RF Sensing for Internet of Things: When Machine Learning Meets Channel State Information

by

Xuyu Wang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
Aug. 04, 2018

Keywords: Internet of Things, Indoor Localization, Vital Sign Monitoring, Deep Learning, Channel State Information

Approved by

Shiwen Mao, Samuel Ginn Endowed Professor of Electrical and Computer Engineering
Jitendra Tugnait, James B Davis Professor of Electrical and Computer Engineering
Mark Nelms, Professor and Chair of Electrical and Computer Engineering
Bo Liu, Assistant Professor of Computer Science and Software Engineering

Abstract

With the rapid development of Internet of Things (IoT) techniques, RF sensing has found wide applications for, e.g., indoor localization, activity recognition, and healthcare. In this dissertation, we investigate the problem of RF sensing for IoT using channel state information (CSI) and machine learning techniques. In particular, our work mainly focuses on indoor localization using deep learning and vital sign monitoring for RF sensing.

In this dissertation, we first study the problem of CSI based indoor localization. For first three works, we exploit deep learning for three different indoor localization systems using CSI amplitudes, CSI calibrated phases, and CSI bimodal data, respectively. Moreover, we study and analyze CSI data, which is stable for indoor localization. We consider deep autoencoder networks to train CSI data, and employ the weights of the deep network to represent fingerprints. A greedy learning algorithm is leveraged to train the weights layer-by-layer to reduce computational complexity, where a sub-network between two consecutive layers forms a Restricted Boltzmann Machine (RBM). In the online stage, we use a probabilistic method for online location estimation.

Then, we exploit deep convolutional neural networks (DCNN) for indoor localization. Since DCNN is a supervised method, it only requires to train one group of weights for all the training data with related labels, which is different with our prior works that requires training weights for every training location. Specially, we use estimated angle of arrival (AOA) images from CSI data as input to the DCNN. By executing four convolutional and subsampling layers, the system can automatically extract the features of the estimated AOA images, to obtain training weights. To improve indoor localization accuracy, we propose deep residual sharing learning for training two channels CSI tensor data. Moreover, we can stack many residual sharing blocks for adding the depth of the deep network, thus achieving higher learning and representation ability for CSI tensor

data. The proposed system can achieve decimeter level location accuracy, which is better than other deep learning methods.

This dissertation also focuses on vital sign monitoring using CSI and machine learning techniques. First, we consider CSI phase difference data to monitor breathing and heart beats with commodity WiFi device. We implement data preprocessing for the collected CSI phase difference data to obtain the denoised breathing signal and the restructured heart signal. Moreover, we leverage the peak detection method for breathing rate estimation and FFT based method for heart signal estimation. To estimate breathing rates for multiple persons with CSI data. We leverage the tensor decomposition technique to handle the CSI phase difference data. This work first uses CSI phase difference data to create CSI tensor data. Then Canonical Polyadic (CP) decomposition is applied to obtain the desired breathing signals. A stable signal matching algorithm is developed to find the decomposed signal pairs, and a peak detection method is applied to estimate the breathing rates for multiple persons. To improve the robustness of breathing signs monitoring, we exploit bimodal CSI data, including amplitude and phase difference, for realtime breathing monitoring. Then, we implement the data preprocessing, adaptive signal selection, and breathing signal monitoring modules, and employ peak detection to estimate breathing rates.

The last work of this dissertation considers a phase based active sonar to monitor breathing rates with smartphones. We implement several signal processing algorithms, including signal generation, data extraction, received signal preprocessing, and breathing rate estimation. Specially, we propose an adaptive median filter approach to remove the static vector in the received signal, which allows to effectively extract the inaudible phase information. Our experimental results validate the superior performance in different indoor environment settings.

Acknowledgments

First of all, I would like to express my sincere gratitude to my major advisor Prof. Shiwen Mao, who provided me the continuous support of my Ph.D research with his patience and extensive knowledge. He spent precious time and effort to polish my writing for this thesis. Moreover, He provided me valuable suggestions on how to find a faculty job, how to become a successful researcher, and how to write a high-quality proposal. These would inspire me for seeking for excellent research work in the future.

I also would like to thank my dissertation committee: Prof. Jitendra Tugnait, Prof. Mark Nelms, and Prof. Bo Liu, for their insightful comments and suggestions for my research works. I am also indebted to Prof. Tao Shu for serving as the university reader, who reviewed my work. In addition, I thank Mr. Wendong Zhu, who provided me the opportunity for internship in the summer of 2017 and provided me with numerous suggestions about deep learning and big data.

I appreciate my friends at Auburn University: Dr. Yi Xu, Dr. Zhifeng He, Dr. Zhefeng Jiang, Dr. Yu Wang, Dr. Hui Zhou, Mingjie Feng, Kefan Xiao, Yu Wang, Ningkai Tang, Lingjun, Gao, Chao Yang, Xiangyu Wang, Runze Huang, Zhitao Yu, Ticao Zhang, Wei Sun, Bo Wu, Jian Zhang, Zhitao Gong, and Keqiang He for the discussions, and assistance during these years. Moreover, I want to thank Dr. Yihan Li for her hospitality and help in my life at Auburn.

Last but not the least, I would like to thank my dear wife, Yang Zhao, my beautiful daughter, Ashley Wang, my adorable son, Jonathan Wang, my parents, my parents in-law, and my brother. They support me spiritually during writing this dissertation.

This work is supported in part by the US NSF under Grant CNS-1702957, and through the Wireless Engineering Research and Education Center (WEREC) at Auburn University.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the fast advances in mobile devices and communication technologies, various machines and devices are capable of interacting with each other within a network. This new generation of information network is the Internet of Things (IoT), which is one of the most important areas of networking and attracts much attention from academia and industry [1, 2]. In the IoT, data generated by sensors or devices is shared with others or stored in the cloud. With the powerful computation capability of cloud servers, the data can be analyzed or processed much more efficiently than ever before.

Many emerging applications benefit from the development of the IoT. In general, IoT based applications consist of three layers: the sensing layer, the gateway layer, and the cloud layer. As shown in Fig. 1.1, the left block represents the *sensing layer*. Sensed data is usually captured by various sensors, such as accelerometer and gyroscope, in this layer. Recently, researchers also utilize RF signals to capture events in the IoT environment (i.e., *RF sensing*). RF signals are transmitted, reflected, blocked, and scattered by objects like walls, furniture, vehicles, or human body. Thus, it is possible to extract useful information, such as position, movement direction, speed, and vital signs of a human subject, from received RF signals. Unlike traditional hardware sensors, RF sensing provides users with low-cost and unobtrusive services. Furthermore, due to the broadcast nature of RF signals, RF sensing can be used not only to monitor multiple subjects, but also to capture changes in the environment over a large area.

The *gateway layer* in Fig. 1.1 (the middle block) is to transfer sensed signals to the *cloud layer* (the right block). Usually captured signals are analyzed in the cloud layer with various signal

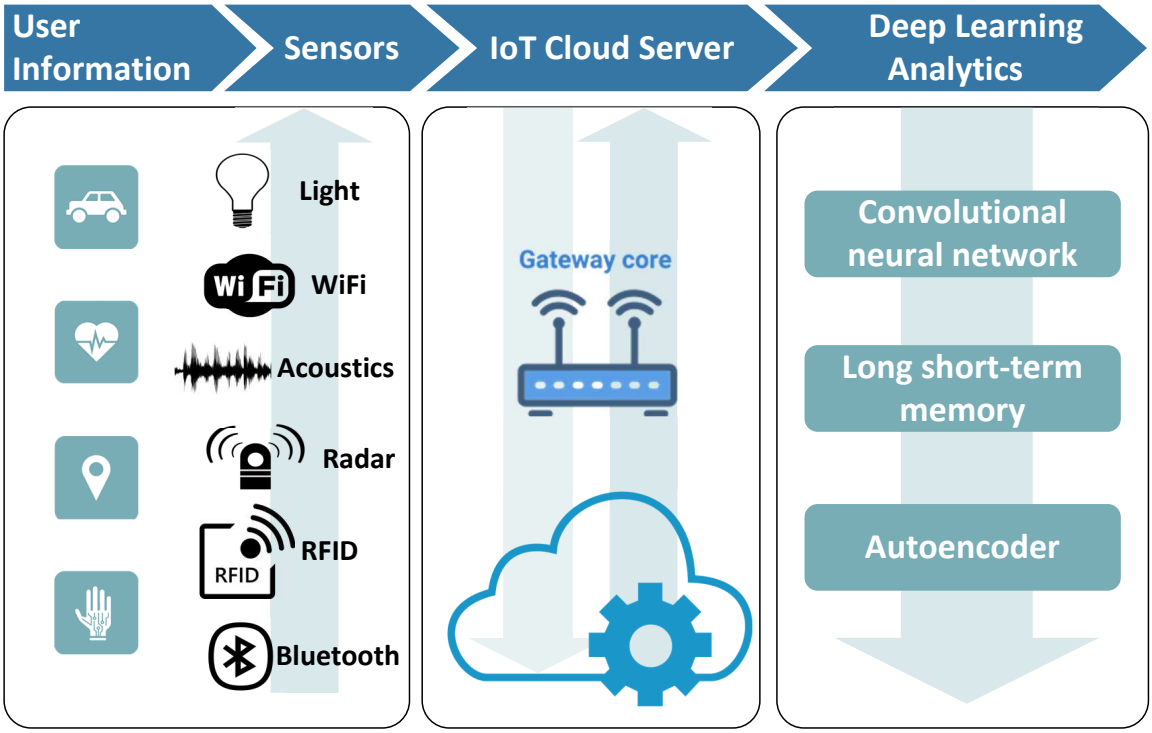


Figure 1.1: The layered architecture for RF sensing in the IoT.

processing techniques or machine learning algorithms. Recently, there has been considerable interest in applying deep learning techniques such as deep autoencoder, convolutional neural network (CNN), and long short-term memory (LSTM) to RF sensing [3, 4]. Traditional machine learning algorithms, such as support vector machine (SVM) and K-nearest neighbor (KNN) are effective for small dataset and easy classification tasks. However, they cannot scale well with the increase number of samples. Moreover, SVM and KNN need careful data preprocessing and parameters selection to avoid over-fitting and under-fitting. For example, PCA is always used for feature extraction for traditional machine learning. However, deep learning can handle the large dataset and complex classification tasks, which can obtain higher classification accuracy. Moreover, deep learning models can predict well, although being highly over-parametrized. Finally, deep learning algorithms also have great potential to process high dimensional data that could not be handled by shallow machine learning algorithms.

1.1 General Deep Learning Framework for RF Sensing

1.1.1 The General Framework

In this section, we present a general framework to leverage deep learning techniques for RF sensing applications. As shown in Fig. 1.2, various types of RF signals can be utilized as inputs to deep learning algorithms, such as WiFi, RFID, UWB, and Acoustics. For RF signals, preprocessing is an essential step before employing deep learning algorithms in the applications, which is only a data preparation step. In other words, compared with traditional shallow machine learning techniques such as SVM and KNN, feature extraction is not necessary in our framework, because deep learning algorithms have an excellent capability to represent data and then extract features from the data. In fact, pre-processing step needs to firstly obtain calibrated data for RF signals, which should remove randomness errors from other factors such as the packet boundary detection (PBD) error, the sampling frequency offset (SFO) and central frequency offset (CFO). For example, calibrated phase or phase difference between two antennas for RF signals should be implemented in preprocessing step. Then, for different deep network architectures including CNN, LSTM, and autoencoder, the inputs for them are different in preprocessing step. For example, when CNN is used, images can be constructed from the calibrated phases or amplitudes of signals in the data preprocessing step [4]. When LSTM is employed, signals can be divided into small time series in this step before it is passed on to the LSTM. When autoencoder is exploited, signals can be directly leveraged for the proposed deep learning framework.

The proposed framework consists of two stages: an offline training stage and an online prediction stage. In the offline stage, training data is used to train the deep leaning model. For different types of applications, the deep network models exhibit different potentials. For example, CNN achieves outstanding performance in image classification and pattern recognition, since it emulates the natural visual perception mechanism. On the other hand, LSTM is effective at processing variable-length inputs sequences, which makes it highly suited for time related applications. In the online stage, the test data is fed into the well trained deep network to provide prediction results.

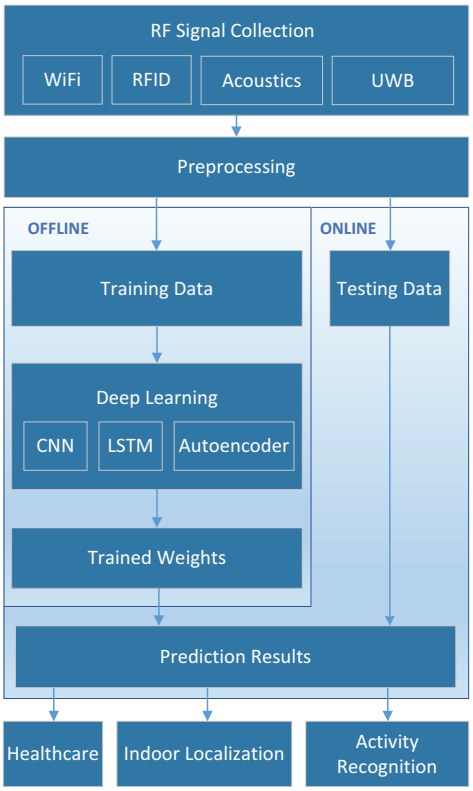


Figure 1.2: A general deep learning framework for RF sensing.

In this stage, strategies such as Bayesian methods have been used to optimize the output of the deep network (e.g., for indoor localization) [3]. Sometimes, the output of the deep network can be directly used as prediction results, such as in some recognition or detection applications. On the other hand, when the surrounding environment changes, the proposed framework can use transfer learning for updating training weights with small measurement dataset. In other words, we can fix weights for low layers and only train weights for the top layer with a classifier, which reduces greatly training time and data collection time.

1.1.2 RF Sensing Techniques

Various wireless signals have been used for RF sensing, such as WiFi, RFID, UWB, and acoustics signals.

WiFi

WiFi is a technology that uses radio waves to provide network connectivity for devices based on the 802.11 standards [5]. Over the years, a series of WiFi standards have been created by the IEEE LAN/MAN Standards Committee (IEEE 802). Initially, the maximum data rate is 2 Mbps with a range of 20 m, which has now been increased to 1.73 Gbps with a range of 35 m, as specified in the IEEE 802.11ac standard [5]. WiFi usually uses the 2.4 GHz and 5.8 GHz bands (while IEEE 802.11ad uses the 60 GHz band). Compared with the 2.4 GHz band, the 5 GHz band provides higher data rates and lower signal interferences, even though the longer wavelength of the 2.4 GHz is beneficial to long-range transmissions.

WiFi has become the dominant data access technology for mobile users. WiFi access is ubiquitous in many indoor and outdoor environments, which makes WiFi an ideal candidate for RF sensing to capture changes in the environment. Compared with traditional sensors, WiFi is capable of monitoring a large and crowded area, but WiFi signals are susceptible to interferences.

RFID

RFID is a technique to automatically identify and track tags attached to objects using electromagnetic fields. The technology has been widely used in almost all industrial sectors, such as storage, retail stores, factories, and supply chain management [6]. Similar to WiFi, RFID signals are also affected by the surroundings. Features of RFID signals, such as RSSI and phase obtained from a reader, can also be utilized to monitor changes of environment.

There are two types of RFID systems, i.e., active and passive RFID systems. An active RFID system depends on the internal power supply to reflect a response to the reader. Although longer ranges can be achieved, active RFID systems usually have a higher cost and larger form factor. Passive RFID tags draw much attention because of its smaller size, lower cost, and no need for power sources. Passive tags also have great potential in privacy protection because they could only be read from a specific direction and at a small distance. However, RFID is also limited by its

extremely simple design. For instance, when a reader attempts to read multiple tags close to each other, there will be collision among the response signals, which causes data loss.

UWB

UWB is a carrierless communication technology, which achieves high date rates by utilizing ultra-short pulses with a duration less than 1 nanosecond [7]. To achieve a high data rate, the ultra-short pulses are transmitted over a wide bandwidth, which is usually larger than 500 MHz. Due to the ultra-short pulses, the power consumption of UWB is much lower than traditional communication systems. Ultra-short pulses also mitigate the multipath effect and enable high precision Time of Flight (TOF) estimation, which is beneficial to many RF sensing applications. UWB signals can penetrate materials, and many through-wall imaging systems are proposed to exploit this feature. Furthermore, because of its unique, wide spectrum, UWB signals are robust to interference from other wireless sources. However, comparing to WiFi and RFID, UWB hardware is usually more expensive.

Acoustic Signals

Acoustic signals attract researchers' attention due to the extensive use of microphones and speakers on mobile devices (i.e., smartphones). Considering the lower propagation speed and narrow bandwidth of an acoustics signal, high speed resolution can be provided by acoustics signals, which means that it is much easier for acoustic signals to capture the small movements of an object. Acoustic signals have been exploited for activity recognition and speed detection based on machine learning and Doppler shift. However, comparing with other signals such as WiFi and RFID, acoustics signals can be easily affected by other sound sources in adjacent frequencies. Acoustics signals transmitted from smartphones also do not have a strong penetration ability. Thus applications with smartphone acoustic signals can only be deployed within small distances (e.g., in a single room). If the acoustic signal is in the audible range, the sound would be annoying to users.

1.1.3 Deep Learning Techniques

Deep learning is a branch of Machine Learning that achieves multiple levels of representation of data using a general purpose learning procedure. Recently, there has been extraordinary interests on applying deep learning to wireless systems, largely motivated by the huge success of deep learning in a variety of areas, such as natural language processing, pattern recognition, image classification, and gaming. We discuss the features of three widely used deep learning models, including autoencoder, CNN, and LSTM.

Autoencoder Neural Network

An autoencoder neural network is an unsupervised learning algorithm. Its aim is to generate the output that is an approximation of the input [8]. The architecture of autoencoder is shown in Fig. 1.3 (top). Generally, an autoencoder is composed with three parts: an input layer, one or more hidden layers, and an output layer. To reconstruct its own input, the output layer has an identical number of nodes as the input layer. On the other hand, the number of nodes in the hidden layers is always smaller than the number of nodes in the input layer, so that a compressed representation can be extracted from the input data.

There are three stages in the training process, including pretraining, unrolling, and fine-tuning. In the pretraining stage, each neighboring set of two layers is modeled as a Restricted Boltzmann Machines (RBM) to approximate a good solution. Then the deep network is unrolled to obtain the reconstructed input with forward propagation. Next, the backpropagation technique is used to fine-tune the results. Like PCA, the purpose of the autoencoder is to find low-dimensional representations of the input data. Naturally, the autoencoder neural network is widely used in data compression and signal denoising. With the proposed deep learning framework in Fig. 1.2, we can also use deep autoencoder for activity recognition and health sensing.

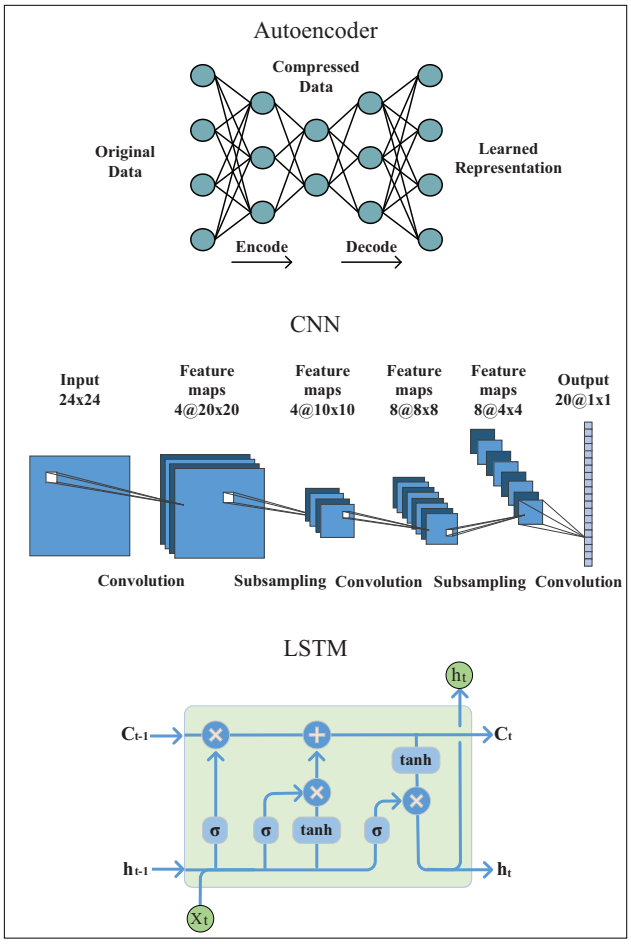


Figure 1.3: Three popular deep learning networks: CNN (top), Autoencoder (middle), LSTM (bottom).

Convolutional Neural Network

CNN is a widely used deep learning technique, which is inspired by emulating the natural visual perception mechanism of living creatures, and consequently, CNN has achieved a great success in computer vision. In 1998, LeCun proposed LeNet-5 [9], which is one of the first architecture of CNN. As shown in Fig. 1.3, the convolution and subsampling operations of LeNet-5 are first applied to the input data in the computation units called convolutional layer and subsampling layer, respectively. After two groups of such computation, the output of the higher layer is processed by a fully connected traditional neural network, where the final classification results are improved.

In 2015, a residual learning framework, called ResNet, was proposed by Microsoft Research [10]. A 152-layer residual network achieves an error rate of 3.57% on the ImageNet test set, and won the 1st place in the ILSVRC 2015 classification competition. To solve the vanishing gradient problem caused by greatly increased depth of the network, the residual module creates a shortcut path between the input and output, which implies an identity mapping.

The great performance of CNNs also attracts RF sensing researchers' attention. For example, ResLoc [4], an indoor localization system with commodity 5 GHz WiFi, uses bimodal CSI tensor data to train a *deep residual sharing learning*. ResLoc has achieved superior performance of indoor localization and outperformed several existing deep learning based methods.

Long Short-Term Memory

Recurrent neural networks (RNN) are developed to process variable-length input sequences, which originated from conventional feedforward neural networks. With feedback loops in the recurrent layer, long-term dependencies could be handled. However, the dependencies also make it hard to train the RNN, because the gradient of the loss function tends to either diminish or explode, which makes gradient-based optimization methods ineffective.

The LSTM model, shown in Fig. 1.3, is proposed to address this problem [11]. Unlike the traditional RNN where the input at each time-step affects every feedback loop, an LSTM unit utilizes three gates to control the data flow. An *input gate* decides if a new value could flow into the memory; a *forget gate* controls if a value should remain in memory; and an *output gate* determines if the value in memory could be used to compute the output of the unit. These gates ensure that gradient-based optimization methods could be used to train the LSTM. LSTM has been used widely for machine translation, speech recognition, and time-series prediction. More and more applications based on LSTM have appeared in the area of RF sensing.

In the work, we investigate the problem of RF sensing for IoT using CSI and machine learning techniques. In particular, our work mainly focuses on indoor localization and vital sign monitoring for RF sensing. For indoor localization, we consider different CSI features as fingerprints

for indoor localization using different deep learning methods. On the other hand, for vital sign monitoring, we consider breathing and heart rates monitoring for single person and multiple persons using CSI phase differences. Moreover, we study the resilient breathing beats monitoring for bad locations. In addition, we also exploit the acoustic signal for breathing rate estimation with smartphones.

The contributions of our work is summarized as follows:

1. We present a novel deep learning based indoor fingerprinting system using CSI, which is termed DeepFi. Based on three hypotheses on CSI, the DeepFi system architecture includes an off-line training phase and an on-line localization phase. In the off-line training phase, deep learning is utilized to train all the weights of a deep network as fingerprints. Moreover, a greedy learning algorithm is used to train the weights layer-by-layer to reduce complexity. In the on-line localization phase, we use a probabilistic method based on the radial basis function to obtain the estimated location. Experimental results are presented to confirm that DeepFi can effectively reduce location error compared with three existing methods in two representative indoor environments.

2. We propose PhaseFi, a fingerprinting system for indoor localization with calibrated CSI phase information. In PhaseFi, the raw phase information is first extracted from the multiple antennas and multiple subcarriers of the IEEE 802.11n network interface card (NIC) by accessing the modified device driver. Then a linear transformation is applied to extract the calibrated phase information, which we prove to have a bounded variance. For the offline stage, we design a deep network with three hidden layers to train the calibrated phase data, and employ the weights of the deep network to represent fingerprints. A greedy learning algorithm is incorporated to train the weights layer-by-layer to reduce computational complexity, where a sub-network between two consecutive layers forms a RBM. In the online stage, we use a probabilistic method based on the radial basis function (RBF) for online

location estimation. The proposed PhaseFi scheme is implemented and validated with extensive experiments in two representation indoor environments. It is shown to outperform three benchmark schemes based on CSI or RSS in both scenarios.

3. We study fingerprinting based indoor localization in commodity 5GHz WiFi networks. We first theoretically and experimentally validate three hypotheses on the CSI data of 5GHz OFDM channels. We then propose BiLoc, bi-modality deep learning for indoor localization using commodity WiFi devices. We develop a deep learning algorithm to exploit bi-modal data, i.e., estimated angle of arrival (AOA) and average amplitude (which are calibrated CSI data with several proposed techniques), in both the offline and online stages of indoor fingerprinting. The proposed BiLoc system is implemented with commodity WiFi devices. Its superior performance is validated with extensive experiments under three typical indoor environments and through comparison with three benchmark schemes.

4. We propose CiFi, deep convolutional neural networks (DCNN) for indoor localization with commodity 5GHz WiFi. First, by leveraging a modified device driver, we can extract phase data of CSI, which is used to estimate AOA. We then create estimated AOA images as input to the DCNN, to train the weights in the offline phase. The location of mobile device is predicted based on the trained DCNN and new CSI AOA image. We implement the proposed CiFi system with commodity Wi-Fi devices in the 5GHz band and verify its performance with extensive experiments in two representative indoor environments.

5. We propose ResLoc, a deep residual sharing learning based system for indoor localization with CSI tensor data. First, we introduce CSI data in wireless systems and discuss how to build CSI tensor data for indoor localization. Then, we design the ResLoc system, which employs two channels CSI tensor data to train the deep network by using the proposed deep residual sharing learning in the offline phase. For online test phase, we use newly received

CSI tensor data to estimate the location of the mobile device based on the enhanced probabilistic method. Finally, the experimental results show the proposed ResLoc system can obtain the decimeter level localization accuracy.

6. We design PhaseBeat system to leverage CSI phase difference data to monitor breathing and heart beats with commodity WiFi device. First, we deeply analyze the measured phase errors and prove the phase difference with the same frequency with breathing rate. Then, we implement data preprocessing for the collected CSI phase difference data to obtain the denoised breathing signal and the restructured heart signal. Moreover, we leverage the peak detection method for breathing rate estimation and FFT based method for heart signal estimation. Our experimental results demonstrate that our PhaseBeat system can obtain better performances in different environmental parameters.

7. We propose TensorBeat, a system to employ CSI phase difference data to intelligently estimate breathing rates for multiple persons with commodity WiFi devices. The main idea is to leverage the tensor decomposition technique to handle the CSI phase difference data. The proposed TensorBeat scheme first obtains CSI phase difference data between pairs of antennas at the WiFi receiver to create CSI tensor data. Then Canonical Polyadic (CP) decomposition is applied to obtain the desired breathing signals. A stable signal matching algorithm is developed to find the decomposed signal pairs, and a peak detection method is applied to estimate the breathing rates for multiple persons. Our experimental study shows that TensorBeat can achieve high accuracy under different environments for multi-person breathing rate monitoring.

8. We present ResBeat, a commodity 5 GHz WiFi based system to exploit bimodal CSI including amplitude and phase difference, for realtime, long-term, and contact-free breathing monitoring. Specifically, we first present an analysis of breathing signal anomaly based on bimodal CSI data. Then, we implement the data preprocessing, adaptive signal selection,

and breathing signal monitoring modules of ResBeat, and employ peak detection to estimate breathing rates. We conduct extensive experiments on breathing rate monitoring under three different environments, where superior performance over two alternative methods is validated.

9. We present a SonarBeat system to leverage a phase based active sonar to monitor breathing rates with smartphones. We design and implement the SonarBeat system, with components including signal generation, data extraction, received signal preprocessing, and breathing rate estimation, with Android smartphones. Our experimental results validate the superior performance of SonarBeat in different indoor environment settings.

Chapter 2

DeepFi: CSI Amplitude based Fingerprinting for Indoor Localization Using Deep Learning

2.1 Introduction

With the proliferation of mobile devices, indoor localization has become an increasingly important problem. Unlike outdoor localization, such as the Global Positioning System (GPS), that has line-of-sight (LOS) transmission paths, indoor localization faces a challenging radio propagation environment, including multipath effect, shadowing, fading and delay distortion [12, 13, 14, 15]. In addition to the high accuracy requirement, an indoor positioning system should also have a low complexity and short online process time for mobile devices. To this end, fingerprinting-based indoor localization becomes an effective method to satisfy these requirements, where an enormous amount of measurements are essential to build a database to facilitate real-time position estimation.

Fingerprinting based localization usually consists of two basic phases: (i) the off-line phase, which is also called the training phase, and (ii) the on-line phase, which is also called the test phase [16]. The training phase is for database construction, when survey data related to the position marks is collected and pre-processed. In the off-line training stage, machine learning methods can be used to train fingerprints instead of storing all the received signal strength (RSS) data. Such machine learning methods not only reduce the computational complexity, but also obtain the core features in the RSS for better localization performance. KNN, neural networks, and support vector machine, as popular machine learning methods, have been applied for fingerprinting based indoor localization. KNN uses the weighted average of K nearest locations to determine an unknown location with the inverse of the Euclidean distance between the observed RSS measurement and

its K nearest training samples as weights [12]. A limitation of KNN is that it needs to store all the RSS training values. Neural networks utilizes the back-propagation algorithm to train weights, but it considers one hidden layer to avoid error propagation in the training phase and needs labeled data as a supervised learning [17]. Support vector machine uses kernel functions to solve the randomness and incompleteness of the RSS values, but has high computing complexity [18]. In the on-line phase, a mobile device records real time data and tests it using the database. The test output is then used to estimate the position of the mobile device, by searching each training point to find the most closely matched one as the target location. Besides such nearest estimation method, an alternative matching algorithm is to identify several close points each with a maximum likelihood probability, and to calculate the estimated position as the weighted average of the candidate positions.

Many existing indoor localization systems use RSS as fingerprints due to its simplicity and low hardware requirements. For example, the Horus system uses a probabilistic method for location estimation with RSS data [19]. Such RSS based methods have two disadvantages. First, RSS values usually have a high variability over time for a fixed location, due to the multipath effects in indoor environments. Such high variability can introduce large location error even for a stationary device. Second, RSS values are coarse information, which does not exploit the many subcarriers in an orthogonal frequency-division multiplexing (OFDM) for richer multipath information. It is now possible to obtain CSI from some WiFi network interface cards (NIC), which can be used as fingerprints to improve the performance of indoor localization [20, 21]. For instance, the FIFS scheme uses the weighted average CSI values over multiple antennas [22]. In addition, the PinLoc system also exploits CSI information, while considering 1×1 m^2 spots for training data [23].

