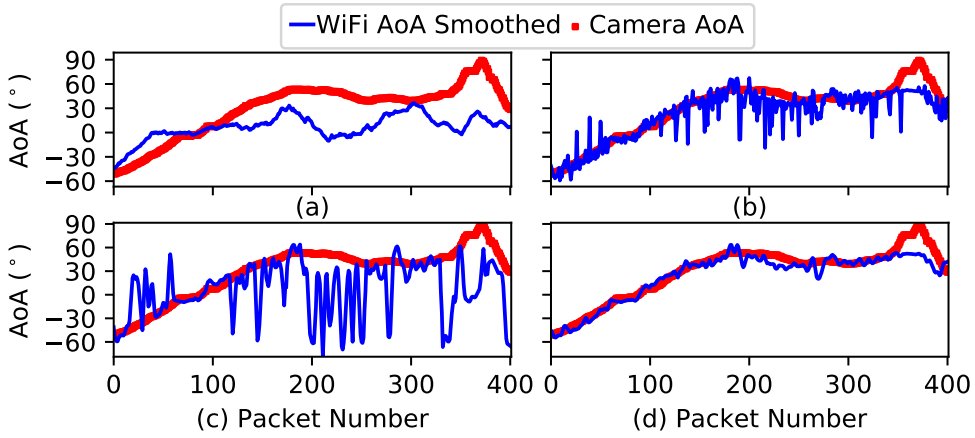


Figure 4.5: Comparison between different smoothing techniques. (a) is smoothed with moving average, (b) is smoothed with our variance based initial smoothing, (c) is AoA data smoothed with LOWESS, and (d) is after applying LOWESS on smoothed data from (b).

of applying moving average algorithm on the data from Figure 4.4(b) is shown in Figure 4.5(a). We can see that the smoothed results are biased toward the noisy direction.

For non-parametric based methods such as Locally Weighted Scatterplot Smoothing (LOWESS) can produce better results in some cases as it does not assume the data fits some specific distribution. However, in our case, the AoA data can become extremely noisy such as around packet number 200 - 250. These abrupt fluctuations can severely distort the smoothed result as shown in Figure 4.5(c) after applying LOWESS. Even though it improves over the moving average approach, it is still too noisy to match with a camera based trajectory.

To achieve the best possible smoothing, we develop a two-stage smoothing pipeline. The first stage addresses the noise causing abrupt fluctuations. We define a smoothing window of length N with the targeted smoothing data point $x_t \in X$ at the middle point, where X is a set of all the N data points within the window. The variance of all the data points in the smoothing window is calculated:

$$\sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2 \tag{4.2}$$

we then calculate the variance without the targeted data point:

$$\sigma_{\hat{X}}^2 = \frac{1}{N}\sum_{i=1, i\neq t}^{N}(x_i - \mu_{\hat{X}})^2 \tag{4.3}$$

If the difference between the two variances $\delta = \sigma_X^2 - \sigma_{\hat{X}}^2$ is larger than a threshold λ (we set to 1), we treat the target data point as an abrupt noise and replace it with:

$$x_t = \begin{cases} \mu_{X_{biased}}, & \text{if } \sigma_X^2 > \lambda_{var} \\ \mu_X, & \text{otherwise} \end{cases} \tag{4.4}$$

The σ_X^2 and λ_{var} are variance of the smoothing window and a threshold (set to 300) to determine if this smoothing window is a volatile region. Our empirical study shows that as the neural network learns the multipaths of SpotFi, the estimated AoAs have larger variances when the phone is within larger AoA ranges. Example of such areas can be seen in Figure 4.4(b) packet 150-225 and 325-375. As they are in the large AoA range, the noise can be large and toward the opposite direction. As a result, we use $\mu_{X_{biased}}$ to replace the targeted data point when $\sigma_X^2 > \lambda_{var}$ is met. X_{biased} is determined as follows:

$$X_{biased} := \begin{cases} \{x \in X : x \geq X_{median}\}, & X_{median} \geq 0 \\ \{x \in X : x < X_{median}\}, & X_{median} < 0 \end{cases} \tag{4.5}$$

The X_{median} is the median value of X (considering all N AoA values) in that smoothing window. Expression $\{x \in X : x \geq X_{median}\}$ means elements in X that are larger than their median value. $\mu_{X_{biased}}$ is the average of X_{biased} from Equation 4.5. For the second case of Equation 4.4, μ_X is the sample mean of all the AoA data points in the smoothing window. Figure 4.5(b) shows the result of the variance-based smoothing performance over the first AoA (i.e., the first element of the output vector) generated by the neural network. As we can see from the figure, the noises causing abrupt fluctuations are largely addressed. For the second stage of smoothing, we apply LOWESS to the smoothed data and the results are shown in Figure 4.5(d). The mean and median values of the absolute differences between the smoothed AoA data and camera generated ground truth AoA of all the figures in Figure 4.5 are shown in Table 4.1. It shows that applying LOWESS after variance based smoothing significantly reduces the mean and median error of AoA estimation. We perform a detailed evaluation in Chapter 4.6.3.

	Moving Average	**Variance based**	**LOWESS**	**Variance based + LOWESS**
Mean (°)	19.45	12.39	22.77	10.14
Median (°)	15.47	8.33	10.93	6.40

Table 4.1: Mean and Median error of AoA estimation after applying different smoothing algorithms on the example data shown in Figure 4.5

4.4.3 Identification Through Trajectory Matching

To identify individuals, EyeFi performs a cross-modal trajectory matching. The simplest approach is to apply Euclidean distance between two trajectories to find the shortest distance,

$$d_{j,k} = \sqrt{\sum_{i=1}^{N} (T_{i,j} - T_{i,k})^2} \tag{4.6}$$

where $T_{i,j}$ and $T_{i,k}$ are the AoA trajectories for subject j (computed using camera) and k (computed using WiFi CSI), respectively from a window of size N. By ranking the $d_{j,k}$ we can find the matched one with the shortest distance.

If each subject walks differently, Euclidean distance could be enough to identify subjects as long as the AoA estimation from each sensing modality is accurate. However, to identify subjects that have a similar path, we also take into consideration of the rate of the change in AoA trajectories. Specifically, we use polynomial functions to represent the trajectories and match with the desired subject. For a matching window of size N, we have can represent the segment with:

$$y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d \tag{4.7}$$

where $x \in X[1, N]$. The polynomial fitting problem is also a smoothing operation, where small noises are filtered out. The estimated y data points are used for calculating the Euclidean distance between the trajectories from WiFi and camera after fitting polynomials. Also, instead of using the standard Euclidean distance function, we apply weights to each AoA data point. Using the polynomial function, we can find the rate of change at every data point R_i. Then, we calculate the absolute rate of change of differences $\hat{R}_i = |R_{i,j} - R_{i,k}|$ for each pair of trajectories. Then the weighted distance is calculated as follows.

$$\hat{d_{j,k}} = \sqrt{\sum_{i=1}^{N} (T_{i,j} - T_{i,k})^2 \otimes \hat{R}_i} \tag{4.8}$$

Here, the operator $\otimes$ represents element-wise multiplication. We only apply weighted Euclidean distance to smaller windows (less than 200 packets) as using it for larger matching windows will smooth out sharp trajectory changes that will deteriorate the performance of the matching. For larger matching windows, the standard Euclidean distance is used as it provides good performance and less computation overhead.

4.5 Data Collection

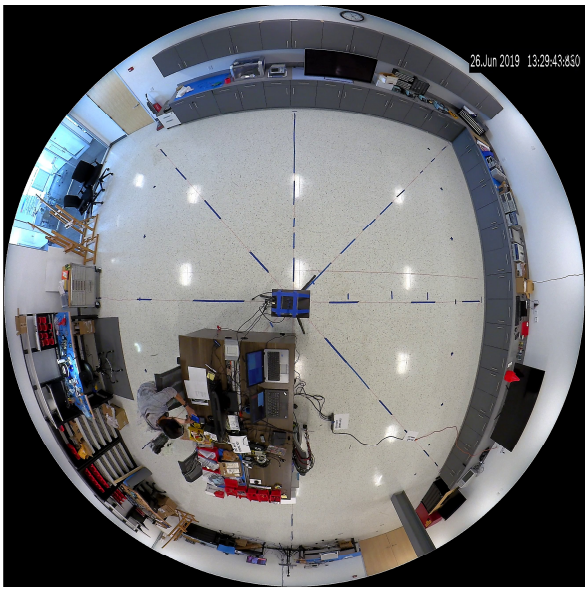
4.5.1 Hardware and Software Setup

In this work, we use Bosch Flexidome IP Panoramic 7000 camera to collect vision data and Intel 5300 WiFi Network Interface Card (NIC) installed in an Intel NUC to collect WiFi data. The camera is mounted on a ceiling at a height of 2.85 meters whereas the WiFi card is located at the same location as the camera but at a height of 1.12 meters forming a unified coordinate system. We collect data in two different locations as shown in Figure 4.6. Figure 4.6a shows the lab area where the majority of the data are collected, and Figure 4.6b shows the Kitchen area where the collected data are used for testing only (not used for training). The lab area is rectangular having dimensions of 11.8m x 8.74m and the kitchen area is irregularly shaped with maximum distances of 19.74m and 14.24m between two walls. The kitchen also has numerous obstacles and different materials that pose different RF reflection characteristics. The change in the environment creates a vastly different RF characteristic that is used to test the robustness and generalizability of the system.

To collect WiFi data, we set up a Google Pixel 2 XL smartphone as an access point and connect the Intel 5300 NIC to it for WiFi communication (both are shown in Figure 4.7). The phone transmits 20-25 packets per second to the NUC. Such a low packet transmission rate simulates realistic scenarios, e.g., apps in the phone are receiving notifications. We use Linux CSI Tool [55] to record the CSI information from the Intel 5300 WiFi NIC on Intel NUC.

4.5.2 Collected Dataset

We collect data over multiple days and vary the number of people present in the scene. In the end, we have transmitted over 1.2 million WiFi packets and collected corresponding CSI values of over 13 hours. We also have over 15 different individuals holding the phone to capture various ways people hold phones, their walking patterns, and different heights. In addition to a single person walking in the scene, we also have multiple people walking simultaneously.



(a) Lab area.

(b) Kitchen area.

Figure 4.6: Data collection environment seen from panoramic cameras. (a) lab area which is large and rectangular shaped, (b) kitchen area which is irregular and has many obstacles. An Intel NUC is located at the middle of each figure forming a unified co-ordinate system of the camera and NUC.

4.6 Evaluation

4.6.1 Evaluation Setup

We evaluate how each component of our system performs. We divide our dataset into different sets for this purpose. Part of the data collected from the lab area is used for our training and algorithm development. Data collected from the kitchen area are only used for testing the robustness and generalizability of the system. To identify the subject with the phone and test the accuracy of identification, we use the data from our camera system as ground truth. The camera detects individuals and provides the (x, y) coordinates of each of the detected subject, which is used to estimate AoA.

We first evaluate the accuracy of the camera in terms of its ability to estimate (x,y) coordinates and AoA by standing in 16 different locations throughout the lab and comparing the differences between the camera and our actual measurements. We find that the average error of the camera is (0.32, 0.29) meters for estimating (x,y) coordinates and 1.03 degree for AoA. It shows

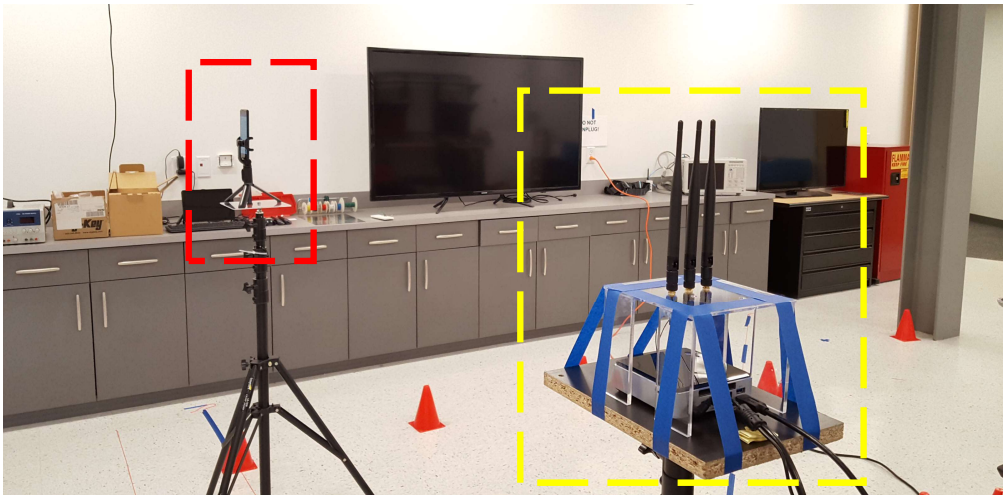


Figure 4.7: Data collection equipment. On the right (yellow box) is the Intel NUC with Intel 5300 WiFi card installed with three external antennas. On the left (red box) is the Google Pixel 2 XL phone for communication. Note that all the data are collected while a subject is holding the phone in his/her hand.

that we can use the location data from the camera system as ground truth for the training and testing.

4.6.2 Neural Network Based AoA Prediction

4.6.2.1 Training Data

As discussed in Chapter 4.4.1, our training data consists of SpotFi generated AoAs and camera generated ground truth AoA. To prepare the dataset, we need to address the phase offsets between the 3 RF chains in the Intel 5300 NIC. [139] states that the phase offsets between these chains are deterministic and the offset between two RF chains only poses two possible values. We determine the two values based on our own measurements using methods stated in [139].

During the data collection, it is impractical to measure the phase offsets each time the system reboots. To address this issue, we apply all possible (four) combinations of the phase offset when calculating AoAs using the SpotFi algorithm. Once all the SpotFi AoA data are generated, we find the correct phase offset by choosing the one with the smallest mean absolution difference

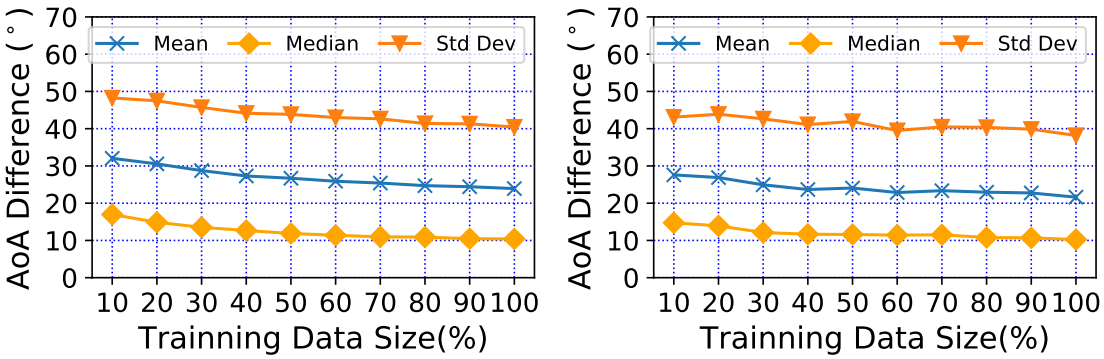
between the AoA from SpotFi and AoA from the camera. Then, we use the SpotFi data with correct phase offsets calibrated to train our neural network models.

4.6.2.2 Neural Network Models

For the AoA estimation, we train our neural network models with different outputs to evaluate the performance of the teacher-student model and whether the neural network can learn the SpotFi algorithm. We train three different neural network models: *SpotFi Only NN* that uses SpotFi generated AoA results for training, *SpotFi + Camera NN* that is our teacher-student model, and *Camera Only NN* that uses only the camera generated ground truth AoA for training. All the neural network models are trained with the same training dataset with roughly 400,000 WiFi-camera AoA pairs.

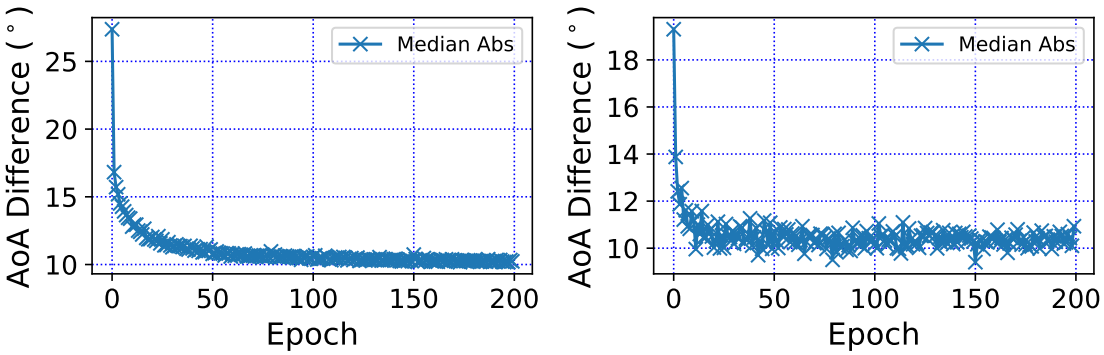
To evaluate how the number of training samples affect the neural network performance, we train our *SpotFi + Camera NN* teacher-student model with different size of the training dataset, and test the performance of the neural network on the validation dataset (roughly 58,000 WiFi-camera AoA pairs from the lab) and a subset of the data collected in the Kitchen area (roughly 22,000 WiFi-camera pairs). The results are shown in Figure 4.8. The X-axis is the percentage of the dataset (roughly 400,000 WiFi-camera AoA pairs) used to train the neural network and Y-axis is the absolute difference between the neural network prediction and ground truth AoA. We report mean, median and standard deviation of the difference here. In Figure 4.8a, we see that the performance of the neural network on the validation dataset improves as the size of the training dataset increases. The same trend are also being observed with the Kitchen area dataset. The improvements become minimal after 80% of the dataset is being used for training. It shows that our training dataset is large enough to train our neural network model.

We also test if our epoch is large enough to complete the training of the network for improving the AoA estimation by calculating the mean and median AoA difference after each epoch with both the datasets from lab and the kitchen. The results are shown in Figure 4.9. As we can see, the performance improvements level out when the number of epochs is larger than 75. The



(a) Result in Lab area. (b) Result in Kitchen area.

Figure 4.8: Neural network performance for different size of training dataset.



(a) Result in Lab area. (b) Result in Kitchen area.

Figure 4.9: Neural network performance during training.

similarity between Figure 4.9a and Figure 4.9b shows that our neural network model generalizes to a difference environment as it learns.

	Lab		**Kitchen**	
	Mean (°)	Median (°)	Mean (°)	Median (°)
SpotFi	59.17	58.08	50.14	44.03
SpotFi Only NN	62.97	63.29	45.55	46.01
Camera only NN	31.87	14.57	36.61	20.63
SpotFi + Camera NN	**30.56**	**13.98**	**35.08**	**18.72**

Table 4.2: Mean and Median on AoA estimation performance of different neural network models and SpotFi on data collected from lab and kitchen area.

We test all three models on an unseen test dataset. Table 4.2 shows the mean and median of the absolute AoA difference between the predicted one and ground truth for the neural network models and SpotFi. From the table, we see similar performance between the *SpotFi* algorithm

and the *SpotFi Only NN* that demonstrates that the neural network can learn the SpotFi algorithm. *Camera Only NN* and *SpotFi + Camera NN* both show improved performance over *SpotFi* and *SpotFi Only NN*, and our teacher-student model *SpotFi + Camera NN* shows the best performance. The neural network learns the underlying relationship between the CSI and AoA, and the teacher-student model further improves the generalizability of the neural network on different test cases and environments.

4.6.2.3 Robustness and Efficiency of AoA Estimation Neural Network

We test the robustness of our teacher-student neural network model on both unseen CSI data collected in the lab and kitchen area using over 30,000 and 20,000 WiFi packets, respectively. The results are shown in Table 4.2. It shows that the performance of the neural network is comparable across different environments and shows better results than SpotFi. The difference between the SpotFi results in two environments can be because of different environment RF characteristics and percentage of the phone in different AoA range (SpotFi performs worse in large AoA range).

In addition to the performance improvement on the AoA estimation, the neural network also improves the execution speed. SpotFi takes around 1.5 seconds to estimate AoA per WiFi packet, thus making it difficult to be useful in a real-time solution. Based on using 22,854 WiFi packets collected from the kitchen, we see that our neural network is around 3809 times faster than SpotFi, which can be further improved using GPU computation and batch data, enabling EyeFi a real-time solution.

4.6.3 Smoothing

We use the kitchen data (different environment from the training) from the previous chapter to evaluate the performance of our smoothing algorithm. We measure the mean and median absolute errors between the smoothed AoA data and camera derived ground truth AoA data. The results are shown in Table 4.3. In this table, *NN* is the AoA estimation from our neural network model, *Variance Based* is the results after our first stage smoothing, *Variance + LOWESS* is our

full smoothing stack, and we also report the result by only applying the LOWESS algorithm in *LOWESS*. From this table, we can see that smoothing improves on the original neural network generated AoA estimation. Our smoothing stack produces the best results with both lowest mean and median absolute errors.