In this chapter, we propose a deep learning based fingerprinting scheme to mitigate the several limitations of existing machine learning based methods. The deep learning based scheme can fully explore the feature of wireless channel data and obtain the optimal weights as fingerprints. It also incorporates a greedy learning algorithm to reduce computational complexity, which has been successfully applied in image processing and voice recognition [24]. The proposed scheme is based on CSI to obtain more fine-grained information about the wireless channel than RSS based

schemes. The proposed scheme is also different from the existing CSI based schemes, in that it incorporates 90 magnitudes of CSI values collected from the three antennas of the Intel's IWL 5300 NIC to train the weights of a deep network with deep learning.

In particular, we present DeepFi, a deep learning based indoor fingerprinting scheme using CSI. We first introduce the background of CSI and present three hypotheses on CSI. We then present the DeepFi system architecture, which includes an off-line training phase and an on-line localization phase. In the training phase, CSI information for all the subcarriers from three antennas are collected from accessing the device driver and are analyzed with a deep network with four hidden layers. We propose to use the weights in the deep network to represent fingerprints, and to incorporate a greedy learning algorithm using a stack of RBMs to train the deep network in a layer-by-layer manner to reduce the training complexity. The greedy algorithm first estimates the parameters of the first layer RBM to model the input data. Then the parameters of the first layer are frozen, and we obtain the samples from the conditional probability to train the second layer RBM and so forth. Finally, we can obtain the parameters of the fourth layer RBM with the above greedy learning algorithm. Moreover, for each layer RBM model, we use the contrastive divergence with 1 step iteration (CD-1) method to update weights, which has lower time complexity than other schemes such as Markov chain Monte Carlo (MCMC) [25]. In the on-line localization phase, a probabilistic data fusion method based on radial basis function is developed for online location estimation using multiply packets. To reduce the computational complexity for online localization, packets are divided into several batches, each of which contains the same number of packets. Because packets are processed in parallel in batches, we can significantly shorten the processing time when dealing with a large amount of packets.

The proposed DeepFi scheme is validated with extensive experiments in two representative indoor environments, i.e., a living room environment and a computer laboratory environment. DeepFi is shown to outperform several existing RSSI and CSI based schemes in both experiments. We also examine the effect of different DeepFi parameters on localization accuracy and execution time, such as using different number of antennas, using different number of test packets,

and different number of packets per batch. Finally, we investigate the effect of different propagation environments on the DeepFi performance, such as replaced obstacles, human mobility, and the training grid size in our experimental study. Our experimental results confirm that DeepFi can perform well in these scenarios.

The remainder of this chapter is organized as follows. We review related work in Section 2.2. The background and hypotheses are presented in Section 2.3. The DeepFi system is presented in Section 2.4 and evaluated in Section 2.5. Section 2.6 concludes this chapter.

2.2 Related Work

There has been a considerable literature on indoor localization [26]. Early indoor location service systems include (i) Active Badge equipped mobiles with infrared transmitters and buildings with several infrared receivers [27], (ii) the Bat system that has a matrix of RF-ultrasound receivers deployed on the ceiling [28], and (iii) the Cricket system that equipped buildings with combined RF/ultrasound beacons [29]. All of these schemes achieve high localization accuracy due to the dedicated infrastructure. Recently, considerable efforts are made on indoor localization systems based on new hardware, with low cost, and high accuracy. These recent work mainly fall into three categories: Fingerprinting-based, Ranging-based and AOA-based, which are discussed in this chapter.

2.2.1 Fingerprinting-based Localization

Fingerprinting-based Localization requires a training phase to survey the floor plan and a test phase to search for the most matched fingerprint for location estimation [30, 31]. Recently, different forms of fingerprint have been explored, including WiFi [19], FM radio [32], RFID [33], acoustic [34], GSM [35], light [36] and magnetism [37], where WiFi-based fingerprinting is the dominant method because WiFi signal is ubiquitous in the indoor environments. The first work based on WiFi is RADAR [16], which builds fingerprints of RSS using one or more access points. It is a deterministic method using KNN for position estimation. Horus [19] is an RSS based

scheme utilizing probabilistic techniques to improve localization accuracy, where the RSS from an access point is modeled as a random variable over time and space. In addition to RSS, channel impulse response of WiFi is considered as a location-related and stability signature, with which the fine-grained characteristics of wireless channels can be exploited to achieve higher localization accuracy. For example, FIFS [22] and PinLoc [23] use CSI obtained through the off-the-shelf IWL 5300 NIC to build reliable fingerprints. Although these techniques achieve high localization precision, they need a large amount of calibration to build the fingerprint database via war-driving, as well as manually matching every test location with the corresponding fingerprint. Currently, time-reversal based methods are proposed for obtaining centimeter-accuracy indoor localization using a frequency hopping approach [38] and a multi-antenna approach [39]. In fact, these methods require a large number of fingerprints collected in training phase and are implemented in a small area.

Crowdsourcing is proposed to reduce the burden of war-driving by sharing the load to multiple users. It consists of two main steps: (i) estimations of user trajectories, and (ii) construction of a database mapping fingerprints to user locations [40]. Recently, Zee [41] uses the inertial sensors and particle filtering to estimate a user's walking trajectory, and to collect fingerprints with WiFi data as crowd-sourced measurements for calibration. Similarly, LiFS [42] also uses user trajectories to obtain fingerprint values and then builds the mapping between the fingerprints and the floor plan. Crowdsourcing can also be used to detect indoor contexts. For example, CrowdInside [43] and Walkie-Markie [44] can detect the shape of the floor plan and build the pathway to obtain the crowdsourced user's fingerprints. Moreover, Jigsaw [45] and Travi-Navi [46] combine vision and mobility obtained from a smartphone to build user trajectories. Although crowdsourcing does not require a large amount of calibration effort, it obtains coarse-grained fingerprints, which leads to low localization accuracy in general.

2.2.2 Ranging-based Localization

Ranging-based localization computes distances to at least three access points and leverages geometrical models for location estimation. These schemes are mainly classified into two categories: power-based and time-based. For power-based approaches, the prevalent log-distance path loss (LDPL) model is used to estimate distances based on RSS, where some measurements are utilized to train the parameters of the LDPL model [47]. For example, EZ [48] is a configuration-free localization scheme, where a genetic algorithm is used for solving the RSS-distance equations. The LDPL model and truncated singular value decomposition (SVD) are used to build a RSS-distance map for localization, which is adaptive to indoor environmental dynamics [47]. CSI-based ranging is proposed to overcome the instability of RSS in indoor environments. For instance, FILA exploits CSI from the PHY Layer to mitigate the multipath effect in the time-domain, and then trains the parameters of LDPL model to obtain the relationship between the effective CSI and distance [49].

Acoustic-based ranging approaches are developed for improving indoor localization precision. H. Liu $etal$ propose a peer assisted localization technique based on smartphones to compute accurate distance estimation among peer smartphones with acoustic ranging [50]. Centour [51] leverages a Bayesian framework combining WiFi measurements and acoustic ranging, where two new acoustic techniques are proposed for ranging under NLOS and locating a speaker-only device based on estimating distance differences. Guoguo [52] is a smartphone-based indoor localization system, which estimates a fine-grained time-of-arrival (TOA) using beacon signals and performs NLOS identification and mitigation.

2.2.3 AOA-based Localization

Indoor localization based on AOA utilizes multiple antennas to estimate the incoming angles and then uses geometric relationships to obtain the user location. This technique is not only with zero start-up cost, but also achieves higher accuracy than other techniques such as RF fingerprinting or ranging-based systems. The challenge of this technique is how to improve the resolution of the antenna array. The recently proposed CUPID system [53] adopts the off-the-shelf Atheros chipset

with three antennas, and can obtain CSI to estimate AOA, achieving a mean error about 20 degrees with the MUSIC algorithm. The relatively large error is mainly due to the low resolution of the antenna array. For high localization accuracy, the Array-Track system [54] is implemented with two WARP systems, which are FPGA-based software defined radios. It incorporates a rectangular array of 16 antennas to compute the AOA, and then uses spatial smoothing to suppress the effect of multipath on AOA. However, Array-Track requires a large number of antennas, which is generally not available for commodity mobile devices.

On the other hand, some systems, such as LTEye [55], Ubicarse [56], Wi-Vi [57], and PinIt [58], use Synthetic Aperture Radar (SAR) to mimic an antenna array to improve the resolution of angles. the main idea of SAR is to use a moving antenna to obtain signal snapshots as it moves along a trajectory, and then to utilize these snapshots to mimic a large antenna array along the trajectory. However, it requires accurate control of the speed and trajectory by using a moving antenna placed on an iRobot Create robot.

2.3 Background and Hypotheses

2.3.1 Channel State Information

Thanks to the NICs, such as Intel's IWL 5300, it is now easier to conduct channel state measurements than in the past when one has to detect hardware records for physical layer (PHY) information. Now CSI can be retrieved from a laptop by accessing the device drive. CSI records the channel variation experienced during propagation. Transmitted from a source, a wireless signal may experience abundant impairments caused by, e.g., the multipath effect, fading, shadowing, and delay distortion. Without CSI, it is hard to reveal the channel characteristics with the signal power only.

Let $\vec{X}$ and $\vec{Y}$ denote the transmitted and received signal vectors. We have

$$\vec{Y} = \text{CSI} \cdot \vec{X} + \vec{N}, \tag{2.1}$$

where vector $\vec{N}$ is the additive white Gaussian noise and CSI represents the channel's frequency response, which can be estimated from $\vec{X}$ and $\vec{Y}$.

The WiFi channel at the 2.4 GHz band can be considered as a narrowband flat fading channel for OFDM system. The Intel WiFi Link 5300 NIC implements an OFDM system with 56 subcarriers, 30 out of which can be read for CSI information via the device driver. The channel frequency response CSI_i of subcarrier i is a complex value, which is defined by

$$\text{CSI}_i = |\text{CSI}_i| \exp\{j \angle \text{CSI}_i\}. \tag{2.2}$$

where $|\text{CSI}_i|$ and $\angle \text{CSI}_i$ are the amplitude response and the phase response of subcarrier i, respectively. In this chapter, the proposed DeepFi framework is based on these 30 subcarriers (or, CSI values), which can reveal much richer channel properties than RSSI.

2.3.2 Hypotheses

We next present three hypotheses about the CSI data, which are validated with the statistical results through our measurement study.

Hypothesis 1

CSI amplitude values exhibit great stability for continuously received packets at a fixed location, compared with RSS values.

CSI amplitude values reflect channel properties in the frequency domain and exhibit great stability over time for a given location. Fig. 2.1 plots the cumulative distribution function (CDF) of the standard deviations of normalized CSI and RSS amplitudes for 150 sampled locations. At each location, CSI and RSS values are measured from 50 received packets with the three antennas of Intel WiFi Link 5300 NIC. It can be seen that for CSI amplitude values, 90% of the standard deviations are below 10% of the average value; for RSS values, however, 60% of the standard deviations are below 10% of the average value. Therefore CSI is much more stable than RSS. Our

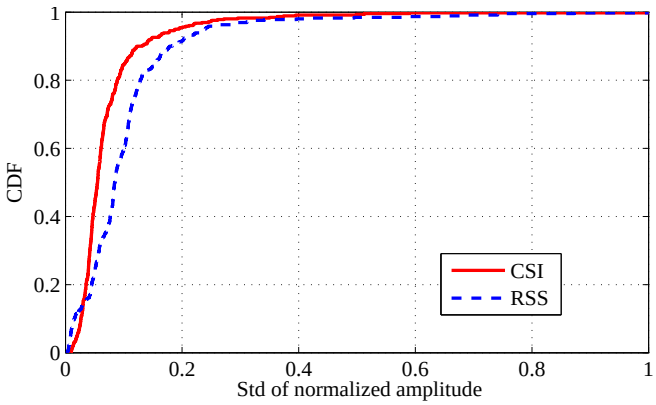


Figure 2.1: CDF of the standard deviations of CSI and RSS amplitudes for 150 sampled locations.

measurements last a long period including both office hours and quiet hours. No obvious difference in the stability of CSI for the same location is observed at different times. On the contrary, RSS values exhibit large variations even at the same position. Therefore, CSI amplitude values are leveraged as the feature of deep learning in the DeepFi system.

Hypothesis 2

Because of the multipath effect and channel fading indoor environment, the number of clusters of CSI values over subcarriers varies at different locations.

CSI amplitude values reflect channel frequency responses with abundant multipath components and channel fading. Our study of channel frequency responses shows that there are several dominant clusters for CSI amplitude values, where each cluster consists of a subset of subcarriers with similar CSI amplitudes values. To find the feature of clusters of CSI amplitudes values, we draw the CDF and the two-dimensional (2D) contour of the number of clusters for CSI amplitude values for 50 different locations in the living room environment in Fig. 2.2 and Fig. 2.3, respectively. For every location, CSI values are measured from 50 received packets with the three antennas of Intel WiFi Link 5300 NIC. Based on Fig. 2.2 and Fig. 2.3, we can see that the number of clusters of CSI amplitude values varies at 50 different locations. Moreover, at most of locations, CSI amplitude values exhibit two or three clusters. Some locations have one cluster because of

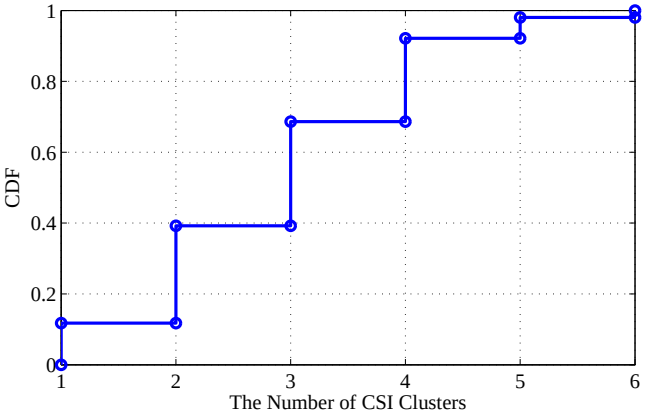


Figure 2.2: CDF of the number of clusters of CSI amplitude values at 50 different locations.

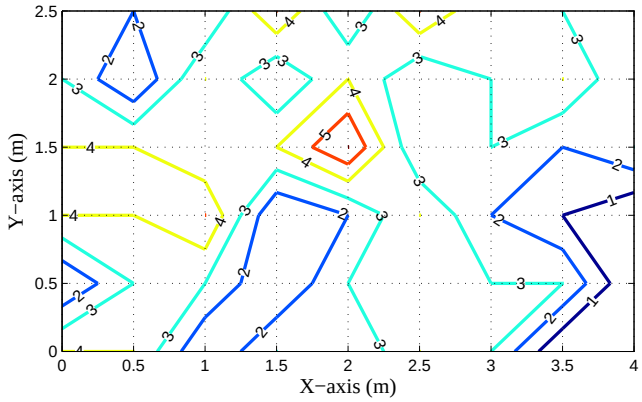


Figure 2.3: 2D contour of the number of clusters of CSI amplitude values at 50 different locations.

less reflection and diffusion. Some other locations with few five or six clusters may suffer from the severe multipath effect.

To detect all possible numbers of clusters, we measure CSI amplitude values from received packets for a long period at each location, which can be used for training weights in deep network. In addition, more packet transmissions will be helpful to reveal the comprehensive properties at each location. In our experiments, we consider 500 and 1000 packets for training in the living room environment and the computer laboratory environment, respectively, more than the 60 packets used in FIFS.

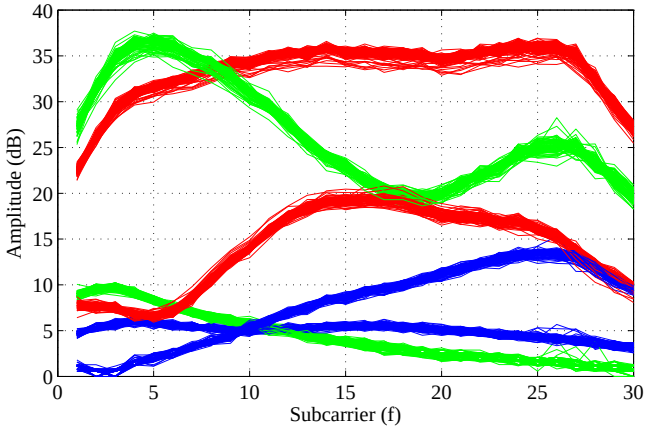


Figure 2.4: Amplitudes of channel frequency response measured at the three antennas of the Intel WiFi Link 5300 NIC (each is plotted in a different color) for 50 received packets.

Hypothesis 3

The three antennas of the Intel WiFi Link 5300 NIC have different CSI features, which can be exploited to improve the diversity of training and test samples.

Intel WiFi Link 5300 is equipped with three antennas. We find that the channel frequency responses of the three antennas are highly different, even for the same packet reception. In Fig. 2.4, amplitudes of channel frequency response from the three antennas exhibit different properties. In FIFS, CSI amplitude values from the three antennas are simply accumulated to produce an average value. In contrast, DeepFi aims to utilize their variability to enhance the training and test process in deep learning. The 30 subcarriers can be treated as 30 nodes and used as input data of visible variability for deep learning. With the three antennas, there are 90 nodes that can be used as input data for deep learning. The greatly increased number of nodes for input data can improve the diversity of training and test samples, leading to better performance of localization if reasonable parameters are chosen.

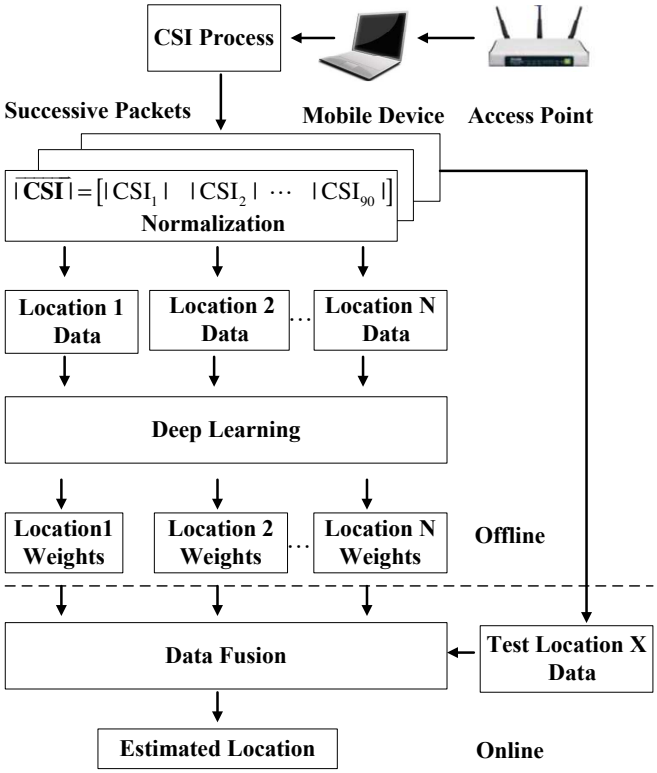


Figure 2.5: The DeepFi architecture.

2.4 The DeepFi System

2.4.1 System Architecture

Fig. 2.5 shows the system architecture of DeepFi, which only requires one access point and one mobile device equipped with an Intel WiFi link 5300 NIC. At the mobile device, raw CSI values can be read from the modified chipset firmware for received packets. The Intel WiFi link 5300 NIC has three antennas, each of which can collect CSI data from 30 different subcarriers. We can thus obtain 90 raw CSI measurements for each packet reception. Unlike FIFS that averages over multiple antennas to reduce the received noise, our system uses all CSI values from the three antennas for indoor fingerprint to exploit diversity of the channel. Since it is hard to use the phases of CSI for localization, we only consider the amplitude responses for fingerprinting In this chapter. On the other hand, since the input values should be limited in the range (0, 1) for effective deep learning, we normalize the amplitudes of the 90 CSI values for both the offline and online phases.

In the offline training phase, DeepFi generates feature-based fingerprints, which are highly different from traditional methods that directly store CSI values. Feature-based fingerprints utilize a large number of weights obtained by deep learning for different locations, which effectively describe the characteristics of CSI for each location and reduce noise. Meanwhile these weights can indirectly extract the feature of clusters hidden in CSI values. The feature-based fingerprints server can store the weights for different training locations. In the online localization phase, the mobile device can estimate its position with a data fusion approach.

2.4.2 Weight Training with Deep Learning

Fig. 2.6 illustrates how to train weights based on deep learning. There are three stages in the procedure, including pretraining, unrolling, and fine-tuning [8]. A deep network with four hidden layers is adopted, where every hidden layer consists of a different number of neurons. In order to reduce the dimension of CSI data, we assume that the number of neurons in a higher hidden layer is more than that in a lower hidden layer. Let K_1, K_2, K_3 and K_4 denote the number of neurons in the first, second, third, and fourth hidden layer, respectively. It follows that $K_1 > K_2 > K_3 > K_4$.

In addition, we propose a new approach to represent fingerprints, i.e., using the weights between connected layers. Define W_1, W_2, W_3 and W_4 as the weights between the normalized magnitudes of CSI values and the first hidden layer, the first and second hidden layer, the second and third hidden layer, and the third and fourth hidden layer, respectively. The key idea is that after training the weights in the deep network, we can store them as fingerprints to facilitate localization in the on-line test stage. Moreover, we define h_i as the hidden variable at layer i, for $i = 1, 2, 3, 4$, respectively, and let v denote the input data, i.e., the normalized CSI magnitudes.

We represent the deep network with four hidden layers with a probabilistic generative model, which can be written as

$$\begin{aligned} & \Pr(v, h^1, h^2, h^3, h^4) \\ = \ & \Pr(v|h^1) \Pr(h^1|h^2) \Pr(h^2|h^3) \Pr(h^3, h^4). \end{aligned} \tag{2.3}$$

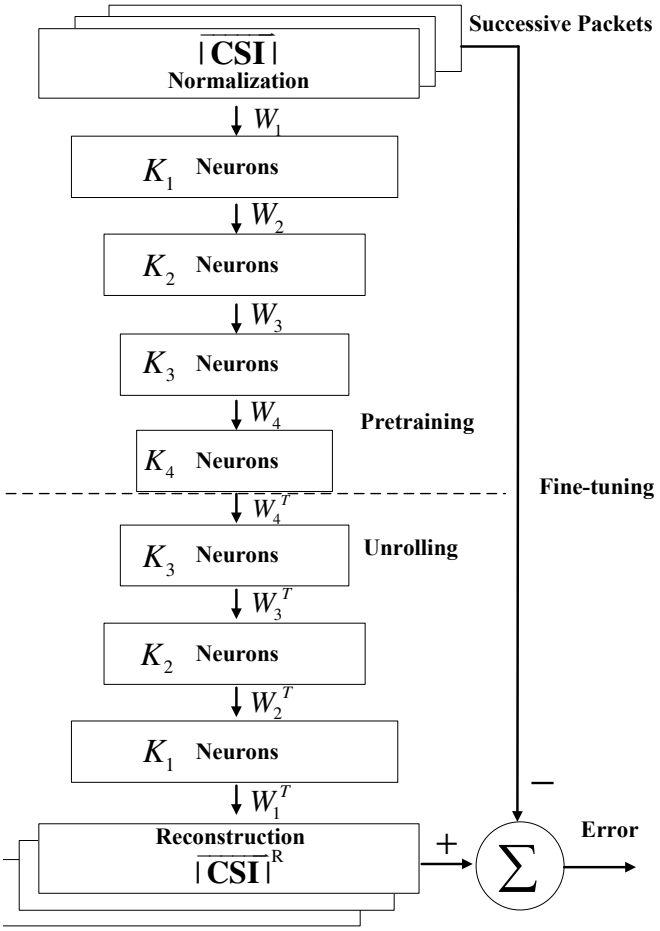


Figure 2.6: Weight training with deep learning.

Since the nodes in the deep network are mutually independent, $\Pr(v|h^1)$, $\Pr(h^1|h^2)$, and $\Pr(h^2|h^3)$ can be represented by

$$\begin{cases} \Pr(v|h^1) = \prod_{i=1}^{90} \Pr(v_i|h^1) \\ \Pr(h^1|h^2) = \prod_{i=1}^{K_1} \Pr(h_i^1|h^2) \\ \Pr(h^2|h^3) = \prod_{i=1}^{K_2} \Pr(h_i^2|h^3). \end{cases} \tag{2.4}$$

In (2.4), $\Pr(v_i|h^1)$, $\Pr(h_i^1|h^2)$, and $\Pr(h_i^2|h^3)$ are described by the sigmoid belief network in the deep network, as

$$\begin{cases} \Pr(v_i|h^1) = 1/\left(1+\exp\left(-b_i^0 - \sum_{j=1}^{K_1} W_1^{i,j} h_j^1\right)\right) \\ \Pr(h_i^1|h^2) = 1/\left(1+\exp\left(-b_i^1 - \sum_{j=1}^{K_2} W_2^{i,j} h_j^2\right)\right) \\ \Pr(h_i^2|h^3) = 1/\left(1+\exp\left(-b_i^2 - \sum_{j=1}^{K_3} W_3^{i,j} h_j^3\right)\right) \end{cases} \tag{2.5}$$

where b_i^0, b_i^1 and b_i^2 are the biases for unit i of input data v, unit i of layer 1, and unit i of layer 2, respectively.

The joint distribution $\Pr(h^3, h^4)$ can be expressed as a RBM [25] with a bipartite undirected graphical model [25], which is given by

$$\Pr(h^3, h^4) = \frac{1}{Z}\exp(-\mathbb{E}(h^3, h^4)), \tag{2.6}$$

where $Z = \sum_{h^3}\sum_{h^4}\exp(-\mathbb{E}(h^3, h^4))$ and $\mathbb{E}(h^3, h^4) = -\sum_{i=1}^{K_3} b_i^3 h_i^3 - \sum_{j=1}^{K_4} b_j^4 h_j^3 - \sum_{i=1}^{K_3}\sum_{j=1}^{K_4} W_4^{i,j} h_i^3 h_j^4$. In fact, since it is difficult to find the joint distribution $\Pr(h^3, h^4)$, we use CD-1 method [25] to approximate it, which is given by

$$\begin{cases} \Pr(h^3|h^4) = \prod_{i=1}^{K_3} \Pr(h_i^3|h^4) \\ \Pr(h^4|h^3) = \prod_{i=1}^{K_4} \Pr(h_i^4|h^3), \end{cases} \tag{2.7}$$

where $\Pr(h_i^3|h^4)$, and $\Pr(h_i^4|h^3)$ are described by the sigmoid belief network, as

$$\begin{cases} \Pr(h_i^3|h^4) = 1/\left(1+\exp\left(-b_i^3 - \sum_{j=1}^{K_4} W_4^{i,j} h_j^4\right)\right) \\ \Pr(h_i^4|h^3) = 1/\left(1+\exp\left(-b_i^4 - \sum_{j=1}^{K_3} W_4^{i,j} h_j^3\right)\right). \end{cases} \tag{2.8}$$

Finally, the marginal distribution of input data for the deep belief network is given by

$$\Pr(v) = \sum_{h^1}\sum_{h^2}\sum_{h^3}\sum_{h^4}\Pr(v, h^1, h^2, h^3, h^4). \tag{2.9}$$

Due to the complex model structure with the large number of neurons and multiple hidden layers in the deep belief network, it is difficult to obtain the weights using the given input data with the maximum likelihood method. In DeepFi, we adopt a greedy learning algorithm using a stack of RBMs to train the deep network in a layer-by-layer manner [25]. This greedy algorithm first estimates the parameters $\{b^0, b^1, W_1\}$ of the first layer RBM to model the input data. Then the parameters $\{b^0, W_1\}$ of the first layer are frozen, and we obtain the samples from the conditional probability $\Pr(h^1|v)$ to train the second layer RBM (i.e., to estimate the parameters $\{b^1, b^2, W_2\}$), and so forth. Finally, we can obtain the parameters $\{b^3, b^4, W_4\}$ of the fourth layer RBM with the above greedy learning algorithm.

For the layer i RBM model, we use CD-1 method to update weights W_i. We first get h^i based on the samples from the conditional probability $\Pr(h^i|h^{i-1})$, and then obtain $\hat{h}^{i-1}$ based on the samples from the conditional probability $\Pr(h^{i-1}|h^i)$. Finally we obtain $\hat{h}^i$ using the samples from the conditional probability $\Pr(h^i|\hat{h}^{i-1})$. Thus, we can update the parameters as follows.

$$\begin{cases} W_i = W_i + \alpha(h^{i-1}h^i - \hat{h}^{i-1}\hat{h}^i) \\ b^i = b^i + \alpha(h^i - \hat{h}^i) \\ b^{i-1} = b^{i-1} + \alpha(h^{i-1} - \hat{h}^{i-1}), \end{cases} \tag{2.10}$$

where α is the step size. After the pretraining stage, we need to unroll the deep network to obtain the reconstructed data $\hat{v}$ using the input data with forward propagation. The error between the input data v and the reconstructed data $\hat{v}$ can be used to adjust the weights at different layers with the back-propagation algorithm. This procedure is called fine-tuning. By minimizing the error, we can obtain the optimal weights to represent fingerprints, which are stored in a database for indoor localization in the on-line stage.

The pseudocode for weight learning with multiply packets is given in Algorithm 1. We first collect m packet receptions for each of the N training locations, each of which has 90 CSI values, as input data. Let $v(t)$ be the input data from packet t. The output of the algorithm consists of N groups of fingerpirnts, each of which has eight weight matrices. In fact, we need to train a deep

network for each of the N training locations. The training phase includes three steps: pretraining, unrolling and fine-tuning. For pretraining, the deep network with four hidden layers is trained with the greedy learning algorithm. The weight matrix and bias of every layer are initialized first, and are then iteratively updated with the CD-1 method for obtaining a near optimal weight, where m packets are trained and iteratively become output as input of the next hidden layer (lines 4-21).

Once weight training is completed, the input data will be unrolled to obtain the reconstructed data. First, we use the input data to compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1} to obtain the coding output h^4, which is a reduced dimension data (lines 23-26). Then, by computing $\Pr(\hat{h}^{i-1}|\hat{h}^i)$ based on the sigmoid with input $\hat{h}^i$, we can sample the reconstructed data $\hat{h}^0$, where the weights of the deep network are only transposed, thus reducing the time complexity of weight learning (lines 27-31). Once the reconstructed data $\hat{h}^0$ if obtained, the unsupervised learning method for the deep network becomes a supervised learning problem as in the fine-tuning phase. Thus, we compute the error between the input data $v = h^0$ and reconstructed data $\hat{h}^0$ to successively update the weight matrix with the standard back-propagation algorithm (lines 33-34).