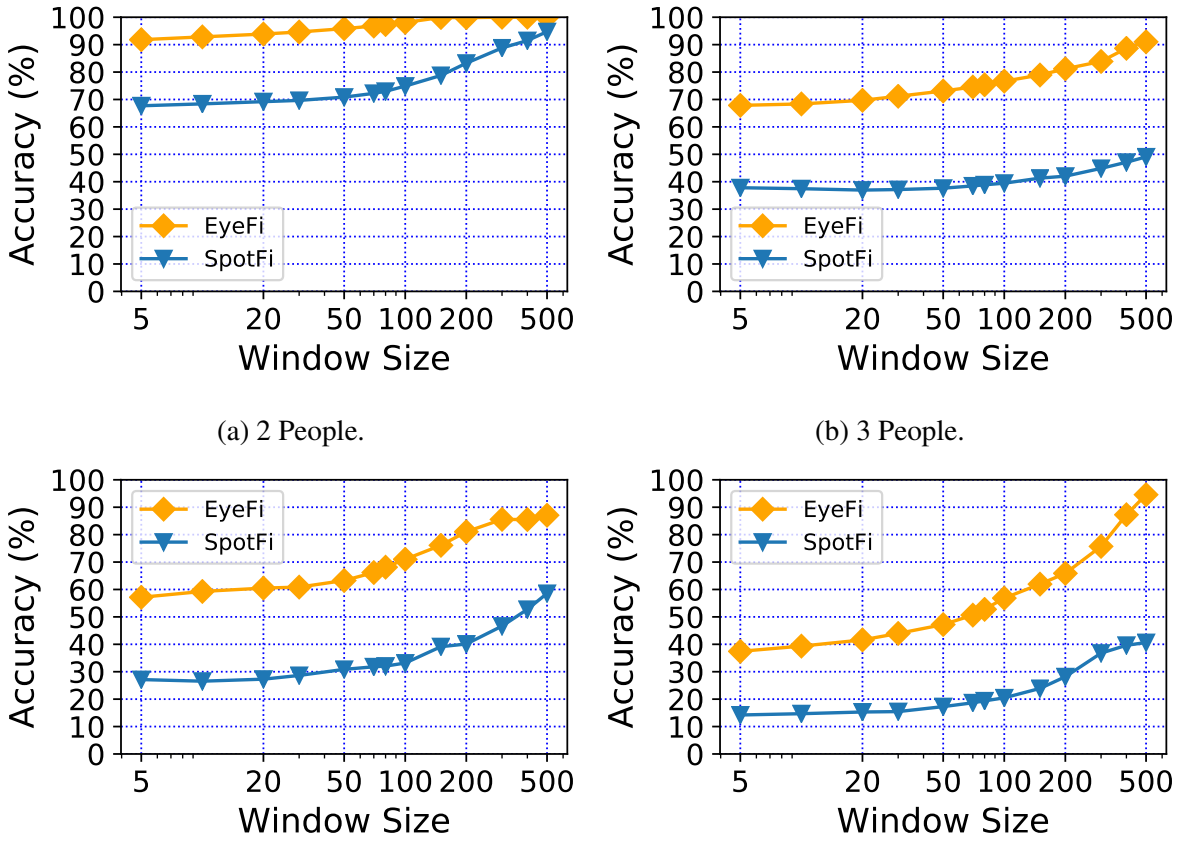
	NN	**Variance Based**	**Variance + LOWESS**	**LOWESS**
Mean (°)	24.61	12.98	**10.80**	12.17
Median (°)	11.76	9.59	**7.91**	8.09

Table 4.3: Smoothing performance with different smoothing techniques and combinations.

The results from LOWESS based smoothing is better than our variance based smoothing in Table 4.3. This is different from the results presented in Table 4.1 which is the statistics for Figure 4.5. This is because in Figure 4.5, we choose a segment of the data where the neural network introduced more noise and the ground truth is in the large AoA range.

4.6.4 Identification

To evaluate the performance of identification, we collect datasets with multiple people walking in the scene simultaneously. The results are reported in Figure 4.10. We consider *SpotFi* as our baseline. It uses AoA data generated by the SpotFi algorithm and identifies the subject through Euclidean distance. *EyeFi* uses weighted Euclidean distance for identification on the two staged smoothed data generated by the teacher-student neural network. As stated in Chapter 4.4.3, we only apply our weighted distance for the sequence size of fewer than 200 packets. The accuracy is calculated by sliding a window of size N along the time-series data, identify the subject with the AoA data within the window range, then computing the percentage of the number of accurately identified time-segments throughout the test dataset. The window size starts at 5, which means using only 5 packets. For example, in Figure 4.10a, we can successfully identify the subject 90% of the time in a normal 2 people scenario. Since we collect data at a rate of 20-25 packets per second, 5 packets represent a duration of 0.25s or less. As more people are present in the scene, the difficulty of identification increases. As shown in Figure 4.10b, the accuracy is



(a) 2 People.

(b) 3 People.

(c) 5 People.

(d) 10 People

Figure 4.10: Identification accuracy for different number of people.

worse than the 2 people case. The performance further drops with 5 people in the scene as shown in Figure 4.10c.

Figure 4.10d shows the result of identification with 10 people walking simultaneously. The results show that EyeFi can still identify even though the accuracy drops in smaller windows. The identification accuracy of 10 people is a bit higher than that of 5 people when the window size is 500. This can be due to the human subject who is holding the phone walks in a different path than the rest of the group. This allows the identification can be 100% accurate when the window size is large. However, in real situations, 500 packets span about 25 seconds during our data collection. With an average human walking speed of 1.1 meters per second, a person can walk about 27.5 meters during that time frame.

Given the number of packets large enough and the path is different enough to be distinguished from other trajectories, EyeFi could achieve near 100% accuracy in identification.

4.7 Discussion

4.7.1 Limitations of Our Proposed Solution

There are two major limitations of EyeFi. First, it requires having a smartphone with WiFi communication. Second, it is unable to identify if multiple people walking together in a way that results in similar AoA trajectories. For the first limitation, the usage of the phone is very common but not all users will connect to public WiFi such as the ones provided by the store in our case. However, EyeFi can still receive WiFi packets generated by phones while discovering local access points. This property allows the system to estimate AoA and track the subject. However, if the WiFi interface is disabled or the smartphone is not generating any WiFi traffic, then EyeFi will not be able to identify people. But the camera can still provide analytics on behavior of people to some extent. For the second limitation, in retail stores, if the trajectories are the same for a few people, identifying either could provide the same analytic regarding customer behavior.

4.7.2 Motion Across Multiple Cameras

In areas where multiple cameras are deployed with or without shared field of view, EyeFi can leverage existing camera-based re-identification algorithm [144] to identify the same user across multiple cameras to provide full trajectory in the covered area. By leveraging WiFi, EyeFi can enable identification at different spatio-temporal segments, thus reducing the search space for the vision based identification and enable more accurate long-term re-identification.

4.7.3 Generalization to Multiple Phones

The presence of multiple phones in the scene will have minimal effect on the performance of the system. This is due to the nature of WiFi communication that enables multiple devices to

talk to each other without interference. This also means that the CSI information collected at the WiFi unit for each device has the same quality. EyeFi only relies on CSI information and does not require a high transmission rate which further reduces the impact of multiple devices. During our data collection, all other WiFi devices and communications are functioning normally and no effects are observed.

4.7.4 Effects of Using Phones

Differs from most previous works that use Intel 5300 NIC for both communication devices, we use a smartphone at one end. The internal antenna design is vastly different from the external antenna used with Intel 5300 NIC devices and the holding of the phone by a human subject also affects the WiFi signal quality. During our experiments, we also observe poor performance and stability issues in AoA estimation with phones in comparison to Intel 5300 NIC with external antennas, and the AoA results are worse than that is reported in previous works such as [69]. Note that Linux CSI Tool [55] offers the best performance and stability using injection mode, which is currently unavailable with the phone.

4.7.5 Effects of Environment

In our evaluation, we test EyeFi in two different environments that shows stable performance among them. This demonstrates the robustness and generalizability of our system. However, different environments can affect the performance of WiFi AoA estimation if the environment is very crowded with obstacles between the phone and WiFi access point to create a non-line-of-sight situation. In such a situation, the WiFi signal is distorted which degrades the AoA estimation performance. However, if the trajectories between different subjects in the scene are different from each other, the system should be able to identify. In our experiment, there are times where the phone is blocked by human bodies which distort the WiFi signal as well. With our smoothing pipeline, the identification can still perform well.

4.7.6 Privacy Issues

Privacy is a major concern nowadays and we design EyeFi with that in mind. Current camera-based systems mostly use facial recognition for user re-identification across time. However, facial recognition is unreliable in many settings (discussed earlier) and can be racial biased. There are also laws [6] to ban facial recognition to prevent such bias and protect privacy. In contrast to vision-based facial recognition systems to track a user across time, EyeFi only collects the MAC address of the phone and hashes that to obtain a consistent identification marker. Given most human subjects do not change phones very frequently, the hashed MAC address can be used as a reliable marker across time. As EyeFi does not keep the link between a hashed MAC address and its particular user, even if the hashed MAC addresses are compromised, it will be very difficult to use that to identify a particular user.

4.8 Summary

In this chapter, we propose EyeFi, a multimodal system that fuses WiFi and camera data to identify individuals by capturing motion trajectories from each modality. We design a teacher-student based neural network model to estimate AoA accurately, which speeds up AoA estimation by over 3800 times with 30% higher accuracy, enabling EyeFi to be a real-time system. We test the performance in two different environments and find the neural network based AoA estimation is robust to a change of the environment. When evaluating the accuracy of person identification, we see that EyeFi can achieve an average of 75% accuracy across all number of packets in a 2 to 10 people scenario. For future works, we will improve performance for each component and build an end-to-end system that can identify people in real-time.

CHAPTER 5: WIFI ENABLED HUMAN LOCALIZATION IN AUTOMOTIVE ENVIRONMENT

5.1 Introduction

Transportation is changing with technology advancement and demographics shifting [9]. The result is fewer people are driving while more utilize public transportation. As people rely on ride-hailing services, e.g., Uber and Lyft, it becomes increasingly important for drivers and passengers to find each other without a hitch. With the anticipated deployment of autonomous vehicles in the future, user experience is tightly connected with how smooth the pick-up process is. Currently, drivers and passengers use smartphones, which rely on GPS or cellular signals, to locate each other while far apart, and require them to recognize each other while nearby. However, in urban cities and areas like downtown, where there are numerous skyscrapers, GPS and cellular signals often do not work. There are places, e.g., in airports, where the drivers need to come indoors (such as parking garage) to pick up passengers where the building structure blocks GPS signals. Also, in crowded environments like stadiums, airports, theatres, and bars, it is challenging to locate the actual passenger among many people. For unfamiliar environments to both the passengers or drivers, it is no easy task to identify which side of the street is the correct one. If the passenger can not recognize the vehicle, he or she does not know which direction to walk to once the vehicle has stopped at a nearby suitable location. The situation can worsen due to lack of visibility, e.g., at night and during bad weather (such as rain, storm, and snow).

For a better user experience, it is beneficial to let both the driver and the passenger know the location of each other so that they can complete the pick-up process efficiently. Numerous technologies can be used to improve such an experience. For example, the vehicle can use a camera and facial recognition [100] to identify the passenger subsequently compute the location.

However, facial recognition requires the passenger to upload his or her photo, which can be privacy-intrusive. Moreover, for facial recognition to work, the passenger needs to be within the camera's field of view and occupy enough pixels to be successfully recognized. This situation is usually when the passenger is already close and unobstructed, thus defeating the purpose of passenger localization. These requirements reduce the practicality of such facial recognition systems to be deployed in the real world. One can also ask the user to scan the surroundings with his or her phone, and then the server can perform 3D reconstruction [62] and matching [77] to the previous established real-world model to compute the exact location of the passenger. This approach can provide much better accuracy and work without the driver and passenger being close to one another. However, this is a computation-intensive approach that can be unrealistic for the server to process all the requests on the ride-hailing platform. Besides the computation load, this method also requires the world to be digitized and constructed to allow such matching.

In this chapter, we propose to utilize smartphones – which the passengers will mostly like to possess for the ride-hailing booking – and WiFi-enabled dashcam – increasingly crucial for safety and legal purposes – to localize where the passenger is relative to the position of the vehicle. We call our proposed solution CarFi. CarFi does not require the passenger to upload any photos of him or her, and neither needs the photo of the surrounding area, which protects the passenger's privacy and reduces the computation load. Instead, CarFi works by providing where the passenger is in the four quadrants surrounding the vehicle and allowing the driver and passenger to find each other more easily. To this end, CarFi uses WiFi communications between the passenger's smartphone and the dashcam onboard the vehicle. The usage of the dashcam is for the purpose of standalone devices that can be installed on any vehicle, but the proposed system does not exclude vehicles that have WiFi already installed to localize the passenger.

CarFi uses the WiFi chipset embedded in the dashcam to receive the WiFi packets sent by the smartphone, which the passenger is holding. The system on the vehicle extracts the Channel State Information (CSI) data from the WiFi chipset and performs feature extraction and localization. This system does not require any modification to the vehicle and the smartphone. The WiFi

packets generation can be handled by the ride-hailing application, which can report the phone's MAC address through the server to the vehicle. Thus, the vehicle can identify which phone is the correct one. To successfully localize where the passenger is, CarFi utilizes a small sequence of the WiFi packets, which are jointly used for feature extractions.

This work has the following contributions:

- First, we design and implement a novel localization system by utilizing WiFi signals in an automotive environment. Our CarFi system does not require privacy-intrusive personal information from the passenger nor requires heavy computation load on the server.
- Second, we collected our dataset with over 1,582,939 WiFi packets in 5 different environments. We also annotate our data with ground truth provided by markers on the floor or videos from a drone. This dataset contains multiple scenarios, including standing still, walking around, and driving around for robust analysis.
- Third, we propose a neural network model that takes in multiple features to estimate the phone's location, which the passenger holds.
- Finally, our method can perform location estimation, whereas existing solutions such as SpotFi [69] fail to do. Our features enable the system to generalize while raw CSI data can not. CarFi framework is also computational lightweight, which can be implemented on constrained devices.

5.2 Usage Scenario

We describe a real-world usage scenario of CarFi in this chapter. When a passenger wants to travel to a specific location, he or she uses the ride-hailing app on the phone to book the trip. The server processes the request and finds the driver. The locations of the vehicle and the passenger are determined by their respective location providers, such as GPS on the phone. Once the trip is confirmed, the driver heads toward the passenger's location. As the driver arrives within a cer-

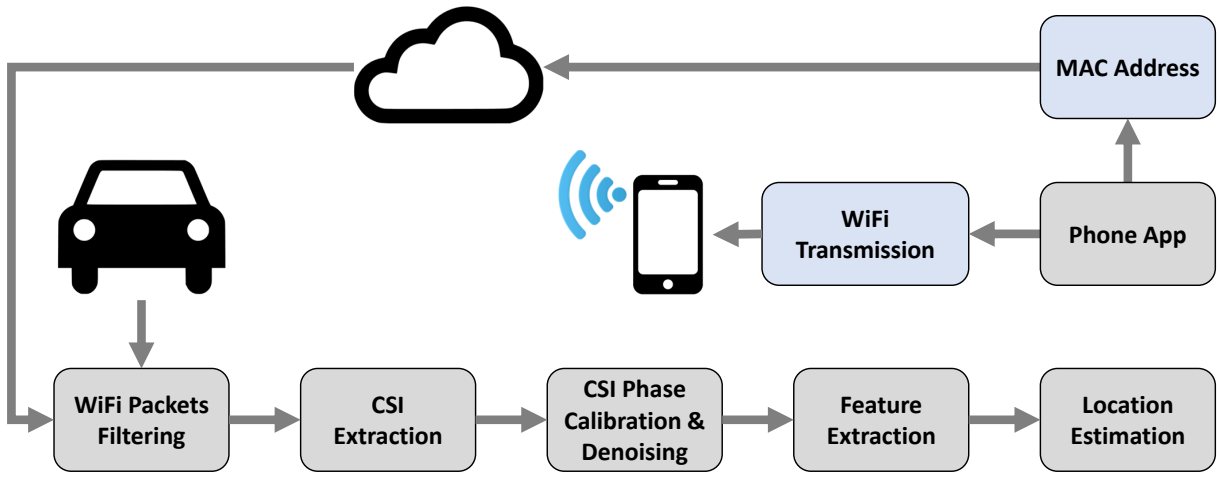


Figure 5.1: CarFi system overview

tain distance from the passenger based on the location data, the passenger's phone will transmit the WiFi packets at a higher packet rate as the ride-hailing app controls it. Meanwhile, the dash-cam starts listening for WiFi packets containing the phone's MAC address. When CarFi system receives the WiFi packets, it extracts the CSI information and calculates the features. Then it estimates where the passenger is, and the driver can proceed to the correct location. If the driver can only park the vehicle at specific parking spots safely, the relative location can also guide the passenger toward the vehicle.

5.3 CarFi Overview

CarFi is a framework that combines the passenger's smartphone, the cloud server, and the WiFi receiver on the vehicle. The phone generates WiFi traffic which can be detected by the WiFi receiver located in the vehicle. The WiFi system in the vehicle receives the WiFi packets, processes them, and localizes where the phone is. The cloud server is the bridge between the vehicle and the phone to exchange the data. This framework does not require the phone to take any photo or video, which minimizes the server's computation, reduces network traffic and power consumption, and protects privacy. Meanwhile, the CarFi system exploits the WiFi signal's properties and estimates the location to provide a better user experience.

The CarFi framework is shown in Figure 5.1. The phone app controls the phone to transmit WiFi packets when the vehicle is within a certain distance. We note that the location of the vehicle and the phone are determined by other methods such as GPS. They are shared via the cloud in real-time. The app also retrieves the phone's MAC address being used and shares it with the vehicle through the cloud server. The WiFi devices in the vehicle constantly monitor the WiFi traffic and filter out the packets containing the specific MAC address. After the WiFi packets filtering, the CSI information is extracted, which is later calibrated and denoised. This step is performed to remove the noises that are introduced by the WiFi hardware and software imperfections. The calibrated and denoised data is used for feature extraction, and these features are used for location estimation.

5.4 Challenges

In this chapter, we discuss the challenges that WiFi localization faces in the automotive environment.

5.4.1 Automotive Environment

When moving WiFi devices from indoor locations to outdoor environments, the characteristics of the environment and effects on the signals change dramatically. One of the biggest issues in an automotive environment is the metal structure of the vehicle body, which can be similar to a Faraday cage. Although the signal of normal radio frequency communication systems has a higher frequency than what the window can block due to its large size (the window opening needs to be smaller than the wavelength), the vehicle's metal surface can still block and redistribute the signal. The effect of the vehicle body on the RF signal is also different from vehicle to vehicle. For example, the front window of the Tesla Model S has metallic costing which interference with RFID transponders (such as toll transponder), so they have an area of the window designed explicitly for such transponders [8].

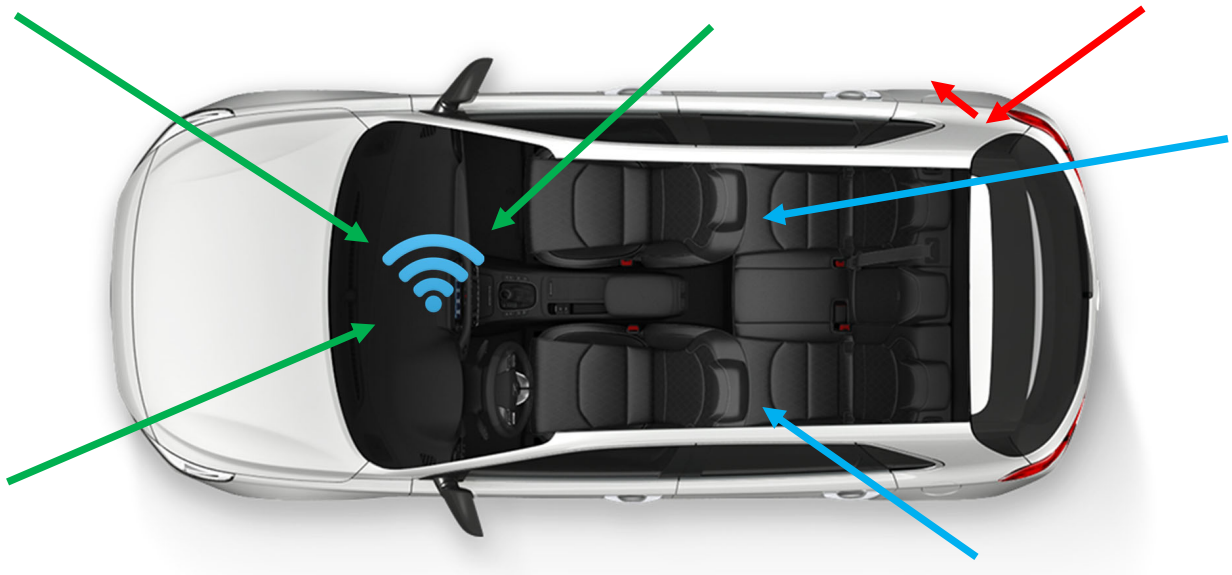


Figure 5.2: Effect of the vehicle body on RF signal. When the receiver is placed on the front dashboard, the signal can come from all directions. Some signals (shown in green) can penetrate the glass window and directly reach the receiver. Some (shown in blue) may penetrate the glass window and be attenuated by the interior seats and human body. Other signals (shown in red) may directly being reflected by the vehicle's mental body.

We show a simple illustration of how the RF signal can be affected by the vehicle's body in Figure 5.2. We can see that some signals (shown in green) can travel directly through the windows and reach the receiver, and some (shown in blue) being attenuated by the interior seats and human body, while others (shown in red) being reflected and blocked by the metal body. We note that this is a simplified illustration of how the RF signal may propagate around the vehicle. Some signals may travel into the vehicle and be reflected, absorbed, attenuated by the metal body, interior seats, and human body.

With such a complex RF environment, the current state-of-the-art method, such as SpotFi [69], can not accurately estimate the angle of the arrival of the WiFi signal. An example of the angle estimated is shown in Figure 5.3. We can see from this plot that when the signal is coming from the front of the vehicle and no other obstacles are blocking or attenuating, the Angle of Arrival (AoA) estimation can work quite well. This is because the WiFi signal can directly travel through the front windshield and reach the receiver antennas. However, when the person who is holding the phone is positioned on the side of the vehicle (roughly 0-3 meters away) and the back of the

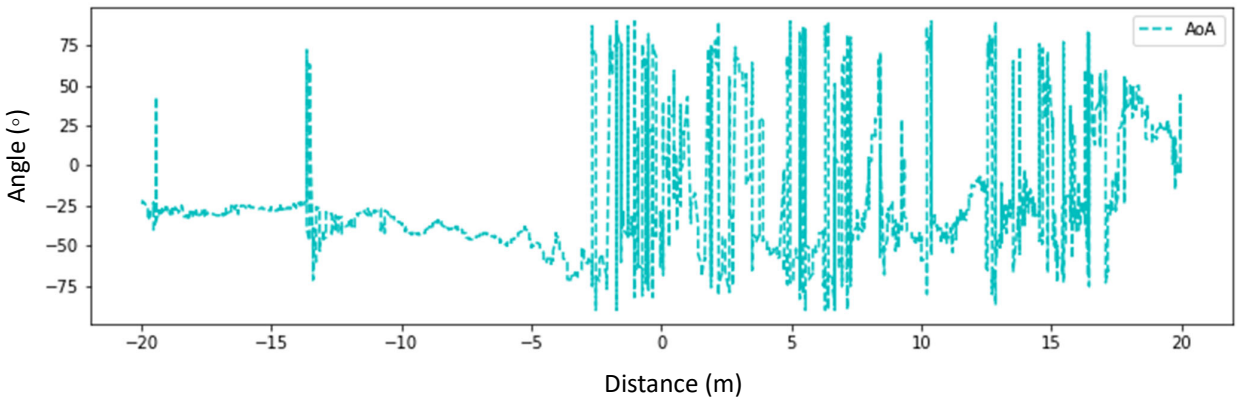


Figure 5.3: SpotFi results with WiFi packets collected inside the vehicle. Only the vehicle and the person holding the phone are present at the scene. The distance is the distance between the WiFi receiver in the vehicle and the phone being held by the passenger. When the distance is smaller than 0, the passenger is in front of the vehicle, and when the distance is larger than 0, the passenger is at the back of the vehicle.

vehicle (beyond 3 meters in the distance), the AoA estimation results become unusable and can not even correctly estimate whether the person is on the left or right side of the vehicle [1]. We note that there are distances where the AoA estimation seems reasonable when the person is at the back of the vehicle. This is because there could be a direct line-of-sight path between the phone and the antenna through the rear windows.