2.4.3 Location Estimation based on Data Fusion

After off-line training, we need to test it with positions that are different from those used in the training stage. Because the probabilistic methods have better performance than deterministic ones, we use the probability model based on Bayes' law, which is given by

$$\Pr(L_i|v) = \frac{\Pr(L_i)\Pr(v|L_i)}{\sum_{i=1}^{N}\Pr(L_i)\Pr(v|L_i)}, \tag{2.11}$$

where L_i is reference location i, $\Pr(L_i|v)$ is the posteriori probability, $\Pr(L_i)$ is the prior probability that the mobile device is determined to be at reference location i, and N is the number of reference locations. In addition, we assume that $\Pr(L_i)$ is uniformly distributed in the set

Algorithm 1: Training for Weight Learning

Input: m packet receptions each with 90 CSI values for each of the N training locations;
Output: N groups of fingerprints each consisting of eight weight matrices;
for $j = 1 : N$ **do**
    // pretraining;
    for $i = 1 : 4$ **do**
        Initialize $W^i = 0.1 \cdot randn$, $b^i = 0$, //$randn$ is the standard Gaussian distribution function;
        for $k = 1 :$ *maxepoch* **do**
            for $t = 1 : m$ **do**
                $h^0 = v(t)$;
                Compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1};
                Sample h^i from $\Pr(h^i|h^{i-1})$;
                Compute $\Pr(h^{i-1}|h^i)$ based on the sigmoid with input h^i;
                Sample $\hat{h}^{i-1}$ from $\Pr(h^{i-1}|h^i)$;
                Compute $\Pr(h^i|\hat{h}^{i-1})$ based on the sigmoid with input $\hat{h}^{i-1}$;
                Sample $\hat{h}^i$ from $\Pr(h^i|\hat{h}^{i-1})$;
                $W_i = W_i + \alpha(h^{i-1}h^i - \hat{h}^{i-1}\hat{h}^i)$;
                $b^i = b^i + \alpha(h^i - \hat{h}^i)$;
                $b^{i-1} = b^{i-1} + \alpha(h^{i-1} - \hat{h}^{i-1})$;
            end
        end
    end
    //unrolling;
    for $i = 1 : 4$ **do**
        Compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1};
        Sample h^i from $\Pr(h^i|h^{i-1})$;
    end
    Set $\hat{h}^i = h^i$;
    for $i = 4 : 1$ **do**
        Compute $\Pr(\hat{h}^{i-1}|\hat{h}^i)$ based on the sigmoid with input $\hat{h}^i$;
        Sample $\hat{h}^{i-1}$ from $\Pr(\hat{h}^{i-1}|\hat{h}^i)$;
    end
    //fine-tuning;
    Obtain the error between input data h^0 and reconstructed data $\hat{h}^0$;
    Update the eight weights using the error with back-propagation;
end

$\{1, 2, \cdots, N\}$, and thus $\Pr(L_i) = 1/N$. It follows that

$$\Pr(L_i|v) = \frac{\Pr(v|L_i)\frac{1}{N}}{\sum_{i=1}^{N} \Pr(v|L_i)\frac{1}{N}} = \frac{\Pr(v|L_i)}{\sum_{i=1}^{N} \Pr(v|L_i)}. \tag{2.12}$$

Based on the deep network model, we define $\Pr(v|L_i)$ as the radial basis function (RBF) in the form of a Gaussian function, which is formulated as

$$\Pr(v|L_i) = \exp\left(-\frac{\|v - \hat{v}\|}{\lambda\sigma}\right), \tag{2.13}$$

where v is the input data, $\hat{v}$ is the reconstructed input data, σ is the variance of input data, λ is the coefficient of variation (CV) of input data. In fact, we use multiple packets to estimate the location of a mobile device, thus improving the indoor localization accuracy. For n packets, we need to compute the average value of RBF, which is given by

$$\Pr(v|L_i) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(-\frac{\|v_i - \hat{v}_i\|}{\lambda\sigma}\right). \tag{2.14}$$

Finally, the position of the mobile device can be estimated as a weighted average of all the reference locations, as

$$\hat{L} = \sum_{i=1}^{N} \Pr(L_i|v)L_i. \tag{2.15}$$

The pseudocode for online location estimation with multiply packets is presented in Algorithm 2. The input to the algorithm consists of n packet receptions, each of which has 90 CSI values, and N groups of fingerprints obtained in the off-line training phase, each of which has eight weight matrices for each known training locations. First, we compute the variance of the 90 CSI values from each packet. We also group the n packets into a batches, each with b packets, for accelerating the matching algorithm (lines 3-4). To obtain the posterior probability for different locations, we need to compute the RBF as likelihood function based on the reconstructed CSI values and input CSI values, where the reconstructed CSI values are obtained by recursively unrolling the deep network using the input data with forward propagation. For batch j, the reconstructed CSI values $\hat{V}_j$ are obtained by iterating the input data V_j based on the eight weight matrices (lines 10-12). Then the sum of the RBFs (i.e., the d_j's) is obtained by summing over the 90 CSI values and

Algorithm 2: Online Location Estimation

1 **Input:** n packet receptions each with 90 CSI values, N groups of fingerprints each with eight weight matrices and the known training location;
2 **Output:** estimated location $\hat{L}$;
3 Compute the variance of CSI values σ;
4 Group the n packets into a batches, each with b packets;
5 **for** $i = 1 : N$ **do**
6 **for** $j = 1 : a$ **do**
7 //compute the reconstructed CSI $\hat{V}_j$ with b packets;
8 $\hat{V}_j = V_j$;
9 //where V_j is the matrix with 90 rows and b columns;
10 **for** $k = 1 : 8$ **do**
11 $\hat{V}_j = 1/(1 + \exp(-\hat{V}_j \cdot W_k))$;
12 **end**
13 $d_j = \sum_{m=1}^{b} \exp\left(-\frac{1}{\lambda\sigma} \sum_{t=1}^{90} \sqrt{(V_j^{tm} - \hat{V}_j^{tm})^2}\right)$;
14 //where V_j^{tm} is the element at row t and column m in matrix V_j, $\hat{V}_j^{tm}$ is the element at row t and column m in matrix $\hat{V}_j$;
15 **end**
16 $P_i = \frac{1}{n} \sum_{j=1}^{a} d_j$;
17 **end**
18 // Obtain the posterior probability for different locations;
19 **for** $i = 1 : N$ **do**
20 $Pr_i = P_i / \sum_{i=1}^{N} P_i$;
21 **end**
22 // Compute the estimated location;
23 $\hat{L} = \sum_{i=1}^{N} Pr_i L_i$;

the b packets in each batch (line 13). In addition, the expected RBF is computed by averaging over all the n packets (line 16). Then, we compute the the posteriori probability Pr_i for every reference location, thus obtaining the estimated position of the mobile device as the weighted average of all the reference locations (lines 19-23).

2.5 Experiment Validation

2.5.1 Experiment Methodology

Our experiment testbed is implemented with two major components, the access point, which is a TP Link router, and the mobile terminal, which is a Dell laptop equipped with the IWL 5300 NIC.

At the mobile device, the IWL 5300 NIC receives wireless signals from the access point, and then stores raw CSI values in the firmware. In order to read CSI values from the NIC driver, we install the 32-bit Ubuntu Linux, version 10.04LTS of the Server Edition on a Dell laptop and modify the kernel of the wireless driver. In the new kernel, raw CSI values can be transferred to the laptop and can be conveniently read with a C program.

At the access point, the TL router is in charge of continuously transmitting packets to the mobile device. Since the router needs to respond to a mobile device who requires localization service, we use Ping to generate the request and response process between the laptop and the router. Initially, the laptop Pings the router, and then the router returns a packet to the laptop. In our experiment, we design a Java program to implement continuous Pings at a rate of 20 times per second. There are two reasons to select this rate. First, if we run Ping at a lower rate, no enough packets will be available to estimate a mobile device position. Second, if too many Pings are sent, there may not be enough time for the laptop to process the received packets. Also, since we need to continuously estimate the device position, it may cause buffer overflow and packet loss. Numerically, we consider that the rate of 20 times per second is the order of magnitude from 10 to 100. Because we need 100 packets for online localization, it takes few seconds to complete the data collection. If we select the rate of 1 time per second, more than 1 minute are required to collect data, which is not acceptable for online phase. On the other hand, if we choose the rate of 1000 times per second for online phase, huge data are obtained, which cannot obviously improve the result of indoor localization. In addition, after the IWL 5300 NIC receives a packet, the raw CSI value will be recorded in the hardware in the form of CSI per packet reception. DeepFi can obtain 90 raw CSI values for each packet reception, which are all used for fingerprinting or for estimating the device position.

We experiment with DeepFi and examine both the training phase and the test phase. During the training phase, CSI values collected at each location are utilized to learn features, which are then stored as fingerprints. In the test phase, we need to use online data to match the closest spot with the similar feature stored in the training phase. In fact, a major challenge in the feature

matching is how to distinguish each spot without overlap or fuzziness. Although CSI features vary for different propagation paths, two spots with a shorter distance and a similar propagation path may have a similar feature. We examine the similarity of CSI feature along with spot interval in Section 2.5.5, where more details are discussed. If the training spots we select are too sparse, it is possible to cause fuzziness in the test phase, resulting in low localization accuracy. For example, a measurement could hardly match any training spot with high similarity, as it in fact has strong similarity with many random spots. On the other hand, if we choose dense training spots, it will cost a lot of efforts on pre-training data collection. Based on our experiments, the distance between two spots is set to 50 cm, which can maintain the balance between localization accuracy and pre-process cost.

Since DeepFi fully explores all CSI features to search for the most matched spot, each packet is able to fit its nearest training spot with high probability. Therefore, in our localization system, only one access point is utilized to implement DeepFi, which can achieve similar precision as other methods such as Horus and FIFS with two or more access points. Although DeepFi has high accuracy with a single access point, it needs more time and computation in the offline training phase in order to learn fine-grained features of the spots. Fortunately, the pre-training process will be performed in the offline phase, while the online test phase can estimate position quickly. We design a data collection algorithm with two parts. In the training phase, we continuously collect 500–1000 packets at each spot and the measurement will lasts for 1 min. When collecting packets in our experiment, the laptop remains static on the floor, while all the test spots are at the same height, which construct a 2D platform. Then all the packets collected at each spot are used in DeepFi to calculate the weights of the deep network, which are stored as a spot feature. In the test phase, since we match for the closest position with weights we have saved in the database, it is unnecessary to group a lot of packets for complex learning processing. We thus use 100 packets to estimate position, thus significantly reducing the operating complexity and cost.

We verify the performance of DeepFi in various scenarios and compare the resulting location errors in different environments with several benchmark schemes. We find that in an open room

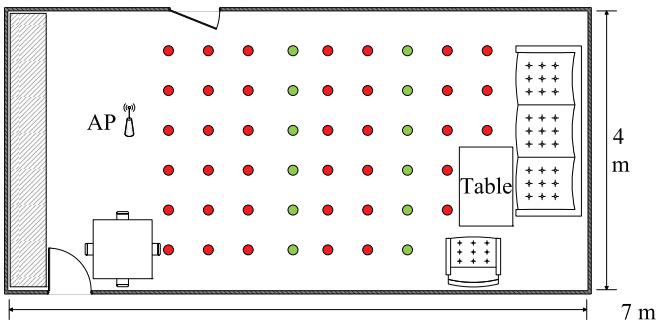


Figure 2.7: Layout of the living room for training/test positions.

where there are no obstacles around the center, the performance of indoor localization is better than that in a complex environment where there are fewer LOS paths. We present the experimental results from two typical indoor localization environments, as described in the following.

Living Room in a House

The living room we choose is almost empty, so that most of the measured locations have LOS receptions. In this 4×7 m^2 room, the access point was placed on the floor, and so do all the training and test points. As shown in Fig. 2.7, 50 positions are chosen uniformly scattered with half meter spacing in the room. Only one access point is utilized in our experiment, which is placed at one end (rather than the center) of the room to avoid isotropy. We arbitrarily choose 12 positions along two lines as test positions and use the remaining positions for training (in Fig. 2.7: the training positions are marked in red and the test positions are marked in green). For each position, we collect CSI data for nearly 500 packet receptions in 60 s. We choose a deep network with structure $K_1 = 300, K_2 = 150, K_3 = 100$, and $K_4 = 50$ for the living room environment.

Computer Laboratory

The other test scenario is a computer laboratory in Broun Hall in the campus of Auburn University. There are many tables and PCs crowded in the 6×9 m^2 room, which block most of the LOS paths and form a complex radio propagation environment. In this laboratory, 50 training positions and 30 test positions are selected, as shown in Fig. 2.8. The mobile device will also be put at these

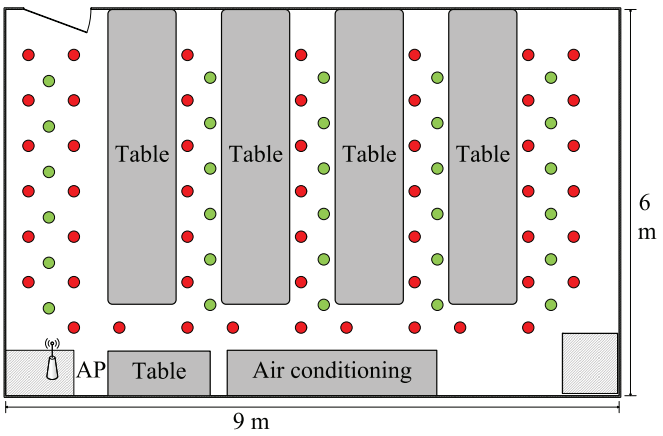


Figure 2.8: Layout of the laboratory for training/test positions.

locations on the floor, with LOS paths blocked by the tables and computers. To obtain fine-grained characteristics of the subcarriers, CSI information from 1000 packet receptions are collected at each training position. We choose a deep network with structure $K_1 = 500, K_2 = 300, K_3 = 150$, and $K_4 = 50$ for the laboratory environment.

Benchmarks and Performance Metric

For comparison purpose, we implemented three existing methods, including FIFS [22], Horus [19], and Maximum Likelihood (ML) [59]. FIFS and Horus are introduced. In ML, the maximum likelihood probability is used for location estimation with RSS, where only one candidate location is used for the estimation result. For a fair comparison, these schemes use the same measured dataset as DeepFi to estimate the location of the mobile device.

The performance metric for the comparison of localization algorithms is the mean sum error $\mathcal{E}$. Assume the estimated location of an unknown user i is $(\hat{x}_i, \hat{y}_i)$ and the actual position of the user is (x_i, y_i). For K locations, the mean sum error is computed as

$$\mathcal{E} = \frac{1}{K} \sum_{i=1}^{K} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}. \tag{2.16}$$

Table 2.1: Mean errors for the Living Room and and Laboratory Experiments

	Living Room		Laboratory	
Method	*Mean error (m)*	*Std. dev. (m)*	*Mean error (m)*	*Std. dev. (m)*
DeepFi	0.9425	0.5630	1.8081	1.3432
FIFS	1.2436	0.5705	2.3304	1.0219
Horus	1.5449	0.7024	2.5996	1.4573
ML	2.1615	1.0416	2.8478	1.5545

2.5.2 Localization Performance

We first evaluate the performance of DeepFi under the two representative scenarios. The mean and standard deviation of the location errors are presented in Table 2.1. In the living room experiment, the mean distance error is about 0.95 meter for DeepFi with a single access point. In the computer laboratory scenario, where there exists abundant multipath and shadowing effect, the mean error is about 1.8 m across 30 test points. DeepFi outperforms FIFS in both scenarios; the latter has a mean error of 1.2 m in the living room scenario and 2.3 m in the laboratory scenario. DeepFi achieves a 20% improvement over FIFS, by exploiting the fine-grained properties of CSI subcarriers from the three antennas. Both CSI fingerprinting schemes, i.e., DeepFi and FIFS, outperform the two RSSI-based fingerprinting schemes, i.e., Horus and ML. The latter two have errors of 2.6 m and 2.8 m, respectively, in the laboratory experiment.

Fig. 2.9 presents the CDF of distance errors with the four methods in the living room experiment. With DeepFi, about 60% of the test points have an error under 1 meter, while FIFS ensures that about 25% of the test points have an error under 1 meter. In addition, most of the test points have distance errors less than 1.5 m in FIFS, which is similar to DeepFi. On the other hand, both RSSI methods, i.e., Horus and ML, do not perform as well as the CSI-based schemes. There are only 80% of the points have an error under 2 m.

Fig. 2.10 plots the CDF of distance errors in the laboratory experiment. In this more complex propagation environment, DeepFi can achieve a 1.7 m distance error for over 60% of the test points, which is the most accurate one among the four schemes. Because the tables obstruct most LOS

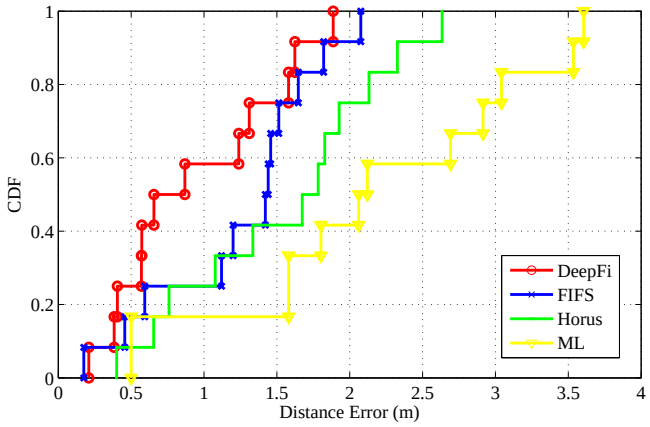


Figure 2.9: CDF of localization errors in the living room experiment.

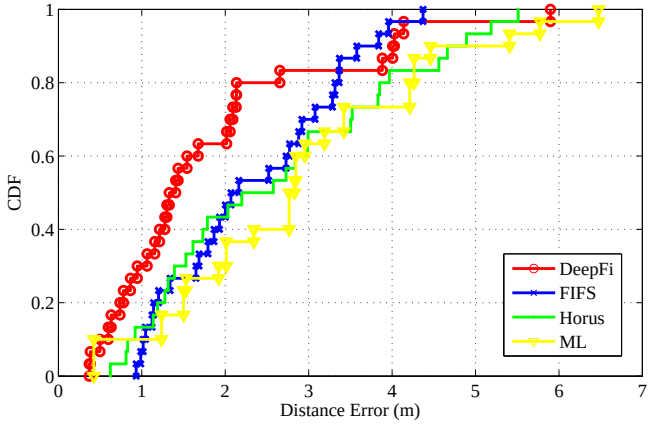


Figure 2.10: CDF of localization errors in the laboratory experiment.

paths and magnify the multipath effect, the correlation between signal strength and propagation distance is weak in this scenario. The methods based on propagation properties, i.e., FIFS, Horus, and ML all have degraded performance than in the living room scenario. In Fig. 2.10, it is noticed that 70% of the test points have a 3 meters distance error with FIFS and Horus. Unlike FIFS, DeepFi exploits various CSI subcarriers. It achieves higher accuracy even with just a single access point. It performs well in this NLOS environment.

2.5.3 Effect of Different System Parameters

Impact of Different Antennas

In order to evaluate the effect of different antennas on DeepFi performance, we consider two different versions of DeepFi: (i) DeepFi with 90 CSI values from the three antennas as input data in both phases (3-antenna DeepFi); (ii) DeepFi with only the 30 CSI values from one of the three antennas in the training phase and estimating the position using 30 CSI values from the same antenna in the test phase (single antenna DeepFi). In addition, we set all the other parameters the same as that in the computer laboratory experiments.

In Fig. 2.11, we compare these two schemes with different antennas in the training and test phases. According to the CDFs of estimation errors, more than 60% of the test points in the 90-CSI scheme have an estimated error under 1.5 m, while the other 30-CSI single antenna schemes have an estimated error under 1.5 m for fewer than 40% of the test points. In fact, the single antenna scheme has a mean distance error around 2.12 m, while the three-antenna scheme has reduced the mean distance error to about 1.84 meters. Thus the 90-CSI scheme achieves better localization accuracy than the 30-CSI schemes, because more environment property of every sampling spot is exploited for location estimation in the test phase as the amount of CSI values is increased from 30 CSI values to 90 CSI values, thus improving the diversity of CSI samples. This experiment validates our Hypothesis 3.

Even though the 3-antennas DeepFi scheme achieves a lower mean error, it takes more time for processing the 90 CSI values as input data for each packet. We evaluate the average processing time to estimate the device position in the test phase using 100 received packets. The processing time is measured as the CPU occupation time for the Matlab program running on a laptop. In Fig. 2.12, we can see that the single antenna schemes take 2.3 s on average to estimate the device position, while the 3-antenna scheme takes around 2.5 s for processing the 100 packets with 90 CSI values per packet as input data to estimate the location. The difference is small, although the latter processes three times input data than that in the single antenna scheme. Although the 3-antenna

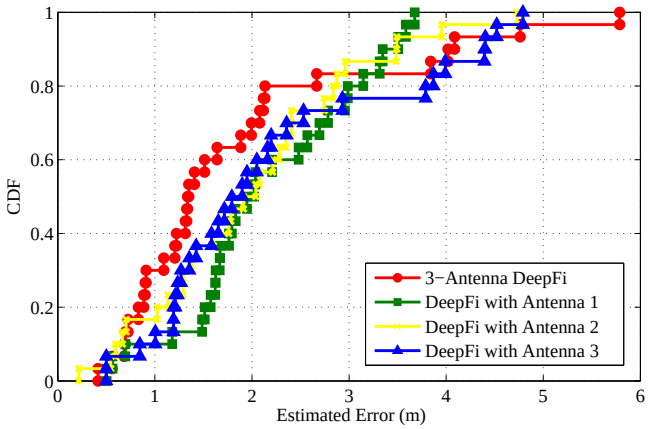


Figure 2.11: CDF of estimated errors for DeepFi with different number of antennas.

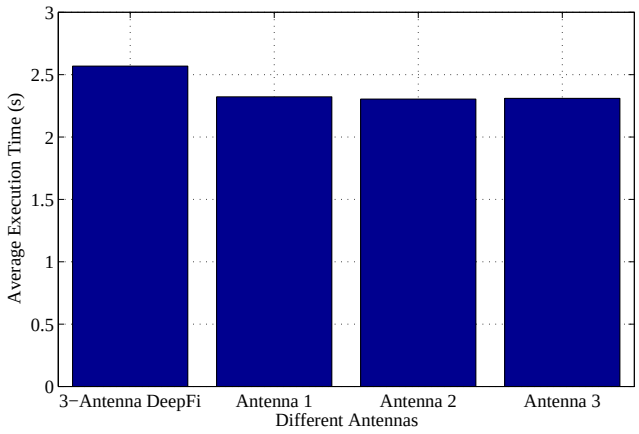


Figure 2.12: The average execution time for DeepFi with different number of antennas.

DeepFi takes about 10 percent extra processing time, it can achieve a 15 percent improvement in localization precision. The latter is generally more important for indoor localization. For 3-antenna approach, it can obtain the most accurate estimation and the average execution time is acceptable for indoor localization. Thus, we consider 3-antenna approach for our DeepFi system.

Impact of the Number of Test Packets

In order to study the impact of the number of test packets, we design a specific experiment by utilizing different numbers of packets to evaluate their effect on both localization accuracy and execution time. In DeepFi, the laptop requests packets from the wireless router every 50 ms, i.e., at a rate of 20 packets per second. In addition, we assume that a user randomly moves with the

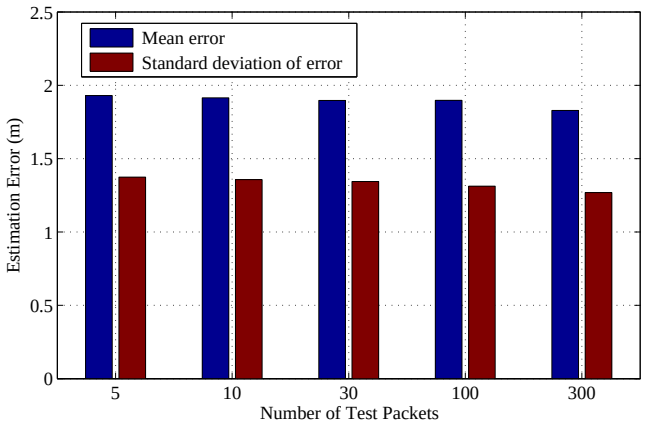


Figure 2.13: The expectation and standard deviation of estimated error for DeepFi using different number of test packets.

speed of about 1 meter per second, then stays in a 1 meter square spot for 1 second, moves again, and so forth. Thus 20 packets per second are received for each test location.

Fig. 2.13 shows the expectation and the standard deviation of localization error of 90 independent experiments. As the number of test packets is increased, the mean localization error tends to decrease. For example, the mean estimated localization error is about 1.83 meters for the case of 300 packets, which is better than the error of 1.93 meter for the case of 5 packets. This is because a large number of test packets provide a stable estimation result, thus mitigating the influence of environment noise on CSI values. Another trend is that the standard deviation of localization error will decrease as the number of packets is increased. This is because that as more samples are available, the standard deviation of samples will be decreased. On the other hand, the characteristic of clusters hidden in CSI values is revealed by increasing the number of packets, thus improving the localization accuracy.

In the case of using 5 test packets, although it takes less than 1/4 s for collecting them, DeepFi can still achieve a good performance of localization. Apart from reducing the collecting time, DeepFi using 5 test packets also simplifies the process of averaging packets in the test phase, thus significantly reducing the execution time for the online phase. We compare the average execution time of position estimation for 90 independent experiments based on recorded CPU occupation time for the cases of using different test packets. In Fig. 2.14, it can be seen that as the number

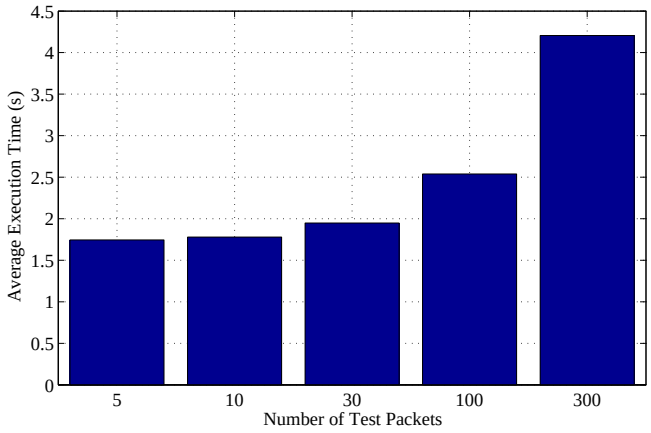


Figure 2.14: The average execution time of position estimation for DeepFi using different number of test packets.

of test packets is increased, the execution time also increases quickly. This is because DeepFi estimates the error of every location by averaging errors of all the test packets. For instance, the execution time with 300 packets is around 4.2 s, which is about 2.5 times of that with 5 packets (about 1.7 s). Therefore, even though more packets contributes to slightly improving the localization precision, we prefer to reduce the number of packets for saving collecting and processing time.

Impact of the Number of Packets per Batch

Since deep learning utilizes n packets in the test phase, how to pre-process these packets is important for DeepFi to reduce the computation complexity. Before the test phase in DeepFi, packets are divided into several batches, each of which contains a same number of packets. Because packets are processed in parallel in batches, we can significantly shorten the processing time when dealing with a large amount of packets. We analyze the impact of the number of packets per batch in this section. We set 1, 3, 5, 10, 50 and 100 packets per batch in the test phase with 100 collected packets. Again, we examine two main effects: the localization error and the test execution time.

Fig. 2.15 shows the expectation and the standard deviation of localization error with different number of packets per batch. As expected, the six experiments maintain approximately the same mean and standard deviation of errors, due to the fact that the parallel processing based on batches

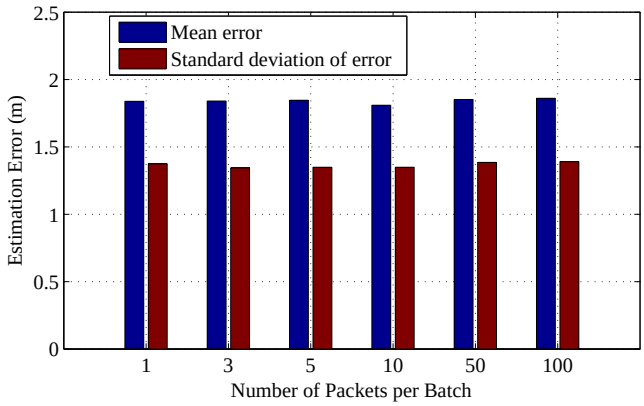


Figure 2.15: The expectation and the standard variation of estimation error for DeepFi with different number of packets per batch.

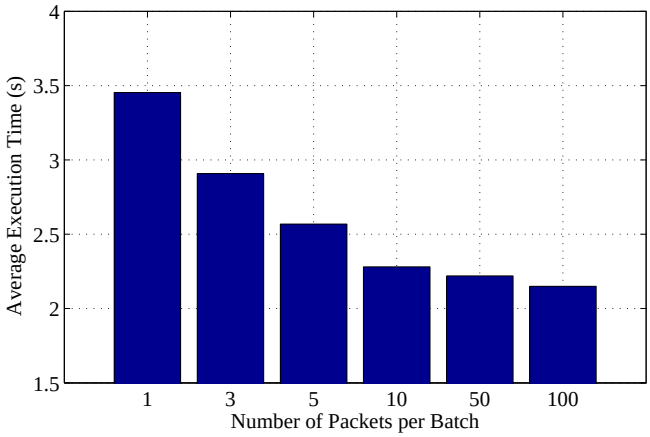


Figure 2.16: The average execution time of position estimation for DeepFi with different number of packets per batch.

only averages the errors of 100 packets. In Fig. 2.16, it is noticed that as the number of packets per batch is increased from 1 to 10, the average execution time decreases quickly. For continuing increasing the number of packets from 10 to 100, we can find that the average computation time is approximately from 2.28 s to 2.15 s, which has smaller change. In addition, we need to average the errors over different batch data to improve the robustness of the localization results. For example, if we consider 100 packets per batch, there is only 1 batch for 100 packets, thus leading that we cannot average the errors. Thus, we employ 10 packets per batch for our DeepFi system, which not only has lower average computation time, but also higher localization results.

2.5.4 Impact of Environment Variation

Since the CFR changes as the indoor propagation environment varies, we examine the effect of varying propagation environment on CSI properties through two specific aspects: replaced obstacles in the room and human mobility. First, because the relative distance between the transmitter and the obstacle can affect the strength and direction of reflection of wireless signal, we consider the impact of replaced obstacles at different relative distances. In the experiment, We place a laptop and a wireless router at two fixed positions, and then add obstacles at different distances to the router, i.e., at 1 meter, 2 meters, and 3 meters locations. Then, we calculate and plot the CDF of the correlation coefficient of (i) the 90 CSI values under this cluttered environment and (ii) the 90 CSI values under the obstacle-free environment.

In Fig. 2.17, we can see that as the distance between the obstacle and the wireless router is increased, the correlation between the two groups of 90 CSI values becomes stronger, which means that the obstacle has less impact on wireless signal transmission when it is farther away. This is due to the fact that when the obstacle is farther from the transmitter, there is lower possibility that it distorts strong signals such as the LOS signal that the laptop receives. In addition, more than 80% of the test points have a correlation coefficient greater than 0.8 when the obstacle is 3 meters away from the wireless router. The high correlation suggests that the obstacle placed more than 3 meter has no significant impact on the 90 CSI values the laptop receives. On the other hand, when the obstacle is very close to the router, the 90 CSI values will slightly change. It leads to a smaller correlation coefficient, which affects the precision of indoor localization in the test phase based on such CSI properties. Therefore, when the obstacle arbitrarily moves in the room, its impact on CSI properties is acceptable, and high localization precision can still be achieved with DeepFi.

In addition to static obstacles, human mobility is another problem we need to consider in practical localization. The experiment of human mobility consists of two scenarios: a user randomly moves (i) near the LOS path, and (ii) near the NLOS path. To demonstrate the effect of human interference on indoor localization, we also plot the CDF of the correlation coefficients between

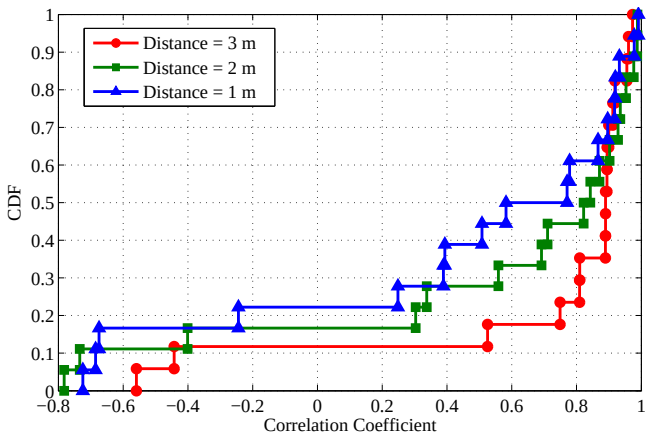


Figure 2.17: CDF of correlation coefficient between the 90 CSI values under cluttered environment and the 90 CSI values measured without obstacles.

(i) the 90 CSI values when a user moves near the LOS path and (ii) the 90 CSI values when a user moves near the NLOS path.

We then present the human mobility experiment results in Fig. 2.18. It can be seen that there are only fewer than 20% of the test points with a correlation coefficient under 0.7, if a user moves near the LOS path. On the other hand, when a user moves apart from the LOS path, approximately 20% of the test points has a correlation coefficient under 0.8. As we can see, the correlation of the two groups of 90 CSI values if a user moves around the LOS path is weaker than that if a user moves around the reflected path, which is about 2 meter away from the wireless router. In fact, due to the stability of CSI values and high correlation coefficients for the above two scenarios, the property of the 90 CSI values will not be significantly affected by human mobility. Therefore, DeepFi can still achieve high localization accuracy even in a busy environment.

2.5.5 Impact of the Training Grid Size

With DeepFi, a mobile device in the test phase uses 90 CSI values it receives to search for the most similar training position. Thus, it is preferable that each training position possesses a unique property for the 90 CSI values. Otherwise, if most of the positions have similar CSI properties, it would be difficult to separate the matched positions from unmatched ones. As a result, these unmatched positions, which randomly scatter in the coverage space, lead to reduced localization

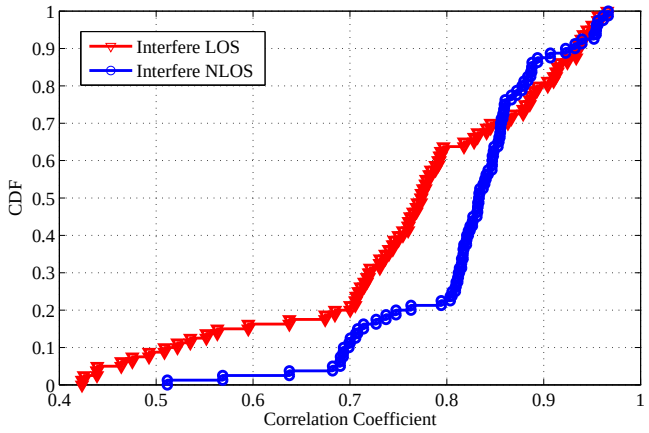


Figure 2.18: CDF of correlation coefficients between the 90 CSI values when a user moves around the LOS path and the 90 CSI values when a user moves around the NLOS path.

accuracy. Therefore, in order to design a suitable training grid size for DeepFi, we study the correlation coefficient of the 90 CSI values between two neighboring training positions as the distance between them is increased. Our experiment records many pairs of positions with different distances, including 15 cm, 30 cm, 60 cm, and 120 cm. In order to mitigate the effect of the direction of the router on the correlation coefficient of the 90 CSI values, we equally place the laptop at four directions facing north, south, west and east.

Figure 2.19 shows that as the grid size is increased, the correlation coefficient of the 90 CSI values between two neighboring positions becomes weaker. In other words, their CSI properties have less similarity due to the larger grid size. In fact, some positions even have low or negative correlation coefficients, even when the grid size is small (i.e., when they are close to each other). This is because the CFR will change as a user moves, as some multipath components may be blocked at near positions and thus some of clusters in received CSI values may be lost. If the CSI values cannot match the corresponding clusters, the correlation will obviously become low. From Fig. 2.19, we find that the localization performance should be acceptable when the grid size is over 30 cm. i.e., most of the training positions can be separated by CSI with the 30 cm range. We thus set the grid size at about 50 cm for the training positions, so that a test position at the center of the square formed by four neighboring training positions has a distance of $50 \times \sqrt{2}/2 = 35$ cm to the nearest training position in the worst case. A larger grid size would fail to match highly similar

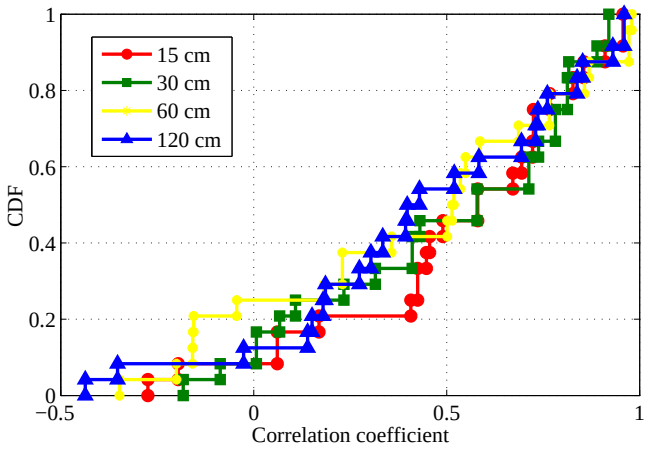


Figure 2.19: CDF of correlation coefficient of the 90 CSI values between two adjacent training positions.

positions because of the scarcity of matched positions, while a smaller grid size requires redundant pre-training work.

2.6 Conclusion

In this chapter, we presented DeepFi, a deep learning based indoor fingerprinting scheme that uses CSI information. In DeepFi, CSI information for all the subcarriers and all the antennas are collected through the device driver and analyzed with a deep network with four hidden layers. Based on the three hypotheses on CSI, we proposed to use the weights in the deep network to represent fingerprints, and incorporated a greedy learning algorithm for weight training to reduce complexity. In addition, a probabilistic data fusion method based on the RBF was developed for online location estimation. The proposed DeepFi scheme was validated in two representative indoor environments, and was found to outperform several existing RSS and CSI based schemes in both experiments. We also examined the effect of different parameters and varying propagation environments on DeepFi performance, and found that DeepFi can achieve good performance under such scenarios.

Chapter 3

PhaseFi: CSI Phase Fingerprinting for Indoor Localization with a Deep Learning Approach

3.1 Introduction

The proliferation of mobile terminals such as smartphones, tablets, and laptops has stimulated enormous interests in indoor localization and location-based services [60, 61, 62, 63]. As one of the popular schemes for indoor localization, a fingerprinting-based approach first establishes a database with thorough measurements of the field and then infers the real-time location by comparing the new measurements with database data. It requires no additional infrastructure support and is thus amenable for indoor deployment.

By modifying the device driver, we can now obtain CSI from some advanced WiFi NIC, such as the Intel WiFi Link 5300 NIC [20, 21]. CSI values provide subcarrier-level channel measurements, which can be helpful for indoor fingerprinting. For example, FIFS [22] utilizes the weighted average CSI values over multiple antennas to improve the performance of RSS-based method for indoor fingerprinting. Another work, DeepFi [64] learns a large amount of CSI data from three antennas for indoor localization based on a deep network. However, these schemes only consider the amplitude of CSI, and the CSI phase information is ignored, which is largely due to the randomness and unavailability of the raw phase information. To the best of our knowledge, CSI-MIMO [65] incorporates both magnitude and phase information of CSI from each sub-carrier for fingerprinting, but the phase information is not calibrated. In fact, the calibrated phase information obtained with a linear transformation is successfully used for LOS identification with WiFi [66] and passive

human movement detection [67]. These two interesting works motivate us to explore calibrated CSI phase information for indoor fingerprinting.

In this chapter, we present PhaseFi, an indoor fingerprinting system based on calibrated phase information of CSI. In PhaseFi, the raw phase information is first extracted from the CSI values from the 30 subcarriers of each of the three antennas of the Intel WiFi Link 5300 NIC (i.e., 90 in total), by accessing the modified device driver. Then, by implementing a linear transformation to remove the phase offset, we obtain the calibrated phase information, which is shown in our measurement study to be considerably more accurate than raw phases. We also provide a phase calibration algorithm and prove an upper bound on the variance of the calibrated phase, which clearly indicates its stability feature.

In the offline stage, unlike traditional shallow learning methods, we design a deep network with three hidden layers to train the calibrated phase data, and use weights to represent fingerprints, which can fully exploit the characteristic of the calibrated phase data. We also develop a greedy learning algorithm to train the weights in a layer-by-layer manner to effectively reduce the computational complexity. With this training approach, a sub-network between two consecutive layers forms a RBM, which is solved by a CD-1 algorithm for sub-optimal solutions. Once the fingerprint database is established, the online stage uses a Bayes method based on the RBF for location estimation.

We implement the PhaseFi system with a laptop computer and an Access Point (AP), and conduct extensive experiments to validate the performance of the PhaseFi system under two representative indoor environments, including a living room in a house and a computer laboratory that is cluttered with metal tables and computers. We find that PhaseFi outperforms three benchmark schemes that are either based on CSI or RSS in both scenarios.

In summary, the main contributions In this chapter include:

1. We propose to use CSI phase information for indoor fingerprinting. Specifically, we theoretically prove and experimentally validate the feasibility of utilizing the calibrated CSI phase

information for indoor localization. To the best of our knowledge, this is the first work to leverage the calibrated CSI phase information for indoor fingerprinting.

2. We design a deep network with three hidden layers to train the calibrated phase data, and utilize the weights of the deep network to represent fingerprints. We also develop a greedy learning algorithm to effective reduce the computational overhead for training. Furthermore, we present a Bayes method based on RBF for probabilistic location estimation.

3. We implement the PhaseFi system with commodity WiFi device and demonstrate its performance in two representative indoor environments. Experimental results show that PhaseFi outperforms several existing RSSI and CSI based schemes at only slightly increased execution time. PhaseFi satisfies the real-time localization requirement for indoor localization.

The remainder of this chapter is organized as follows. The preliminaries and phase sanitization are introduced in Section 3.2. We present PhaseFi in Section 3.3 and our experimental study in Section 3.4. Section 3.5 concludes this chapter.

3.2 Preliminaries and Phase Sanitization

3.2.1 Channel State Information

In modern digital wireless communication systems, OFDM is widely used (e.g., in WiFi standards such as IEEE 802.11a/g/n) to combat frequency selective fading in multipath propagation environments. As shown in Fig. 3.1, at the OFDM transmitter, data is encoded and mapped into multiple orthogonal subcarriers and then transmitted over the subcarriers. With inverse Fast Fourier Transform (IFFT), the subcarriers are converted from the frequency domain to the time domain. To reduce the inter-symbol interference (ISI), the cyclic prefix is added in the time domain. Then, in-phase and quadrature (I-Q) modulation is used for transmission in the multi-path channel. The digital data is converted into analog data with the Digital to Analog Converter (DAC). Finally, the analog signal is up-converted and amplified by the high power amplifier (HPA). At the OFDM receiver, the signal is down-converted to the baseband. The Automatic Gain Controller (AGC) can

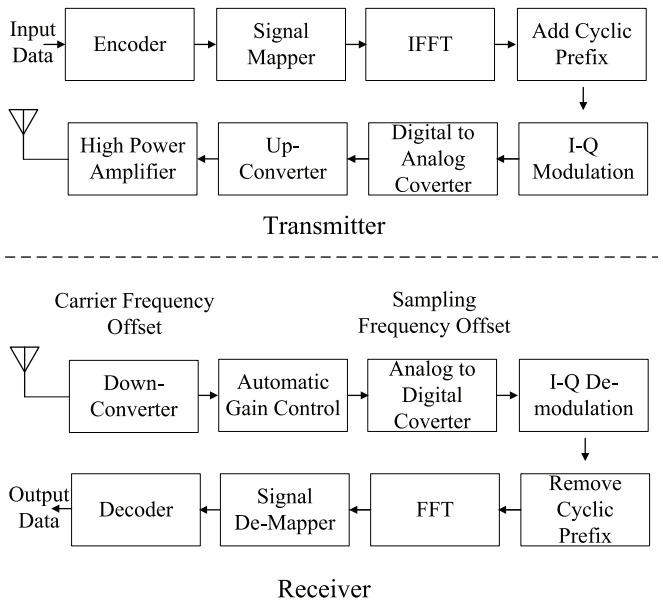


Figure 3.1: The block diagram of an OFDM transceiver

compensate for the signal amplitude attenuation. The inverse process of that at the transmitter is implemented for recovering the data at the receiver.

By modifying the device driver of off-the-shelf NICs, i.e., Intel's IWL 5300, we are able to obtain CSI as fine-grained PHY information, which represents the subcarrier-level channel measurements. In addition, CSI describes the channel properties experienced by the packet. For example, a wireless signal in propagation may undergo considerable impairments due to shadowing, multipath propagation, and distortion, which are reflected in the CSI.

3.2.2 Phase Sanitization

Although the phase of CSI is available from the IWL 5300 NIC, they have not been exploited for indoor localization yet. The problem is mainly due to the hardware imperfection, which leads to measured phase errors. In fact, there are two main causes for the above errors for the system in Fig. 3.1. The first one is CFO generated by the down-converter for receiver signal, because the central frequencies between the receiver and the transmitter cannot be perfectly synchronized. The other one is the SFO generated by the ADC, because of non-synchronized clocks. Moreover,

for SFO, the measured phase errors are different for different subcarriers. Thus, the raw phase information is of limited use for indoor localization.

In this chapter, we propose a simple yet effective approach to mitigate the random phase offsets by implementing a linear transformation. Let $\angle\widehat{CSI}_i$ denote the measured phase of subcarrier i. It can be written as

$$\angle\widehat{CSI}_i = \angle\text{CSI}_i + 2\pi\frac{m_i}{N}\Delta t + \beta + Z, \tag{3.1}$$

where $\angle\text{CSI}_i$ is the genuine phase, Δt is the time lag due to SFO, m_i is the subcarrier index of the ith subcarrier, N is the FFT size, β is the unknown phase offset due to CFO, and Z is the measurement noise. We can obtain the subcarrier indices m_i for $i = 1$ to 30, and the FFT size N from the IEEE 802.11n specification [21]. In fact, because of the unknown Δt and β, it is impossible to obtain the genuine phase information. However, considering the phase across the total frequency band, we can implement a linear transformation on the raw phases to remove the Δt and β terms [67].

Let k and b denote the slope of phase and the offset across the entire frequency band, respectively. It is noticed that the phase error $2\pi\frac{m_i}{N}\Delta t + \beta$ is a linear function of the subcarrier index m_i. We can estimate the slope of phase k and the offset b with the following expressions.

$$k = \frac{\angle\widehat{CSI}_{30} - \angle\widehat{CSI}_1}{m_{30} - m_1} \tag{3.2}$$

$$b = \frac{1}{30}\sum_{i=1}^{30}\angle\widehat{CSI}_i. \tag{3.3}$$

Subtracting $km_i + b$ from the raw phase $\angle\widehat{CSI}_i$, we can obtain the calibrated phase $\angle\widetilde{CSI}_i$, which is given by

$$\angle\widetilde{CSI}_i = \angle\widehat{CSI}_i - km_i - b. \tag{3.4}$$

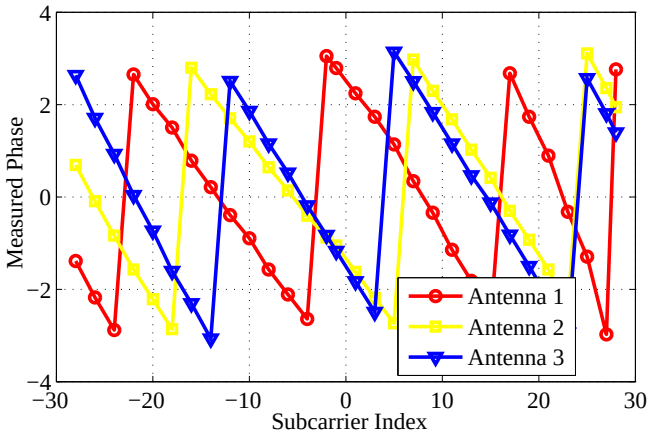


Figure 3.2: Measured phase values for three different antennas.

Although the above expression (3.4) can be used for calibrating phase information, the measured phase is folded due to the recurrence characteristic of phase. Thus, we need to transform the measured phase into the true value. In Fig. 3.2, we plot the measured phase values of CSI for the three antennas at the receiver. It is noticed that the measured phase of each of the three antennas is folded with the increase of subcarrier index and the range of the phase is $[-\pi\ \pi]$. In order to obtain the true measured phase, the folded phase can be recovered by subtracting multiple 2π. Thus, we propose a new phase calibration algorithm in Algorithm 3. In lines (8-13) of the algorithm, the measured phase is compensated for multiple 2π's by judging whether the measured phase change between the adjacent subcarriers is greater than the given threshold such as π. In lines (14-18), the calibrated phase is obtained based on the above phase calibration analysis.

Figure 3.3 presents the true measured phase values for three different antennas. We can see that with the increase of subcarrier index, the true measured phase gradually decreases for all the three different antennas. Fig. 3.4 shows the calibrated phase values for three different antennas. It is noticed that the range of the calibrated phase becomes much smaller than the measured phase for three antennas. On the other hand, we present an upper bound on the variance of the calibrated phase in the following theorem.

Theorem 1. *When the indices of 30 subcarriers are symmetric (i.e., ranging from -28 to 28 as in IEEE 802.11n) and the true phases of the 30 subcarriers are i.i.d., an upper bound of the variance*

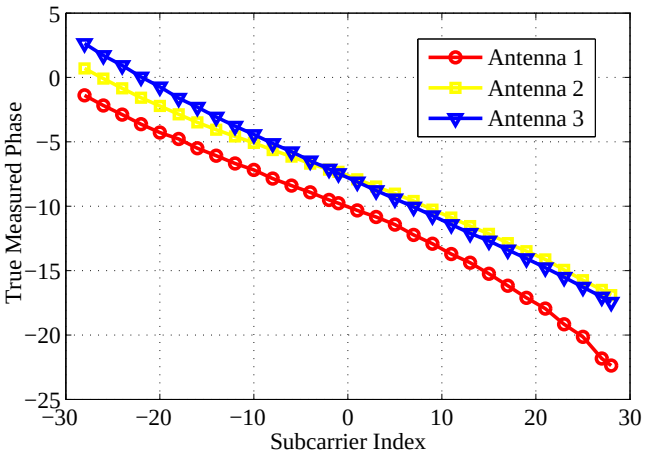


Figure 3.3: True measured phase values for three different antennas.

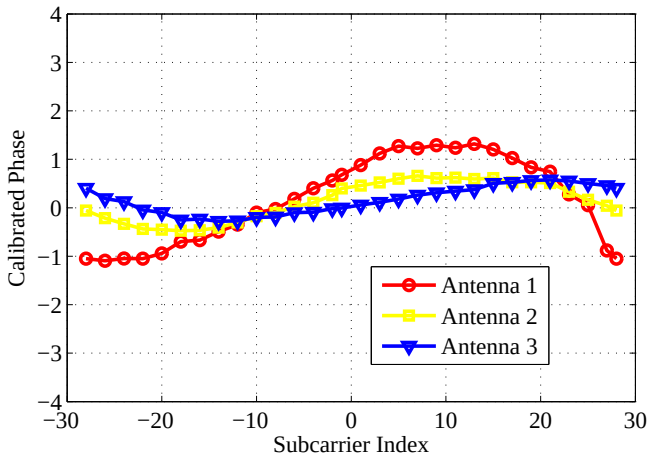


Figure 3.4: Calibrated phase values for three different antennas.

of the calibrated phase is given by

$$\text{Var}(\angle\widetilde{CSI}_i) \leq \frac{23}{15}\text{Var}(\angle CSI_i). \tag{3.5}$$

Proof. We can compute the slope of the phase $k = \frac{\angle CSI_{30} - \angle CSI_1}{m_{30} - m_1} + \frac{2\pi}{N}\Delta t$, and the offset across the total frequency band $b = \frac{1}{30}\sum_{i=1}^{30}\angle CSI_i + \frac{2\pi\Delta t}{30N}\sum_{i=1}^{30} m_i + \beta + Z$. Since the indices of the 30 subcarriers are symmetric for IEEE 802.11n [67], we have $\sum_{i=1}^{30} m_i = 0$. It follows that $b = \frac{1}{30}\sum_{i=1}^{30}\angle CSI_i + \beta$+$Z$. Substituting the slope of the phase, k, the offset, b, and the measured

Algorithm 3: Phase Calibration

Input: measured phase values M_P of 30 subcarriers;
Output: calibrated phase values C_P of 30 subcarriers;
Set T_P as a vector as the same size of M_P;
Set m as a vector from -28 to 28;
Set diff $= 0$;
Set $\eta = \pi$;
Set $T_P(1)$=$M_P(1)$;
for $i = 2 : 30$ **do**
    if $M_P(i) - M_P(i-1) > \eta$ **then**
        diff $=$ diff $+ 1$;
    end
    $T_P(i) = M_P(i) -$ diff $* 2 * \pi$;
end
Compute $k = \frac{T_P(30)-T_P(1)}{m(30)-m(1)}$;
Compute $b =$ sum$\{T_P\}/30$;
for $i = 1 : 30$ **do**
    $C_P(i) = T_P(i) - k * m(i) - b$;
end

phase of subcarrier i, $\angle\widehat{CSI}_i$, into (3.4), the calibrated phase is given by

$$\angle\widetilde{CSI}_i = \angle CSI_i - \frac{\angle CSI_{30} - \angle CSI_1}{m_{30} - m_1} m_i - \frac{1}{30}\sum_{i=1}^{30} \angle CSI_i.$$

Note that the calibrated phase is a linear combination of the true phases, with the random offset β and time lag Δt removed. Since the true phases of the 30 subcarriers are i.i.d., the variance of the calibrated phase is Var$(\angle\widetilde{CSI}_i) =$ Var$(\angle CSI_i) + \frac{m_i^2}{(m_{30}-m_1)^2}(Var(\angle CSI_{30}) +$ Var$(\angle CSI_1)) +$ Var$(\frac{1}{30}\sum_{i=1}^{30} \angle CSI_i)$. Since the subcarrier indices are symmetric, we have $m_i \leq m_{30}$ and $m_{30} = -m_1$, and it follows that $\frac{m_i^2}{(m_{30}-m_1)^2} \leq \frac{m_{30}^2}{(2m_{30})^2} = \frac{1}{4}$. Furthermore, since the true phases of the 30 subcarriers are i.i.d., we have Var$(\frac{1}{30}\sum_{i=1}^{30} \angle CSI_i) = \frac{1}{30}Var(\angle CSI_i)$ and Var$(\angle CSI_i) =$ Var$(\angle CSI_{30}) =$ Var$(\angle CSI_1)$. We thus have Var$(\angle\widetilde{CSI}_i) \leq \frac{23}{15}Var(\angle CSI_i)$, which completes the proof. □

Theorem 1 provides an upper bound on the variance of the calibrated phase, and indicates that the calibrated phase is relatively more stable. In Fig. 3.5, we plot the raw phases (as blue crosses) and the calibrated phases (as red dots) in the polar coordinate system for 100 CSI data units for the

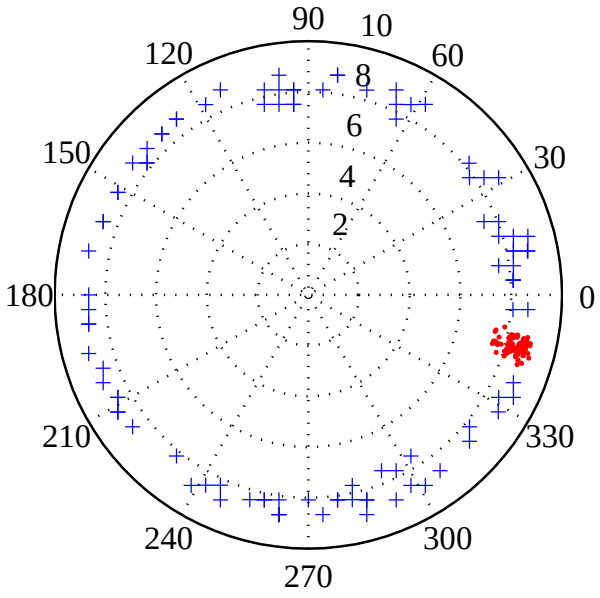


Figure 3.5: Raw phase and calibrated phase measurements.

8th subcarrier in the first antenna of the IWL 5300 NIC. It can be easily seen that the raw phases scatter randomly over all feasible angles. This is why it is not useful for indoor localization. However, the calibrated phases, after the proposed linear transformation, all concentrate into a sector between 330° and 0°. Thus, the proposed linear transform does remove the phase offset.

On the other hand, another characteristic of CSI phase is the great variability at different locations. Fig. 3.6 plots the calibrated phase for 100 packet receptions from three different positions, from which we can observe that calibrated phases are different for three locations. The calibrated phase not only is more stable in one given location, but also varies in different locations, which can be very useful for indoor fingerprinting.

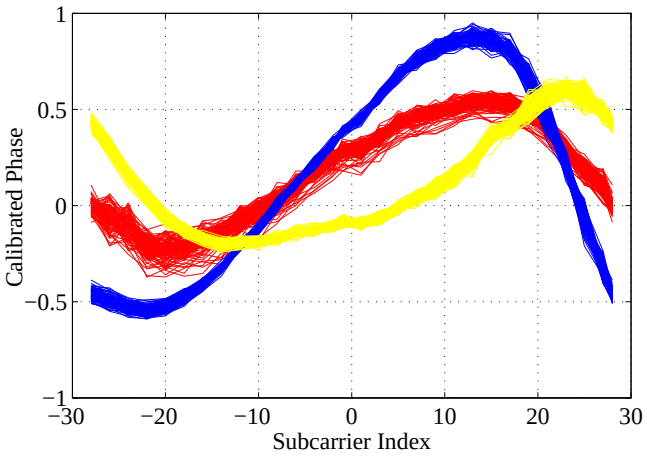


Figure 3.6: Calibrated phase values for three different locations.

3.3 The PhaseFi System

3.3.1 System Architecture

The architecture of PhaseFi is presented in Fig. 3.7. In our design, PhaseFi requires one mobile device equipped with an IWL 5300 NIC, which can read CSI data from the slight modified device driver. The IWL 5300 NIC has three antennas, each of which receives from 30 subcarriers. Thus we can collect 90 CSI data units for one packet reception. Since all the subcarriers are utilized, PhaseFi can effectively improve the diversity of training samples in deep learning, and is thus effective in exploiting the location features for building the fingerprint database. Then, the calibrated phases are obtained by implementing the proposed linear transformation on the the raw phases extracted from CSI data. PhaseFi considers the phase data for indoor fingerprinting for two reasons. First, when a signal encounters obstacle blockages, the amplitude of the signal will be strongly weakened, but the phase of the signal with the periodical change over the propagation distance is relatively more robust. Second, the calibrated phase information is relatively more stable for a given position.

The calibrated phases are then used for both offline training and online testing. In the offline training stage, PhaseFi employs a deep network with three hidden layers to train the calibrated

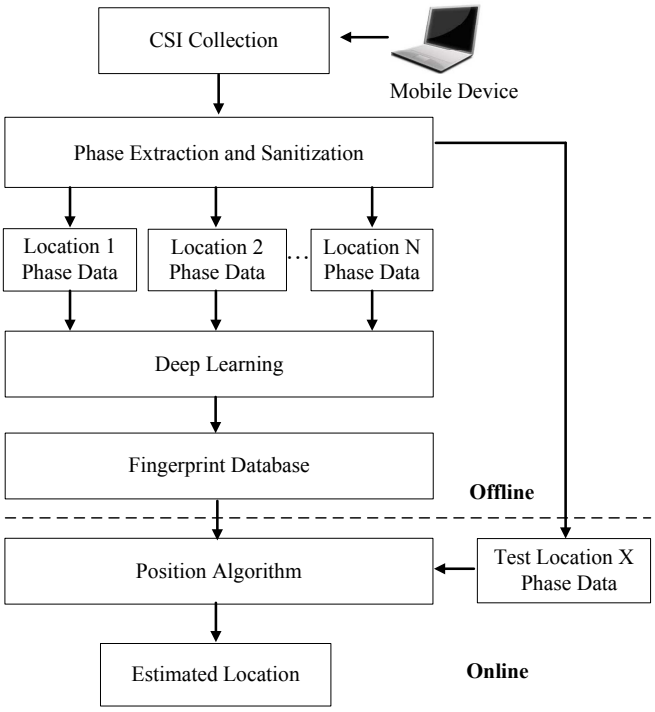


Figure 3.7: Architecture of the proposed PhaseFi system.

phases. It incorporates deep learning to generate feature-based fingerprints. This approach is different from the traditional methods that directly store the measurement data as fingerprints, which are easily influenced by the complex indoor propagation environment. In addition, a large number of weights in the deep network are used as feature-based fingerprints, which effectively represent the characteristics of the calibrated phases for each position. We create the fingerprint database by training the weights of the deep networks with calibrated phases for different positions. In the online test stage, a probabilistic data fusion approach is used to estimate the mobile device location based on the fingerprint database and the new calibrated phase data from the mobile device.

3.3.2 Offline Training

In the offline training stage, PhaseFi incorporates deep learning to train weights and then stores them as the feature-based fingerprint database. The training procedure consists of three stages: pretraining, unrolling, and fine-tuning [8] as shown in Fig. 3.8. In the pretraining stage, we use a deep network with one input layer (with K_0 inputs) and three hidden layers (each with K_i nodes,

$k = 1, 2, 3$). The final weight value mainly depends on input data and deep network structure including the number of hidden layers and the number of nodes in each hidden layer. For PhaseFi, we employ three hidden layers to train and test CSI calibrated phase data. There are two reasons for the chosen structure. First, the deep network with the three hidden layers can achieve near real-time online localization performance, where the mean execution times are 0.3780 s and 0.3770 s for living room and laboratory, respectively. If we use the deep network with four or more layers, it leads to a higher time complexity for online localization, which is not traditionally effective for systems with real-time requirement. Second, the deep network with the three hidden layers can also achieve low localization errors, which are 1.0800 m and 2.0134 m for the living room and laboratory scenarios, respectively. If we use a network with less than or equal to two layers, the deep network will become a shallow network, where deep learning may not be necessary for these networks and the localization accuracy will become low. Let h^i denote the hidden variable with K_i nodes at layer i, $i = 1, 2, 3$, and h^0 denote the calibrated phase data. In addition, let W_1, W_2 and W_3 be the weights between the calibrated phase data and the first hidden layer, the first and second hidden layer, and the second and third hidden layer, respectively.