5.4.2 Speed and Time

While we are not assuming the vehicle will approach the passenger at highway speed when they are nearby, we assume that the vehicle will be traveling around a reasonable city driving speed. This means that a speed limit of 40 Km/h (25 miles per hour) can be assumed. This is roughly equivalent to 11 m/s. We also consider the transmission range of the WiFi signal to be around 50 to 100 meters in the real world. Given the human response time is about 1 to 1.5 seconds, we determine that the CarFi system should have a response time of less than 1 second to allow adequate time for the driver to respond and stop safely. As WiFi packets can be transmitted

[1]The antenna are in a linear array and is perpendicular to the length of the vehicle, thus when the phone is on one side of the vehicle, the Angle of Arrival (AoA) will be between 0°to 90°, and 90°to 180°on the other side of the vehicle.

at different rates, we found that the phone can achieve a rate of 100 to 150 packets per second reliably in our empirical study. Therefore, we determined that our system uses 50 WiFi packets, and all of them need to be transmitted within 1 second. These constraints pose extra challenges to the system as more time and packets can usually achieve higher accuracy.

5.5 Features

This chapter describes how some features can be extracted from the WiFi packets and may be used in the estimation.

5.5.1 RSSI and RSS

WiFi signal strength is affected by multiple factors such as the distance between the transmitter and the receiver, obstacles, transmitter signal strength, and the receiver antenna's property. Therefore, we can use RSSI and RSS to represent the signal strength at the receiver antenna. The Received Signal Strength Indicator (RSSI) is an estimate of power in the received signal at the receiver. RSSI is an indicator that is a relative measurement as defined by the manufacture and is normally positive numbers. Received Signal Strength (RSS) is the actual measurement, usually with a unit of dB in WiFi transmission. As there are three antennas in the CarFi system, the signal strength at each antenna varies from one another. This variation is partly caused by the difference in distance to the transmitter, where the path loss of the signal in free space can be estimated with Friis Free Space Propagation loss:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{4\pi^2 d^2 L}, \quad \text{where} \quad G = \frac{4\Pi A_e}{\lambda^2} \tag{5.1}$$

Where the P_t is the transmitting power, $P_r(d)$ is the received power at a distance d, G_t is the transmitting antenna power gain, G_r is the receiving antenna power gain, λ is the wavelength. As we can see from this equation, the power loss is in inverse proportion to the square of the distance; thus, the power loss is severe.

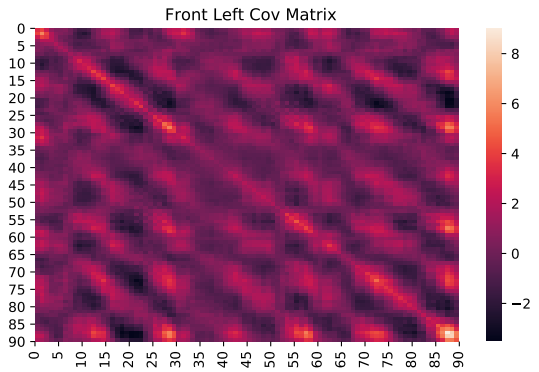
The variance in the signal strength across different antennas can also be caused by how each antenna is affected differently by the environment. For example, as the signal travels to the vehicle where it can be reflected and attenuated, each antenna can be located at constructive or destructive locations, which results in differences in received antenna signal strength.

For each packet P, the WiFi network card reports three RSSI values: $RSSI_a$, $RSSI_b$, and $RSSI_c$, where each value corresponds to one of the three receiving antennas. In addition, the Linux CSI Tool also reports Automatic Gain Control (AGC) which can be used to calculate the actual signal strength in dB.

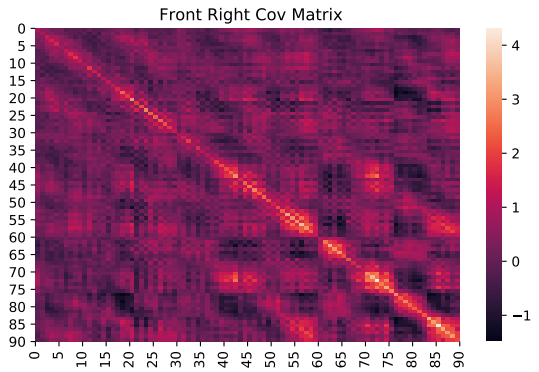
5.5.2 CSI Covariance Matrix

As there are 3 antennas on the receiver side and each antenna is placed at different locations (they are $\lambda/2$ apart, where λ is the wavelength), how the signal is changing over time can be different as they are being affected slightly differently. When the relative location of the phone to the vehicle changes, the signal propagation paths also change. The change in propagation paths can affect the amplitude of the signals arrived at each antenna. Given the signal strength is also inverse to the square of the distance, the closer the antenna can result in more significant changes in the amplitude. Thus the correlation between nearby subcarriers and itself can potentially be higher.

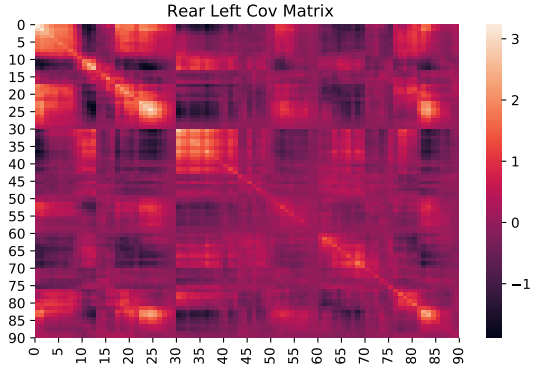
We show an example of the covariance matrix calculated from our collected data in Figure 5.4. These covariance matrices are computed with 50 consecutive packets from each case. In this figure, we can see that the patterns of the covariance matrices are different from each other which are clearly distinguishable. We also show the covariance matrices calculated with all the packets from the same dataset in Figure 5.5. Each covariance matrix is calculated with 10,000 to 18,000 packets. They show that when the phone is on the right side of the vehicle, covariance values between the rightmost antenna subcarriers are higher than the rest, and covariance values between left most antenna subcarriers are higher when the phone is on the left side of the vehicle.



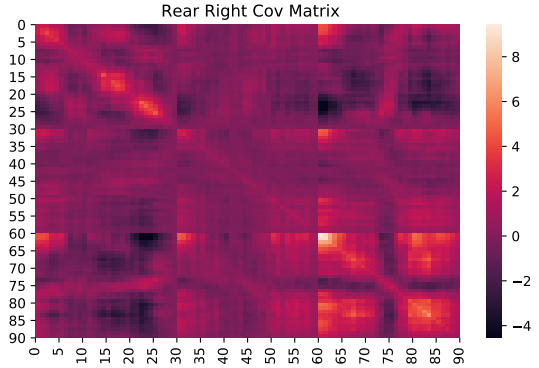
(a) Covariance matrix from front left side of the vehicle.



(b) Covariance matrix from front right side of the vehicle.



(c) Covariance matrix from rear left side of the vehicle.



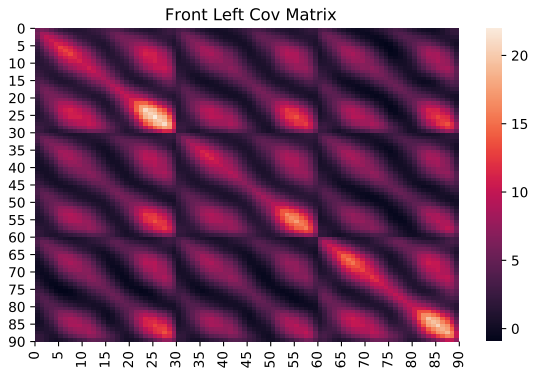
(d) Covariance matrix from rear right side of the vehicle.

Figure 5.4: Example CSI covariance of the received WiFi packets. The covariance is calculated with all 30 subcarriers of each antenna, so the total number of variables is 90. 50 consecutive packets are used for the calculation in each case.

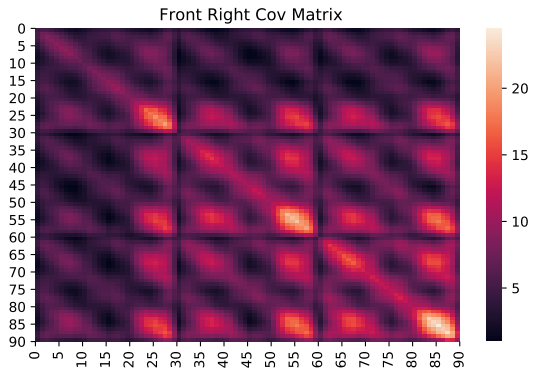
These results show that one can estimate the location of the phone by extract features from the covariance matrices.

5.5.3 Multipath Profile

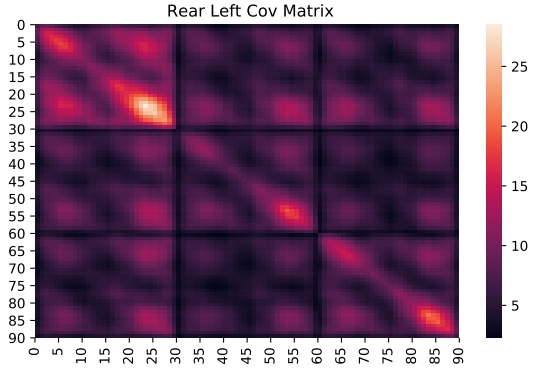
Since WiFi CSI data contains multipath attenuation caused by the environment, the multipath profile extracted from the CSI data can be very useful in location estimation. The most significant factor on the multipath profile is the vehicle itself, as the WiFi receiver is inside and the metal structure severely distorts the signals. As the WiFi antenna array is in linear formation and is perpendicular to the front-back axis of the vehicle, it can not identify whether the signal is com-



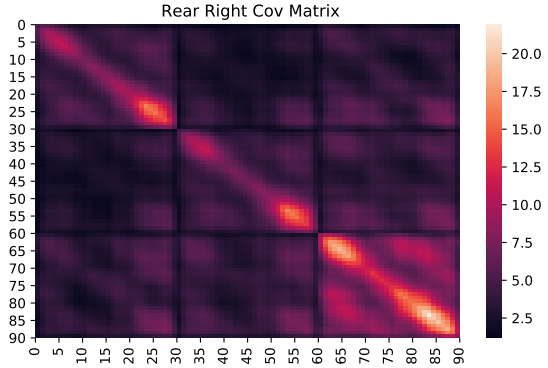
(a) Covariance matrix from front left side of the vehicle.



(b) Covariance matrix from front right side of the vehicle.



(c) Covariance matrix from rear left side of the vehicle.



(d) Covariance matrix from rear right side of the vehicle.

Figure 5.5: Example CSI covariance of all the received WiFi packets. The covariance is calculated with all 30 subcarriers of each antenna, so the total number of variables is 90. All packets from the same dataset are used for the calculation in each case.

ing from the front or the back. This is because the Angle of Arrival (AoA) will be symmetric as discussed in Chapter 2.4.4. To differentiate whether the signal is coming from the front or the back, one can explore the difference in multipath profiles due to the vehicle is affecting the signals differently. Assuming the antenna is placed near the front windshield and in the center position of the dashboard, when the signal is coming from the front of the vehicle, most of the signal will travel directly through the window and be received by the antennas. In this situation, the number of multipath will be smaller as they are less attenuated by the vehicle body, resulting in a stronger dominant path, especially in the line-of-sight scenario. On the other hand, when the signal is coming from the back of the vehicle, it is most likely heavily attenuated by the vehicle's

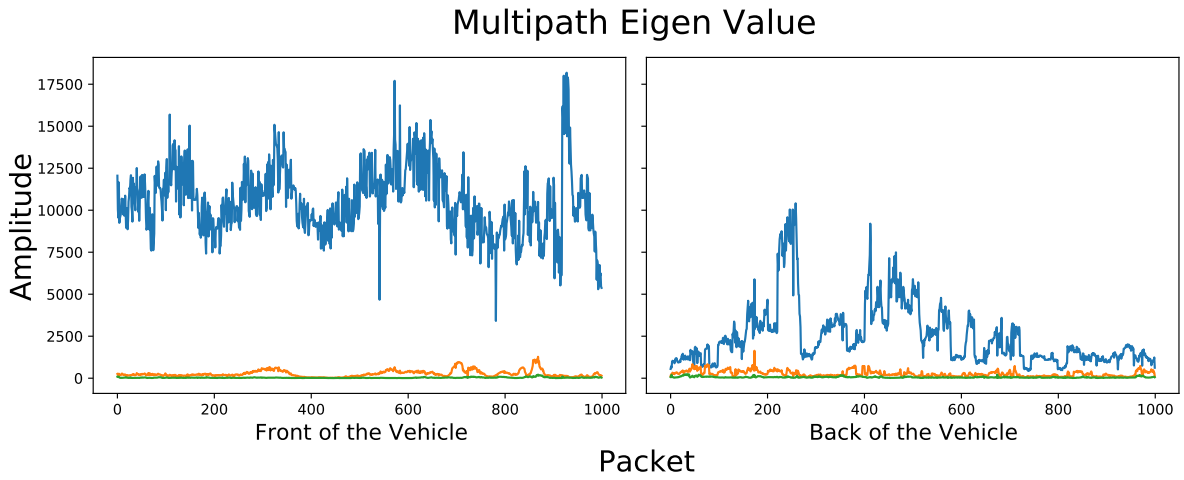


Figure 5.6: Example CSI multipath Eigenvalue. Left shows the Eigenvalue calculated when the phone is in the front of the vehicle. Right shows the Eigenvalues calculated when the phone is in the back of the vehicle.

body with reflection and absorption. Thus the multipath profile is more complex, and the number of multipath tends to be higher. With more complex multipath, the differences between each multipath signal tend to be smaller as well.

To extract the multipath profile of the CSI data, one can explore how MUSIC [108] and SpotFi [69] algorithms extract signals and estimate their Angle of Arrival. They first isolate multiple possible signals by performing Eigen decomposition of matrix XX^H, where X is the CSI measurement, and X^H is the conjugate transpose of X. The eigenvectors and eigenvalues can be used as features as they are affected by the environment and the vehicle. For example, as shown in Figure 5.6, the eigenvalues when the phone is in the front of the vehicle have a higher dominant path (especially the ratio between paths), whereas eigenvalues in the back of the vehicle have a lower ratio. Thus, the multipath profile can help identify where the signal is coming from, especially exploiting the asymmetry introduced by the placement and structure of the vehicle.

5.6 CarFi Algorithm

5.6.1 CSI Calibration and Denoising

CSI contains how the RF signal propagates through the environment as they are being affected during transmission. The CSI data collected at the receiver side contains those affected and encoded in the complex form with amplitude and phase information. Each CSI data point is also the Channel Frequency Response (CFR):

$$H(f;t) = \sum_{n}^{N} a_n(t) e^{-j2\pi f \tau_n(t)} \tag{5.2}$$

Where $a_i(t)$ is the amplitude attenuation factor, $\tau_i(t)$ is the propagation delay, and f if the carrier frequency [122] [80]. However, due to real-world hardware and software imperfections, the CSI data contains noises. For more accurate location estimation, such noises need to be reduced. To remove the antenna phase offset introduced by hardware imperfections, we first measure the antenna phase offset by transmitting WiFi packets through an RF splitter, in which all three receiver antennas will receive the signal at the same time. By removing the offset we measured, we correct the antenna phase offset. The system also introduces Sampling Time Offset (STO) and Sampling Frequency Offset (SFO) as the sampling clocks and frequencies are unsynchronized between the receiver and transmitter. We follow SpotFi [69] method to remove STO and SFO through linear regression. Once CSI data are calibrated and denoised, we extract features as described in the previous chapter.

5.6.2 Neural Network for Location Estimation

CarFi improves the experience by providing relative location between the vehicle and passenger to help them find each other faster. However, the WiFi signal is severely distorted by the metal structure of the vehicle, which makes it extremely difficult to estimate the accurate Angle of Arrival (AoA) of the phone's location. To enable realistic application, CarFi estimate where

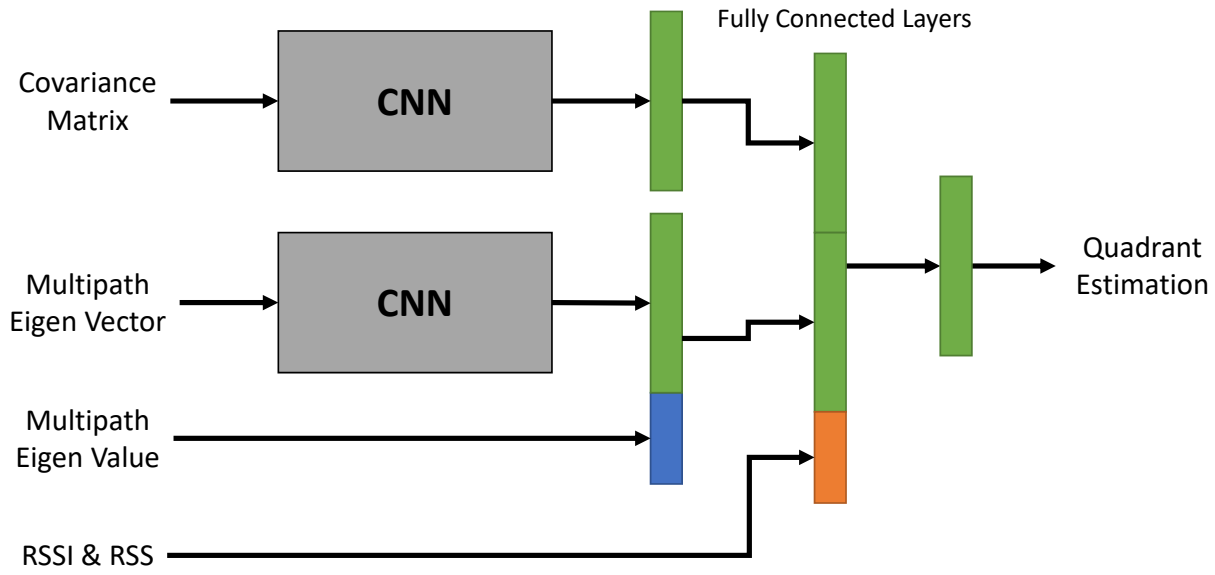


Figure 5.7: Neural network model for our WiFi location estimation.

the phone is in the 4 quadrants of the vehicle: **Front Left**, **Front Right**, **Rear Left**, and **Rear Right**.

We propose a neural network model with the features described above as input to estimate the location. An illustration of the neural network model is shown in Figure 5.7. This model uses convolutional neural network (CNN) layers to process the covariance and multipath features. The filter size is set to 3x3 with a stride size of 1 and padding of 1. Each CNN layer is followed by a Batch Normalization layer [61] and ReLU (Rectified Linear Unit) activation function [107] layer. While the number of filters doubles every layer starting from 32, the input size halves every layer through the max polling layer after the ReLU layer. The last convolutional layer reduces from a 2D feature space to a vector. Since covariance eigenvalues are vector feature inputs, they are concatenated with feature output vector from the CNN layers. We then concatenate the feature vector of the covariance matrix, multipath profile, RSSI, and RSS as one feature vector. We then use two extra fully connected layers connect to the output of dimension 4. Cross-Entropy loss function [36] is used to calculate the loss and Adam [68] optimizer for backpropagation optimization.

Figure 5.8: In-vehicle system setup. The left picture shows the antenna viewed from outside. The middle picture shows WiFi antennas placed on the central dashboard. The right picture shows a laptop with Intel 5300 NIC and connected with antennas through cables.

For evaluation, we also test the performance of location estimation using a subset of the features to analyze the feature's usefulness. To perform those evaluations, we remove the layers that process the features we are not using and modify the concatenation layer to remove the connections.

5.7 Data Collection

5.7.1 System Setup

To extract the WiFi data in an automotive environment, we utilize a laptop with Intel 5300 WiFi Network Interface Cards (NICs) for portability, as shown in Figure 5.8. We use the Linux CSI tool [54] to collect PHY layer CSI information from received WiFi packets. We set up a Pixel 2 XL phone as the Access Point (AP), which the laptop is connected to, and generate WiFi traffic by pinging the laptop. An Android App controls the WiFi traffic generation, as shown in Figure 5.9a, that we developed and can achieve a packets rate of up to 300 packets per second. The actual average transmission rate is around 150 packets per second during our data collection.

5.7.2 Ground Truth

We hold the phone while standing or walking around the vehicle when the vehicle is safely parked to collect the data. However, the phone's location service, such as GPS or cellular, does

(a) Android APP developed to generate WiFi traffic.

(b) View on how phone is located relative to the car.

Figure 5.9: Data collection equipment and environment.

not provide location data with high accuracy and a high refresh rate. Thus, we need to determine the ground truth location data through our equipment. To this end, we used two methods to annotate the ground truth. The first method uses small cones placed on the ground at predefined locations as shown in Figure 5.9b (they are 1,3,5,10,20,30 meters away from the vehicle and two meters away from the middle line of the vehicle). This method is used to collect data where we are standing still while holding the phone. We use a commercial drone hovering above the data collection site for the second method to record the data collection process. For easier drone footage processing, we conduct data collection at night and place LED lights on the top of the vehicle, human subject, and predetermined locations. When the video is recorded at a lower exposure, we can use a simple threshold computer vision technique to extract the pixel locations of the LEDs as they are the bright spots. Then we transform the pixel locations into real-world locations.