Let $\Pr(h^0, h^1, h^2, h^3)$ denote the probabilistic generative model for the deep network with one input layer and three hidden layers. To obtain the optimal weights in the pretraining stage, we need to maximize the marginal distribution of the calibrated phase data for the deep network, which is formulated by

$$\max_{\{W_1, W_2, W_3\}} \sum_{h^1} \sum_{h^2} \sum_{h^3} \Pr(h^0, h^1, h^2, h^3). \tag{3.6}$$

Due to the complex model structure with multiple hidden layers and a large number of nodes in the deep network, it is challenging to obtain the optimal weights using the calibrated phase data with the maximum likelihood method. In PhaseFi, we develop a greedy learning algorithm to train the weights layer-by-layer by using a stack of RBMs to reduce complexity [25]. For the layer i RBM model, $i = 1, 2, 3$, the joint distribution $\Pr(h^{i-1}, h^i)$ is expressed by an RBM as a bipartite

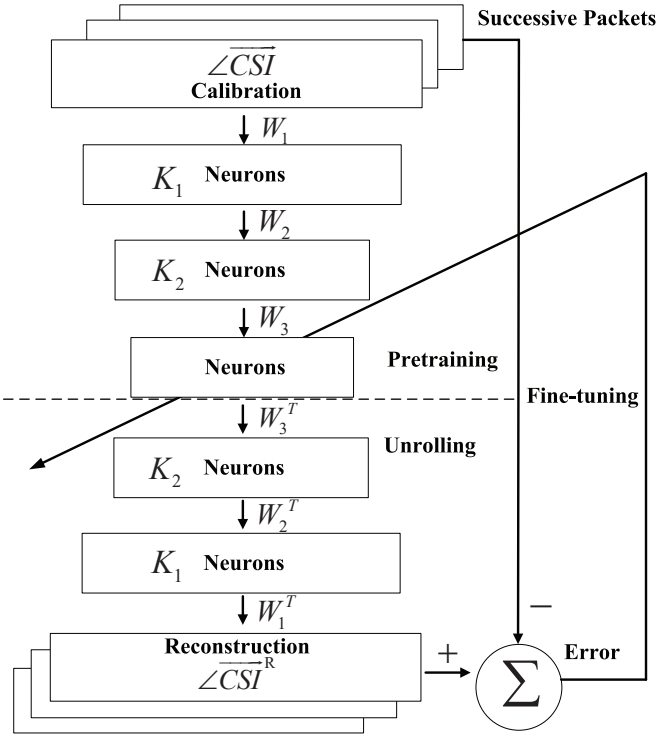


Figure 3.8: The training procedure of PhaseFi.

undirected graphical model [25], which is given by

$$\Pr(h^{i-1}, h^i) = \frac{\exp(-\mathbb{E}(h^{i-1}, h^i))}{\sum_{h^{i-1}} \sum_{h^i} \exp(-\mathbb{E}(h^{i-1}, h^i))}, \tag{3.7}$$

where $\mathbb{E}(h^{i-1}, h^i)$ represents the free energy between layer $i-1$ and layer i. $\mathbb{E}(h^{i-1}, h^i)$ is defined as

$$\mathbb{E}(h^{i-1}, h^i) = -b^{i-1}h^{i-1} - b^i h^i - h^{i-1} W_i h^i, \tag{3.8}$$

where b^{i-1} and b^i are the biases for units of layer $i-1$ and units of layer i, respectively. In fact, since it is difficult to find the joint distribution $\Pr(h^{i-1}, h^i)$, we use the CD-1 algorithm to approximate

it as follows.

$$\begin{cases} \Pr(h^{i-1}|h^i) = \prod_{j=1}^{K_{i-1}} \Pr(h_j^{i-1}|h^i) \\ \Pr(h^i|h^{i-1}) = \prod_{j=1}^{K_i} \Pr(h_j^i|h^{i-1}), \end{cases} \tag{3.9}$$

where $\Pr(h_j^{i-1}|h^i)$, and $\Pr(h_j^i|h^{i-1})$ are described by sigmoid belief network, that are

$$\begin{cases} \Pr(h_j^{i-1}|h^i) = \left[1 + \exp\left(-b_j^{i-1} - \sum_{t=1}^{K_i} W_i^{j,t} h_t^i\right)\right]^{-1} \\ \Pr(h_j^i|h^{i-1}) = \left[1 + \exp\left(-b_j^i - \sum_{t=1}^{K_{i-1}} W_i^{j,t} h_t^{i-1}\right)\right]^{-1}. \end{cases} \tag{3.10}$$

We use the greedy algorithm to estimate the parameters of all weights for a stack of RBMs. First, given the calibrated phase data, the parameters $\{b^0, b^1, W_1\}$ of the first layer RBM are estimated by using CD-1 method. Then we freeze the parameters $\{b^0, W_1\}$ of the first layer, and sample from the conditional probability $\Pr(h^1|h^0)$ to train the parameters $\{b^1, b^2, W_2\}$ of the second layer RBM. Next, the parameters $\{b^0, b^1, W_1, W_2\}$ of the first and second layers are frozen, and then we sample from the conditional probability $\Pr(h^2|h^1)$ to train the parameters $\{b^2, b^3, W_3\}$ of the third layer RBM.

To update the weights in each RBM, the CD-1 method is adopted to approximate them. For the layer i RBM model, First, $\hat{h}^{i-1}$ is estimated by sampling from the conditional probability $\Pr(h^{i-1}|h^i)$. Then $\hat{h}^i$ is obtained by sampling from the conditional probability $\Pr(h^i|\hat{h}^{i-1})$. Finally, the parameters are updated as follows.

$$\begin{cases} \Delta W_i = \epsilon(h^{i-1}h^i - \hat{h}^{i-1}\hat{h}^i) \\ \Delta b^i = \epsilon(h^i - \hat{h}^i) \\ \Delta b^{i-1} = \epsilon(h^{i-1} - \hat{h}^{i-1}), \end{cases} \tag{3.11}$$

where ϵ is the step size.

Once the pretraining stage is completed, we obtain the near-optimal weights for the deep network. Then, in the unrolling stage, the reconstructed calibrated phase data are obtained by unrolling the deep network with forward propagation. Finally, we use the back-propagation algorithm to train all weights in the deep network by computing the error between the input calibrated phase data and the reconstructed calibrated phase data. In addition, the error can be used to iteratively optimize the weights layer-by-layer based on the back-propagation algorithm. This stage is called fine-tuning. After minimizing the error, the optimal weights are stored in the fingerprint database.

The pseudocode for weight training with multiple received packets is presented in Algorithm 4. We first receive n packet for each of the N training positions, each of which has 90 CSI calibrated phase data units as input data. Let $v(t)$ be the input data from packet t. The output of the training includes N groups of fingerpirnts, each of which owns six weight matrices. Moreover, a deep network for each of the N training locations should be trained. The training phase consists of three steps: pretraining, unrolling and fine-tuning. For pretraining, the greedy learning algorithm is used to train the deep network with three hidden layers. The weight matrix are initialized first, and are then iteratively updated with the CD-1 method for obtaining initial weights, where m packets are learned and iteratively generate output as input of the next hidden layer (lines 4-21).

After weights training is finished, the input data will be unrolled to get the reconstructed data. First, we utilize the input data to compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1} to get the coding output h^3, which is a reduced dimension data (lines 23-26). Then, by computing $\Pr(\hat{h}^{i-1}|\hat{h}^i)$ based on the sigmoid with input $\hat{h}^i$, the reconstructed data $\hat{h}^0$ is sampled, where the weights of the deep network are only transposed, thus reducing the time complexity of weights training (lines 27-31). Once the reconstructed data $\hat{h}^0$ is obtained, a supervised learning method based on back-propagation algorithm is used for the deep network as in the fine-tuning phase. Thus, we compute the error between the input data $v = h^0$ and reconstructed data $\hat{h}^0$ to successively update the weight matrix (lines 33-34).

Algorithm 4: Weights Training

Input: n packet receptions each with 90 CSI calibrated phase values for each of the N training locations;
Output: N groups of fingerprints each consisting of six weight matrices;
for $j = 1 : N$ **do**
    // pretraining;
    for $i = 1 : 3$ **do**
        initialize $W^i = 0$, $b^i = 0$;
        for $k = 1 :$ *maxepoch* **do**
            for $t = 1 : n$ **do**
                $h^0 = v(t)$;
                Compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1};
                Sample h^i from $\Pr(h^i|h^{i-1})$;
                Compute $\Pr(h^{i-1}|h^i)$ based on the sigmoid with input h^i;
                Sample $\hat{h}^{i-1}$ from $\Pr(h^{i-1}|h^i)$;
                Compute $\Pr(h^i|\hat{h}^{i-1})$ based on the sigmoid with input $\hat{h}^{i-1}$;
                Sample $\hat{h}^i$ from $\Pr(h^i|\hat{h}^{i-1})$;
                $W_i = W_i + \alpha(h^{i-1}h^i - \hat{h}^{i-1}\hat{h}^i)$;
                $b^i = b^i + \alpha(h^i - \hat{h}^i)$;
                $b^{i-1} = b^{i-1} + \alpha(h^{i-1} - \hat{h}^{i-1})$;
            end
        end
    end
    //unrolling;
    for $i = 1 : 3$ **do**
        Compute $\Pr(h^i|h^{i-1})$ based on the sigmoid with input h^{i-1};
        Sample h^i from $\Pr(h^i|h^{i-1})$;
    end
    Set $\hat{h}^i = h^i$;
    for $i = 3 : 1$ **do**
        Compute $\Pr(\hat{h}^{i-1}|\hat{h}^i)$ based on the sigmoid with input $\hat{h}^i$;
        Sample $\hat{h}^{i-1}$ from $\Pr(\hat{h}^{i-1}|\hat{h}^i)$;
    end
    //fine-tuning;
    Obtain the error between input data h^0 and reconstructed data $\hat{h}^0$;
    Update the six weights using the error with back-propagation;
end

3.3.3 Position Algorithm

In the online test stage, a probabilistic method is developed to estimate the location of the mobile device based on the fingerprint database and new calibrated phase data. We compute the posteriori

probability $\Pr(l_i|h^0)$ based on Bayes' law, which is given by

$$\Pr(l_i|h^0) = \frac{\Pr(l_i)\Pr(h^0|l_i)}{\sum_{i=1}^{N}\Pr(l_i)\Pr(h^0|l_i)}, \tag{3.12}$$

where N is the number of reference locations, l_i is reference location i in the fingerprint database, $\Pr(l_i)$ is the prior probability that the mobile device is determined to locate at the reference location l_i. We assume that $\Pr(l_i)$ follows an uniformly distribution, and then the posteriori probability $\Pr(l_i|h^0)$ can be simplified as follows.

$$\Pr(l_i|h^0) = \frac{\Pr(h^0|l_i)}{\sum_{i=1}^{N}\Pr(h^0|l_i)}. \tag{3.13}$$

Based on the deep network model, we consider $\Pr(h^0|l_i)$ as the RBF in the form of a Gaussian function to measure the degree of similarity between the reconstructed calibrated phase data $\hat{h^0}$ and the input calibrated phase data h^0, which is given by

$$\Pr(h^0|l_i) = \exp\left(-\frac{1}{\lambda\sigma}\left\|h^0 - \hat{h^0}\right\|\right), \tag{3.14}$$

where σ is the variance and λ is the parameter of the variance of the input calibrated phase data. Finally, the position of the mobile device can be computed as a weighted average of all the reference locations, as

$$\hat{l} = \sum_{i=1}^{N}\Pr(l_i|h^0)l_i. \tag{3.15}$$

3.4 Experimental Validation

3.4.1 Experiment Methodology

We examine the performance of PhaseFi with extensive experiments. In our experiments, a TP Link router serves as AP and the mobile device is a Dell laptop equipped with an Intel WiFi Link

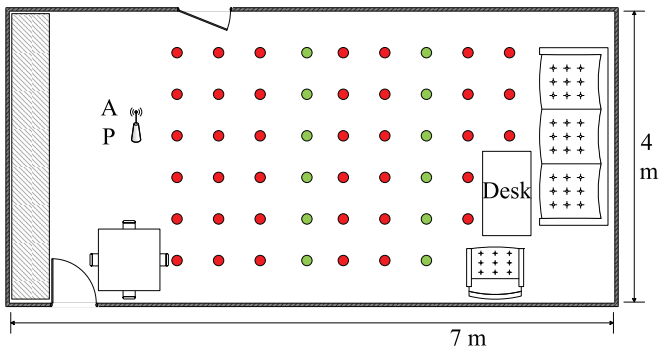


Figure 3.9: Layout of the living room for training/test positions.

5300 NIC. We also modify the NIC's device driver to read CSI values that are recorded in the hardware in the form of CSI for each packet reception. The phase data are extracted from the CSI and calibrated for training and testing.

At the access point, the router needs to respond to a mobile device for the localization service. Thus, the Ping command is employed to implement the request and response process between the laptop and the router. The laptop Pings the router, and then the router returns packets to the laptop. In our localization experiment, we write a Java program to implement continuous Pings at a rate of 20 times per second. There are two reasons to choose this rate. First, if we run Ping at a lower rate, no enough packets will be available to determine a mobile device position. The rate of 20 times per second is proper for the online phase in PhaseFi. Second, if too many Pings are run, it is difficult for the laptop to process the received packets with the short time. Also, because we need to continuously determine the mobile device position, it may cause packet loss and buffer overflow. Moreover, once the IWL 5300 NIC receives a packet, the CSI value will be recorded in the hardware in the form of CSI per packet reception. PhaseFi can obtain 90 CSI values and calibrate them for each packet reception, which are all used for weights training or for determining the mobile device position.

In this section, we validate the performance of PhaseFi in two representative indoor environments as follows. First, we conduct experiments in a 4×7 m^2 living room where there are no outstanding obstacles around the center so that most of the measured locations can have LOS receptions. Fig. 3.9 shows the layout of the living room as well as the training/test points. The

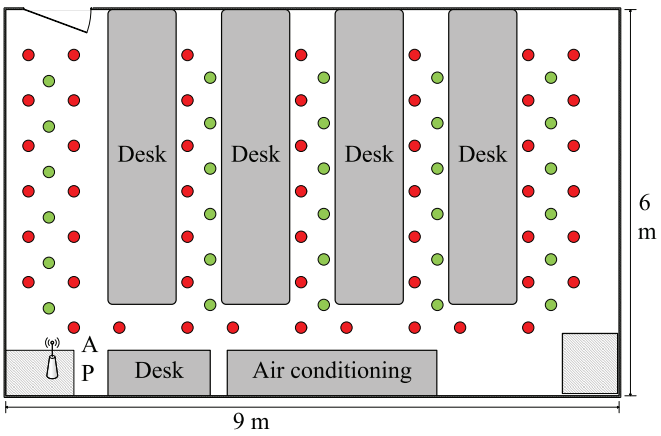


Figure 3.10: Layout of the laboratory for training/test positions.

AP is placed at one end (rather than the center) of the living room on the floor to avoid isotropy. We set 38 points as training points (in red) and 12 points as test points (in green). In addition, we collect CSI data for 400 packet receptions for each training point, and 20 packet receptions for each test point. A deep network with structure 90 inputs, $K_1 = 60$, $K_2 = 30$, and $K_3 = 15$ is used for the living room environment. We use a deep network with structure 90 inputs, K1 = 60, K2= 30, and K3 = 15 for both the living room and laboratory environments. The main reason is that for PhaseFi, we need to satisfy the near real-time requirement for indoor localization so that the number of nodes should be not be very large. Compared with the deep network structure, we also tried other structures, such as K1 = 150, K2 = 100, K3 = 50. We found that although the mean execution time varies from 0.37s to 0.65s, the indoor localization errors do not change much, i.e., around 1m and 2m for living room and laboratory environments, respectively. Thus, we choose the deep network structure with the smaller mean execution time.

Second, we chose a computer laboratory in Broun Hall in the campus of Auburn University. In this 6×9 m^2 laboratory, there are PCs and many desks crowded in the room such that most of the LOS paths are blocked, thus leading to a complex radio propagation environment. Fig. 3.10 shows the layout of the laboratory, where we select 50 training points and 30 test points. The AP is placed on the left bottom corner. To obtain integrated characteristics of the subcarriers, we read CSI data for 800 packet receptions for each training point, and 20 packet receptions for each test

Table 3.1: Mean errors and execution time (Living Room)

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean exe. time (s)*
PhaseFi	1.0800	0.4046	0.3780
FIFS	1.2436	0.5705	0.2362
Horus	1.5449	0.7024	0.2297
ML	2.1615	1.0416	0.2290

Table 3.2: Mean errors and execution time (Laboratory)

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean exe. time (s)*
PhaseFi	2.0134	1.0139	0.3770
FIFS	2.3304	1.0219	0.2439
Horus	2.5996	1.4573	0.2214
ML	2.8478	1.5545	0.2220

point. The structure of the deep network in the laboratory environment is the same as that in the living room environment.

For comparison purpose, we implement three existing methods, including FIFS [22], Horus [19], and ML [59]. FIFS and Horus are discussed in introduction. In ML, based on RSS measurements, only one reference location with maximum posterior probability is considered as the estimated result. For a fair comparison, all schemes use the same measured data set to estimate the position of the mobile device.

3.4.2 Localization Performance

Tables 3.1 and 3.2 present the mean location errors, the standard deviations, and the average execution time of the living room and the laboratory experiments, respectively. In the living room environment, we find PhaseFi to achieve a mean location error of 1.08 m and a standard deviation of 0.4046 m for the 12 test points. In the laboratory scenario, the error is higher due to abundant multipath and shadowing effect. Our system achieves a mean error of 2.0134 m and a standard deviation of 1.0139 m for the 30 test points. For indoor localization accuracy, PhaseFi based on calibrated phases outperforms all the other schemes (i.e., FIFS, Horus and ML) that are based on amplitudes. PhaseFi also demonstrates robust performance for different locations with the smallest

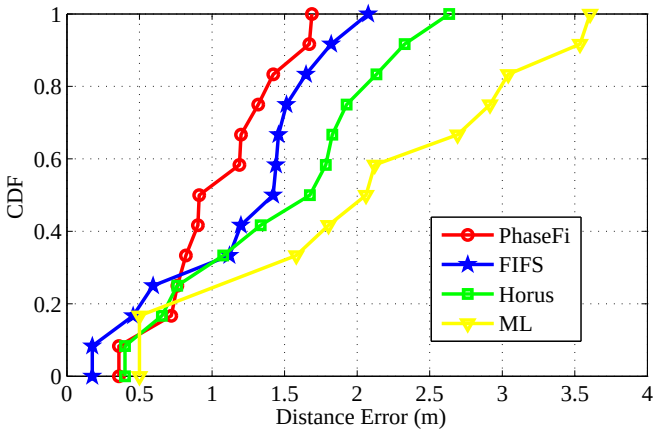


Figure 3.11: CDF of localization errors of the living room experiments.

standard deviation. We also examine the computational complexity of the schemes. Although the mean execution time for PhaseFi is higher than the benchmark schemes, the 0.38 s average execution time of PhaseFi for both scenarios still satisfies the real-time requirement for most indoor localization applications. In fact, by optimizing the parameters and reducing the number of nodes in the deep network, the average execution time of PhaseFi can be further reduced.

Fig. 3.11 shows the CDF of distance errors with the four schemes in the living room scenario. For PhaseFi, more than 50% of the test points have an error under 0.9 m using one AP, while the other schemes guarantee that 30% of the test points have an error under 0.9 m. Moreover, PhaseFi and FIFS have approximate 80% of the test points with mean location errors under 1.5 m, while Horus and ML have the same test points with mean error under 2.0 m and 3.0 m, respectively. The CSI-based schemes such as PhaseFi and FIFS can utilize the fined-grained subcarrier information, and are thus more stable than the RSS-based schemes.

Fig. 3.12 presents the CDFs of distance errors achieved with the four schemes in the laboratory environment. In this more complex propagation environment, for PhaseFi, about 60% of the test points have a distance error under 2 m, while the other schemes have the same portion of test points with error under 2.7 m. We find PhaseFi to be the most accurate among the four schemes, because the phase of the signal periodically changes over the propagation distance, which is more robust than amplitude, especially in cluttered propagation environments. The signal amplitude is

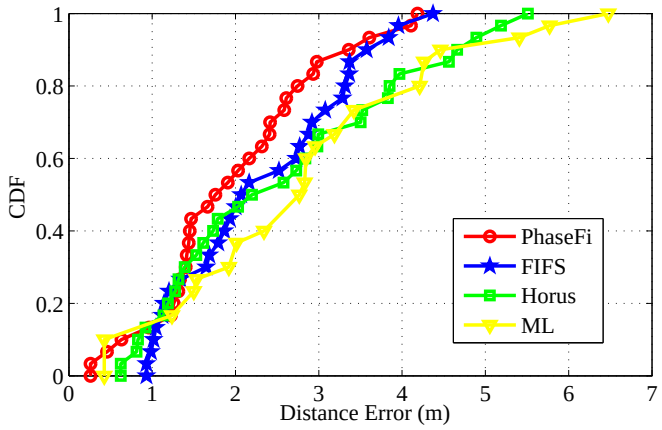


Figure 3.12: CDF of localization errors of the laboratory experiments.

usually more vulnerable to transmission impairments, and the correlation between signal strength and propagation distance is usually weak in indoor scenarios. Thus PhaseFi outperforms the three amplitude based schemes (based on either CSI or RSS).

Fig. 3.13 shows the mean localization errors versus different number of packets in the laboratory and living room experiments. In both experiments, the distance error is decreased as more packets are used. In particular, the mean distance error is decreased from 1.21 m to 1.03 m in the living room experiment, and from 2.23 m to 1.98 m in the laboratory experiment, when the number of packets is increased from 5 to 50. Only small reduction in localization error is achieved when the number of packets is increased for 10 times. Thus, we use 20 packets for online test in PhaseFi, which achieves not only a good localization accuracy, but also a low computational complexity for real-time localization applications.

3.5 Conclusion

In this chapter, we proposed PhaseFi, a phase fingerprinting system for indoor localization. In the system, the phase information was first extracted and calibrated from the three antennas of the Intel WiFi Link 5300 NIC by accessing the modified device driver. In the offline stage, we designed a deep network with three hidden layers to train the calibrated phase data, and used weights to represent fingerprints. To reduce complexity, a greedy learning algorithm was incorporated to

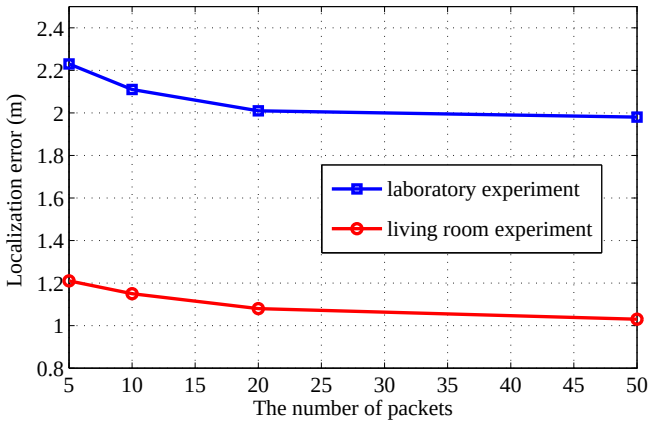


Figure 3.13: Mean localization errors for different number of packets in the laboratory and living room experiments.

train the weights layer-by-layer, where a sub-network between two consecutive layers formed an RBM approximately and solved by a CD-1 algorithm. In the online stage, a Bayes method based on RBF was used for location estimation. The proposed PhaseFi scheme was validated in two representative indoor environments, and was shown to outperform three benchmark schemes based on either CSI or RSS in both scenarios.

Chapter 4

BiLoc: Bi-modal Deep Learning for Indoor Localization with Commodity 5GHz WiFi

4.1 Introduction

The proliferation of mobile devices has fostered great interest in indoor location-based services, such as indoor navigation [68, 69, 70, 71, 72], robot tracking in the factories [73], locating workers in construction sites [74], and activity recognition [75], all requiring accurately identifying locations of mobile devices indoors. The indoor environment poses a complex radio propagation channel, including multipath propagation, blockage, and shadow fading, and stimulates great research efforts on indoor localization theory and systems [12]. Among various indoor localization schemes, *WiFi-based fingerprinting* is probably one of the most widely used techniques. With fingerprinting, a database is first built with data collected from a thorough measurement of the field in the offline training stage. Then, the position of a mobile user can be estimated by matching the newly received data in the pre-built database. A unique advantage of this approach is that no extra infrastructure needs to be deployed.

Recently, for the Intel 5300 NIC in 2.4GHz, two effective methods are proposed to remove the randomness in raw CSI phase data. In [67], the measured phases from 30 subcarriers are processed with a linear transformation to mitigate the randomness in CSI phase, which is then employed for passive human movement detection. In [66], in addition to the linear transformation, the difference of the sanitized phases of two antennas is obtained and used for LOS identification. Although both approaches can stabilize the phase information, the mean value of phase will be zero (i.e., lost) after such processing. This is actually caused by the firmware design of the Intel 5300 NIC when

operating on the 2.4GHz band [76]. To address this issue, Phaser [76] is the first to exploit CSI phase in 5GHz WiFi. Phaser constructs an AOA pseudospectrum for phase calibration in single Intel 5300 NIC. Motivated the above works, we explorer effectively cleansed phase data for indoor localization with commodity 5GHz WiFi.

In this chapter, we consider the problem of fingerprinting-based indoor localization with commodity 5GHz WiFi. We first present three hypotheses on CSI amplitude and phase information for 5GHz OFDM channels. *First*, the average amplitude over two antennas is more stable than that from a single antenna as well as RSS. *Second*, CSI phase difference values from two antennas in 5GHz are highly stable. Due to the firmware design of Intel 5300 NIC, the phase differences of consecutively received packets form four clusters when operating in 2.4GHz. Such ambiguity makes measured phase difference unusable. However, we find this phenomenon does not exit in the 5Ghz band, where all the phase differences concentrate around one value. We further design a simple multi-radio hardware for phase calibration which is greatly different from the technique [76] that uses AOA pseudospectrum searching with high computation complexity to calibrate phase in single Intel 5300 NIC. As a result, the randomness from the frequency and time difference between the receiver and transmitter, and the unknown phase offset can all be removed; and stable phase information can be obtained. *Third*, the calibrated phase difference in 5GHz can be translated into AOA with considerable accuracy when there is a strong LOS component. We validate these hypotheses with both extensive experiments and simple analysis.

We then design BiLoc, **Bi**-modal deep learning for indoor **loc**alization using commodity WiFi devices, to incorporate the three hypotheses in an indoor fingerprinting system. In BiLoc, we first extract raw amplitude and phase data from the three antennas, each with 30 subcarriers, with a modified firmware. We then obtain bi-modal data, including average amplitudes over pairs of antennas and estimated AOAs, with the calibration procedure discussed above. In the training phase, we adopt a deep autoencoder network to extract the unique channel features hidden in the bi-modal data, and leverage the weights of the deep autoencoder network as the extracted features (i.e., fingerprints). To reduce the computational complexity, a greedy learning scheme is leveraged

to train the deep autoencoder network using a RBM model. In the test phase, bi-modal test data is first collected from a mobile device. Based on the RBF, a Bayesian probability model is employed to estimating position.

The main contributions of this chapter are summarized below.

- We theoretically and experimentally validate the feasibility of using bi-modal CSI data for indoor localization. In particular, we deeply analyze the measured phase errors and design a multi-radio hardware for calibrating the unknown phase offset difference in single Intel 5300 NIC. To the best of our knowledge, we are the first to employ both average amplitudes and estimated AOAs for indoor fingerprinting in commodity 5GHz WiFi networks.

- We propose a deep learning approach for indoor fingerprinting. In particular, we leverage a deep autoencoder network to extract OFDM channel features hidden in the rich CSI bi-modal data, and use weights to build the bi-modal fingerprint database. Further, we propose a probability fusion method for accurately estimating position with bi-modal test data.

- We implement the BiLoc system with commodity 5GHz WiFi and show its superior performance in three typical indoor scenarios with extensive experiments. Our test results demonstrate that BiLoc outperforms three representative existing schemes on localization accuracy.

In the rest of this chapter, the preliminaries and hypotheses are given in Section 4.2. We present the BiLoc system in Section 4.3 and validate its performance in Section 4.4. Section 4.5 summarizes this chapter.

4.2 Preliminaries and Hypotheses

4.2.1 Distribution of Amplitude and Phase

In general both $\mathcal{I}_i$ and $\mathcal{Q}_i$ can be modeled as i.i.d. AWGN of variance σ^2. The amplitude response is $|\text{CSI}_i| = \sqrt{\mathcal{I}_i^2 + \mathcal{Q}_i^2}$, which follows a Rician distribution when there is a strong LOS component [77]. The probability distribution function (PDF) of the amplitude response is defined

by

$$f(|\text{CSI}_i|) = \frac{|\text{CSI}_i|}{\sigma^2} \exp\left(-\frac{|\text{CSI}_i|^2 + |\text{CSI}_0|^2}{2\sigma^2}\right) \times I_0\left(\frac{|\text{CSI}_i| \cdot |\text{CSI}_0|}{\sigma^2}\right), \tag{4.1}$$

where $|\text{CSI}_0|$ is the amplitude response without noise, $I_0(\cdot)$ is the zeroth order modified Bessel function of the first kind. When the signal to noise ratio (SNR) is high, the PDF $f(|\text{CSI}_i|)$ will converge to the Gaussian distribution as $\mathcal{N}(\sqrt{|\text{CSI}_0|^2 + \sigma^2}, \sigma^2)$ [77].

The phase response of subcarrier i is computed by $\angle\text{CSI}_i = \arctan(\mathcal{Q}_i/\mathcal{I}_i)$ [77]. The phase PDF is given by

$$\begin{aligned} & f(\angle\text{CSI}_i) \\ = & \frac{1}{2\pi} \exp\left(-\frac{|\text{CSI}_0|^2}{2\sigma^2}\right)\left(1 + \frac{|\text{CSI}_0|}{\sigma}\sqrt{2\pi}\cos(\angle\text{CSI}_i) \times \right. \\ & \left. \exp\left(\frac{|\text{CSI}_0|^2\cos^2(\angle\text{CSI}_i)}{2\sigma^2}\right)\left(1 - Q\left(\frac{|\text{CSI}_0|\cos(\angle\text{CSI}_i)}{\sigma}\right)\right)\right), \end{aligned}$$

where $Q(\cdot)$ is the Q-function. In the high SNR regime, the PDF $f(\angle\text{CSI}_i)$ also converges to a Gaussian distribution as $\mathcal{N}\left(0, (\sigma/|\text{CSI}_0|)^2\right)$ [77]. The distribution of amplitude and phase of the subcarriers would be useful to guide the design of localization algorithms.

4.2.2 Hypotheses

We consider three important hypotheses about the 5GHz CSI data, which are demonstrated and tested with our measurement study and theoretical analysis.

Hypothesis 1

The average CSI amplitude value over two adjacent antennas for the 5GHz OFDM channel is highly stable at a fixed location.

We find CSI amplitude values exhibit great stability for continuously received packets at a given location. Fig. 4.1 presents the CDF of the standard deviations (STD) of (i) the normalized CSI amplitude averaged over two adjacent antennas, (ii) the normalized CSI amplitude from a single antenna, and (iii) the normalized RSS amplitude from a single antenna, for 90 positions. At each position, 50 consecutive packets are received by the Intel 5300 NIC operating on the 5GHz band. It can be seen that 90% of the testing positions are blow 10% of the STD in the case of averaged CSI amplitudes, while the percentage is 80% for the case of single antenna CSI and 70% for the case of single antenna RSS. Thus, averaging over two adjacent antennas can make CSI amplitude highly stable for a fixed location with 5GHz OFDM channels. We conduct the measurements over a long period of time, including midnight hours and business hours. No obvious difference in the stability of CSI is observed over different times, while RSS values exhibit large variations even for the same position. This finding motives us to use average CSI amplitudes of two adjacent antennas as one of the features of deep learning in the BiLoc design.