Location	Number of WiFi Packets	Ground Truth Method	Train/Test
High School Parking Lot 1	310,340	24 Cones	Train & Test
UNC Parking Lot	308,909	24 Cones	Train & Test
Lake Side Parking Lot	49,142	Drone	Test
High School Parking Lot 2	171,993	Drone	Test
High School Parking Lot 3	742,555	Drone	Train & Test

Table 5.1: CarFi collected dataset.

5.7.3 Collected Dataset

We collect data over several days and vary the locations to provide more environmental variations. In the end, we collected 1,582,939 packets over 5 different locations. The number of packets and ground truth method for each location is shown in Table 5.1. We also specify the training or testing of the data collected at that location. After splitting, we have a total of 1,082,803 packets for training and 491,136 packets for testing.

We split data collected from *High School Parking Lot 1*, *UNC Parking Lot*, and *High School Parking Lot 3* into 80% as training and 20% as testing. The data collected at *High School Parking Lot 2* location are with the vehicle driving around while the person who is holding the phone being static. Data collected from other locations is when the vehicle is stationary, and the person holding the phone is either standing at different locations or walking around. The number of packets of each testing dataset and the content inside each dataset are listed in Table 5.2. In this table, the datasets in the ***Driving*** scenario are all from the *High School Parking Lot 2* location.

We note that the data we collected and reported here are all in a linear antenna array, and the number of packets reported here is all processed ones. Using the location of the antennas as the origin of the coordinate system, we define the front-back axis as the y-axis and the left-right axis as the x-axis. For the WiFi packets we collected, those in the location range of $-1 < x < 1$ or $-1 < y < 1$ (in meters) are removed. These locations represent the person is standing directly in the front or back of the vehicle on the road, or they are adjacent to the vehicle, which should be able to identify the vehicle and already passed through the 4 quadrants estimation. As our

	Dataset	Number of Packets	Content
Walking	High School 1	59,674	The vehicle is stationary at an empty high school parking lot. The phone is held by a person while standing still at 24 locations.
	UNC Parking	61,187	The vehicle is stationary at an empty UNC parking lot. The phone is held by a person while standing still at 24 locations.
	Lake Side Parking	49,142	The vehicle is stationary at an empty lake-side parking lot. The phone is held by a person while walking around the vehicle. The drone provides ground truth.
	High School 3	149,140	The vehicle is stationary at an empty high school parking lot. The phone is held by a person while walking around the vehicle. The drone provides ground truth.
Driving	Single Person	34,781	The phone is held by a person standing still in an empty parking lot. The vehicle drives around. Only the vehicle and the person is present.
	Three People 1	27,293	The phone is held by a person standing still in an empty parking lot. The vehicle drives around. The vehicle and three people are present, with three people standing side by side in a line.
	Three People 2	14,110	The phone is held by a person standing still in an empty parking lot. The vehicle drives around. The vehicle and three people are present, with the extra two people standing in front of the one who is holding the phone.
	Single Person + Vehicle	69,857	The phone is held by a person standing still in an empty parking lot. The vehicle drives around. Three vehicles and a single people are present, with the extra two vehicles parked in front of the one who is holding the phone.
	Three People + Vehicle	25,952	The phone is held by a person standing still in an empty parking lot. The vehicle drives around. Three vehicles and three people are present, with the extra two people standing in front of the one who is holding the phone and two vehicles parked in front of the people. Refer Figure **??** for reference.

Table 5.2: CarFi test dataset size and content

Figure 5.10: Drone image of three people and two vehicles in between the WiFi transmitter and receiver.

features include covariance of 50 consecutive WiFi packets, we also remove packets when there are less than 50 packets in 1 second.

5.8 Evaluation

5.8.1 Features

We first evaluate the location estimation performance of a different subset of the features. Using the neural network architecture described in Chapter 5.6.2, we train multiple ones with different features and report their best estimation accuracy in Table 5.3. From the table, we can see that both the multipath profile and covariance matrix can be used to estimate the location of the vehicle. Combining these two features can improve the performance, which results in better generalization as the accuracy improves with the vehicle driving data. An interesting note is that the covariance matrix feature has better performance than the multipath profile feature. This could be that while multipath profile estimates different signals about their amplitude and phase, there are only 3 antennas present in the system, which is less than the number of multipath exits.

This becomes a compressed sensing problem with a lot of estimations. Furthermore, covariance matrices look at a consecutive of 50 packets that contain more information than the single packet used in multipath profile calculation.

	Dataset	Accuracy (%)		
		Multipath	Covariance	Multipath + Covariance
Walking	High School 1	85.92	94.12	92.58
	UNC Parking	86.1	94.68	90.23
	Lake Side Parking	51.44	52.97	57.19
	High School 3	45	76.13	75.61
Driving	Single Person	32.09	37.62	37.45
	Three People 1	29.61	37.36	38.3
	Three People 2	27.63	42.66	45.97
	Single Person + Vehicle	45.23	46.23	46.01
	Three People + Vehicle	37.25	32.81	35.63

Table 5.3: Neural network performance with individual features.

The quadrant localization using RSSI and RSS is not performed. They contain much less information about where the signal is coming from, especially identifying if the person is standing in front of the vehicle or in the back. This is because the received signal strength is affected both by the metal structure of the vehicle and the distance between the transmitter and receiver. When the phone is further away from the vehicle while in the front, the received signal strength can be similar when the phone is in the back at a closer distance as the signal can be partly reflected and absorbed by the vehicle's body. We also analyze if raw CSI data can provide similar performance in Chapter 5.9.1.

5.8.2 Localization Performance

The result from the CarFi neural network model is shown in Table 5.4. The results show that CarFi can predict where the phone is in the 4 quadrants even in scenarios that have not been seen before. Furthermore, the localization accuracy with all the features generalized better than when only a subset of the features is used. For the *High School 1* and *UNC Parking* data, the person who is holding the phone is standing still in 24 locations, the dataset is split with the first 80%

packets being training and the rest 20% as testing. While they are not consecutive packets, the small body movement does not cause large changes in the signal propagation paths. As a result, the localization accuracy can be very good due to similarity to the training data.

	Dataset	Accuracy (%)
Walking	High School 1	91.38
	UNC Parking	92.66
	Lake Side Parking	57.84
	High School 3	74.33
Driving	Single Person	36.76
	Three People 1	39.63
	Three People 2	44.61
	Single Person + Vehicle	48.15
	Three People + Vehicle	39.97

Table 5.4: Neural network performance for localization.

During the data collection, the person who is holding the phone walks around the vehicle in *Lake Side Parking* and *High School 3* scenarios. While there may be similar walking patterns and WiFi packets collected at similar locations, the variations within the packets are large. The data from *High School 3* is split the same way as before, with the first 80% as training and the rest 20% as testing. Thus the difference between the training and testing is more significant. We can observe the accuracy decreases but does show the framework can generalize to unseen locations. For the testing data collected while the vehicle is driving and the person who is holding the phone is stationary, the **Driving** dataset is unseen from all the training data. These datasets are difficult as environmental obstacles, the instability of the antennas, and the movement of the vehicle all introduce additional noises and variations. When the results are compared to using only a subset of features, the overall accuracy improves, which demonstrates they can generalize to these difficult situations. These results show that by increasing the number of packets collected and provide more variations in the training dataset, the overall accuracy can be further enhanced.

5.9 Discussion

5.9.1 What About CSI Value?

In Chapter 4, we used raw CSI value to estimate the Angle of Arrival (AoA) in the indoor environment. However, as the difference between the indoor and automotive environment differs dramatically, this method does not generalize easily. We trained a fully connected neural network with CSI real and imaginary numbers as a vector to predict the quadrant of where the phone is located. The results are shown in Table 5.5. In this table, we can see that the localization accuracy is high and comparable to using the features in *High School 1* and *UNC Parking* scenarios. As reason stated before, this is most likely the environment is stable, and signal propagation paths do not vary much over a short period of time. Thus the CSI data are similar to each other. However, the accuracy in other environments is very low as the environment is changing significantly, and using raw CSI data is very hard to extract meaningful features and generalize.

	Dataset	Accuracy (%)
Walking	High School 1	91.94
	UNC Parking	93.02
	Lake Side Parking	56.63
	High School 3	32.4
Driving	Single Person	25.99
	Three People 1	18.74
	Three People 2	17.53
	Single Person + Vehicle	13.68
	Three People + Vehicle	35.88

Table 5.5: Neural network performance with CSI as input.

5.9.2 What Happens When There Are Vehicles Blocking?

In the evaluation, the accuracy is lower in vehicle-driving scenarios. There are multiple reasons for this result. First, the environment is different from when the vehicle is stationary, and the person is either standing or walking around. For example, the antennas can vibrate, resulting in changing their relative positions to each other and the vehicle. Second, the speed of the vehicle is

different from what a person can achieve while walking. This can result in frequency shifting due to the Doppler effect. Last but not least, the signal can be severely distorted when other vehicles are parked between the person who is holding the phone, and the vehicle carries the antennas.

5.9.3 Privacy?

As privacy concerns keep rising among consumers, personal information needs to be protected. Unlike camera-based sensor solutions, using WiFi for localization does not require any personal information such as personal photos. Furthermore, the MAC address can be randomized while transmitting the signal each time to conceal the true hardware MAC address. As a result, CarFi can preserve user privacy while improving convenience.

5.10 Summary

In this chapter, we investigate the feasibility of using WiFi devices as localization equipment to identify where the passenger is relative of the vehicle. This method can potentially enable a smoother and hassle-free riding experience. To this end, we propose CarFi, a system that can be incorporated with WiFi-enabled dashcam to identify where the passenger is located. We designed our neural network model which can be used to localize the person who is holding the phone in four quadrants. The features used by the neural network model are extracted from the WiFi CSI data and show the ability to generalize in different environments. We collect a large dataset including the person who's holding the phone, either standing at various locations or walking around the vehicle. In addition, we also collect a dataset for when the person is stationary and the vehicle is driving around.

CHAPTER 6: WIFI SENSING FOR HUMAN PRESENCE DETECTION IN BUILDING CORNERS

6.1 Introduction

With the increased deployment of robots in real-world scenarios, practical human-robot interaction issues such as safety and sociability are becoming increasingly important. With recent advances in robotic planning and control, although the reaction of robots is becoming more and more agile, for many robotics applications it is still not on par with the dynamics of humans. Given the limited capability of fast reaction, the demand for better sensing arises. We hypothesize that – for a robot equipped with better sensing capabilities, its required reaction time can be relaxed, and safety and sociability can be increased.

Most on-board sensors of a robot, such as cameras, LIDAR, infrared, and ultrasonic sensors, require line-of-sight between the sensor and the target. These sensors may serve perfectly in the *seen* world, but in a crowded environment or places that have many blind spots, the line-of-sight requirement of these sensors makes it challenging for a robot to navigate safely and sociably. If the sensors were capable of *X-ray vision* for non-line-of-sight sensing, a robot could overcome these blind spots, plan its actions better, and ensure safer and more natural human-robot interaction for social robots.

Existing works have studied the *seeing around the corner* problem in robotics using vision [105] [22] and acoustics [19]. Cameras used in these systems are expensive, special, and require adequate lighting in the environment. Sound of footsteps is used to localize humans [19], but these solutions are not effective unless the environment is quiet and humans are walking as opposed to just standing around the corner. Recent works have exploited radio frequency (RF)

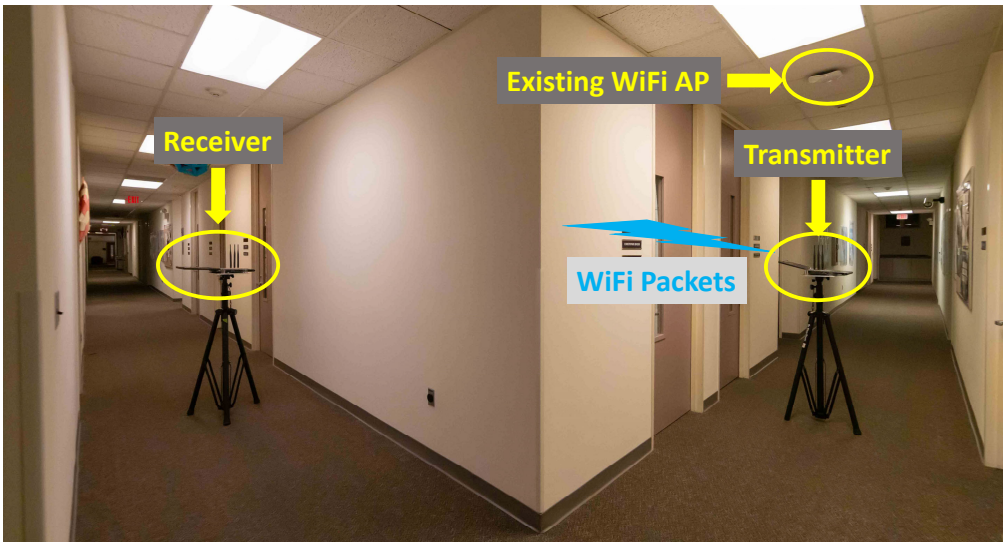


Figure 6.1: Our system consisting two WiFi devices, a transmitter which can be exiting WiFi Access Point (AP) and a receiver which can be WiFi card on the robot. The receiver receives packets transmitted by the transmitter then analyze the properties of the received packet to estimate whether human is present or not.

signal's capability to penetrate walls to detect human figures [13] [143] [142], but these solutions require large antenna arrays and moving human figures.

In this paper, we study the problem of determining whether a human is present around the corner in non-line-of-sight situations by using only commodity WiFi devices. We propose a solution that relies upon commodity WiFi devices and extracts features from received WiFi signals [38]. In addition to standard coarse-grained signal power measurements such as Received Signal Strength Indicator (RSSI), we perform multipath estimation using fine-grained Channel State Information (CSI). We also extract features that represent signal variability over time. Utilizing all these features, we model and classify the state of the environment.

Our solution works in real-time and is not dependent on the environments. Unlike prior works, we require only 50ms for sensing, which makes it suitable for indoor mobile robot applications. Since we rely on commodity WiFi devices, the solution can be easily incorporated into mobile robots. The extended sensing capability does not add significant cost as many robots already use WiFi for connectivity. We evaluate our solution in multiple unseen environments (i.e., training and testing environments are different) to demonstrate the robustness of our system. We

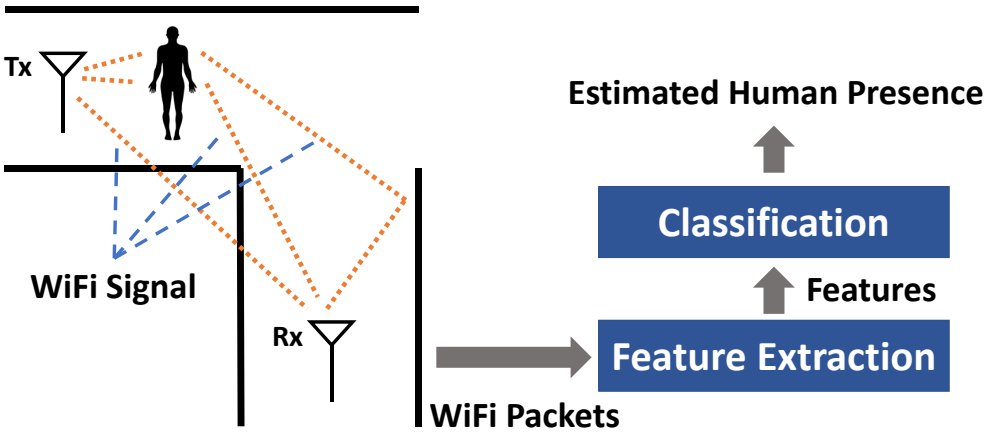


Figure 6.2: System Overview. The transmitter sends WiFi packets where the signal can travel directly, pass through the human body, and reflected to the receiver. The receiver receives the packet and calculates the properties of the signal. Then our system extracts the features which are later used for classification on human presence.

compare our solution with existing ones and show that in challenging environments, the proposed solution outperforms the existing ones.

6.2 System Overview

We study the problem of detecting human presence in *around-the-corner* situations using commodity WiFi devices. An example scenario is depicted in Figure 7.5a. A WiFi transmitter, such as a WiFi AP in a building, and a WiFi receiver, such as a WiFi device mounted on a robot, are placed on an L-shaped corridor in a non-line-of-sight setup. The goal of the receiver is to determine whether there are humans on the other side where the WiFi transmitter is.

An overview of how the proposed solution is shown in Figure 6.2. Tx and Rx refer to the transmitter and the receiver, respectively. When the transmitter sends a packet, the WiFi signal is broadcasted in all directions. The signal travels through the walls and reaches the receiver on the shortest path. Some of the signals get attenuated by the human body (if present) before reaching the receiver.[1] Signals also get reflected multiple times before the receiver receives them. Hence,

[1]The human is occluded by the corner, which results in no direct line-of-sight (LOS) path for the WiFi signal to pass through the human body and then directly received by the receiver. All such signals are attenuated further by the walls and other objects in the environment.

the received signal is a combination of all these signals traveling on different paths. A number of consecutive WiFi packets are retrieved, a set of features is computed, and classified to determine if a human is present around the corner.

6.3 WiFi Feature Extraction

This chapter describes the WiFi features that the system uses to represent the RF environment. All features are extracted from the same number of received packets.

6.3.1 Received Signal Strength Features

The *Received Signal Strength Indicator* (RSSI) is an estimate of power in the received signal at an RF client. The signal strength is affected by multiple factors such as the distance between Tx and Rx, obstacles, Tx strength, and Rx antenna's property. Since the human body attenuates WiFi signals, e.g., by absorbing signals, it changes the RSSI value. Hence, we use RSSI as one of the features to represent the RF environment.

For each packet P, the WiFi card reports three RSSI values: $RSSI_a$, $RSSI_b$, and $RSSI_c$, where each value corresponds to one of the three receiving antennas. We calculate the mean and the variance of each RSSI from the N received packets: $(x \in \{a, b, c\})$

$$\mu_{RSSI_x} = \frac{1}{N}\sum_{i=1}^{N} RSSI_{x_i} \ , \ \sigma^2_{RSSI_x} = \frac{1}{N}\sum_{i=1}^{N}(RSSI_{x_i} - \mu_{RSSI_x}) \tag{6.1}$$

6.3.2 Effective Signal to Noise Ratio Features

The *Effective Signal Noise Ratio* (SNR), which measures the quality of the WiFi signals, is affected by environmental changes. We calculate the effective SNR for four modulation schemes, i.e., BPSK, QPSK, 16QAM, and 64QAM. The effective SNR is calculated from the channel state information (CSI), which is described next. Since we only use one transmission antenna, it is

a *Single Input Multiple Output* (SIMO) system. The calculated effective SNR is a 1×4 vector $(SNR_0, SNR_1, SNR_2, SNR_3)$, where each element corresponds to a modulation scheme.

Similar to RSSI, we calculate the mean and the variance for each effective SNR across the N received packets, which become parts of the feature vector: $(j = 0, 1, 2, 3)$

$$\mu_{SNR_j} = \frac{1}{N}\sum_{i=0}^{N} SNR_{j_i} \quad , \quad \sigma^2_{SNR_j} = \frac{1}{N}\sum_{i=1}^{N}(SNR_{j_i} - \mu_{SNR_j}) \tag{6.2}$$

6.3.3 Signal Tendency Index

The Signal tendency index (STI) [146] [147] is based on Procrustes analysis to compare shape similarity across different packets. It is calculated from the *Channel State Information* (CSI) containing fine-grained information of both the magnitude and the phase of each subcarrier between each transmitter-receiver antenna pair [135].

For the CSI vector $(H_1^t, H_2^t, \ldots, H_n^t)$, at time step, $t = 1, 2, \ldots, N$, we standardize the values by subtracting the mean and dividing by the standard deviation:

$$\hat{H}^t = \frac{[H_1^t - \bar{H}^t, H_2^t - \bar{H}^t, \ldots, H_n^t - \bar{H}^t]}{\sigma(H^t)} \tag{6.3}$$

$$\bar{H_1^t} = \frac{1}{n}\sum_{i=0}^{n} H_i^t \quad , \quad \sigma(H^t) = \sqrt{\frac{\sum_{i=0}^{t}(H_i^t - \bar{H}^t)^2}{n}} \tag{6.4}$$

The STI between two consecutive packets is $S = \left\| \hat{H}^t - \hat{H^{t-1}} \right\|$, which is the Euclidean distance between the curves in STI metrics and larger STI value means greater difference. An example of STI values between different packets is shown in Figure 6.3. The changes between packets 20-40 are larger than the rest as they are collected while a person walks, which severely distorts the WiFi signal between packets. As for the empty space (0-20) and a person standing still (40-60), the difference is similar to the environment is almost static. We also observe that even in the empty space, CSI is not stable.

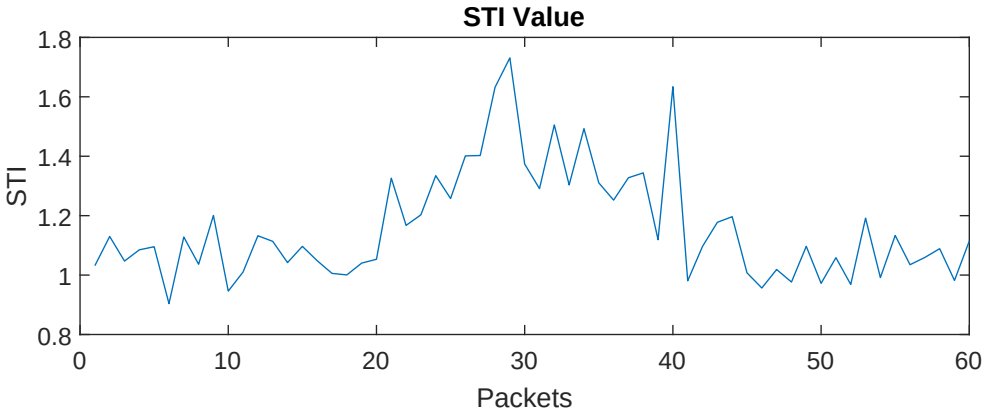


Figure 6.3: Example of STI value across different packets. Packets 0-20: empty environment. Packets 20-40: a person is walking. Packets 40-60: a person is standing still.