Recall that the PDF of the amplitude response of a single antenna is Gaussian in the high SNR regime. Assuming that the CSI values of the two antennas are i.i.d. (true when two adjacent antennas are more than a half wavelength apart [66]), the average CSI amplitudes also follow the Gaussian distribution, as $\mathcal{N}(\sqrt{|\text{CSI}_0|^2+\sigma^2}, \sigma^2/2)$, but with a smaller variance. This proves that stability can be improved by averaging CSI amplitudes over two antennas [78](as observed in Fig. 4.1). On the other hand, we consider the average CSI amplitudes over two antennas instead of three antennas or only one antenna, because BiLoc system employs a bi-model data, such as estimated AOAs and average amplitudes. This requires that we use the same number of nodes as the input for a deep autoencoder network.

Hypothesis 2

The difference of CSI phase values between two antennas of the 5GHz OFDM channel is highly stable, compared to that of the 2.4GHz OFDM channel.

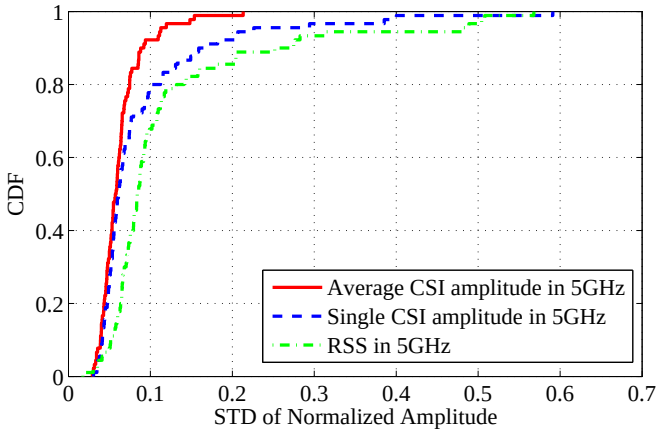


Figure 4.1: CDF of the standard deviations of average CSI amplitude, a single CSI amplitude, and a single RSS in the 5GHz OFDM channel for 90 positions.

Although the CSI phase information is also available from the Intel 5300 NIC, it is highly random and cannot be directly used for localization, due to noise and the unsynchronized time and frequency of the transmitter and receiver. Recently, two useful algorithms are used to remove the randomness in CSI phase. The first approach is to make a linear transform of the phase values measured from the 30 subcarriers [67, 79]. The other one is to exploit the phase difference between two antennas in 2.4GHz and then remove the measured average [66]. Although both methods can stabilize the CSI phase in consecutive packets, the average phase value they produce is always near zero, which is different from the real phase value of the received signal.

Switching to the 5GHz band, we find the phase difference becomes highly stable. Fig. 4.2 shows the measured phase differences of the 30 subcarriers between two antennas for 200 consecutively received packets in the 5GHz (in blue) and 2.4GHz (in red) bands. The phase difference of the 5GHz channel varies between [0.5, 1.8], which is considerably more stable than that of the 2.4GHz channel (varies between $[-\pi, \pi]$). To further illustrate this finding, we plot the measured phase differences between the 5th subcarrier of two antennas using polar coordinates in Fig. 4.4. We find that all the 5GHz measurements concentrate around 30°, while the 2.4GHz measurements form four clusters around 0°, 90°, 180°, and 270°. It is because of the firmware design of the Intel 5300 NIC when operating on the 2.4GHz band, which reports the phase of channel modulo $\pi/2$

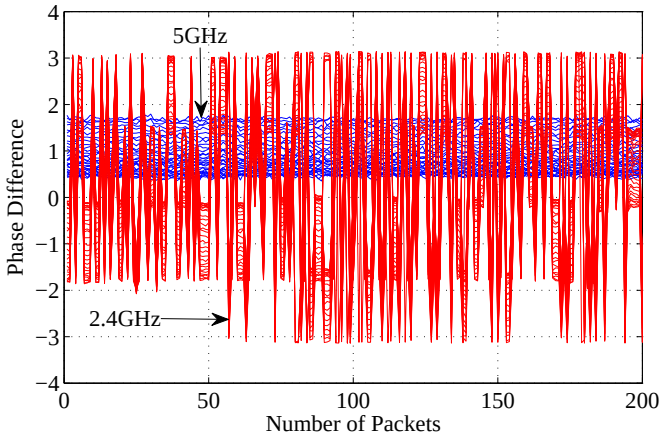


Figure 4.2: The measured phase differences of the 30 subcarriers between two antennas for 200 consecutively received packets in the 5GHz (blue) and 2.4GHz (red) bands.

rather than 2π on the 5GHz band [76]. Comparing to the ambiguity in the 2.4GHz band, the highly stable phase difference in the 5GHz band could be very useful for indoor localization.

As in Hypothesis 1, we also provide an analysis to validate the observation from experiments. Let $\angle\widehat{CSI}_i$ denote the measured phase of subcarrier i, which is given by [80, 81]

$$\angle\widehat{CSI}_i = \angle\text{CSI}_i + (\lambda_p + \lambda_s)m_i + \lambda_c + \beta + Z, \tag{4.2}$$

where $\angle\text{CSI}_i$ is the true phase from wireless propagation, Z is the measurement noise, β is the initial phase offset because of the phase-locked loop (PLL), m_i is the subcarrier index of subcarrier i, λ_p, λ_s and λ_c are phase errors from PBD, SFO and CFO, respectively [80], which are expressed by

$$\begin{cases} \lambda_p = 2\pi\frac{\Delta t}{N} \\ \lambda_s = 2\pi(\frac{T'-T}{T})\frac{T_s}{T_u}n \\ \lambda_c = 2\pi\Delta f T_s n, \end{cases} \tag{4.3}$$

where Δt is the packet boundary detection delay, N is the FFT size, T' and T are the sampling periods from the receiver and the transmitter, respectively, T_s is the total length of the data symbol and the guard interval, T_u is the length of the data symbol, n is the sampling time offset for current

packet, Δf is the center frequency difference between the transmitter and receiver. It is noticed that we cannot obtain the exact values about Δt, $\frac{T'-T}{T}$, n, Δf, and β. Moreover, λ_p, λ_s and λ_c vary for different packets with different Δt and n. Thus, the true phase $\angle \text{CSI}_i$ cannot be derived from the measured phase value.

However, note that the three antennas of the Intel 5300 NIC use the same clock and the same down-converter frequency. Consequently, the measured phases of subcarrier i from two antennas have identical packet detection delay, sampling periods and frequency differences (and the same m_i) [76]. Thus the measured phase difference on subcarrier i between two antennas can be approximated as

$$\Delta\angle\widehat{CSI}_i = \Delta\angle\text{CSI}_i + \Delta\beta + \Delta Z, \tag{4.4}$$

where $\Delta\angle\text{CSI}_i$ is the true phase difference of subcarrier i, $\Delta\beta$ is the unknown difference in phase offsets, which is in fact a constant [76], and ΔZ is the noise difference. We can find that $\Delta\angle\widehat{CSI}_i$ is stable for different packets because of the above equation (4.4) without Δt and n.

In the high SNR regime, the PDF of the phase response of subcarrier i for each of the antennas is $\mathcal{N}(0, (\sigma/|\text{CSI}_0|)^2)$. Due to the independent phase responses, the measured phase difference of subcarrier i is also Gaussian with $\mathcal{N}(\Delta\beta, 2\sigma^2(1 + 1/|\text{CSI}_0|^2))$. Note that although the variance is higher comparing to the true phase response, the uncertainty from the time and frequency differences is removed, leading to much more stable measurements (as shown in Fig. (4.4)).

Hypothesis 3

The calibrated phase difference in 5GHz can be translated into the AOA with considerable accuracy when there is a strong LOS component.

The measured phase difference on subscriber i can be translated into an estimation of AOA, as

$$\theta = \arcsin\left(\frac{\Delta\angle\widehat{CSI}_i\lambda}{2\pi d}\right), \tag{4.5}$$

where λ is the wavelength and d is the distance between the two antennas (set to $d = 0.5\lambda$ in our experiments). Although the measured phase difference $\Delta\angle\widehat{CSI}_i$ is highly stable, we still wish to remove the unknown phase offset difference $\Delta\beta$ to further reduce the error of AOA estimation. For commodity WiFi devices, the existing approach for a single NIC is to search for $\Delta\beta$ within an AOA pseudospectrum in the range of $[-\pi, \pi]$, which, however, has a high time complexity [76].

For the proposed Biloc system, we design a simple method to remove the unknown phase offset difference $\Delta\beta$ using two Intel 5300 NICs. As in Fig. 4.3, we use one Intel 5300 NIC as transmitter and the other as receiver, while a *signal splitter* is used to route signal from antenna 1 of the transmitter to antennas 1 and 2 of the receiver through cables of the same length. Since the two antennas receive the same signal, the true phase difference $\Delta\angle\text{CSI}_i$ of subcarrier i is zero. We can thus obtain $\Delta\beta$ as the measured phase offset difference between antennas 1 and 2 of the receiver. We also use the same method to calibrate antennas 2 and 3 of the receiver, to obtain the unknown phase offset difference between them as well. We notice that the unknown phase offset difference is relatively stable over time.

After calibrating the unknown phase offset differences for the three antennas, we then use the MUSIC algorithm for AOA estimation [82]. In Fig. 4.5, the AOA estimation using MUSIC with the calibrated phase information for the 30 subcarriers is plotted for a high SNR signal with a known incoming direction of 14°. We can see that the peak occurs at around 20° in Fig. 4.5, indicating an AOA estimation error of about 6°. In fact, there are multiple paths indoor environments. Thus, only using three antennas cannot still obtain accurate angle estimation. Moreover, we implemented more experiments for angles estimation by using MUSIC indoor environments; we find that the estimated angles are changing for different locations because of NLOS paths. Although

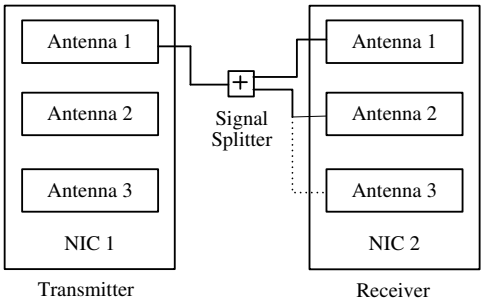


Figure 4.3: The multi-radio hardware design for calibrating the unknown phase offset difference $\Delta\beta$.

the calibrated phase differences are not available for estimating angles by using three antennas, we believe that the new phase calibrated method can be used for future WiFi systems such as IEEE 802.11 ac that has more than three antennas.

We can obtain the true incoming angle with MUSIC when the LOS component is strong. To deal with the case with strong NLOS paths (typical in indoor environments), we adopt a deep autoencoder network to learn the estimated AOAs and the average amplitudes of adjacent antenna pairs as fingerprints for indoor localization. As input to the deep network, the estimated AOA is obtained as follows.

$$\theta = \arcsin\left(\left(\Delta\angle\widehat{CSI}_i - \Delta\beta\right)\frac{\lambda}{2\pi d}\right) + \frac{\pi}{2}, \tag{4.6}$$

where $\Delta\beta$ is measured with the proposed multi-radio hardware experiment. The estimated AOA is in the range of $[0, \pi]$.

4.3 The BiLoc System

4.3.1 BiLoc System Architecture

The overall architecture of BiLoc is illustrated in Fig. 4.6. The BiLoc design uses one mobile device and one access point, each equipped with an Intel 5300 NIC, servicing as receiver and transmitter, respectively [83, 84]. All the communications are on the 5GHz band. Based on the Intel 5300 NIC with three antennas, we can collect 90 CSI data for every received packet. We then

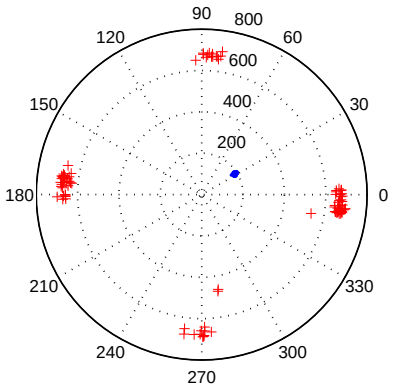


Figure 4.4: The measured phase differences of the 5th subcarrier between two antennas for 200 consecutively received packets in the 5GHz (blue dots) and 2.4GHz (red crosses) bands.

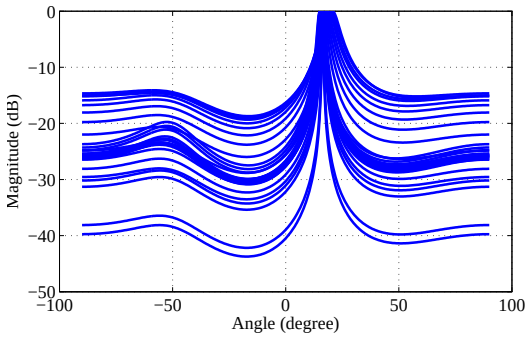


Figure 4.5: The estimated AOAs from the 30 subcarriers using the MUSIC algorithm, while the real AOA is 14°.

calibrate the phase information of the received CSI data using our multi-radio hardware design (see Fig. 4.3). Both the estimated AOAs and average amplitudes of two adjacent antennas are used as location feature for building the fingerprint database.

A unique feature of BiLoc is its bi-modal design. With the three receiving antennas, we can obtain two groups of data: (i) 30 estimated AOAs and 30 average amplitudes from antennas 1 and 2, and (ii) that from antennas 2 and 3. BiLoc utilizes estimated AOAs and average amplitudes for indoor fingerprinting for two main reasons. First, these two types of CSI data are highly stable for any given position. Second, they are usually complementary to each other under some indoor circumstances. For example, when a signal is blocked, the average amplitude of the wireless signal will be significantly reduced; but the estimated AOA becomes more effective. On the other hand,

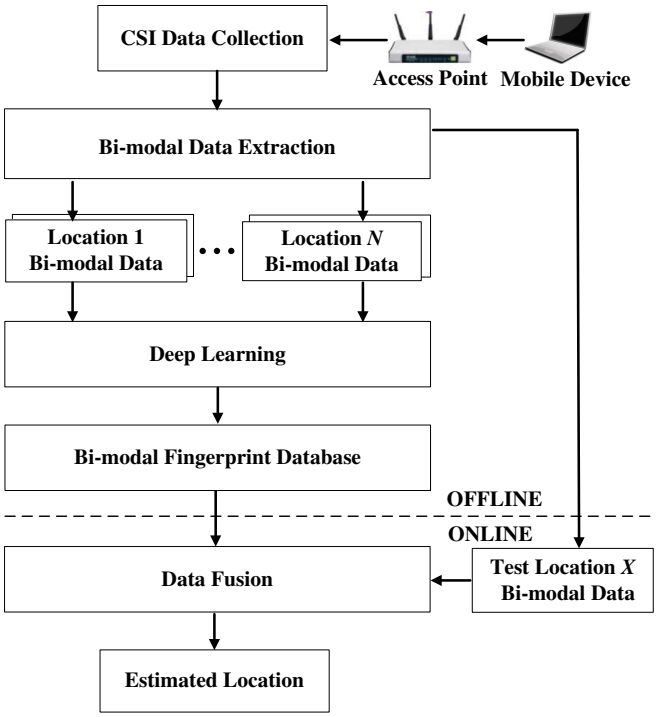


Figure 4.6: The BiLoc system architecture.

when the NLOS components are stronger than the LOS component, the average amplitude will help to improve the localization accuracy.

Another unique characteristic of BiLoc is the use of deep learning to produce feature-based fingerprints from the bi-modal data in the offline training stage, which is quite different from the traditional approach of storing the measured raw data as fingerprints. Specifically, we leverage the weights in the deep autoencoder network as the features-based fingerprints for every position. By obtaining the optimal weights with the bi-modal data on estimated AOAs and average amplitudes, we can establish a bi-modal fingerprint database for the training positions. The third feature of BiLoc is the probabilistic data fusion approach for location estimation based on received bi-modal data in the online test stage.

4.3.2 Offline Training for Bi-Modal Fingerprint Database

In the offline stage, BiLoc leverages deep learning to train and store the weights to build a bi-modal fingerprint database, which is a deep autoencoder network that involves three phases: pretraining,

unrolling, and fine-tuning [8, 64]. In the pretraining phase, a deep autoencoder network with three hidden layers and one input layer is used to learn the bi-modal data. We denote h^i as the hidden variable with K_i nodes at layer i, $i = 1, 2, 3$, and h^0 as the input data with K_0 nodes at the input layer. Let the average amplitude data be v^1 and the estimated AOA data be v^2. To build the bi-modal fingerprint database, we set $h^0 = v^1$ and $h^0 = v^2$ for database 1 and 2, respectively, each of which is a set of optimal weights. We denote W_1, W_2 and W_3 as the weights between input data and the first hidden layer, the first and second hidden layer, and the second and third hidden layer, respectively.

To reduce the computational complexity for obtaining training weights, a greedy learning algorithm for the proposed BiLoc system is developed to learn the weights *layer by layer* based on a stack of RBMs [25]. We develop a greedy algorithm to train the weights and biases for a stack of RBMs, which is the same scheme in PhaseFi system [85]. To train the weights and biases of each RBM, we use the CD-1 approach to solve them. For the layer i RBM model, we estimate $\hat{h}^{i-1}$ by sampling from the conditional probability $\Pr(h^{i-1}|h^i)$; by sampling from the conditional probability $\Pr(h^i|\hat{h}^{i-1})$, we can estimate $\hat{h}^i$.

After the pretraining phase, we then unroll the deep autoencoder network using *forward propagation* to obtain the reconstructed input data in the unrolling phase. Finally, in the fine-tuning phase, the *backpropagation* algorithm is used to train the weights in the deep autoencoder network according to the error between the reconstructed data and the input data. The optimal weights are obtained by minimizing the error. In BiLoc, we use estimated AOAs and average amplitudes as input data, and obtain two sets of optimal weights for the bi-modal fingerprint database.

4.3.3 Online Data Fusion for Position Estimation

In the online phase, we adopt a probabilistic approach to location estimation using the bi-modal fingerprint database and the bi-modal test data. We derive the posteriori probability $\Pr(l_i|v^1, v^2)$

using Bayes' law as

$$\Pr(l_i|v^1, v^2) = \frac{\Pr(l_i)\Pr(v^1, v^2|l_i)}{\sum_{i=1}^{N}\Pr(l_i)\Pr(v^1, v^2|l_i)}, \tag{4.7}$$

where l_i is the ith reference location in the bi-modal fingerprint database, N is the number of reference positions, and $\Pr(l_i)$ is the prior probability, which is uniformly distributed for any reference position l_i [85]. The posteriori probability $\Pr(l_i|v^1, v^2)$ becomes

$$\Pr(l_i|v^1, v^2) = \frac{\Pr(v^1, v^2|l_i)}{\sum_{i=1}^{N}\Pr(v^1, v^2|l_i)}. \tag{4.8}$$

In BiLoc, we approximate $\Pr(v^1, v^2|l_i)$ with an RBF to consider the degree of similarity between the reconstructed bi-modal data and the test bi-modal data, given by

$$\Pr(v^1, v^2|l_i) = \exp\left(-(1-\rho)\frac{\|v^1 - \hat{v}^1\|}{\eta_1\sigma_1} - \rho\frac{\|v^2 - \hat{v}^2\|}{\eta_2\sigma_2}\right), \tag{4.9}$$

where $\hat{v}^1$ and $\hat{v}^2$ are the reconstructed average amplitude and reconstructed AOA, respectively; σ_1 and σ_2 are the variance of the average amplitude and estimated AOA, respectively; η_1 and η_2 are the parameters of the variance of the average amplitude and estimated AOA, respectively; and ρ is the ratio for the bi-modal data.

For the Eq. (4.9), the average amplitudes $\hat{v}^1$ and the estimated AOAs $\hat{v}^2$ are as the input of the deep autoencoder network, where the different nodes of the input can express the different CSI channels. Then, by employing the test data $\hat{v}^1$ and $\hat{v}^2$, we compute the reconstructed average amplitude $\hat{v}^1$ and reconstructed AOA $\hat{v}^2$ based on database 1 and database 2, respectively, which is used to compute the likelihood function $\Pr(v^1, v^2|l_i)$.

The location of the device is finally determined as a weighted average of all the reference positions, that is

$$\hat{l} = \sum_{i=1}^{N}\Pr(l_i|v^1, v^2)\cdot l_i. \tag{4.10}$$

4.4.1 Test Configuration

We present our experimental study with BiLoc in the 5GHz band in this section. In the experiments, we use a Dell laptop as a mobile device and a desktop computer as an access point, both of which are equipped with an Intel 5300 NIC. In fact, we use the desktop computer instead of the commodity routers that are not equipped with the Intel 5300 NIC nowadays. Our implementation of BiLoc is executed on the Ubuntu desktop 14.04 LTS OS for both the access point and mobile device. We use QPSK modulation and a 1/2 coding rate for the OFDM system. For the access point, it is set in monitor model and the distance between two adjacent antennas is $d = 2.68$ cm. It is half of a wavelength for the 5GHz band. For the mobile device, it transmits packets at 100 packets per second using only one antenna in injection mode. 5GHz CSI data can be obtained by using packet injection technique based on LORCON version 1. Then, we extract bi-modal data for training and test stages as described in Section 5.3.2.

We implement three representative schemes from the literature, i.e., Horus [19], FIFS [22], and DeepFi [64]. Moreover, all the schemes employ the same dataset captured in the 5GHz band for a fair comparison. Moreover, we can find that the three methods do not leverage phase difference information. We conduct extensive experiments with the schemes in the following three representative indoor environments.

Computer Laboratory: This is a 6×9 m^2 computer laboratory, a cluttered environment with metal tables, chairs, and desktop computers, blocking most of the LOS paths. The floor plan is shown in Fig. 4.7, with 15 chosen training positions (marked as red squares) and 15 chosen test positions (marked as green dots). The distance between two adjacent training positions is 1.8 m. The single access point is put close to the center of the room. We collect bi-modal data from 1000 packets for each training position, and from 25 packets for each test position. The deep autoencoder network used for this scenario is configured as $\{K_1 = 150, K_2 = 100, K_3 = 50\}$. Also, the ratio ρ for the bi-modal data is set as 0.5.

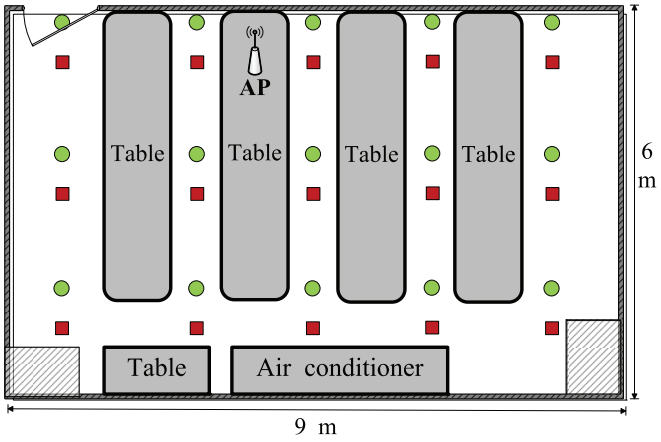


Figure 4.7: Layout of the computer laboratory: training positions are marked as red squares and testing positions are marked as green dots.

Corridor: This is a 2.4×24 m^2 corridor, as shown in Fig. 4.8. In this scenario, the AP is placed at the left end of the corridor and there are plenty of LOS paths. Ten training positions (red squares) and 10 test positions (green dots) are arranged along a straight line. The distance between two adjacent training positions is also 1.8 m. We also collect bi-modal data from 1000 packets for each training position and from 25 packets for each test position. The deep network used for this scenario is configured as $\{K_1 = 150, K_2 = 100, K_3 = 50\}$. Also, the ratio ρ for the bi-modal data is set as 0.1.

Two Corridors: These are a 2.4×24 m^2 corridor and a 2.4×20 m^2 corridor, as shown in Fig. 4.9. In this scenario, six APs are placed two corridors and there also are enormous LOS paths. Eighteen training positions (red squares) are arranged along two corridors. And plenty of test positions are randomly in the two corridors, which are not shown in Fig. 4.9. The distance between two adjacent training positions is also 1.8 m. We also measure bi-modal data from 1000 packets for each training position and from 25 packets for each test position. Other parameters in deep network are set as the same with the above one corridor. For the deployment, we only leverage it to study the impact of the number of APs on indoor localization results for different schemes.

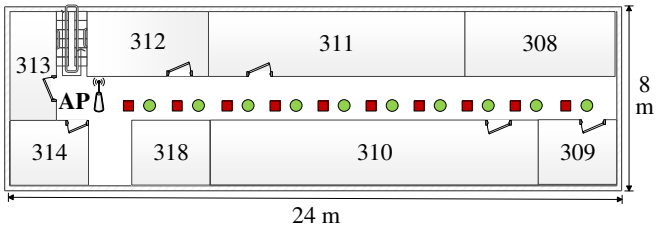


Figure 4.8: Layout of the corridor: training positions are marked as red squares and testing positions are marked as green dots.

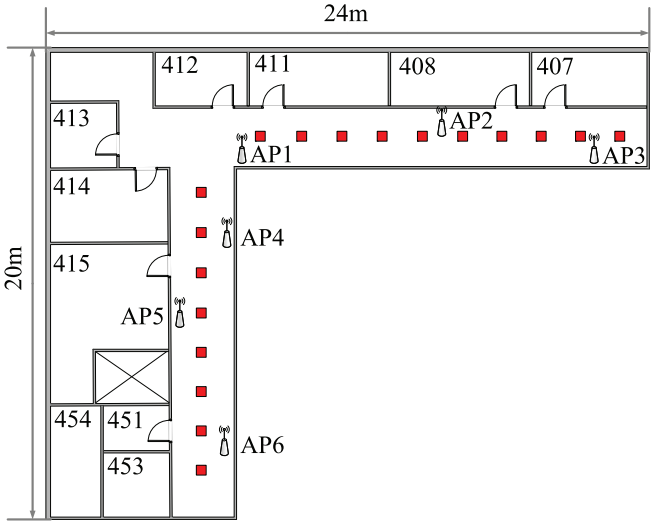


Figure 4.9: Layout of the two corridors: training positions are marked as red squares.

4.4.2 Accuracy of Location Estimation

Tables 4.1 and 4.2 show the mean and STD of localization errors, and the execution time of the four schemes for the two scenarios, respectively. In the laboratory environment, BiLoc achieves a mean error of 1.5743 m and an STD error of 0.8312 m across the 15 test points. In the corridor experiment, because only one access point is used for this larger space, BiLoc achieves a mean error of 2.1501 m and an STD error of 1.5420 m across the 10 test points. BiLoc outperforms the other three benchmark schemes with the smallest mean error, as well as with the smallest STD error, i.e., being the most stable scheme in both scenarios. We also compare the online test time of all the schemes. Due to the use of bi-modal data and the deep network, the mean executing time of BiLoc is the highest among the four schemes. However, the mean execution time is 0.6653

Table 4.1: Mean/STD error and execution time of the laboratory experiment

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean execution time (s)*
BiLoc	1.5743	0.8312	0.6653
DeepFi	2.0411	1.3804	0.3340
FIFS	2.7151	1.0805	0.2918
Horus	3.0537	1.0623	0.2849

Table 4.2: Mean/STD errors and execution time of the corridor experiment

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean execution time (s)*
BiLoc	2.1501	1.5420	0.5440
DeepFi	2.8953	2.5665	0.3707
FIFS	4.4296	3.4256	0.2535
Horus	4.8000	3.5242	0.2505

s for the laboratory case and 0.5440 s for the corridor case, which are sufficient for most indoor localization applications.

Fig. 4.10 shows the CDF of distance errors of the four methods in the laboratory scenario. In this complex propagation environment, BiLoc has 100% of the test positions with an error under 2.8 m, while DeepFi, FIFS, and Horus have about 72%, 52%, and 45% of the test positions with an error under 2.8 m, respectively. For a much smaller error of 1.5 m, the percentage of test positions having a smaller error are 60%, 45%, 15%, and 5% for BiLoc, DeepFi, FIFS, and Horus, respectively. BiLoc achieves the highest precision among the four schemes, due to the use of bi-modal CSI data (i.e., average amplitudes and estimated AOAs). In fact, when the amplitude of a signal is strongly influenced in the laboratory environment, the estimated AOA can be utilized to mitigate this effect by BiLoc. However, the other methods based solely on CSI or RSS amplitudes will be affected.

Fig. 4.11 presents the CDF of distance errors of the four schemes for the corridor scenario. Only one access point is used at one end for this 24 m long corridor, making it hard to determine the position of the mobile device. For BiLoc, more than 90% of the test positions have an error under 4 m, while DeepFi, FIFS, and Horus have about 70%, 60%, and 50% of the test positions

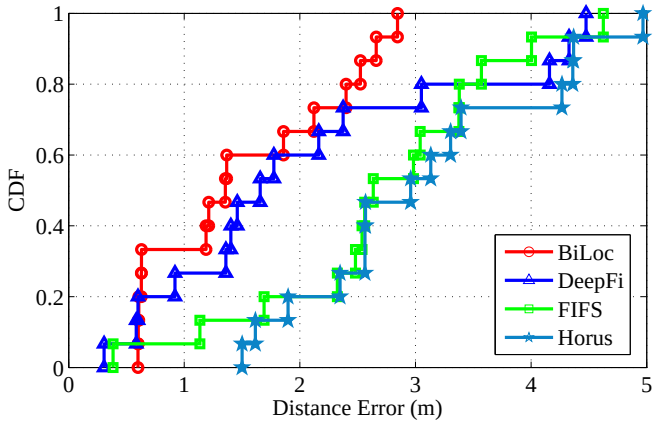


Figure 4.10: CDF of localization errors in 5GHz for the laboratory experiment.

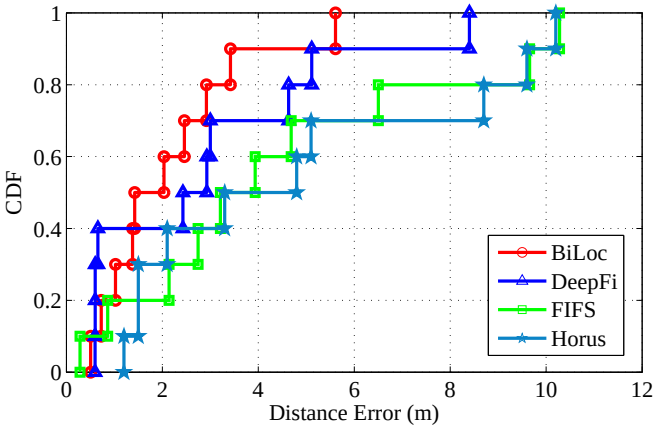


Figure 4.11: CDF of localization errors in 5GHz for the corridor experiment.

with an error under 4 m, respectively. For a tighter 2 m error threshold, BiLoc has 60% of the test positions with an error below this threshold, while it is 40% for the other three schemes. For the corridor scenario, BiLoc mainly utilizes the average amplitudes of CSI data, because the estimated AOAs are similar for all the training/test positions (recall that they are aligned along a straight line with the access point at one end). This is a challenging scenario for differentiating different test points and the BiLoc mean error is 0.5758 m higher than that of the laboratory scenario.