With N packets received, we calculate $N-1$ STI values. The mean and variance of the STI are then computed and used as a feature.

6.3.4 Multipath

As the human body attenuates WiFi signals, the multipath profile of the WiFi environment changes. When we have many reflectors in the environment (e.g., a crowded scene), the number of multipath is large, but the difference in energy of different paths is small. For a relatively less crowded scene, the number of multipath is less, but the energy differences are high. This property of multipath can be modeled from the WiFi CSI values.

The CSI measurement, X, which contains both magnitude and phase information, can be used to compute the angle of arrival (AoA) [108] of a signal. For example, in recent WiFi-based localization algorithms [69] [34], the AoA of the direct path (which is relevant to the localization problem) is isolated by taking the eigenvector of the matrix, XX^H, for which, the eigenvalue is zero. The eigenvector goes through further processing to obtain the direct path.

Inspired by this, one can compute the AoA corresponding to each eigenvalue to create a multipath profile of the WiFi environment. However, we propose that, in order to model the multipath profile, we do not necessarily have to compute the AoAs. Instead, we can take the top k largest eigenvalues of XX^H to have the simplest way to create a feature that inherently

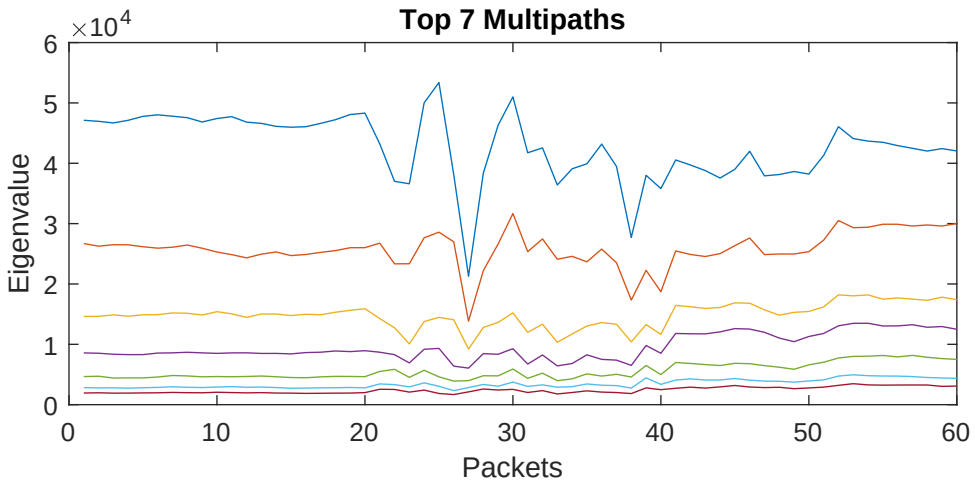


Figure 6.4: Example of eigenvalues for multipath analysis in different packets. Packet 0-20: empty environment. Packet 20-40: a person is walking. Packet 40-60: a person is standing still.

represents the multipath profile. An example of the proposed multipath-based features is shown in Figure 6.4 where we use top seven eigenvalues. We observe that the three behind-the-wall cases: empty space, a person walking, and a person standing is clearly distinguishable by the proposed feature.

6.4 Human Presence Detection

We formulate the non-line-of-sight around the corner human detection as a classification problem having three classes: `empty`, `standing`, and `moving`. We implement a Random Forest [76] classifier that uses the feature vector described in the previous chapter. We empirically determine that extracting features from a total of $N = 50$ packets result in the best classification accuracy while keeping the sensing delay as low as 50ms.

6.5 Implementation

We implement the proposed system using two laptops having Intel 5300 WiFi Network Interface Cards (NICs). We use Linux CSI tool [54] for collecting PHY layer CSI information from

transmitted packets. We operate in the 5GHz WiFi spectrum to avoid firmware limitations [48]. The transmitter operates in the injection mode and the receiver operates in the monitor mode.

To train and test the classifier, we collect WiFi data from seven different corners across different floors of a four-storied building by placing the transmitter and the receiver at varying locations. Some example setups are shown in Figure 6.5. The laptop used as AP transmits WiFi packets over one antenna ($N_{TX} = 1$) and the receiver receives on three antennas ($N_{RX} = 3$). The packets are transmitted at 1000Hz, which results in a sensing delay of 50ms for 50 packets used for feature extraction. Each received packet contains the RSSI and CSI for each antenna. The CSI is a $N_{Tx} \times N_{Rx} \times 30$ matrix where 30 is the number of subcarriers in the WiFi channel reported by the Linux CSI tool.

We consider three scenarios: 1) no one is in the environment, 2) a person is standing around the corner who is occluded by the wall from the receiver's viewpoint, and 3) a person walking in the occluded area.

6.6 Results

To evaluate and examine the robustness of our system, the training and testing datasets are collected from different floors. Hence, our model has not seen any example from the testing environments. We report the precision, recall, and F1 score for each class, and the overall accuracy in Table 6.1.

Class	**Precision**	**Recall**	**F1**	**Overall Accuracy**
`Empty`	0.48	0.59	0.53	61.37%
`Standing`	0.47	0.36	0.4	
`Moving`	0.89	0.89	0.89	

Table 6.1: Classification results with three classes.

We also show the result of binary classification: `empty` and `occupied` by combining the standing and moving classes into a single class. The result is shown in Table 6.2.

Figure 6.5: Data collection environments. Each corner is different from one another, the layout and materials around each corner creates totally different WiFi propagation characteristics.

Class	Precision	Recall	F1	Overall Accuracy
`Empty`	0.5	0.33	0.4	66.47%
`Occupied`	0.71	0.83	0.77	

Table 6.2: Classification results with two classes.

6.7 Discussion

At first glance, the results reported in this paper may seem to have fallen short of what similar systems have reported [147] [132]. However, there is a major difference in how the evaluation is done and how the experiment (Tx and Rx) was setup.

We conduct experiments to demonstrate the expected performance of the proposed system in real-world scenarios that have a completely new environment. In other words, we train and test on completely different environments. On the other hand, existing works [147] [132] collect a single dataset and then split it into training and testing sets. Due to the limited number of locations,

their training and testing sets contain the same transmitter and receiver location pairs–which makes the classification task easier. If we split our dataset into training and testing, and redo the experiment, we achieve results similar to [132], which is reported in Table 6.3. We also perform binary classification, i.e., `empty` vs. `occupied`, by splitting the dataset. The result is shown in Table 6.4.

Furthermore, the experimental setup in existing works considers line-of-sight situations where the human is *not* occluded. These systems use 1500 packets at 50Hz, which results in 30s sensing time, which is 600X slower than our solution.

The difference in the two evaluation methods revealed that if our system is deployed to a specific area, it can continuously collect more data for training and can potentially see a significant performance boost. Thus, the results reported in this paper should be treated as the lower bound if the system is deployed to a completely new environment, which can still provide important information to improve safety.

Class	**Precision**	**Recall**	**F1**	**Overall Accuracy**
`Empty`	0.95	0.95	0.95	94.49%
`Standing`	0.96	0.94	0.95	
`Moving`	0.92	0.94	0.93	

Table 6.3: Classification results with 3 classes and evaluated through splitting training and testing data set.

Class	**Precision**	**Recall**	**F1**	**Overall Accuracy**
`Empty`	1	0.97	0.98	98.85%
`Occupied`	0.98	1	0.99	

Table 6.4: Classification results with 2 classes and evaluated through splitting training and testing data set.

6.8 Summary

We present a system that detects human presence in non-line-of-sight around-the-corner situations using only commodity WiFi devices. We evaluate our system in different unseen environments to demonstrate its robustness. In our future work, we plan to explore the possibility of

extracting more information from CSI measurements by analyzing multipath profile, which can be combined with the 3D model of the environment to better estimate the effects of the human body on the WiFi signal. We also plan to collect more data and conduct experiments to observe whether deep learning models can increase performance.

CHAPTER 7: ENHANCED MMWAVE RADAR SENSING

7.1 Introduction

In recent years, many 3D graphics and vision algorithms have been proposed to model and understand 3D scenes using off-the-shelf cameras and depth sensors. These algorithms have found their uses in robotics, virtual reality (VR), augmented reality (AR), and mixed reality (MR) applications. However, the fundamental limitations of these systems are that they are practically useless when there are occlusions, non-ideal lighting, and difficult environmental conditions such as fog and rain. Radio Frequency (RF) signals, on the other hand, can penetrate certain types of obstacles and are more robust to environmental conditions. Therefore, RF sensing has emerged as an alternative or a complementary solution to 3D imaging and vision—leading to interesting applications such as human activity recognition [125], keystroke detection [18], sign language recognition [75], lip motion recognition [126], localization [128], 3D tracking [15], direction finding [70], range estimation [123], heartbeat detection [16], respiration monitoring [11], emotion detection [141], sleep apnea detection [11], and fall detection [129], multi-person gestures [14], and 2D/3D pose estimation [142] [143].

Typically, RF sensing systems work by modeling the changes in an RF environment when a certain event or an object of interest is present in the scene versus when they are not. These systems are trained to identify and learn the least amount of information from RF signals that are sufficient to distinguish between an event A and an event B, where A and B are significantly different, e.g., running vs. standing. When events of interest are similar, these algorithms fail to model their fine-grained differences due to the lack of enough information in the training data. Placing the RF transceiver close to the objects is a common leeway to handle such cases, but in general, there is a lack of research on creating a rich intermediate representation of an RF

scene that can be used to infer fine-grained gesture or to describe a scene with minute detail. Such a representation can be used in a wide range of applications such as monitoring a patient, describing an open or a concealed scene, and context-aware navigation of autonomous systems.

Among RF imaging systems, synthetic aperture radar (SAR) [87] is a well-known technique to generate a reasonably high-resolution representation of a scene. However, this technique requires us to move the radar linearly over a sizable distance to simulate a large (synthetic) aperture. A downside of SAR is the time it takes to move the radar, and thus, they are not robust to capturing scenes that contain moving objects. While we can address this issue by building a gigantic radar, such a system will not be scalable, cost-effective, computationally-efficient, and practical for use in indoor mobile environments.

For emerging applications such as augmenting the sensing capability of an indoor social robot, the RF sensor needs to be small in size to fit on the robot, especially when the size of the robot can be constrained by the application scenarios. The ideal size of the sensor should be such that it should be easy to fit on most robots. Smaller sensors also allow us to attach multiple of these on the same robotic platform – which increases the sensing coverage and helps explore more surrounding area. In order to enable an adequate 3D sensing and imaging of an indoor environment, the sensor also needs to have a high-resolution since an indoor environment is often cluttered with many irregularly shaped objects which pose additional challenges to high-quality sensing. We also require that these systems should be able to detect moving objects, such as humans and pets, and be able to generate a representation of the scene in near real-time. Fast sensing and computation are crucial to safety-critical applications where a few ms slower response time can be catastrophic.

Inspired by recent works in computer vision domain that aims at increasing the resolution of images to achieve *super-resolution* [91] [72], we propose a framework and an implementation of it, namely the *SuperRF* [45] [44], that enables fast sensing, and generates high-resolution 3D representation of a scene using mmWave radars – while being low-cost and small in size. Our

two-stage framework enhances the RF measurements from a low-cost mmWave radar to achieve a resolution that is close to a SAR's.

- The first stage of our proposed framework includes a training phase that helps the low-cost mmWave radar system learns how to produce SAR-like imagery from the low-resolution SAR imagery. We collect radar snapshots of different objects by sliding the radar linearly like a SAR system. The collected radar data are synthesized to generate SAR imagery. Using these generated images, we train a specially-designed deep neural network that enhances RF images using only *two* low-resolution RF raw snapshots. Our approach mimics the use of an array of radars working simultaneously, which eliminates the time required to move the radar or the physical size requirement of a large antenna.

- The second stage of our framework employs a compressed sensing-based operation to extract the underlying antenna signal for better estimating the desired high-resolution SAR imagery. This iterative method results in the best possible high-resolution imagery that fits the actual measurements we obtain from the limited number of samples. As this problem is ill-posed, given the limited amount of measurements we have, there are numerous possible solutions (RF images) that fits the actual measurements. By exploiting the sparsity of the signals and applying the compressed sensing technique, SuperRF estimates the most likely and a better quality RF imagery than the one generated by the neural network in the earlier stage.

Using empirical data collected with an off-the-shelf 77 GHz mmWave radar, we demonstrate that SuperRF is able to generate RF representations of objects using only two snapshots to achieve similar-quality imagery produced by SAR operation that uses as high as 64 snapshots (or 64 antennas in the SAR direction). This operation is object-invariant and time-saving as snapshots are taken simultaneously. To the best of our knowledge, SuperRF is the first system that enhances RF sensing to be SAR-like—which is impossible to achieve even with existing high resolution angle of arrival estimation algorithms[108] [70] [97].

The contribution of this chapter are as follows:

- We propose a deep neural network architecture to enhance simultaneously captured two snapshots from mmWave radars into a high-resolution SAR-like imagery which is a low-cost, fast, and software-based alternative to a SAR.
- We propose a second stage of RF signal enhancement that employs a compressed sensing-based method to further improve the neural network generated RF imagery.
- We implement the proposed framework using off-the-shelf low-cost mmWave radar and open-source software. We collect our own data set for training and evaluation.[1]

7.2 SuperRF System Design

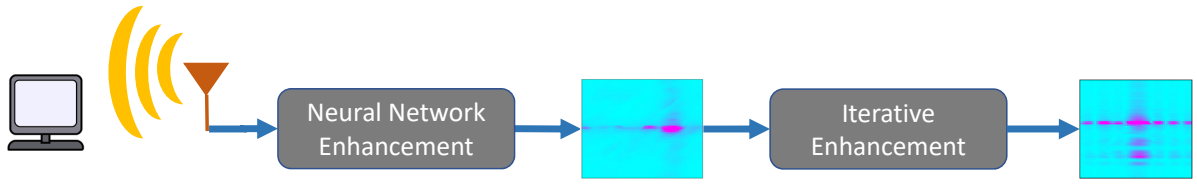


Figure 7.1: An overview of SuperRF's two-stage signal processing pipeline. The RF snapshot after each stage shows the intensity of RF signals (the X and Y axes represent the horizontal and elevation angles, respectively). Pink (darker color) represents a higher intensity. The (x, y) location of intensity values indicates the angle (horizontal, elevation) where the reflection is coming from.

An overview of SuperRF's two-stage RF signal processing pipeline is shown in Figure 7.1. The goal of SuperRF is to take low-resolution, sparse RF signals from the mmWave radar at the input and to generate a higher-resolution, feature-rich representation of the 3D scene at a near-real-time speed.

Prior to entering the SuperRF's processing pipeline, signals undergo the standard preprocessing step of a mmWave radar. The mmWave radar (hardware) samples the intermediate frequency (IF) signals and performs FFT on the samples. This is often called the *range FFT* since each FFT

[1]More info about the dataset can be found at `https://bitbucket.org/embedded_intelligence/superrf_dataset/src/master/`

bin corresponds to a range (i.e., distance) from the radar, and a high value in a bin denotes the presence of one or more objects at that distance. Signals after the range FFT become the input to the SuperRF processing pipeline.

RF signals from each range bin is passed through the first RF enhancement stage of SuperRF– shown as *Neural Network Enhancement* in Figure 7.1 – where a generative deep neural network (DNN) enhances the resolution of the RF signals. This higher resolution RF signal can be directly used in applications such as object detection and occupancy detection. To enhance the output of the DNN further, the second stage of SuperRF– shown as *Iterative Enhancement* – employs an iterative algorithm that takes a compressed sensing-based approach.

7.2.1 Stage 1 – Neural Network Enhancement

The goal of the first stage of SuperRF is to increase the resolution of RF sensor data. This step is necessary since the angular resolution of the input signals is generally poor when the number of antennas used in a mmWave radar system is limited. While a large number of antennas can increase the angular resolution, SuperRF's goal is to increase the angular resolution *without* requiring additional antennas beyond what is feasible (space-wise) in the given system.

For a mmWave radar that does not have an elevation antenna, the radar can not differentiate two objects with the same distance to the radar but placed at different heights. Hence, to capture a scene, either the radar has to move along the vertical axis to take and stitch multiple readings – which we call *snapshots*, or we have to use multiple mmWave radars placed on the vertical axis to simultaneously capture multiple snapshots and then stitch them all together to obtain the SAR-like 3D RF representation.

Examples of RF imaging after SAR operation for a different number of snapshots are shown in Figure 7.2. Consecutive snapshots are taken by moving the radar upwards by $\lambda/2$. We observe that the quality of RF images increases with the number of snapshots, and the improvement in RF imaging quality along the vertical direction is clearly observable.

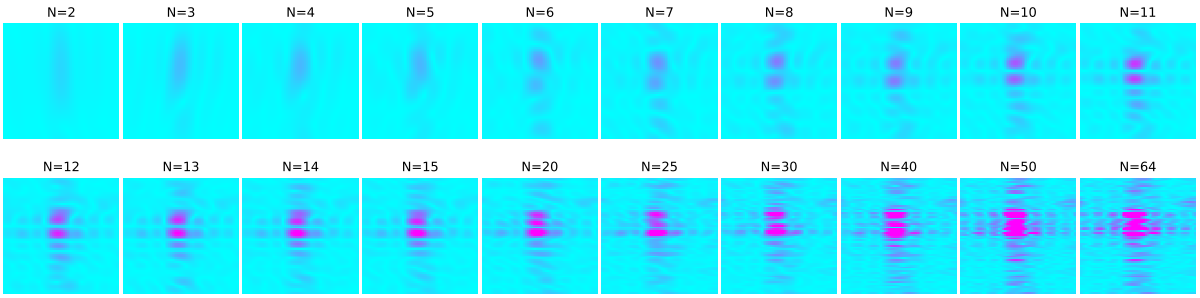


Figure 7.2: RF signal with different number of snapshots. The x-axis denotes the azimuth angle and the y-axis denotes the elevation angle.

For a better understanding of the scene, we need high resolution RF image. However, to achieve this, we need either a large number of antennas or SAR operation which means to physically move the radar.

The study above confirms that for an increased resolution of RF images, we either need to run the SAR operation many times by physically moving the radar, or we need to use multiple radars (or, antennas). Neither of these two options is preferred as a large number of antennas increase the radar size and cost, and a SAR operation takes a significant amount of time due to the need to physically move the radar. For example, in our experiment, each snapshot takes about two seconds due to the communication delay between the radar and the signal processing board along with the delay due to the mechanical movement of the radar on a slider. For 64 snapshots, the entire process takes nearly two minutes. Even with faster communication and faster mechanical operation of the slider, the expected delay in such a SAR system is typically very long.

In order to generate a high-quality (e.g., similar to 64 snapshots SAR) RF imagery from sparse, low-quality RF input (e.g., only 2 snapshots as input), we employ a generative DNN – which is described in Chapter 7.3. The network enables high-resolution RF imaging using a fast, low-cost, and compact radar system, and boosts the RF signal resolution dramatically.

7.2.2 Stage 2 – Iterative Enhancement

With our deep neural network generated RF images, one can directly compute the 3D RF intensity of the scene. However, no deep learning neural network can generate perfect results even with enough data and training time.

A DNN-based generative approach is several orders of magnitude faster than the traditional SAR approach. However, we observe that the output of the DNN is often a noisy version of the desired high-resolution RF imagery.

To overcome this limitation, we propose to enhance the DNN's output by applying an iterative algorithm that is based on the Griffin-Lim phase recovery algorithm [51] and compressed sensing. We treat the neural network generated results as a noisy model of the true signal and iteratively improves its quality. Given that we have a small number of actual measurements (e.g., 2 snapshots) which is neither enough for compressed sensing nor the RF signal is sparse enough as the radar receives reflected signals from different angels, we optimally enhance the output based on our knowledge of the actual measurements, and then combine them with the information sampled from the inferred values. The details of this method are described in Chapter 7.4.

We note that the proposed two-stage RF signal processing framework is application agnostic. The enhanced RF imagery produced by SuperRF can be used in numerous applications such as occupancy detection, obstacle detect, and height estimation that achieves better accuracy and robustness than the state-of-the-art RF sensing systems. In Chapter 7.6.4, we provide an evaluation of SuperRF using object recognition as an application.

7.3 Neural Network Enhancement Algorithm

A typical RF sensor generates coarse signals due to their poor resolution. For high-resolution imaging, one can choose high-frequency RF sensors that operate in the range of over 100 GHz. However, such a high-frequency signal loses its signal power quickly as it propagates through the medium – which makes them unsuitable in many real-world applications that require a rea-

sonable range. Furthermore, due to the propagation loss, high-frequency RF signals normally do not penetrate objects – which makes them less attractive to our intended application scenarios. Hence, in this paper, we use a mmWave radar whose operating frequency is 77 GHz. Algorithms developed in this paper, however, can be applied to other types of radars.

Recall that one can move the radar to collect multiple measurements and then apply SAR operation to obtain a high-resolution RF imagery. However, this operation is both time- and space- expensive. To address this, we propose enhancement of the received signals only from a couple of snapshots to be matched with the ones generated through SAR operation after a large number of snapshots. In other words, we generate 64-snapshot SAR-like RF imagery from only 2 simultaneous snapshots by employing a generative CNN. We focus on improving the resolution in the vertical direction where the resolution is the worst (without any SAR operation). The radar is moved vertically to collect as many as 64 snapshots for training and testing of the CNN.

The CNN learns the information contained in the input signal. We first use the available limited number of snapshots to generate an intensity matrix through FFT operation – which is enhanced by a convolutional neural network (CNN). The CNN is trained with SAR images having a limited number of snapshots as the input and high-resolution RF intensity matrix generated through SAR operation with a large number of snapshots as desired output.

7.3.1 Analytical Model

We denote the desired intensity matrix generated through SAR operation as I_{SAR}, and the intensity matrix generated by the CNN as I_G. We define the loss function of the CNN as:

$$L = \frac{1}{N}\|I_{SAR} - I_G\|^2 \tag{7.1}$$

The above loss function represents the mean squared error function where N is the number of intensity pixels. This loss function is used in numerous super-resolution neural network mod-

els [136, 72], which minimizes the distance between the generated 2D matrices with the desired ones.

7.3.2 Network Architecture

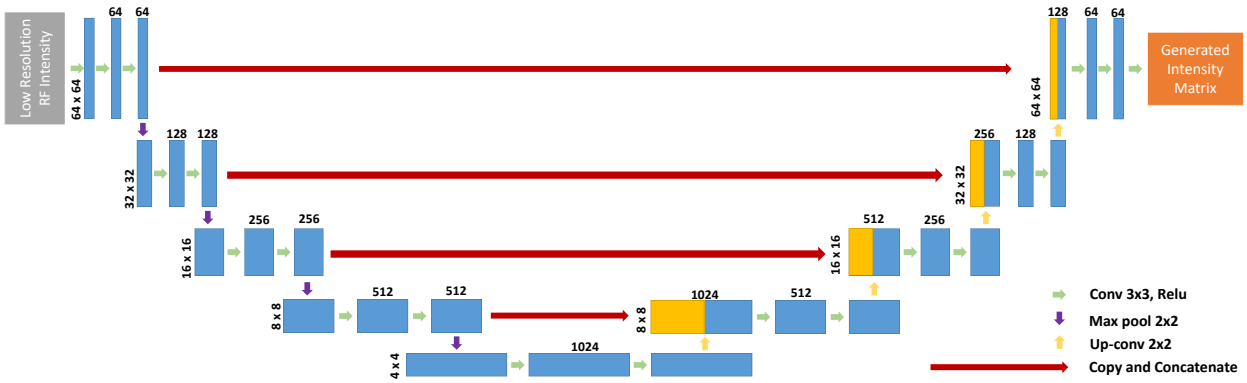


Figure 7.3: SuperRF's neural network architecture. Each box represent a convolutional feature map. The last 2D convolutional layer has only 1 feature map which generates the enhanced RF intensity matrix. The skip connection model allow global structure being used for later stage reconstruction.

The architecture of the proposed deep neural network, which is similar to U-net [102], is shown in Figure 7.3. We choose an encoder-decoder type network which has been shown to perform well in many super-resolution and de-noising problems in computer vision literature [133]. The encoder is a typical convolutional architecture, which consists of 3x3 convolutional filters, followed by ReLU [89] layers. The network is down-sampled with 2x2 max pooling layer with a stride of 2. The feature channels are doubled every time down-sampling happens. The decoder consists of up-sampling the feature map with 2x2 up-convolution, which also reduces the feature channel. The 3x3 convolutional filters and ReLU are also employed here. The final layer is a 1x1 convolution which generates the desired high-resolution intensity matrix. We also incorporate skip connection (residual layer) which helps avoid the vanishing gradients problem and reduces the neural network's size. The residual layers are concatenated to the network.

In general, more and complex features can be extracted by deeper convolutional layers. However, a larger neural network causes over-fitting and increases computational complexity. Based

on our experiments, we choose five stages with a kernel size of 3x3. The 3x3 sized kernel has been proven to be effective in the literature, and multiple 3x3 convolutional layers together can be used to substitute other kernel sizes (such as 5x5) which has higher computational cost.

7.4 Iterative Enhancement Algorithm

7.4.1 Motivation

A neural network is highly dependent on its training and its accuracy beyond what it has seen in the training data is not generally guaranteed. In SuperRF, we observe that while the DNN is fast, its output is often noisy, and depending on the amount of noise, the target application's performance can be significantly degraded. For applications such as robot navigation, in order to ensure that a robot is able to *see* and complement its camera sensors effectively through its RF imaging capability to safely and efficiently navigate the environment, a high degree of accuracy in DNN's output is necessary. However, with limited antenna measurements, the method to further enhance the DNN's output is not straight forward – as one cannot simply make up data out of nowhere. To solve this problem, we take a compressed sensing-based approach in SuperRF. Compressed sensing has recently been proven useful to show that data can be reconstructed with a lower sampling rate as long as the underlying signal is sparse. By treating the number of snapshots as the samples of a compressed sensing problem and by considering the intensity matrix as the intended signal, SuperRF reconstructs a less noisy version of the generated angular intensity representation.

7.4.2 Compressed Sensing

We describe the steps to optimize the intensity matrix generated by the neural network.

Step 1. We estimate the antenna measurements from the neural network generated intensity matrix by performing the inverse Fourier transform (iFFT). The result is an estimation of what the 64-snapshot, full SAR measurement could be.

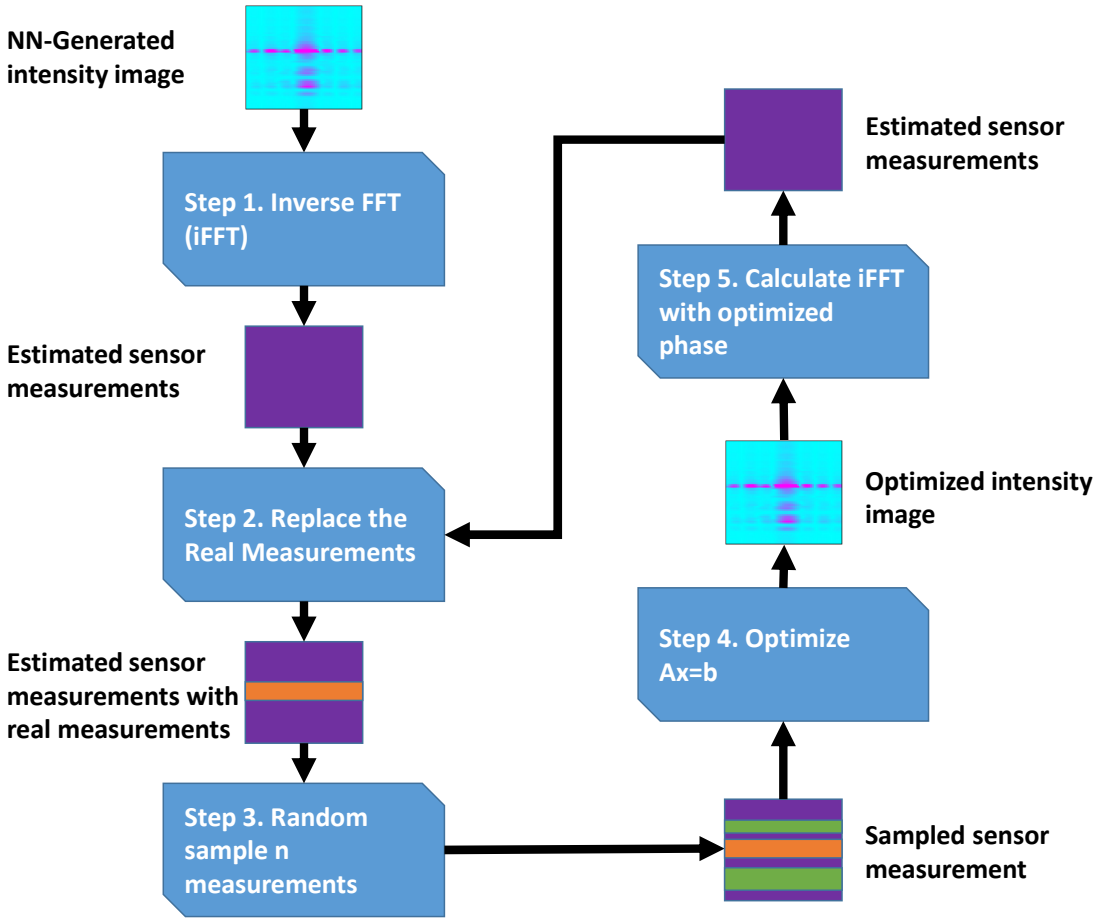


Figure 7.4: A flowchart of the iterative enhancement step.

Step 2. We replace the corresponding antenna measurements with the actual antenna measurements that are available. In our experiment, the middle two snapshots are replaced. This step provides us more accurate antenna measurements.

Step 3. For the newly constructed antenna measurements, we randomly sample n measurements containing the true antenna measurements to form compressed sensing samples. For example, the iFFT step gives us 64 measurements (or, 64 snapshots). We replace the middle two measurements that we have used for the neural network enhancement. The random sampling process takes the two real measurements and samples another $n-2$ measurements from the remaining 62 snapshots to get n synthetic antenna measurements.

Step 4. For compressed sensing, we optimize $Ax = b$, where b denotes the sampled antennas, x denotes the signal we are trying to estimate, corresponding to the intensity matrix. A is the inverse Fourier transform. This step is based on the fact that we have two real measurements. As the real measurements are inserted into the iFFT result, the optimization is performed where the real measurements serve as the constraints.

Step 5. After optimization, we get an estimation of $\hat{x}$ which is a possible explanation of the signals we sampled (note that there is a small number of real measurements while others are estimated from the neural network generated intensity image). Inspired by Griffin-lim algorithm [51], we assume that the amplitude of the neural network generated intensity matrix is correct. Therefore, we replace the amplitude of $\hat{x}$ while maintaining the angle value of it. With the newly constructed $\hat{x}_{new}$, we perform inverse Fourier on it to estimate new antenna measurements.

At this point, we have finished one iteration of our iterative enhancement. In the subsequent iterations, we start from **step 2** to replace the corresponding measurements with the real measurements (the middle two snapshots in our experiments), and then randomly sample another $n - 2$ measurements for the remaining 62 snapshots (which could be the same position as the previous iteration but the value will be different due to optimization) for the new optimization step. A flowchart is shown in Figure 7.4 where we use the intensity image and purple box to represent the intermediate results. The real measurements are shown in orange and the $n - 2$ random sampled measurements are shown in green.

With each iteration, we find a possible solution that contains the real antenna measurements. At the end of the user-defined m iterations, we sort the generated $\hat{x}$ by the order of the inverse Fourier transform of $\hat{x}_{new}$ that has the minimal distance to the real measurements. The distance is calculated as the Euclidean distance between the two snapshots from the iFFT of $\hat{x}_{new}$ and the two real measurements. From the top t samples (t is user defined), we find an average of those samples to get an estimate of the true x – which is the desired intensity matrix. The ranking of the top t samples is based on the fact that there are numerous possible explanations during each

optimization – some are closer to the real one while some are going to be far away. We use the distance as an indicator to find the most likely ones.

7.5 System Implementation

7.5.1 Data Collection

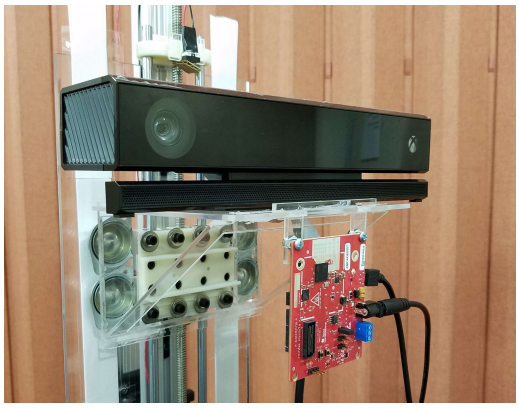
We use an off-the-shelf, low-cost Ti mmWave AWR1443 EVM [1] to collect radar signals and a Kinect V2 sensor to record the ground truth. The setup is shown in Figure 7.5a. We build a motorized lead screw linear slide rail that travels at an interval of $\lambda/2$. Both the Kinect and the Ti mmWave radar are attached to a rigid base support as shown in the figure. The distance between the origin of the Kinect's coordinate system and the center of the antennas is 6.4 cm. We develop necessary control software to automatically move the slider, collect the data, and save the data. We include a Kinect sensor to capture high-quality depth images of the scene which is used as the ground truth for scene reference.

The Ti mmWave radar EVM, as shown in Figure 7.5c, has 3 TX antennas (2 TX at the same level for azimuth angle estimation and 1 TX for elevation capability) and 4 RX antennas. For SAR operation, we collect data from the mmWave radar with 2 TX and 4 RX, which gives us 8 receiving antennas in MIMO mode[2]. The field of view of the radar along the azimuth and elevation directions are $120°$ and $30°$, respectively. The field of view of the Kinect V2 is $70°$ and $60°$.

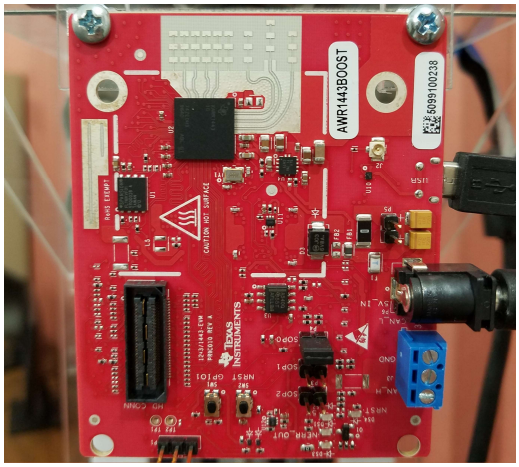
During the data collection phase, we save the color image, depth image, point cloud, and camera intrinsic parameters of the Kinect sensor. For the Ti mmWave Radar, we collect raw RF signals from all antennas for each range index and configurations of the antenna. After saving data for a snapshot, the system moves up the linear slider by $\lambda/2$ and collects new data. For the Ti mmWave radar, we modify the out-of-the-box software to expose the full RF data. Due to the bandwidth constraint of the serial communication between the radar and PC, we collect RF data

[2]We collect data from all 3 TX and 4 RX antennas, however, in this paper, we only use data from the 2 azimuth TX antennas.

(a) Data collection Setup



(b) Kinect and radar



(c) Ti mmWave EVM

Figure 7.5: Data collection system: (a) the full setup where the Kinect and Ti mmWave radar EVM are attached to the linear slider; (b) both the Kinect and the mmWave radar are attached to a rigid support; (c) close up of Ti mmWave radar.

from Doppler bin 0 (static objects). Considering indoor environments, the radar is configured to have approximately 10m maximum working distance.

7.5.2 Objects and Environments

We use a total of 11 objects in our experiment. These objects are made of different materials such as metal, glass, wood, and fiber. We place these objects at different locations (1-2 meters away from the radar) within the field of view of SuperRF and repeat the experiment in different labs and offices in a typical school building. Some of these objects are shown in Figure 7.6.

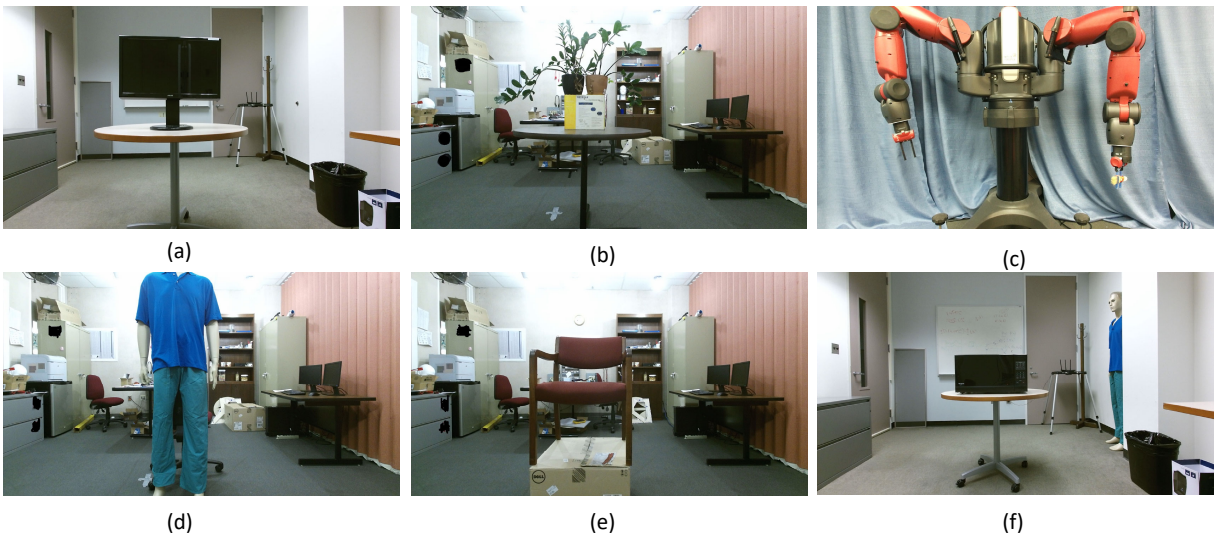


Figure 7.6: Examples of objects used in the experiment: (a) Computer monitor, (b) plants, (c) robot, (d) dummy, (e) chair, and (f) microwave. Note that the objects are placed in three different indoor environments.

To experiment with the see-through capability of the radar, we collect data in non-line-of-sight situations as well. One such scenario is shown in Figure 7.7. The mmWave radar sensor and the object are separated by an office wall. Although the Kinect is not able to see the object, the RF signals penetrate the wall and some of the signals are reflected back from the object, and thus we are able to image the non-line-of-sight object. We use different separators such as walls, closed doors, whiteboards, and wood. We note that the system may not see anything in the non-line-of-sight situations when the separators are made of materials such as metal or thick concrete that block RF signals.

7.5.3 Data Pre-processing

We combine consecutive 64 snapshots to form a 64x8 matrix. The number 8 denotes the number of antennas we have in our radar as described in Chapter 7.5.1. With 64 bins on each axis, 2D FFT is applied on the antenna data – which results in a 64x64 intensity map where each bin represents an angle of arrival. The 64x64 dimensions denote the azimuth and the elevation for the x-axis and the y-axis, respectively. The generated intensity matrix is used as the desired output of the DNN. The 2D FFT represents the SAR operation.

Figure 7.7: An example of occluded sensing where the radar (on the left) and the monitor (on the right) are separated by the office wall.

Out of the 64 snapshots, we choose the middle two snapshots – which is a 2x8 subset of the 64x8 matrix used for the SAR operation. We calculate the 2D FFT of the 2x8 matrix after zero padding to generate 64x64 intensity maps. Zero padding does not increase the angular resolution. The low-resolution intensity matrix becomes the input to the neural network.

An illustration of the data pre-processing is shown in Figure 7.8. We note that the use of the middle two snapshots does not affect the system's performance. The middle two snapshots are chosen because they cover most of the area that the full 64 snapshots cover and they are easier to extract. One can use any two snapshots as long as they are at the same relative position across all intensity images.

7.5.4 Training and Inference

The neural network is trained on a PC having an Nvidia GTX 1080 GPU. We use PyTorch [95] to implement the model. During the training, a batch size of 64 and Adam[68] optimizer are used to optimize the neural network.

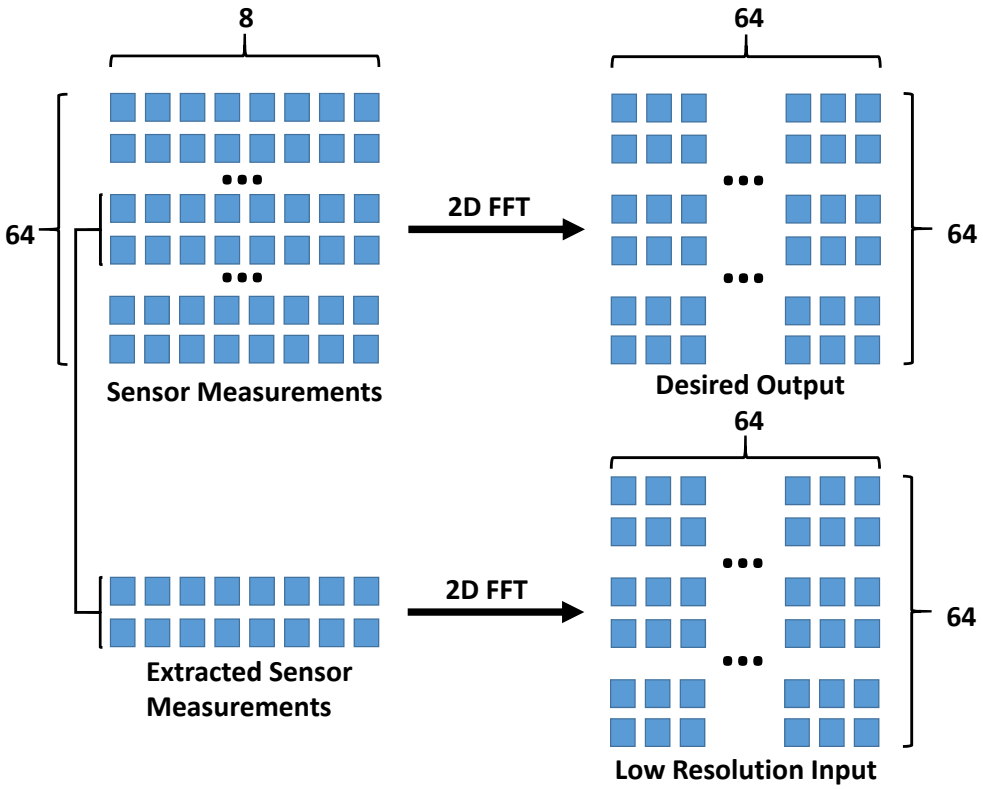


Figure 7.8: Choosing a subset of the 64 measurements as the input. The *Sensor Measurements* are the 64-snapshot radar measurements (the full SAR operation) and the *Desired Output* is the target output after applying the 2D FFT on the 64 snapshots. We retrieve two snapshots and apply the 2D FFT – which is the input to the neural network.

During the training, we use only the line-of-sight data and split the data into two subsets: 80% for training and 20% for validation. A total of 2,914,048 intensity images are in the line-of-sight data, of which, 2,331,264 intensity images are used for training. We train for 50 epochs and each epoch takes around 2.3 hours. All training data, which are the same 2,331,264 intensity images, are used in each epoch. The trained model is used for inference in both line-of-sight and non-line-of-sight scenarios.

7.5.5 Iterative Method

As described in Chapter 7.4.2, we use an iteration of $m = 100$ and randomly choose $n = 15$ samples on the synthetic antenna value. This means during each iteration, we randomly sample 13 antenna measurements and combined with the 2 real measurements for optimization. With

all the generated samples (here we have 100 samples), we choose the $t = 5$ closest to the real antenna measurements for averaging. With our current non-parallel and non-optimized implementation, the time required to perform SuperRF operation requires around 3 seconds, whereas SAR operation requires over 30 seconds (excluding the time needed for physical movement).