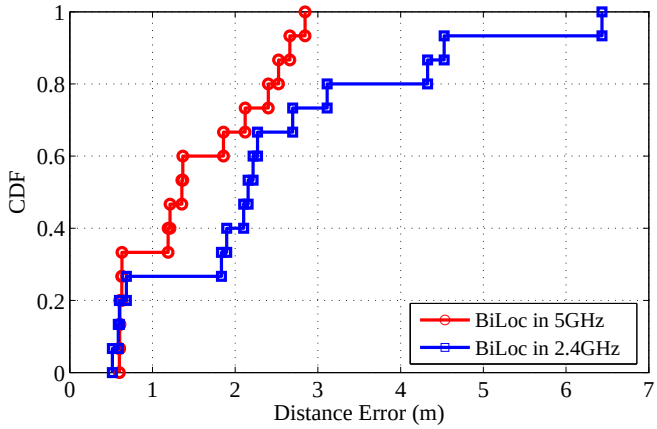


Figure 4.12: CDF of localization errors in 5GHz and 2.4GHz for the laboratory experiment.

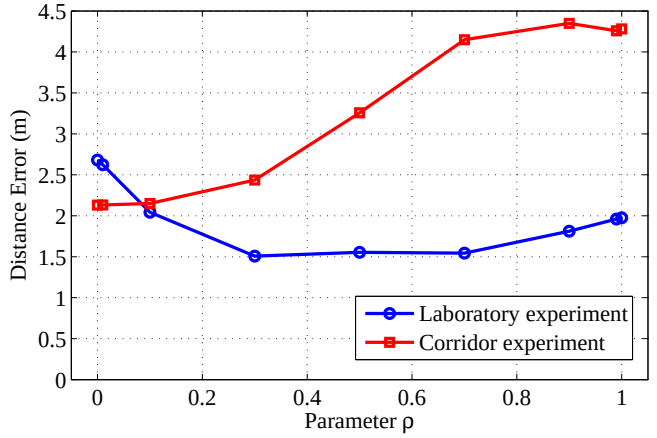


Figure 4.13: Mean localization errors versus parameter ρ for the laboratory and corridor experiments.

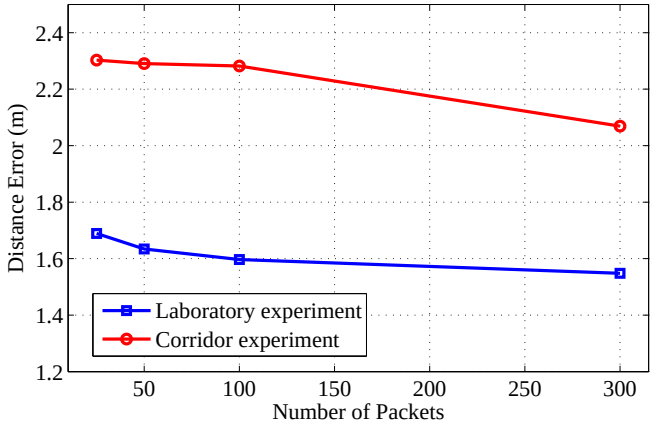


Figure 4.14: Mean localization errors versus the number of packets used in the online test stage for the laboratory and corridor experiments.

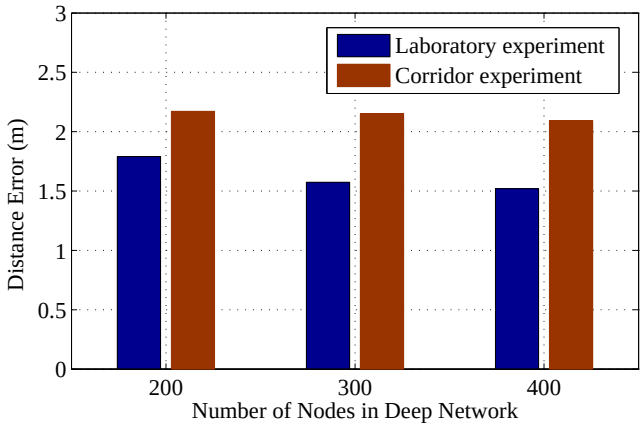


Figure 4.15: Mean localization errors versus the number of nodes in deep networks for the laboratory and corridor experiments.

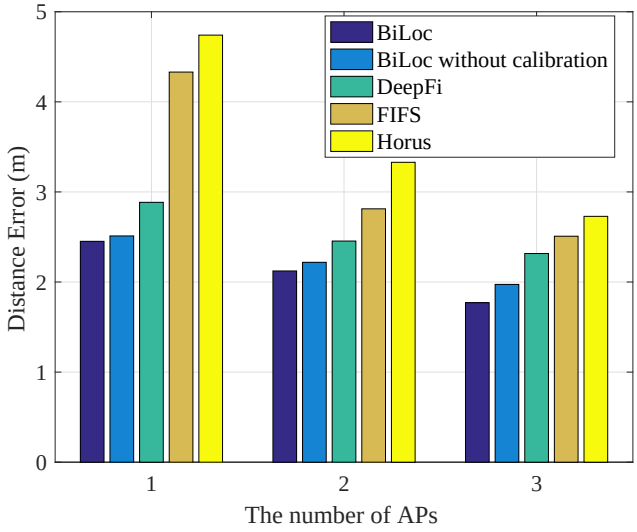


Figure 4.16: Mean localization errors versus the number of APs for two corridors experiment.

4.4.3 2.4GHz versus 5GHz

We also compare the 2.4GHz channel and 5GHz channel with the BiLoc scheme. For a fair comparison, we conduct the experiments at night, because the 2.4GHz band is much more crowded than the 5GHz band during the day.

Fig. 4.14 presents the CDF of localization errors in the 2.4GHz and 5GHz band in the laboratory environment, where both average amplitudes and estimated AOAs are effectively used by BiLoc for indoor localization. We can see that for BiLoc, about 70% of the test positions have an error under 2 m in 5GHz, while 50% of the test positions have an error under 2 m in 2.4GHz. In

addition, the maximum errors in 2.4GHz and 5GHz are 6.4 m and 2.8 m, respectively. Therefore, the proposed BiLoc scheme achieves much better performance in 5GHz than 2.4GHz. In fact, the phase difference between two antennas in 2.4GHz exhibits great variations, which lead to lower localization accuracy. This experiment also validates our Hypothesis 2.

4.4.4 Impact of Parameter ρ

Recall that the parameter ρ is used to trade off the impacts of average amplitudes and estimated AOAs in location estimation as in (4.9). We consider the impact of ρ on localization accuracy under the two environments. With BiLoc, we use bi-modal data for online testing, and ρ directly influences the likelihood probability $\Pr(v^1, v^2|l_i)$ (4.9), which in turn influences the localization accuracy.

Fig. 4.13 presents the mean localization errors for increasing ρ for the laboratory and corridor experiments. In the laboratory experiment, when ρ is increased from 0 to 0.3, the mean error decreases from 2.6 m to 1.5 m. Furthermore, the mean error remains around 1.5 m for $\rho \in [0.3, 0.7]$, and then increases from 1.5 m to 2 m when ρ is increased from 0.6 to 1. Therefore, BiLoc achieves its minimum mean error for $\rho \in [0.3, 0.7]$, indicating that both average amplitudes and estimated AOAs are useful for accurate location estimation. Moreover, BiLoc has higher localization accuracy with the mean error of 1.5m, compared with individual modality such as the estimated AOAs with that of 2.6m or the average amplitudes with that of 2.0m.

In the corridor experiment, we can see that the mean error remains around 2.1 m when ρ is increased from 0 to 0.1. When ρ is further increased from 0.1 to 1, the mean error keeps on increasing from 2.1 m to about 4.3 m. Clearly, in the corridor experiment, the estimated AOAs provide similar characteristics for deep learning, and are not useful for distinguishing the positions. Therefore BiLoc should mainly use the average amplitudes of CSI data for better accuracy. These experiments provide some useful guidelines on setting the ρ value for different indoor environments.

4.4.5 Impact of the Number of Packets

We study the impact of the number of packets used in the online test stage of BiLoc. In this experiment, we estimate the location of the mobile device using different number of packets for the two indoor environments. Although 1000 test packets are received for each position, we only use 25, 50, 100, and 300 of them in the online test for location estimation. We also randomly select the parameter ρ value to guarantee the consistency of localization results obtained with different number of packets.

In Fig. 4.14, we plot the mean localization errors for different number of packets in the corridor and laboratory experiments. We can see that the mean distance error in the laboratory experiment is lower than that in the corridor experiment for different packets. Moreover, with the increase of packets, the mean distance error for both experiments is decreased. Also, we can find that the maximum distance errors for the laboratory and corridor experiments are 1.7 m and 2.3 m, respectively, while the minimum distance errors for the laboratory and corridor experiments are 1.58 m and 2.1 m, respectively. In fact, with the increase of the packets, the decrease of mean distance error is small for both experiments. Therefore, we choose 25 packets for the test phase in the proposed BiLoc system, which can obtain a lower computational complexity and a good localization performance.

4.4.6 Impact of the Number of Nodes in Deep Network

We study the impact of the number of nodes in deep network on localization results of our BiLoc system. Although there are lots of values we can set for the parameters K_1, K_2 and K_3, the number of all nodes ($K_1 + K_2 + K_3$) in deep network is considered. In addition, we set the number of nodes as 200, 300 and 400, respectively, for the two indoor environments.

Fig. 4.15 presents the mean distance errors for increasing number of nodes in deep network for the laboratory and corridor experiments. We can see that the mean distance error is decreased with the increasing number of nodes in deep network for both experiments. It is noticed that the difference of the mean errors is small for different number of nodes, where the mean error is

from 1.8 m to 1.5 m in the laboratory experiment, and that is from 2.2 m to 2.1 m in the corridor experiment. This demonstrates that our BiLoc is robust for different number of nodes in deep network. Thus, we select the proper number of nodes with 300 that is $K_1 = 150$, $K_2 = 100$ and $K_3 = 50$, thus obtaining the lower cost.

4.4.7 Impact of the Number of APs

Finally, we study the impact of the number of APs on localization results for different methods, where we consider the two corridors deployment with six APs. For multiple APs, we consider the online localization scheme by multiplying equation (4.8) for all APs to obtain the fusion likelihood function for every location [22]. Moreover, we implement the proposed BiLoc system without calibration for indoor localization. Also, we consider the transmitter device can access the maximum three APs.

Fig. 4.16 presents the mean distance errors for increasing number of APs based on five different schemes in the two corridors environment. It is noticed that, the mean error is decreased with the increasing number of APs for all schemes. Moreover, for BiLoc system, we can see that with the increase of the number of APs from 1 to 3, the distance error is decreased from approximate 2.4 m to 1.8 m, which can improve the localization accuracy. However, we can see that compared with other traditional methods such as Horus and FIFS, the improvement for localization accuracy is small. Moreover, we can find that the proposed method with single AP can even obtain higher accuracy than other methods. Furthermore, we also find that the BiLoc system has better localization performance than BiLoc system without calibration. Thus, we consider one AP for BiLoc system based on the proposed method, which reaches the lower localization error and device cost.

4.5 Conclusions

We proposed BiLoc, a bi-modal deep learning system for fingerprinting-based indoor localization with 5GHz commodity WiFi NICs. In BiLoc, we first extracted and calibrated CSI data to obtain bi-modal CSI data, including average amplitudes and estimated AOAs, which were used in both

the offline and online stages. In the training phase, we leveraged a deep autoencoder network to train the bi-modal data, and the weights were used to represent the bi-modal fingerprints. In the test phase, a Bayesian approach based probability model was employed for estimating position with bi-model test data. We evaluated the performance of BiLoc with extensive experiments under three representative indoor environments. The experimental results validated the superior performance of BiLoc over several benchmark schemes.

Chapter 5

CiFi: Deep Convolutional Neural Networks for Indoor Localization with CSI Images

5.1 Introduction

The rapid development of mobile devices and wireless techniques has promoted location-based services for internet of things, such as indoor tracking, robot navigation in the industry, health sensing, and activity recognition [86, 87, 88, 89, 90]. These applications require accurately determining the location of a mobile device indoors. Because of the complex wireless propagation in indoor environments, due to shadow fading, multipath propagation, and blockage, indoor localization with wireless signals is a challenging problem that has attracted considerable research effects. Recently, indoor fingerprinting based on Wi-Fi signals has become a research hot-spot, which first builds a database with a large amount of Wi-Fi measurements in the offline phase, and then determines the location of a mobile device by matching the newly received Wi-Fi data with that in the database.

In this chapter, phase difference data with 5 GHz Wi-Fi is used to estimate AOA, which can be useful for indoor localization. Estimated AOA values for a given location are relatively more stable due to the stability of phase difference data. Thus AOA estimation is robust for complexity indoor environments. For example, when Wi-Fi signal is blocked by, e.g., chairs or computers, the CSI amplitudes will be strongly weakened. However, the estimated AOA remains the same when the transmitter location is not changed. Furthermore, we employ the DCNN [91] to train the AOA data from all the training locations as a supervised learning. DCNN is a powerful deep learning technique that has been successfully applied for image recognition [24], human activity

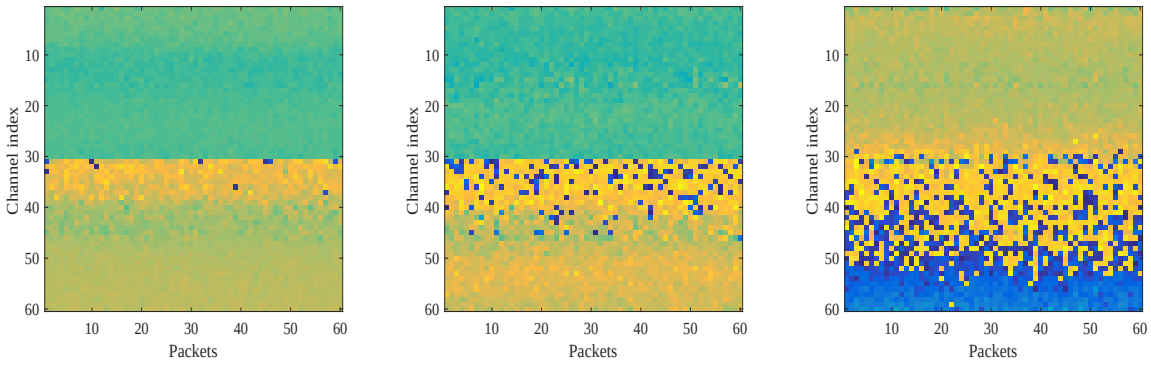
recognition based on sensors [92, 93] and social networks [94]. Specifically, we create AOA images based on a large number of received packets as input to the DCNN. The proposed method is to exploit the time-frequency feature of AOA data for improving localization performance. Moreover, since DCNN is a supervised method, *it only requires to train one group of weights for all the training data with related labels*, which is different with our prior work DeepFi that requires training weights for every training location [64, 3]. Thus, the proposed method can greatly reduce the storage requirement.

In particular, we present CiFi, a deep **C**onvolutional neural networks (DCNN) based scheme for **i**ndoor localization with commodity 5 GHz Wi**Fi**. In CiFi, we first obtain 90 CSI data from the three antennas for every received packet from the modified Intel 5300 firmware, and extract all phase information. Then, we compute two sets of CSI data, each including 30 phase differences, from antennas 1 and 2, and from antennas 2 and 3, respectively. The phase difference data is used to estimate AOA. Next, CiFi uses the estimated AOA values from 960 received packets to construct 16 images with size 60×60. These images are then used as input to the DCNN. For offline training, we use all the constructed images from all training locations to train the DCNN, which consists of a convolutional layer, a subsampling layer, and a fully-connected layer. For the convolutional layer, we obtain the feature map and extract the time-space feature for AOA images. The mean pooling function is implemented in the subsampling layer to reduce training time. We use the squared error loss function based on back propagation (BP) to train the convolutional weights. In the online stage, we propose a probabilistic method to predict the location of the mobile device based on the trained DCNN and the new CSI AOA images received from the device.

The main contributions of this chapter are summarized below.

- We theoretically and experimentally verify the feasibility of exploiting AOA values of CSI data for indoor localization. In particular, we derive a model for measured phase and analyze phase errors. We prove that phase difference is stable, and can be used to estimate AOA.

- This is also the first work to employ DCNN for indoor localization with Wi-Fi. We use estimated AOA image from CSI data as input to the DCNN. By executing four convolutional



(a) (b) (c)

Figure 5.1: CSI images for three different locations: (a) CSI image for location 1, (b) CSI image for location 2, (c) CSI image for location 3.

and subsampling layers, CiFi can automatically extract the features of the estimated AOA image, to obtain training weights with the BP algorithm. Furthermore, we implement DCNN training algorithm for CSI images. In the online phase, we present a probability method for location estimation.

- We implement the proposed CiFi system with commodity 5 GHz Wi-Fi, and verify its performance in two representative indoor environments with extensive experiments. The results show that CiFi achieves better location accuracy than three existing schemes. Moreover, we consider the impact of various system parameters on localization performance.

In the remainder of the chapter, we provide the preliminaries in Section 5.2. We present the CiFi design in Section 5.3 and performance evaluation in Section 5.4. Section 5.5 concludes this chapter.

5.2 Preliminaries

The Intel 5300 NIC provides readings from 90 subcarriers from the three antennas. Then, we compute two sets of CSI data, including 30 phase differences from antennas 1 and 2, and 30 phase differences from antennas 2 and 3. Thus, 60 estimated AOA values for each received packet can be obtained using (4.5). We take 960 packet samples for every training location, and construct 16

images with size 60 $\times$ 60 based on the estimated AOA values. Each image consists of 60 packets (x-axis) and the corresponding 60 estimated AOA values for each packet (y-axis). For example, Fig. 5.1 shows CSI images for three different locations. It is noticed that three CSI images have different data distribution, which can be used as fingerprints for indoor localization. For CiFi system, the constructed images will then be used to train the DCNN.

5.3 The CiFi System

5.3.1 CiFi System Architecture

Figure 5.2 shows the CiFi system architecture. The CiFi system uses one mobile device and one access point as Wi-Fi transmitter and receiver, respectively, both equipped with the Intel 5300 NIC. Using the packet injection technique, the transmitter and receiver are set to the injection and monitor modes, respectively. The 5 GHz band is used for improving channel stability. CiFi exploits the constructed images for two reasons. First, the estimated AOA values are highly stable and robust for each given location. When the Wi-Fi signal is blocked by a wall or chair, the CSI amplitudes will be strongly weakened, which influence the localization accuracy. However, the estimated AOA values are more robust if the transmission distance is not changed. Second, the constructed image can leverage all subcarrier information from all received packets, which contains rich time and frequency features of the CSI data.

The CiFi procedure includes two stages: offline training and online location predication. In the offline phase, the constructed images from all locations are used to train the DCNN. This method is quite different from traditional fingerprinting based methods, where a database is established for every training location, and either the measured raw data or leant features are stored as fingerprints. However, *our CiFi system only trains one group of weights for all the training locations*, which is analogous to a classification or regression problem in machine learning. This proposed method can not only decrease the amount of stored data, but also improve the robustness of the system. In the online phase, we employ an enhanced probabilistic approach for location estimation based on the constructed images of newly received CSI data.

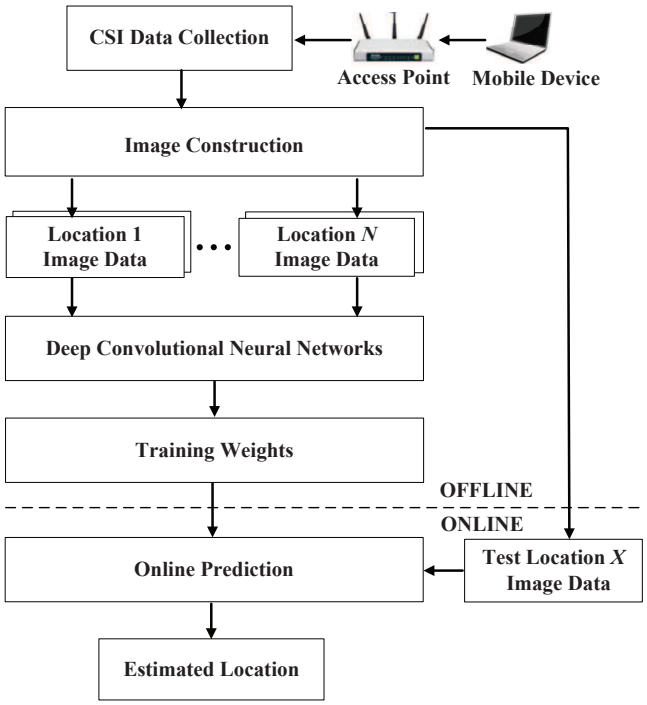


Figure 5.2: The CiFi system architecture.

5.3.2 Offline Training

The DCNN incorporates several convolutional and subsampling layers as well as one or more fully connected layers. It can exploit local correlations by sharing the same weights between neurons of adjacent layers, thus reducing the training time. DCNN can also obtain the local dependency and scale invariant feature from input data. More importantly, it can extract more abstract representation of the input image data from the lower layers to the higher layers in the hierarchical architecture of DCNN, which can strengthen the feature extraction of CSI AOA data for indoor localization. We introduce three main components of DCNN in the following.

The convolutional layer can extract feature maps within local regions in the previous layer's feature maps with linear convolutional filters followed by nonlinear activation functions. Denote θ_i^l as the ith feature map in layer l of the DCNN, which is defined as

$$\theta_i^l = \sigma\left(\sum_{m \in S_{l-1}} w_{im}^l * \theta_m^{l-1} + b_i^l\right), \tag{5.1}$$

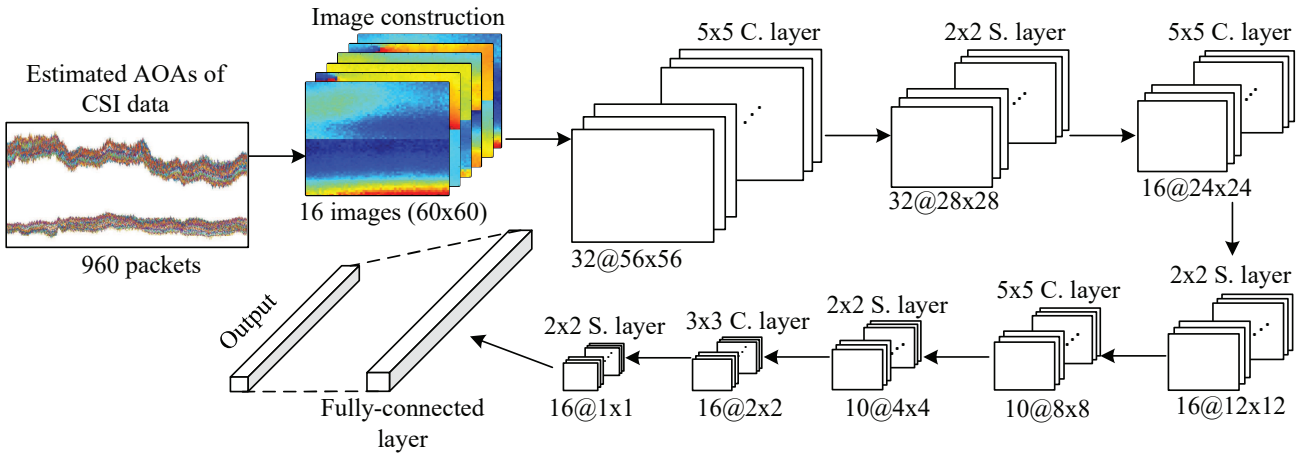


Figure 5.3: CSI data training based on deep convolutional neural networks (C. and S. are short for convolutional and subsampling, respectively).

where $\sigma(t) = \frac{1}{1+\exp(-t)}$ is the sigmoid function, b_i^l is the bias of the ith feature map in layer l, S_{i-1} is the set of feature maps in layer $(i-1)$ connected to the current feature map, w_{im}^l is the convolutional kernel to generate the ith feature map in layer l, which is the same for different m due to local weights sharing. The convolution operation can obtain the shift-invariance of input data and extract robust features. Then, the activation function $\sigma(t)$ is used to avoid obtaining trivial linear combinations of input data.

The subsampling layer or the pooling layer can reduce the resolution of the feature maps by downsampling over a local neighborhood in the feature maps of the previous layer. It is invariant to distortions on the inputs. The feature maps in the previous layer are pooled over a local temporal neighborhood by the mean pooling function, as

$$\theta_{ij}^{l+1} = \frac{1}{\left|G_j^l\right|} \sum_{k \in G_j^l} \theta_{ik}^l, \tag{5.2}$$

where G_j^l is the set of pooling region for the jth value in the ith feature map in layer l, $\left|M_j^l\right|$ is the number of elements in set G_j^l, θ_{ik}^l is the kth value of the ith feature map in layer l. Other methods such as the sum or max pooling function can be also used in this stage for reducing the training time.

For the fully-connected layer, we utilize a basic neural network with one hidden layer to train the output data after all the convolutional and subsampling layers. Moreover, the loss function is employed to measure the difference between the true location label and the output data of DCNN. By minimizing the values of the loss function with the BP algorithm, we can update the convolutional weights with the stochastic gradient descent method. In the proposed DCNN, we use the squared error loss function for training these parameters, which is defined as

$$E = \frac{1}{2K} \sum_{i=1}^{K} (y_i - o_i)^2, \tag{5.3}$$

where K is the number of training locations, y_i is the true label for the ith location, and o_i is the DCNN output for the ith location.

Fig. 5.3 illustrates CSI data training for the DCNN. To obtain the input AOA images, we first estimate AOA values from 960 received packets as in (4.5). Then, we construct 16 images with size 60×60 out of the 960 AOA values. The images are convenient for DCNN to process in its convolution and subsampling layers. For each input image in the first convolutional and subsampling layer, we employ 32 convolutional filters with size 5×5 to obtain the same number of feature maps with size 56×56, which can extract different characteristics. To reduce training data and guarantee the invariance of feature maps, the same number of feature maps with size 28×28 can be obtained by executing the subsampling with size 2×2. Then, by implementing other three convolutional and subsampling layers as in Fig. 5.3, we can obtain 16 feature maps with size 1×1, which can be fully-connected in the next layer. Finally, we can obtain the forward output results and then combine the label of training data, which can be used to update the training weights such as the convolutional filers based on the loss function with the BP algorithm.

The pseudocode for offline training of CiFi is presented in Algorithm 5 and Algorithm 6. The inputs to the Algorithm 5 are CSI images from all training locations, location labels, Max_epoch and learning rate. First, we randomly initiate all weights and biases (step 3). Then, for each epoch, we randomly select a mini-batch from CSI images from all training locations, which are passed

into DCNN defined by the network architecture (step 5). In the proposed CiFi system, the first layer and the last layer are used as the input layer and the output layer respectively. From the second layer, the input data are processed by the convolutional layer and down sampling layer in sequence (step 8-16). The outputs of the last third layer are compressed as the inputs of fully-connected layer (step 18-19). Based on the outputs of fully-connected layer and location labels, loss function is used to measure the difference between the true location label and the output data of DCNN (step 21). After the forward propagation, the errors between network outputs and labels are used as inputs of BP algorithm to train DCNN.

The pseudocode for DCNN BP Algorithm is given in Algorithm 6. For DCNN BP algorithm, we calculate the values of delta for every layer and convolutional kernel, which are used to update weights and biases. First, the errors for the output layer are calculated with the difference between outputs of neural network and labels, which are employed for obtaining the values of delta in the $L-1$ layer using step 2. Because inputs of the fully connected layer are compressed data from the previous layer in the forward propagation, its shape should be restored in DCNN BP algorithm, which is implemented by step 4. Furthermore, to obtain the values of delta for current layer, if the current layer is a sub-sampling layer, the weights of the later layer are rotated 180 degrees and convoluted with values of delta from later layer (step 10). Significantly, connected to the feature maps in the current layer, only kernels from the later layer are calculated in this step. When the current layer is a convolutional layer, the values of delta for the later layer are upsampled by Kronecker product (step 15). Then, the values of delta for the current layer are obtained by the element-wise product between the upsampled delta values and derivatives of sigmoid function (step 18). We define the values of scale as the quotient of the size of feature maps in the previous layer and the current layer. Depending on delta values for each layer, the training weights are updated (step 22-37). The learning rate α controls the speed of adjusting the weights of the DCNN. We will discuss it in the experimental section. It is noticed that the mean gradient over the mini-batch is calculated because a random mini-batch from CSI images from all training locations is passed into DCNN for each epoch.

5.3.3 Online Algorithm

In the online test phase, we adopt a probabilistic method to predict the location of the mobile device based on the trained DCNN and newly received CSI AOA images from the test location. Let M denote the number of images from one location, and o_{ij} be the prediction output of the DCNN for the ith location using the jth image. We can obtain a matrix $\boldsymbol{O}$ as the output of the DCNN for K training locations by using M images, which is given by

$$\boldsymbol{O} = \begin{bmatrix} o_{11} & o_{12} & o_{13} & \dots & o_{1M} \\ o_{21} & o_{22} & o_{23} & \dots & o_{2M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ o_{K1} & o_{K2} & o_{K3} & \dots & o_{KM} \end{bmatrix}. \tag{5.4}$$

With matrix $\boldsymbol{O}$, we propose a greedy method to select R candidate locations and compute a weighted average of these locations as the estimated location for the mobile device. We first select location indexes of the R largest outputs from the DCNN for every column of matrix $\boldsymbol{O}$, thus producing a new matrix $\boldsymbol{S}$ with size $R \times M$ as

$$\boldsymbol{S} = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1j} & \dots & s_{1M} \\ s_{21} & s_{22} & \dots & s_{2j} & \dots & s_{2M} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{R1} & s_{R2} & \dots & s_{Rj} & \dots & s_{RM} \end{bmatrix}, \tag{5.5}$$

where s_{ij} is the location index of the ith largest output for the jth image. Every element of matrix S belongs to the set of location indexes $\{1, 2, \dots, K\}$. The R largest location indexes are obtained by computing the frequencies of all location indexes in matrix $\boldsymbol{S}$. Moreover, the weight of the ith location index can be computed by averaging all the selected outputs for the ith location index, which is denoted as p_i.

Algorithm 5: Weights Training of CiFi System

Input: CSI images from all training locations, location labels, network architecture, Max_epoch, and learning rate α ;
Output: Trained weights w and b;
Randomly initialize w and b;
while $epoch < Max\_epoch$ **do**
    Randomly select a mini-batch from inputs;
    //Forward propagation;
    // L is the number of layers for DCNN;
    for $l = 2 : L - 2$ **do**
        if *the current layer is a convolution layer* **then**
            $\theta_i^l = \sigma\left(\sum_{m \in S_{l-1}} w_{im}^l * \theta_m^{l-1} + b_i^l\right)$;
        end
        else
            //The current layer is a subsampling layer layer;
            $\theta_{ij}^{l+1} = \frac{1}{|G_j^l|} \sum_{k \in G_j^l} \theta_{ik}^l$;
        end
    end
    // The last layer is a fully-connected layer;
    $v = Dense(\theta^{L-1})$;
    $o = \sigma(w^L \times v + b^L)$;
    //Loss function;
    $E = \frac{1}{2K} \sum_{i=1}^{K} (y_i - o_i)^2$;
    Call DCNN BP algorithm;
end

Finally, the position of the mobile device can be estimated as a weighted average of the R selected locations, as

$$\hat{L} = \sum_{i=1}^{R} l_i \times \frac{p_i}{\sum_{i=1}^{R} p_i}, \tag{5.6}$$

where l_i is the ith training location. In our experiment, we set $R = 2$ for better localization performance.