7.6 Evaluation

7.6.1 Performance of Neural Network Enhancement

To evaluate the performance of the neural network, we measure the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) of the generated intensity matrix. We report the average error of randomly chosen 15,000 intensity images in Table 7.1. We observe that the neural network generalizes to both line-of-sight and non-line-of-sight scenarios.

	Mean Squared Error (MSE)	**Mean Absolute Error (MAE)**
LOS	31589908.8	1079.2
NLOS	55585840.1	1152.7

Table 7.1: The MSE and MAE for the neural network model in terms of the absolute intensity values.

7.6.2 Performance of Iterative Enhancement

To evaluate the performance of the iterative method, we measure the MSE and MAE of the neural network only and the end-to-end SuperRF that includes the iterative method respectively. The results are shown in Table 7.2. We observe that the iterative method greatly improves on MAE while reducing on MSE slightly. This is due to outliers that contribute to large errors as such errors are squared. Lower MAE means that the overall difference between the ground truth and the generated representation is smaller. These results are computed across different types of objects and with different distances between the object and the radar. The ground truth that these results compare to is the intensity images produced by the SAR operation with 64 snapshots.

	Mean Squared Error (MSE)	**Mean Absolute Error (MAE)**
NN Only	15112451.54	1159.16
SuperRF	15465346.03	1034.23
Percentage	-2.34%	10.78%

Table 7.2: The MSE and MAE with and without the iterative method. The percentage denotes the performance gain due to the iterative method over neural network-only method.

7.6.3 Generated Images

We present some examples of the generated images in Figure 7.9. We observe that SAR operation with only 2 snapshots (column (d)) has very poor resolution, whereas SuperRF generated ones are closer to the ground truth. Note that instead of using 64 snapshots, we only utilized 2 snapshots in SuperRF.

7.6.4 Evaluation of Application Scenario

To better understand the advantage of SuperRF, we train a neural network to perform object recognition on the generated voxelized (3D pixel) representations. The 3D voxel is generated in two ways, one is directly converting intensity matrices to occupancy grid representation through a threshold, another is we trained a separate neural network to perform the conversion.

We divide the objects in our collection into training and testing sets and train the same neural network, which is a standard 3D convolutional neural network (one can use a more advanced neural network such as [59] for better performance), multiple times and measure the object recognition accuracy for each type of data. For the training and evaluation, we use only the area where the objects are placed (thus, no background information is used). The result is shown in Figure 7.10. *Depth* denotes the case where the depth sensor data is used for training and testing. As expected, it shows the highest accuracy. *AI voxel* denotes the voxelized representation generated a neural network with intensity matrices produced by SuperRF. Its accuracy is very close to what depth sensor achieves. The *AI RF* is the non-voxelized RF representation generated by SuperRF. The *SAR* denotes the neural network voxelized RF representation generated through SAR oper-

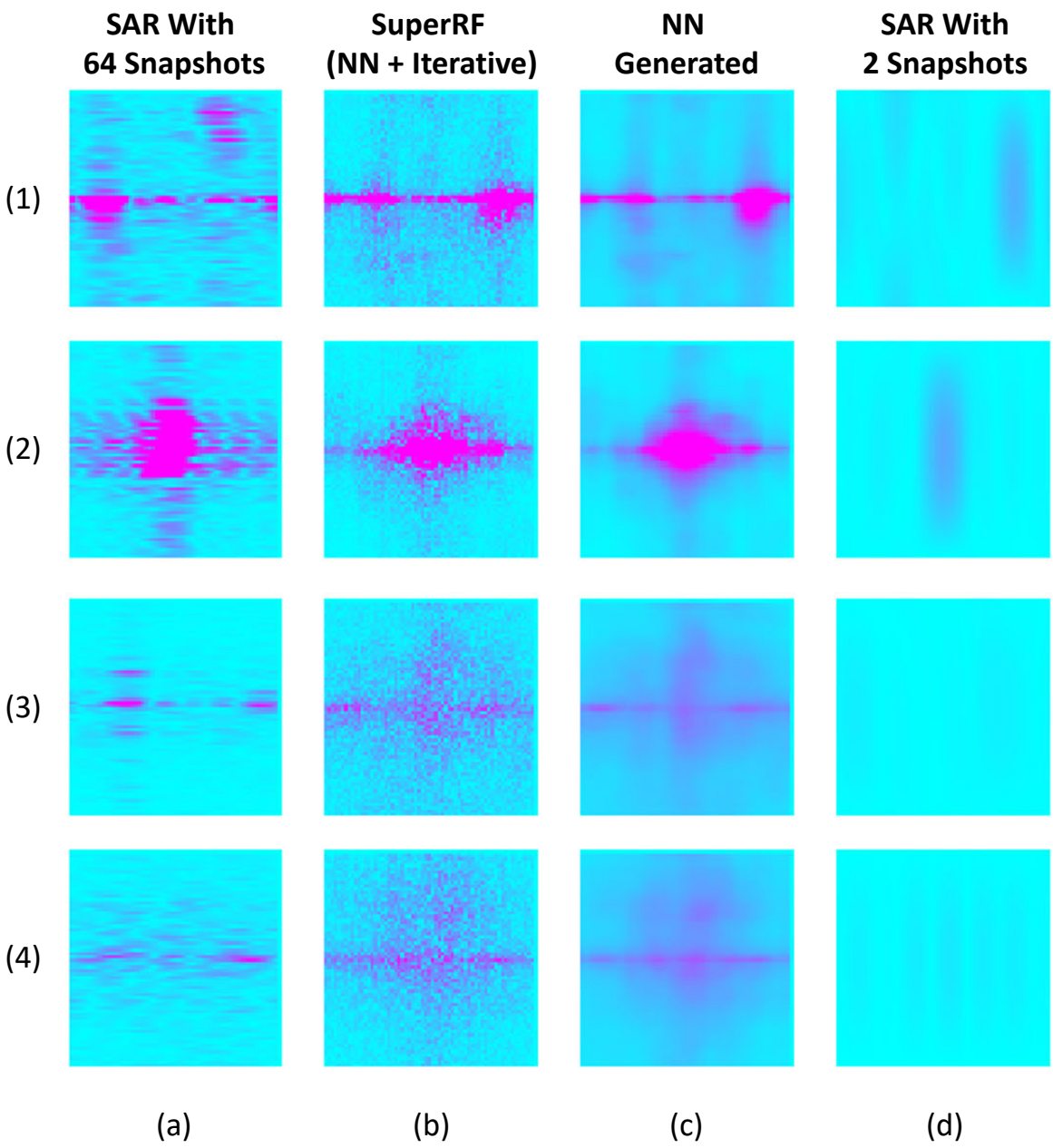


Figure 7.9: Examples of generated intensity matrices. Each row detnoes one example. Column (a) is the SAR operation with 64 snapshots, column (b) is obtained by the full SuperRF algorithm, column (c) shows the intensity matrices directly from the neural network, and column (d) shows the results of SAR operation with only two snapshots.

ation using only 2 snapshots. The *AI RF T* and *SAR T* means to convert the SuperRF generated intensity matrices and SAR generated intensity matrices to the occupancy grid respectively.

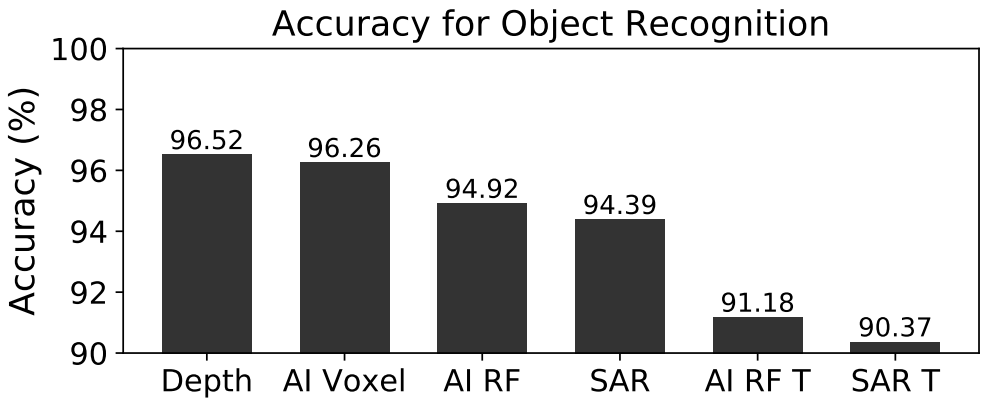


Figure 7.10: Object recognition accuracy for different types of data. *Depth* is the object recognition with Kinect depth image, others are SuperRF generated or through SAR operation. See Chapter 7.6.4 for detail.

In this experiment, the SuperRF performance is evaluated in comparison with Kinect and SAR operation. For different comparisons, such as *AI Voxel* vs. *SAR* and *AI RF T* vs. *SAR T*, SuperRF increases the performance of other applications by generating higher quality representations.

7.7 Summary

In this chapter, we explore the problem of enhancing mmWave radar signals to a higher resolution using a software-based solution. To this end, we propose a two-stage framework. The first stage generates an intensity matrix from only two RF snapshots using a neural network. The second stage further enhances the RF measurements by using a compressed sensing-based iterative method. From this work, we achieved super-resolution on mmWave radar signal for imaging. The enhanced intensity images can improve performance on different applications such as occupancy detection and object recognition. To the best of our knowledge, this is the first of its kind system that enhances mmWave signals to high-quality intensity matrices to represent a scene.

CHAPTER 8: CONCLUSION AND FUTURE WORKS

8.1 Conclusion

This dissertation thesis examines how some of the camera-based sensor systems' fundamental limitations – such as extreme power consumption, privacy-intrusive, and inability to see-through obstacles and other non-ideal visual conditions – can be mitigated. To improve the efficiency as well as the capability of widely deployed camera sensors in various indoor and outdoor spaces, we propose several solutions to tackle the key challenges we identified. First, we propose and implement an autonomous, continuous-vision camera – such as a smart eyeglass or a body camera – which guarantees an extended battery life while minimizing the loss of captured information. This system shows that we can extract meaningful features from the encoded video domain and low-power IMU sensors to infer the scene dynamics and adjust the video settings to achieve a comparable viewing experience. Second, we propose to detect human presence, localize, and identify/re-identify humans in indoor and outdoor spaces such as a warehouse and near a car – where cameras can lack enough pixel values per person to achieve the same or be blocked by obstacles – through WiFi devices. Our proposed solutions leverage existing WiFi infrastructures and show that sensing systems' capabilities can be improved through a combination of other ubiquitous devices. Third, we propose to extend the sensing capability of camera-based systems in non-ideal situations such as bad lighting conditions, smoke, and occlusion by utilizing low-cost, off-the-shelf mmWave radars. Our two-stage algorithm can improve the resolution of mmWave radars by benefiting from the advantages of a neural network and a compressed sensing algorithm while mitigating their respective limitations. These solutions are evaluated through extensive experiments, and we collected a large amount of data which is opened to the community.

8.2 Future Directions

Based on the works done in this thesis, we believe there are multiple aspects that can be improved or extended in the future.

Video Encoding. While we can exploit existing video encoding features such as motion vectors and residuals, new applications can be benefited from a more flexible and adaptive video encoding framework. This can allow users to configure the encoding parameters and better use the features to extract information.

RF Sensing. We explored WiFi and mmWave radar in this thesis, and they provided promising results in practical usage. However, RF sensing modalities still lack performance, which can be improved through a hardware and software co-design in the future.

Multi-modal Sensing Framework. It is possible to incorporate different types of sensors to achieve our goal. Still, we lack a unified multi-modal sensing framework that can be used to combine and extract data easily. Future research can look at how a wide variety of sensors can be used under the same framework, and data can be extracted, processed, compressed, and time-synced to provide easy access.

APPENDIX A: WIFI CSI EXTRACTION

A.1 WiFi CSI Data Collection

There are multiple tools to collect WiFi CSI data, for example, Linux CSI tool [55] and Nexmon [109]. To use the Linux CSI tool, Intel 5300 WiFi NiC card is required, which has already been discontinued and can not long purchase a new copy. This WiFi card can be installed in a compatible desktop motherboard with an m.2 WiFi interface. For portable data collection, several laptops, such as ThinkPad W500 and R400, can be used for portable applications. For use with ThinkPad laptops, firmware may need to be flashed with white-list being removed, so non-ThinkPad certified 5300 chip can be used. As Intel 5300 WiFi chips have 3 antennas, it is advised to attach external antennas for easy antenna placement and choose the distance between them.

A.2 WiFi CSI Data Procession

Both the Linux CSI tool and Nexmon provide MatLab code to process the collected data. It is advised that use the provided code to save the data into a universal format such as HDF5, which can be used in other languages such as Python.

A.3 WiFi CSI Dataset

A.3.0.1 Human Subjects

Human subjects keep holding the phone while moving during the entire data collection. Instead of a single participant holding the phone, the phone is being held by a different person from time to time, introducing variations in height, gait, body shape, and how the phone is being held. In addition to a single person moving in the environment, we also included cases where multiple people are present simultaneously. In such cases, they walk randomly by themselves and create multiple trajectories that can be used to develop and test trajectory matching solutions. The

presence of multiple people also introduces human body shadowing effects in the RF signals. To diversify human subjects, a total of 15 different individuals held the phone.

A.3.0.2 Time Variation

The dataset is collected over the course of several days in the lab. The data collection hardware (NUC and the phone) is rebooted each day that causes variations in hardware offsets and captures how the WiFi signal may change over time. Other aspects, such as placements of items in the lab can also vary slightly from day to day as the lab was shared with other teams. The time variation provides a realistic emulation of long-term deployment in the real world.

A.3.0.3 Time Synchronization

To get the corresponding CSI packets and locations of individuals detected by the camera, Pixel 2 XL is used to time synchronize the phone, panoramic camera, and Intel NUC. The clock of the camera is shown in each camera frame in milliseconds (e.g., at the top right of Figure4.6a). In order to time synchronize, the phone is held under the panoramic camera showing its clock in milliseconds, thus capturing the clocks of the phone and the camera in a single frame allowing us to compute the difference. The same technique is used with the NUC and the phone, where a USB camera is attached to the NUC to capture the phone's clock. During the data collection, in addition to the recording of the CSI values of each received packet, we also record the corresponding timestamps when each packet is received on the NUC. When the dataset is generated, the camera and the NUC's timestamps are converted to the same time reference of the panoramic camera. The location of the transmitter of each CSI packet is determined by the (x,y) location detected by the camera at the nearest time.

A.3.0.4 Phase Offset

Due to the WiFi Network Interface Card (NIC) imperfections, phase offset exists in the measured WiFi CSI. Zhang et al. [139] state that the phase offsets between the RF chains on Intel

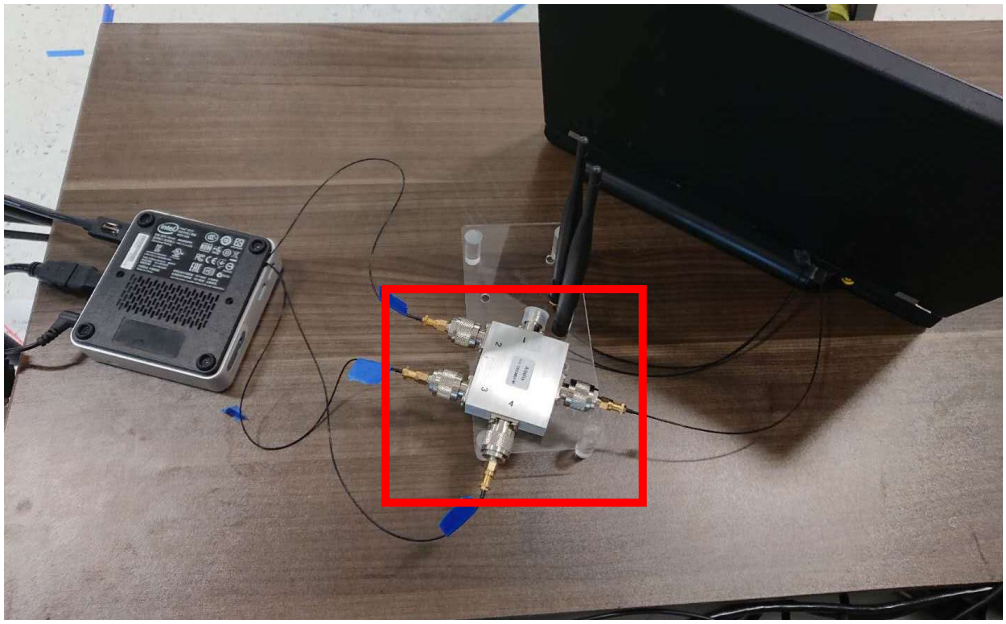


Figure A.1: Phase offset measurement setup. The red box highlights the RF-splitter used in the phase offset measurement, where a transmitter cable is connected to the input of the splitter, and three receiving antennas are connected to the output of the splitter.

5300 WiFi NIC are deterministic, and the offset between two RF chains only poses two possible values. We determine the two values based on our own measurements by connecting the antenna cables of two Intel 5300 WiFi NICs through an RF splitter, as shown in Figure A.1, and operate the WiFi devices in injection and monitoring mode. The RF splitter is used to minimize the impact of the differences in the length of WiFi transmission paths and enables the reception of the same WiFi packet at the same time. The phase offsets are calculated as the phase difference between two antennas and across multiple reboots to measure the two possible values, as stated in [139].

A.3.1 Collected Dataset

A.3.1.1 Overview

We collect WiFi packets and corresponding CSI values of over 13 hours. We also have over 15 different individuals holding the phone to capture various ways people hold phones, their walking patterns, and different heights [43]. In addition to a single person walking in the scene,

Folder Name	Description
1_person	This folder contains data collected when only a single person who is also holding the phone in the scene. Collected in the lab area.
x_people	x (x = 2,3,5,10) is the number of people present in the scene. Only one person is holding the phone at a time. All data under these folders are collected in the lab area.
Kitchen —— 1_person —— 3_people	This folder contains all the data collected in the Kitchen area. Includes subfolder with only 1 person in the scene and 3 people in the scene.
training	This is a randomly shuffled dataset with data from the 1_person folder.

Table A.1: Folders in the dataset.

we also have multiple people walking simultaneously. In the dataset, we include the raw CSI information, timestamp, angle of arrival (AoA), and Cartesian coordinates of each WiFi packet transmitter. AoAs calculated from SpotFi are also included that we will discuss more in Section A.3.1.3.

The dataset is organized into different folders, as shown in Table A.1. All files under each folder are stored in HDF5 format, and they contain the following information (except in the training folder):

1. **CSI:** Raw CSI data retrieved with Linux CSI Tool [55]. They are stored separately with *csi_imag* and *csi_real* representing real and imaginary parts of the CSI values, respectively. There is a total of 1,020,167 WiFi packets, with 774,897 packets collected in the lab and 245,270 in the kitchen area. Out of the packets collected in the lab, 551,027 packets are for a single person (one person at a time, but over 15 persons' data), 119,441 are for 2 persons, 48,981 are for 3 persons, 13,448 for 5 persons, and 12,000 for 10 persons simultaneously. From the kitchen area, 202,843 packets are collected from a single person, and 42,427 are collected from 3 persons simultaneously.

2. **Number of Human Subjects:** The number of human subjects present in the scene is included for a quick reference. Despite many people present in the scene, only one person holds the phone at a time that transmits WiFi packets.

3. **Coordinates and AoAs of Human Subjects:** The camera generates Cartesian coordinates of all the detected human subjects. We calculate the Angle of Arrival using the unified coordinate system (Camera, NUC) to provide each person's ground truth location.

4. **Timestamp:** The timestamp when each WiFi packet is received is also included to help estimating the duration of time slices.

5. **AoA Generated by SpotFi:** We include the AoAs generated by using the SpotFi algorithm and corresponding MATLAB implementation [7].

Data in the training folder are random shuffled and splitted data from 1_person folder, when a single person is present in the scene and holding the phone. The shuffled data only contains ground truth AoA, raw CSI values, and AoA generated from SpotFi. It includes train (70%), validation (20%), and test (10%) split so that different solutions can benchmark their performances.

A.3.1.2 Camera Performance

As we use camera-provided coordinates and subsequent AoAs as ground truth, we evaluate the camera-based localization accuracy. The evaluation is done by standing at 16 different locations in the lab, as shown in Figure A.2, and estimating the difference between the measured ground truth and camera provided (x,y) coordinates. The results are shown in Table A.2. As we see in the table, the average error from the camera is only 0.32 meters, which translates to 1.62 °in absolute AoA difference. So, it can be used as almost ground truth.

		Median Error	Mean Error
Coordinates (m)	x	0.1344	0.3169
	y	0.24	0.2864
Angle of Arrival (AoA) (°)	$\angle$	1.022	1.032
	$\lvert\angle\rvert$	1.299	1.618

Table A.2: Error between ground truth and camera provided locations.

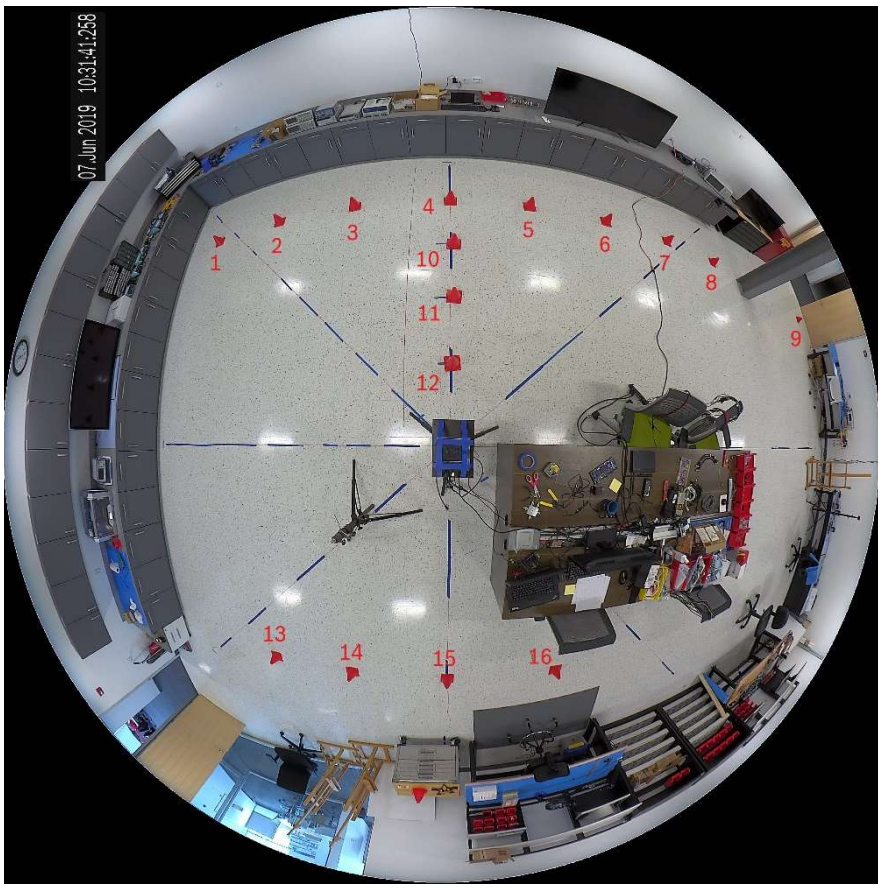


Figure A.2: The performance of the camera system is measured by standing at 16 different locations as shown in this figure, and measure the maximum and average error.

A.3.1.3 Angle of Arrival (AoA) with SpotFi Algorithm

SpotFi [69] is a state-of-the-art solution to estimate the Angle of Arrival (AoA) based on WiFi CSI measurements. In this dataset, we provide the SpotFi calculated AoA with our CSI values with and without applying the phase offsets. The calculation is performed with code provided by the original author in MATLAB [7]. The performance of the SpotFi can be used as a baseline for future solutions.

APPENDIX B: MMWAVE SENSING

B.1 mmWave Data Collection

TI (Texas Instrument) mmWave radar can be used to collect point cloud data using stock firmware. However, to collect raw antenna complex values, either a data capture card is required, or modified firmware is needed. To modify the firmware, one can output the value after range and Doppler FFT, which is being computed onboard. The data transfer speed through the USB port is slow, limiting the amount of data to be transferred. Only one Doppler bin result can be transferred and takes around 0.5 to 1 second. The transfer over USB is standard serial communication, and the decoding of the data can be found with TI examples.

B.2 mmWave Data Procession

The collected data can be processed use FFT to assign intensity to each FFT bin. Each FFT bin corresponds to the angle where the signal is coming from, and each bin represents a range of angles. The range of angle is defined by the field of view of the radar and the number of FFT bins. Therefore, while increasing the number of FFT bins can increase the resolution of the FFT, but it does not increase the actual resolution of the radar, which is defined by the number of antennas.

B.3 mmWave Dataset

The dateset collected in SuperRF project is available at `https://bitbucket.org/embedded_intelligence/superrf_dataset/src/master/`

REFERENCES

[1] http://www.ti.com/product/awr1443. [Online; accessed 24-Aug-2018].

[2] Face Recognition. https://github.com/ageitgey/face_recognition.

[3] Google clips. https://store.google.com/us/product/google_clips.

[4] Google clips review: A smart, but unpredictable camera. https://www.engadget.com/2018/02/27/google-clips-ai-camera-review/.

[5] Myriad 2 ma2x5x vision processor product brief. https://uploads.movidius.com/1463156689-2016-04-29_VPU_ProductBrief.pdf.

[6] San Francisco Banned Facial Recognition. https://www.nytimes.com/2019/07/01/us/facial-recognition-san-francisco.html.

[7] SpotFi Matlab implementation. https://bitbucket.org/mkotaru/spotfimusicaoaestimation/src/master/.

[8] Tesla Model S Owner's Manual. https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf.

[9] The Clearest Explanation Yet for Why Millennials Are Driving Less - Bloomberg. https://www.bloomberg.com/news/articles/2015-07-13/why-millennials-are-driving-less-than-previous-generations-did-at-the-same-age.

[10] S. Aarthi and S. Chitrakala. Scene understanding—a survey. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–4. IEEE, 2017.

[11] H. Abdelnasser, K. A. Harras, and M. Youssef. Ubibreathe: A ubiquitous non-invasive wifi-based breathing estimator. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, pages 277–286, New York, NY, USA, 2015. ACM.

[12] H. Abdi. The kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, pages 508–510, 2007.

[13] F. Adib, C.-Y. Hsu, H. Mao, D. Katabi, and F. Durand. Capturing the human figure through a wall. *ACM Transactions on Graphics (TOG)*, 34(6):219, 2015.

[14] F. Adib, Z. Kabelac, and D. Katabi. Multi-person localization via rf body reflections. In *NSDI*, pages 279–292, 2015.

[15] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller. 3d tracking via body radio reflections.

[16] F. Adib, H. Mao, Z. Kabelac, D. Katabi, and R. C. Miller. Smart homes that monitor breathing and heart rate. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 837–846, New York, NY, USA, 2015. ACM.

[17] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[18] K. Ali, A. X. Liu, W. Wang, and M. Shahzad. Keystroke recognition using wifi signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 90–102, New York, NY, USA, 2015. ACM.

[19] I. An, M. Son, D. Manocha, and S.-e. Yoon. Reflection-aware sound source localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 66–73. IEEE, 2018.

[20] R. V. Babu, M. Tom, and P. Wadekar. A survey on compressed domain video analysis techniques. *Multimedia Tools and Applications*, 75(2):1043–1078, Jan 2016.

[21] R. V. Babu, M. Tom, and P. Wadekar. A survey on compressed domain video analysis techniques. *Multimedia Tools and Applications*, 75(2):1043–1078, 2016.

[22] B. Bai, Y. He, J. Liu, Y. Zhou, H. Zheng, S. Zhang, and Z. Xu. Imaging around corners with single-pixel detector by computational ghost imaging. *Optik*, 147:136–142, 2017.

[23] C. Basri and A. El Khadimi. Survey on indoor localization system and recent advances of wifi fingerprinting technique. In *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pages 253–259. IEEE, 2016.

[24] S. Biswas and R. V. Babu. H. 264 compressed video classification using histogram of oriented motion vectors (homv). In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2040–2044. IEEE, 2013.

[25] S. Biswas and R. V. Babu. Real time anomaly detection in h. 264 compressed videos. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4. IEEE, 2013.

[26] D. Bliss and K. Forsythe. Multiple-input multiple-output (mimo) radar and imaging: degrees of freedom and resolution. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 1, pages 54–59. IEEE, 2003.

[27] W. Bolton. *Instrumentation and control systems*. Newnes, 2015.

[28] S. Chen, W. Tang, X. Zhang, and E. Culurciello. A 64×64 pixels uwb wireless temporal-difference digital image sensor. *IEEE transactions on very large scale integration (VLSI) systems*, 20(12):2232–2240, 2011.

[29] T.-C. Chen, Y.-W. Huang, and L.-G. Chen. Analysis and design of macroblock pipelining for h. 264/avc vlsi architecture. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512)*, volume 2, pages II–273. IEEE, 2004.

[30] J. Choi, S. Park, J. Cho, and E. Yoon. A 3.4 μw cmos image sensor with embedded feature-extraction algorithm for motion-triggered object-of-interest imaging. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 478–479. IEEE, 2013.

[31] W. S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. In *Statistical theory and computational aspects of smoothing*, pages 10–49. Springer, 1996.

[32] F. Coudert, D. Butin, and D. Le Métayer. Body-worn cameras for police accountability: Opportunities and risks. *Computer law & security review*, 31(6):749–762, 2015.

[33] J. C. Curlander and R. N. McDonough. *Synthetic aperture radar*, volume 11. Wiley, New York, 1991.

[34] N. Czink, M. Herdin, H. Özcelik, and E. Bonek. Number of multipath clusters in indoor mimo propagation environments. *Electronics letters*, 40(23):1498–1499, 2004.

[35] L. Czúni, G. Császár, and A. Licsár. Estimating the optimal quantization parameter in h. 264. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 330–333. IEEE, 2006.

[36] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[37] S. Di Domenico, M. De Sanctis, E. Cianca, and M. Ruggieri. Wifi-based through-the-wall presence detection of stationary and moving humans analyzing the doppler spectrum. *IEEE Aerospace and Electronic Systems Magazine*, 33(5-6):14–19, 2018.

[38] S. Fang, R. Alterovitz, and S. Nirjon. Non-line-of-sight around the corner human presence detection using commodity wifi devices. In *Proceedings of the 1st ACM International Workshop on Device-Free Human Sensing*, pages 22–26, 2019.

[39] S. Fang, T. Islam, S. Munir, and S. Nirjon. Eyefi: Fast human identification through vision and wifi-based trajectory matching. In *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 59–68. IEEE, 2020.

[40] S. Fang, K. Mayer-Patel, and S. Nirjon. Distributed adaptive model predictive control of a cluster of autonomous and context-sensitive body cameras. In *Proceedings of the 2017 Workshop on Wearable Systems and Applications*, pages 35–40. ACM, 2017.

[41] S. Fang, K. Mayer-Patel, and S. Nirjon. Zencam: Context-driven control of autonomous body cameras. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 41–48. IEEE, 2019.

[42] S. Fang, K. Mayer-Patel, and S. Nirjon. Exploiting scene and body contexts in controlling continuous vision body cameras. *Ad Hoc Networks*, 113:102373, 2021.

[43] S. Fang, S. Munir, and S. Nirjon. Person tracking and identification using cameras and wi-fi channel state information (csi) from smartphones: dataset. In *Proceedings of the Third Workshop on Data: Acquisition To Analysis*, pages 26–30, 2020.

[44] S. Fang and S. Nirjon. Ai-enhanced 3d rf representation using low-cost mmwave radar. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 414–415. ACM, 2018.

[45] S. Fang and S. Nirjon. Superrf: Enhanced 3d rf representation using stationary low-cost mmwave radar. In *International Conference on Embedded Wireless Systems and Networks (EWSN)...*, volume 2020, page 120. NIH Public Access, 2020.

[46] H. C. Folts. Open systems interconnection reference model. In *Telecommunications Engineer's Reference Book*, pages 12–1. Elsevier, 1993.

[47] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell. *Feedback control of dynamic systems*, volume 3. Addison-Wesley Reading, MA, 1994.

[48] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson. Phaser: Enabling phased array signal processing on commodity wifi access points. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 153–164. ACM, 2014.

[49] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E.-h. Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors*, 16(12):1966, 2016.

[50] M. Graphics and M. L. V. Group). MSU Video Quality Measurement Tool. `http://www.compression.ru/video/quality_measure/video_measurement_tool.html`, 2017. [Online; accessed 30-Nov-2017].

[51] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[52] P. Guillotel and C. Chevance. Comparison of motion vector coding techniques. In *Visual Communications and Image Processing'94*, volume 2308, pages 1594–1604. International Society for Optics and Photonics, 1994.

[53] A. C. V. Gummalla and J. O. Limb. Wireless medium access control protocols. *IEEE Communications Surveys & Tutorials*, 3(2):2–15, 2000.

[54] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: Gathering 802.11 n traces with channel state information. *ACM SIGCOMM Computer Communication Review*, 41(1):53–53, 2011.

[55] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM CCR*, 41(1):53, Jan. 2011.

[56] O. Hamdoun, F. Moutarde, B. Stanciulescu, and B. Steux. Person re-identification in multi-camera system by signature based on interest point descriptors collected on short video sequences. In *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6. IEEE, 2008.

[57] S. Han, R. Nandakumar, M. Philipose, A. Krishnamurthy, and D. Wetherall. Glimpsedata: Towards continuous vision-based personal analytics. In *Proceedings of the 2014 workshop on physical analytics*, pages 31–36. ACM, 2014.

[58] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136. ACM, 2016.

[59] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[60] IEEE. Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, 2016.

[61] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[62] Y. Jin, D. Jiang, and M. Cai. 3d reconstruction using deep learning: a survey. *Communications in Information and Systems*, 20(4):389–413, 2020.

[63] C. Käs and H. Nicolas. An approach to trajectory estimation of moving objects in the h. 264 compressed domain. In *Pacific-Rim Symposium on Image and Video Technology*, pages 318–329. Springer, 2009.

[64] C. M. Katz, D. E. Choate, J. R. Ready, and L. Nuño. Evaluating the impact of officer worn body cameras in the phoenix police department. *Phoenix, AZ: Center for Violence Prevention and Community Safety, Arizona State University*, 2014.

[65] O. Katz, P. Heidmann, M. Fink, and S. Gigan. Non-invasive single-shot imaging through scattering layers and around corners via speckle correlations. *Nature photonics*, 8(10):784–790, 2014.

[66] O. Katz, E. Small, and Y. Silberberg. Looking around corners and through thin turbid layers in real time with scattered incoherent light. *Nature photonics*, 6(8):549–553, 2012.

[67] N. Khalil, D. Benhaddou, O. Gnawali, and J. Subhlok. Sonicdoor: scaling person identification with ultrasonic sensors by novel modeling of shape, behavior and walking patterns. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, page 3. ACM, 2017.

[68] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. Spotfi: Decimeter level localization using wifi. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 269–282. ACM, 2015.

[70] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. Spotfi: Decimeter level localization using wifi. *SIGCOMM Comput. Commun. Rev.*, 45(4):269–282, Aug. 2015.

[71] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi. *Adaptive control*, volume 51. Springer New York, 1998.

[72] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[73] S.-M. Lei and M.-T. Sun. An entropy coding system for digital hdtv applications. *IEEE transactions on circuits and systems for video technology*, 1(1):147–155, 1991.

[74] Q. Leng, M. Ye, and Q. Tian. A survey of open-world person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(4):1092–1108, 2019.

[75] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang. 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, pages 250–261, New York, NY, USA, 2016. ACM.

[76] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[77] L. Liu, H. Li, and Y. Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2372–2381, 2017.

[78] S. Liu, M. Li, S. Zhu, and B. Zeng. Codingflow: Enable video coding for video stabilization. *IEEE Transactions on Image Processing*, 26(7):3291–3302, July 2017.

[79] D. Lyon. *Surveillance society: Monitoring everyday life*. McGraw-Hill Education (UK), 2001.

[80] Y. Ma, G. Zhou, and S. Wang. Wifi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)*, 52(3):1–36, 2019.

[81] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[82] M. Marcus and B. Pattan. Millimeter wave propagation: spectrum management implications. *IEEE Microwave Magazine*, 6(2):54–62, 2005.

[83] R. Mautz and S. Tilch. Survey of optical indoor positioning systems. In *2011 international conference on indoor positioning and indoor navigation*, pages 1–7. IEEE, 2011.

[84] M. Mehrabi, F. Zargari, and M. Ghanbari. Compressed domain content based retrieval using h. 264 dc-pictures. *Multimedia Tools and Applications*, 60(2):443–453, 2012.

[85] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. An overview of tiles in hevc. *IEEE journal of selected topics in signal processing*, 7(6):969–977, 2013.

[86] Y. H. Moon, G. Y. Kim, and J. H. Kim. An efficient decoding of cavlc in h. 264/avc video coding standard. *IEEE Transactions on Consumer Electronics*, 51(3):933–938, 2005.

[87] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou. A tutorial on synthetic aperture radar. *IEEE Geoscience and remote sensing magazine*, 1(1):6–43, 2013.

[88] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan. Glimpse: A programmable early-discard camera architecture for continuous mobile vision. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 292–305. ACM, 2017.

[89] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[90] Y. Nakaya and H. Harashima. Motion compensation based on spatial transformations. *IEEE Transactions on circuits and systems for video technology*, 4(3):339–356, 1994.

[91] K. Nasrollahi and T. B. Moeslund. Super-resolution: a comprehensive survey. *Machine vision and applications*, 25(6):1423–1468, 2014.

[92] Y. Ninomiya and Y. Ohtsuka. A motion-compensated interframe coding scheme for television pictures. *IEEE Transactions on Communications*, 30(1):201–211, January 1982.

[93] S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao. Musicalheart: A hearty way of listening to music. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 43–56. ACM, 2012.

[94] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Modeling the impact of frame rate on perceptual quality of video. In *2008 15th IEEE International Conference on Image Processing*, pages 689–692, Oct 2008.

[95] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[96] L. Patrick, C. Posch, and T. Delbruck. A 128x 128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43:566–576, 2008.

[97] A. Paulraj, R. Roy, and T. Kailath. Estimation of signal parameters via rotational invariance techniques-esprit. In *Nineteeth Asilomar Conference on Circuits, Systems and Computers, 1985.*, pages 83–89. IEEE, 1985.

[98] M. H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, Sept 2004.

[99] W. H. Press and S. A. Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990.

[100] I. D. Raji and G. Fried. About face: A survey of facial recognition evaluation. *arXiv preprint arXiv:2102.00813*, 2021.

[101] I. E. Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[102] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[103] A. Roy. *On–Officer Video Cameras: Examining the Effects of Police Department Policy and Assignment on Camera Use and Activation*. PhD thesis, Arizona State University, 2014.

[104] G. Satat, M. Tancik, and R. Raskar. Towards photography through realistic fog. In *2018 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10. IEEE, 2018.

[105] C. Saunders, J. Murray-Bruce, and V. K. Goyal. Computational periscopy with an ordinary digital camera. *Nature*, 565(7740):472, 2019.

[106] A. Saxena and F. C. Fernandes. Dct/dst-based transform coding for intra prediction in image/video coding. *IEEE Transactions on Image Processing*, 22(10):3974–3981, 2013.

[107] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[108] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation*, 34(3):276–280, 1986.

[109] M. Schulz, D. Wegemer, and M. Hollick. Nexmon: The c-based firmware patching framework, 2017.

[110] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[111] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1442–1468, 2013.

[112] N. Snavely. Scene reconstruction and visualization from internet photo collections: A survey. *IPSJ Transactions on Computer Vision and Applications*, 3:44–66, 2011.

[113] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[114] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[115] V. Sze and M. Budagavi. High throughput cabac entropy coding in hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1778–1791, 2012.

[116] Y. Taki, M. Hatori, and S. Tanaka. Interframe coding that follows the motion. *Proc. Institute of Electronics and Communication Engineers Jpn. Annu. Conv.(IECEJ)*, page 1263, 1974.

[117] I. TEKTRONIX. Wi-fi: Overview of the 802.11 physical layer and transmitter measurements. Brochure, 2016. `https://download.tek.com/document/37W-29447-2_LR.pdf`, Accessed: 12-1-2020.

[118] N. Telecommunications and I. Administration. U.s. frequency allocation chart. Brochure, 2016. `https://www.ntia.doc.gov/files/ntia/publications/january_2016_spectrum_wall_chart.pdf`, Accessed: 12-1-2020.

[119] V. Thilak and C. D. Creusere. Tracking of extended size targets in h. 264 compressed video using the probabilistic data association filter. In *2004 12th European Signal Processing Conference*, pages 281–284. IEEE, 2004.

[120] S. Thomas, C. Bredendiek, and N. Pohl. A sige-based 240-ghz fmcw radar system for high-resolution measurements. *IEEE Transactions on Microwave Theory and Techniques*, 67(11):4599–4609, 2019.

[121] M. Tom, R. V. Babu, and R. G. Praveen. Compressed domain human action recognition in h. 264/avc video streams. *Multimedia Tools and Applications*, 74(21):9323–9338, 2015.

[122] D. Tse and P. Viswanath. *Fundamentals of wireless communication*. Cambridge university press, 2005.

[123] D. Vasisht, S. Kumar, and D. Katabi. Decimeter-level localization with a single wifi access point. In *NSDI*, volume 16, pages 165–178, 2016.

[124] A. Velten, T. Willwacher, O. Gupta, A. Veeraraghavan, M. G. Bawendi, and R. Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature communications*, 3(1):1–8, 2012.

[125] A. Virmani and M. Shahzad. Position and orientation agnostic gesture recognition using wifi. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, pages 252–264, New York, NY, USA, 2017. ACM.

[126] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni. We can hear you with wi-fi! In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 593–604, New York, NY, USA, 2014. ACM.

[127] L. Wang. *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[128] X. Wang, L. Gao, and S. Mao. Csi phase fingerprinting for indoor localization with a deep learning approach. *IEEE Internet of Things Journal*, 3(6):1113–1123, 2016.

[129] Y. Wang, K. Wu, and L. M. Ni. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581–594, 2017.

[130] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[131] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[132] C. Wu, Z. Yang, Z. Zhou, X. Liu, Y. Liu, and J. Cao. Non-invasive detection of moving and stationary human with wifi. *IEEE Journal on Selected Areas in Communications*, 33(11):2329–2342, 2015.

[133] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[134] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[135] Z. Yang, Z. Zhou, and Y. Liu. From rssi to csi: Indoor localization via channel response. *ACM Computing Surveys (CSUR)*, 46(2):25, 2013.

[136] X. Yu and F. Porikli. Ultra-resolving face images by discriminative generative networks. In *European conference on computer vision*, pages 318–333. Springer, 2016.

[137] J. H. Zar. Spearman rank correlation. *Encyclopedia of biostatistics*, 7, 2005.

[138] B. Zeng and J. Fu. Directional discrete cosine transforms—a new framework for image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(3):305–313, 2008.

[139] D. Zhang, Y. Hu, Y. Chen, and B. Zeng. Calibrating phase offsets for commodity wifi. *IEEE Systems Journal*, PP:1–4, 03 2019.

[140] S. Zhang, S. C. Liew, and P. P. Lam. Hot topic: Physical-layer network coding. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 358–365, 2006.

[141] M. Zhao, F. Adib, and D. Katabi. Emotion recognition using wireless signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 95–108. ACM, 2016.

[142] M. Zhao, T. Li, M. Abu Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7356–7365, 2018.

[143] M. Zhao, Y. Tian, H. Zhao, M. A. Alsheikh, T. Li, R. Hristov, Z. Kabelac, D. Katabi, and A. Torralba. Rf-based 3d skeletons. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 267–281. ACM, 2018.

[144] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang. Camera style adaptation for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5157–5166, 2018.

[145] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld. Impact of frame rate and resolution on objective qoe metrics. In *2010 Second International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 29–34, June 2010.

[146] H. Zou, B. Huang, X. Lu, H. Jiang, and L. Xie. A robust indoor positioning system based on the procrustes analysis and weighted extreme learning machine. *IEEE Transactions on Wireless Communications*, 15(2):1252–1266, 2016.

[147] H. Zou, Y. Zhou, J. Yang, W. Gu, L. Xie, and C. Spanos. Freedetector: Device-free occupancy detection with commodity wifi. In *2017 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–5. IEEE, 2017.

[148] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.