Algorithm 6: DCNN BP Algorithm

//Compute δ^{L-1} as the delta value of the $L-1$ layer;
$\delta^{L-1} = \left(w^L\right)^T \times (o-y) \odot (o \odot (1-o))$;
// $\odot$ is used to denote the element-wise product;
$\delta_i^L = Reshape\left(\delta^{L-1}\right)$;
//Reshape δ^{L-1} into feature map style, i is the index of feature maps in $L-1$ layer ;
for $l = L-2:2$ **do**
    if *the current layer is a subsampling layer layer* **then**
        for $i = 1:M_l$ **do**
            // M_l is the number of feature maps in the layer l ;
            $\delta_i^l = \sum_{m \in S_l} \delta_m^{l+1} * rot_{180}\left(w_{i,m}^{l+1}\right)$;
        end
    end
    else
        for $i = 1:M_l$ **do**
            $Upsampling(\delta_i^{l+1}) = \delta_i^{l+1} \otimes \varphi$;
            // φ is an all-ones matrix with the size of scale $\times$ scale;
            // $\otimes$ is denoted as the Kronecker product ;
            $\delta_i^l = Upsampling(\delta_i^{l+1}) \odot \sigma'(\theta_i^l)$;
        end
    end
end
// Update weights;
for $l = 2:L-1$ **do**
    if *the current layer is a convolution layer* **then**
        for $j = 1:M_l$ **do**
            while $i \in S_j$ **do**
                $w_{i,j}^l = w_{i,j}^l - \alpha \times (Mean_{\{mini-batch\}}(rot180(\theta_i^{l-1}) * \delta_j^l))$;
                //$Mean_{\{mini-batch\}}$ means the average of the results over mini-batch data;
            end
            $b_j^l = b_j^l - \alpha \times (Mean_{\{mini-batch\}}(\delta_j^l))$;
        end
    end
    else
        $w^l = w^l - \alpha \times Mean_{\{mini-batch\}}(E \odot (o \odot (1-o) \times v^T)$;
        $b^l = b^l - \alpha \times Mean_{\{mini-batch\}}(E \odot (o \odot (1-o)))$;
    end
end

5.4 Experimental Study

5.4.1 Experiment Configuration

We implement CiFi with 5 GHz commodity Wi-Fi devices and carry out extensive experiments to valid its performance. In particular, we utilize a desktop computer and a Dell laptop as access point

and mobile device, respectively. Both devices are equipped with an Intel 5300 NIC. The operating system is Ubuntu desktop 14.04 LTS OS. We set the PHY parameters as QPSK modulation and 1/2 coding rate for the OFDM system. We set the access point in the monitor model and the distance between its two adjacent antennas is $d = 2.68$ cm, i.e., a half wavelength for 5.58 GHz Wi-Fi on channel 116. The mobile device is set in the injection model with one antenna. Using the packet injection technique with LORCON version 1, we can extract 5 GHz CSI data at the receiver.

We compare CiFi with three representative approaches, including DeepFi [64, 3], FIFS [22], and Horus [19]. To guarantee a fair comparison, we implement these three methods with the same CSI dataset in the 5 GHz band for estimating the position of the mobile device. We experiment with the four schemes in the following two indoor environments.

Computer Laboratory: This is a 6×9 m^2 computer laboratory in the Broun Hall in the Auburn University campus. The indoor space is a cluttered environment with many desktop computers, chairs, metal tables, which block most of the LOS paths. The floor plan is shown in Fig. 5.4. We use 15 training locations (marked as red squares) and 15 test locations (marked as green dots). The access point is put at the center of the room. We set the distance between two adjacent training locations to 1.8 m, and obtain CSI data from 1000 packets for each training position and test position.

Corridor: This is a long corridor in Broun Hall with dimension about 2.4×24 m^2. As in Fig. 5.5, we place the access point at one end on the floor to measure 5 GHz CSI data. There are main LOS paths in this scenario. We use 10 training locations (red squares) and 10 test locations (green dots) along a straight line. The distance between two adjacent training locations is also 1.8 m. We extract 5 GHz CSI data from 1000 packets for each training and test location.

5.4.2 Accuracy of Location Estimation

Figure 5.6 presents the training errors over iterations of the DCNN, for the laboratory and corridor experiments. We set the threshold of training error to 0.06 to guarantee successful training and to avoid overfitting for input AOA images. Moreover, the iterations indicate the times of training

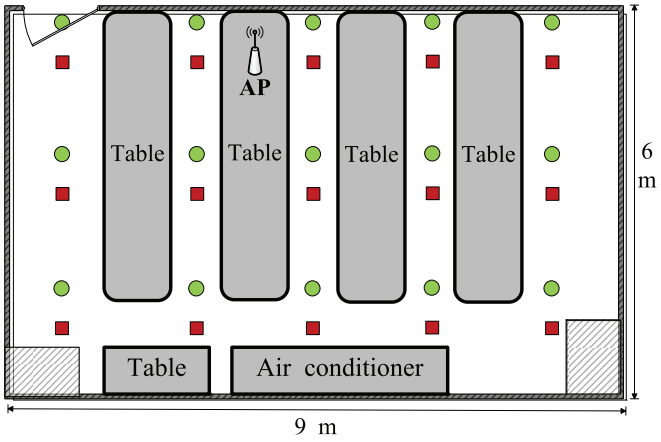


Figure 5.4: Layout of the computer laboratory: training locations are marked as red squares and testing locations are marked as green dots.

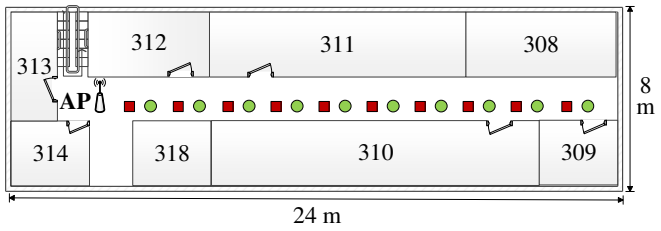


Figure 5.5: Layout of the corridor: training locations are marked as red squares and testing locations are marked as green dots.

input AOA images with the DCNN. For the laboratory experiments, the training error curve starts to converge after 1.48×10^4 iterations, which finally reaches the preset threshold with about 0.06 training error after 4.85×10^4 iterations. For the corridor experiments, the training error curve begins to converge after 3.33×10^4 iterations, which is slower and eventually reaches the preset threshold after 4.86×10^4 iterations.

Tables 5.1 and 5.2 present the mean and STD of localization errors, as well as the execution time for the four schemes in the two indoor environments. In the laboratory experiments, the proposed CiFi scheme achieves a mean error of 1.7882 and an STD error of 1.2489 m. For the corridor environment, CiFi achieves a mean error of 2.3863 m and an STD error of 1.4575 m for the 10 test locations. It is noticed that the performance of CiFi is better than the other three schemes. This is because the CiFi system utilizes the AOA estimations, which is more stable and robust in complexity indoor environments, compared to other RSS or CSI amplitude based

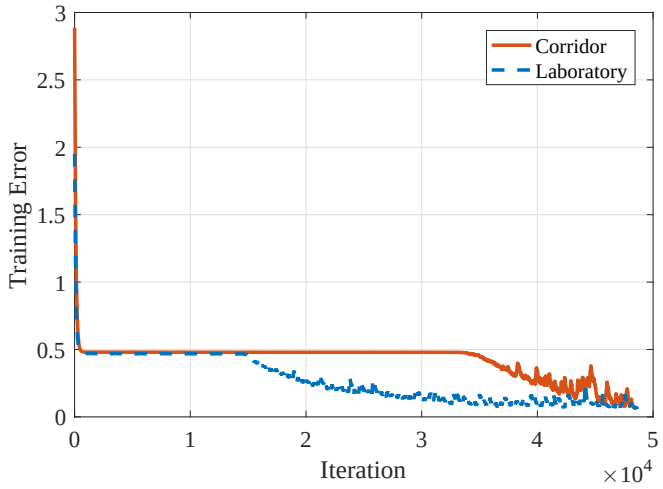


Figure 5.6: Training errors for the laboratory and corridor experiments.

Table 5.1: Localization Error And Execution Time (laboratory)

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean execution time (s)*
CiFi	1.7882	1.2489	0.5496
DeepFi	2.0411	1.3804	0.3340
FIFS	2.7151	1.0805	0.2918
Horus	3.0537	1.0623	0.2849

Table 5.2: Localization Error And Execution Time (Corridor)

Algorithm	*Mean error (m)*	*Std. dev. (m)*	*Mean execution time (s)*
CiFi	2.3863	1.4575	0.6484
DeepFi	2.8953	2.5665	0.3707
FIFS	4.4296	3.4256	0.2535
Horus	4.8000	3.5242	0.2505

methods. Since more test packets are used to construct the AOA images in the online phase, the mean execution time of CiFi is the highest among all the schemes. The mean execution time of CiFi for the computer laboratory and corridor cases are 0.5496 s and 0.6484 s, respectively, which, however, are still quite sufficient for realtime indoor localization.

Figure 5.7 presents the CDF of distance errors of four schemes in the computer laboratory case. For this environment with the complex multipaths, CiFi can utilize the unique multiple path

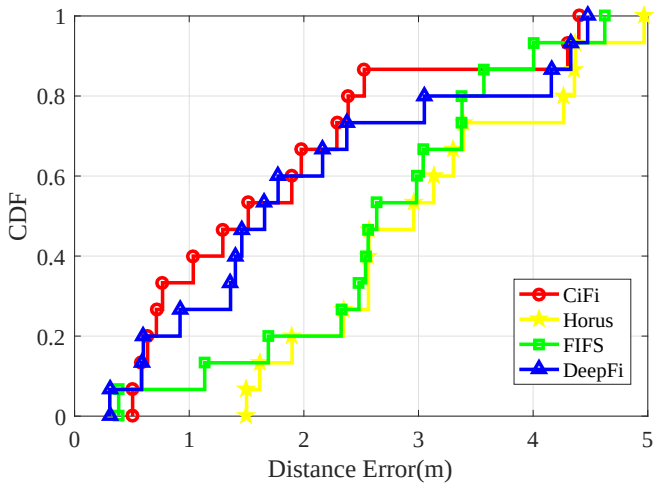


Figure 5.7: CDF of localization errors for the laboratory experiment.

feature for location estimation, which is different for different locations. CiFi has 40% of the test locations having an error less than or equal to 1 m, while that for the other schemes is 30%. We also find that about 87% of the test locations for CiFi have an error under 3 m, while the percentage of test locations having a smaller error than 3 m are 73%, 60%, and 52% for DeepFi, FIFS, and Horus, respectively. Thus, CiFi achieves the best performance in this experiment. This is because, when the magnitude of wireless signal is always influenced by obstacles such as computers in the laboratory environment, the estimated AOA values of CiFi are more robust to the indoor multipath environment, which results in smaller localization errors.

Figure 5.8 presents the CDF of localization errors of all the scheme in the corridor environment. We can see that the maximum error for CiFi is 4.8 m, while that for the other schemes is more than 8 m. This validates that the CiFi system is more robust than the other three schemes. Moreover, about 60% of the test locations for CiFi and DeepFi have an error under 3 m, while it is 40% for FIFS and Horus. This result shows that the CiFi system achieves a close localization performance to that of DeepFi, while both outperform the other two schemes. However, different from DeepFi, the proposed CiFi system does not require to build up a database for every training location, thus greatly reducing the storage requirement.

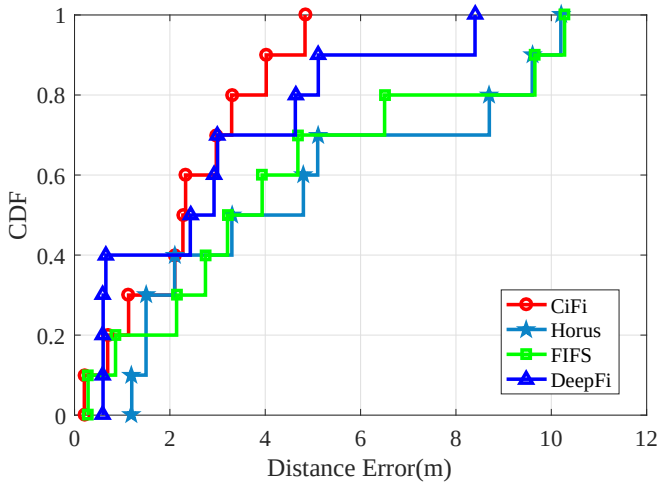


Figure 5.8: CDF of localization errors for the corridor experiment.

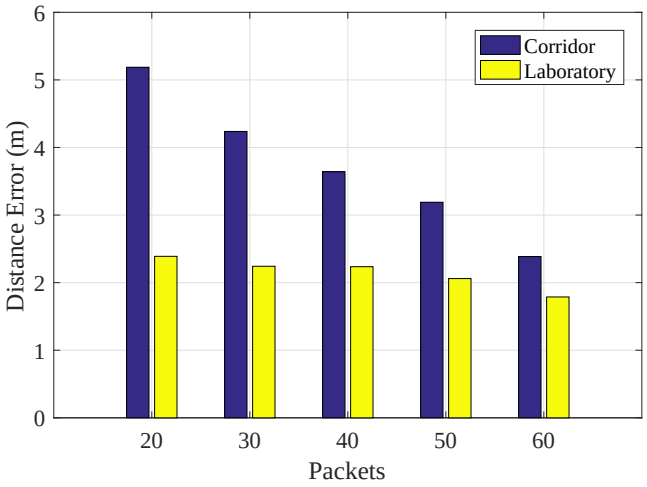


Figure 5.9: Mean localization errors for different number of packets in the laboratory and corridor experiments.

5.4.3 Impact of Various System Parameters

Impact of the number of training packets

To evaluate the effect of different training packets on indoor localization, we construct training CSI images with different numbers of packets. For each CSI packet, it contains 30 phase values extracted from 30 subcarriers of each antenna. In the proposed CiFi system, the size of an image used as inputs of DCNN is considered as 60x60, which means AoA values estimated by two antenna pairs compose y-axis and 60 packets compose x-axis. 60 packets from 2 antenna pairs could generate input image with the size of 60x60 perfectly. When the number of packets is fewer

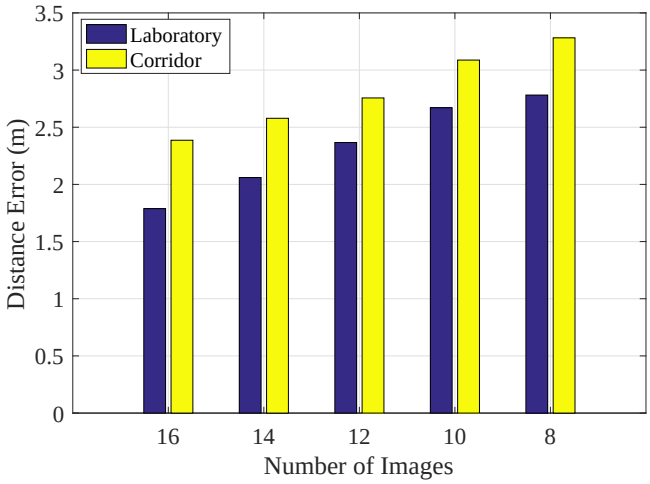


Figure 5.10: Mean localization errors for different number of images in the laboratory and corridor experiments.

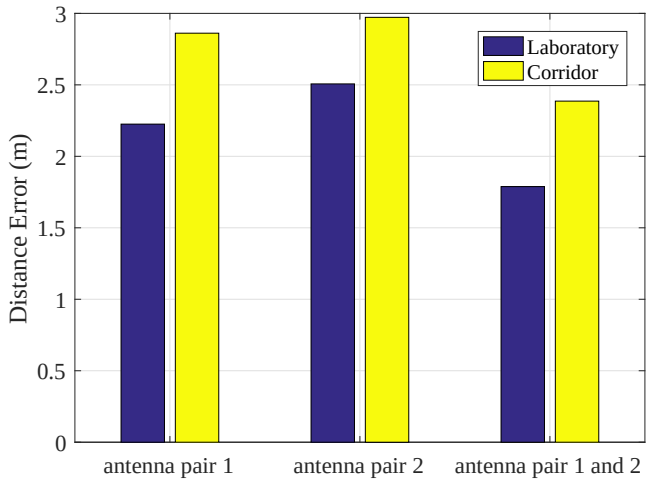


Figure 5.11: Mean localization errors for different antenna pairs in the laboratory and corridor experiments.

than 60, AoA values estimated by packets are duplicated in sequence until the image size is 60x60. For example, to generate a 60x60 input image with 40 packets, the image with a size of 60x40 is produced firstly, and then the image constructed by first 20 packets is duplicated and concatenated with previous 60x40 image. Thus, different training images contain different AoA information, even though the sizes of all images are identical.

Figure 5.9 shows mean localization errors for different number of packets in the laboratory and corridor experiments, respectively. In two indoor environments, we evaluate the performance of CiFi, respectively, with 5 training datasets that contain training images constructed by a different number of packets. As we can see the distance error will decrease with the number of packets

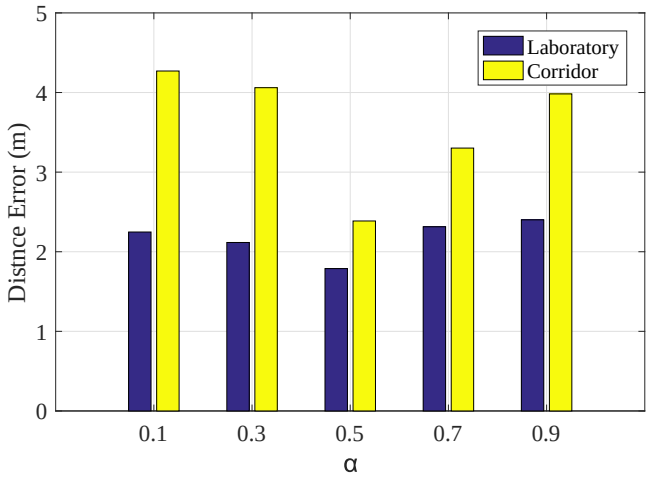


Figure 5.12: Mean localization errors for increasing α in the laboratory and corridor experiments.

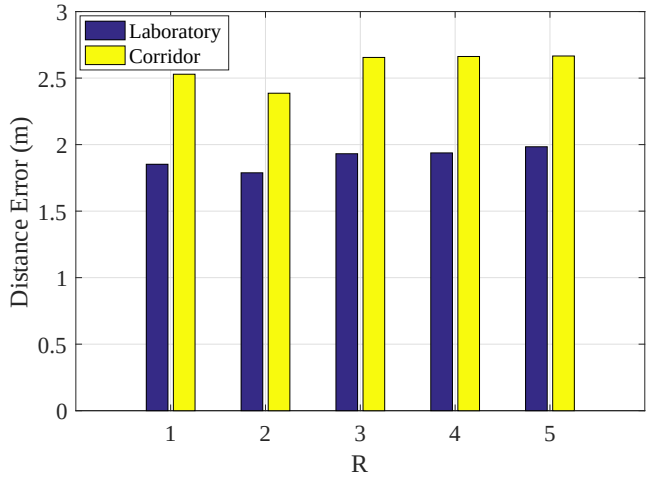


Figure 5.13: Mean localization errors for increasing R in the laboratory and corridor experiments.

increases in both scenarios. The minimum distance errors, 2.386m in the corridor and 1.788m in the lab, are reached when 60 packets are used to generate a training image. Therefore, more AoA information contributes to improving the localization precision.

Impact of the number of images per training location

To study the impact of the number of training images, we build 5 datasets, which contain different numbers of images for each training location, in two indoor environments respectively. For the

sake of fairness, images in all datasets are generated by 60 AoA values estimated from two antenna pairs. We set the packet transmit rate as 1000Hz, which guarantees that 16 images could be generated within 1 second.

Figure 5.11 illustrates mean localization errors for different number of images in the laboratory and corridor experiments, respectively. With the decrease of the number of images for one training location, the mean localization error will increase. When 16 images are generated for one training location, mean distance errors reach 1.788m and 2.386m in two indoor environments, respectively. As we can see, the highest distance errors are 2.781m and 3.282m in the lab and the corridor respectively, which are acceptable for these two scenarios. In other words, our CiFi system could not only achieve a better performance with a larger input dataset but also obtains an acceptable localization precision with a limited number of training images.

Impact of antenna pairs

Since Intel Wi-Fi Wireless Link 5300 has three antennas and the CSI data could be collected from all three antennas simultaneously, we construct three datasets to study the impact of different antenna pairs. For each packet, 30 AoA values could be estimated from 30 subcarriers of each antenna pair. Similar with the method about the impact of the number of training packets, we also construct a training image with the size of 60x60. If only one antenna pair is used to estimate AoA values, the image generated by AoA values is duplicated and concatenated together i.e., a 30x60 image is generated firstly with 60 packets collected from an antenna pair and then the image is duplicated and concatenated with itself to generate a 60x60 image.

Figure 5.10 presents mean localization errors for different antenna pairs in the laboratory and corridor experiments, respectively. It is obvious that the best localization precisions with 2.386m in the corridor and 1.788m in the lab are reached, respectively, when CSI AoA values from all antennas are leveraged to construct training images. Furthermore, we notice that our CiFi system performs well even input images are produced by one antenna pair. The highest distance errors in

the lab and corridor are 2.507m and 2.972m, respectively, where antenna 2 and 3 are leveraged to generate training images.

Impact of different learning rate α

To study the effect of different learning rate, we design a specific experiment by setting different learning rates to evaluate their effect on localization precision. In the experiment, the number of epoch is set in 1200 to guarantee the fairness.

Figure 5.12 illustrates mean localization errors for increasing α in the laboratory and corridor experiments, respectively. As the learning rate increases from 0.1 to 0.5, the minimum distance errors for two scenarios are obtained when the learning rate is 0.5. After that, the mean distance error goes up with the increase of the learning rate. Basically, a low learning rate induces that DCNN could not achieve the convergence within 1200 epochs. However, for a higher leaning rate such as 0.7 or 0.9, DCNN could not reach the best convergence point because the BP algorithm hops back and forth over the valley repeatedly. For our CiFi system, training time does not jeopardize user experience in the offline stage. Thus, in order to reach the lowest distance error, the learning rate is set as 0.5 for two scenarios.

Impact of different value of R

In the proposed CiFi system, we propose a greedy method to select R candidate locations to compute a weighted average of these locations as the estimated location. In our experience, we find that most of correct location predictions are always included in the top five outputs of DCNN. Thus, to improve the localization precision, only the top five outputs are leveraged to calculate location estimation in the proposed CiFi system.

Figure 5.13 shows mean localization errors for increasing R in the laboratory and corridor experiments, respectively. As we can see, when the value of R is 2, the distance errors reach the lowest values for two scenarios. Thus, we set R as 2 in the CiFi system. Furthermore, with the

value of R increases, the mean distance error rises slightly, which means that our CiFi system is robust to different values of R.

5.5 Conclusions

In this chapter, we proposed CiFi, a DCNN based fingerprinting system for indoor localization with 5 GHz Wi-Fi. We theoretically and experimentally verified the feasibility of using AOA values for indoor localization. We then presented the CiFi system, which first formed AOA images to train the DCNN, and then used newly received AOA images to estimate the location of the mobile device. Through extensive experiments, we demonstrated the superior performance of the proposed CiFi system.

Chapter 6

ResLoc: Deep residual sharing learning for indoor localization with CSI tensors

6.1 Introduction

With the remarkable development in mobile devices and wireless techniques [95, 96, 97, 98, 99], location-based services for internet of things, like activity recognition and health sensing, has been enhanced significantly [100, 54, 75]. To fulfill the requirement of these applications, a high precision location information is indispensable. Considering the fact that the wireless propagation is much more complex in the indoor environment, indoor localization with wireless signals faces lots of unsolved issues, which draw so much attention from researchers. Lately, fingerprinting-based indoor localization has become a research hot-spot, which builds a database with a large amount of Wi-Fi measurements in the offline phase, and then computes the position of a mobile device by comparing the newly received Wi-Fi data with that in the database.

Recently, several 802.11n measurement and experimentation tools are released, such as Intel Wi-Fi Link 5300 NIC [21] and the Atheros AR9580 chipset [81], which can extract CSI from received packets by the modified firmware. Comparing with RSS, CSI is the fine-grained channel information, including subcarrier-level channel measurements in OFDM systems. Moreover, CSI is much more stable than RSS for a given location [23]. Based on the CSI information, several fingerprinting systems exhibit better localization performance. FIFS exploits the weighted average of CSI amplitudes over three antennas to achieve fine-grained localization [22]. In addition, CSI amplitudes and calibrated phases information are leveraged by DeepFi [3] and PhaseFi [85], respectively. These two schemes collect CSI from all the subcarriers at all the three antennas and

generate fingerprinting with deep autoencoder networks. Moreover, to improve the localization accuracy, BiLoc system is proposed based on average CSI amplitude and phase difference information for indoor localization by using bimodal deep autoencoder network [101]. Although these three localization systems based on deep network can obtain better localization performance, they need to build a database to store training feature as fingerprints for every training location, which increases the training time and storage space.

In this chapter, we consider bimodal CSI tensor data including estimated AOA and amplitude information that are obtained from the 5GHz band. Firstly, AOA and amplitude information are stable, which can be effective features for fingerprinting based indoor localization. Moreover, AOA and amplitude information are complementary to each other under different indoor environments. For example, when LOS component for wireless signal is weaker than other AOAs, the amplitude information is useful for improving the localization performance. On the other hand, once the signal is blocked by objects such as wall, the estimated AOA values will help to strengthen the localization accuracy because the amplitude information is greatly weakened. Moreover, we present a new deep residual sharing learning for improving the training capacity with two channels CSI tensor data. The proposed method is different from the original deep residual unit without sharing the residual function. Moreover, we can stack many residual blocks for adding the depth of the deep network, thus achieving higher learning and representation ability. The residual learning method has been successfully applied for image recognition [24, 10, 102]. The proposed method only requires for training one group of weights in deep residual network for all training locations as a classification problem in statistical learning, thus significantly reducing the amount of the stored data.

In particular, we present ResLoc, a deep **Res**idual sharing learning for indoor **Loc**alization with CSI Tensor. In ResLoc, we first construct a CSI tensor including three images, each of which has the same size with 30×30 based on the estimated AoA values and the CSI amplitude values. For CSI tensor, we consider two images from estimated AoA values between antenna 1 and 2, and antenna 2 and 3. Another image is from the amplitude values from one antenna. Thus, by using

990 packets, we can obtain 33 CSI tensor data for one training location. Moreover, we consider two channels CSI tensor data, where the difference between two CSI tensor data is that they have different amplitude information from different antennas for creating the third image in CSI tensor. In ResLoc system, we consider the amplitude information from antenna 1 and antenna 2 for two channel CSI tensor data. For offline training phase, all the constructed two channels CSI tensor data from all training locations are leveraged to train the weights of the deep network based on the proposed deep residual sharing learning, which includes the input block, the residual block and the output block. The new idea for the residual block is that the proposed scheme shares the residual functions for two channels, which can effectively exploit the CSI tensor data. Moreover, we also analyze the proposed deep residual sharing learning for forwarding and backward propagation. For the online stage, we use newly received CSI tensor data to compute the location of the mobile device based on the probabilistic method.

The main contributions of this chapter are summarized below.

- This is the first work to use CSI tensor data for indoor localization, which can exploit the rich frequency and time features of the CSI data including the amplitude and phase difference information.

- We propose deep residual sharing learning for training two channels CSI tensor data. Moreover, we can stack many residual sharing blocks for adding the depth of the deep network, thus achieving higher learning and representation ability for CSI tensor data. Moreover, the proposed scheme is analyzed for forwarding and backward propagation. In the online test, we consider a probability method for location prediction.

- We implement the proposed ResLoc system with commodity 5 GHz Wi-Fi in two representative indoor environments with extensive experiments. The results show that ResLoc achieves decimeter level location accuracy, which is better than other deep learning methods.

6.2 Layout

In the remainder of the chapter, we introduce the preliminaries and CSI tensor in Chapter 6.3. We design the ResLoc design in Chapter 6.4 and performance evaluation in Chapter 6.5. Chapter 6.6 summaries this chapter.

6.3 CSI Tensor

To build CSI tensor as the input of the ResLoc system, we compute bimodal CSI data including estimated AOAs and amplitude information that are obtained from the 5GHz band. Besides the amplitude information from three antennas, it is also easy to estimate the corresponding AoA values between two adjacent antennas from each subcarrier and each received packet by using the same method in BiLoc system [101]. Then, for every 30 packets(x-axis) with 30 subcarriers (y-axis), we can construct a CSI tensor including three images, each of which has the same size with 30×30 based on the estimated AoA values and the CSI amplitude values. For CSI tensor, we consider two images from estimated AoA values between antenna 1 and 2, and antenna 2 and 3; Another image is formed from extracted amplitude values from one antenna. Thus, by using 1500 packets, we can obtain 50 CSI tensor data for one training location or one test location. For ResLoc system, we consider two channels CSI tensor data, where the difference between two CSI tensor data is that they have different amplitude information from different antennas for creating the third image in CSI tensor. In ResLoc system, we consider the amplitude information from antenna 1 and antenna 2 for two channel CSI tensor data.

There are three reasons for using the CSI tensor as the input of ResLoc. First, by using CSI tensor with three dimensional data it can strengthen the performance of deep network for classification problem with indoor localization. Moreover, all subcarrier information from all packet sample are exploited by three images in CSI tensor, which contains rich frequency and time features of the CSI data. We can thus extract more effective features from CSI tensor. Third, we leverage bimodal

CSI data including the estimated AoA values and the CSI amplitude values for indoor localization, which are complementary to each other under different indoor environments [4].

6.4 The ResLoc System

6.4.1 ResLoc System Architecture

In Figure 6.1, the ResLoc system is composed by one transmitter, which is a mobile device, and one receiver, which is an access point. Both devices are equipped with the Intel 5300 NIC. To collect the CSI data, the transmitter is set to the injection mode, and the receiver works in the monitor mode. The Intel 5300 NIC reports CSI from 30 groups of subcarriers from each antenna. After CSI data collection, we can build two channels CSI tensor based on estimated AOAs and amplitude information. ResLoc system employs the fingerprinting based method, which includes the offline training and online location prediction. For training data with two channels CSI tensor in the offline phase, we propose a deep residual sharing learning for obtaining the optimal weights of deep residual network. For online location prediction, we utilize newly received CSI tensor data to compute the location of the mobile device based on an enhanced probabilistic approach. Our ResLoc system is totally different from traditional fingerprinting based methods, which build the database for every training location based on raw data or training features as the fingerprints. In fact, ResLoc system only requires for training one group of weights in deep residual network for all training locations like the regression or classification problem in statistical learning. Apparently, this method reduces the amount of the stored data significantly. On the other hand, it also contributes to the improvement of the robustness for indoor localization based on the proposed deep residual sharing learning, which can effectively represent the features of CSI data.

6.4.2 Offline Training

We propose a deep residual shared learning for training the deep network with bi-modal CSI tensor, which includes the input block, the residual block, and the output block in Fig. 6.2.

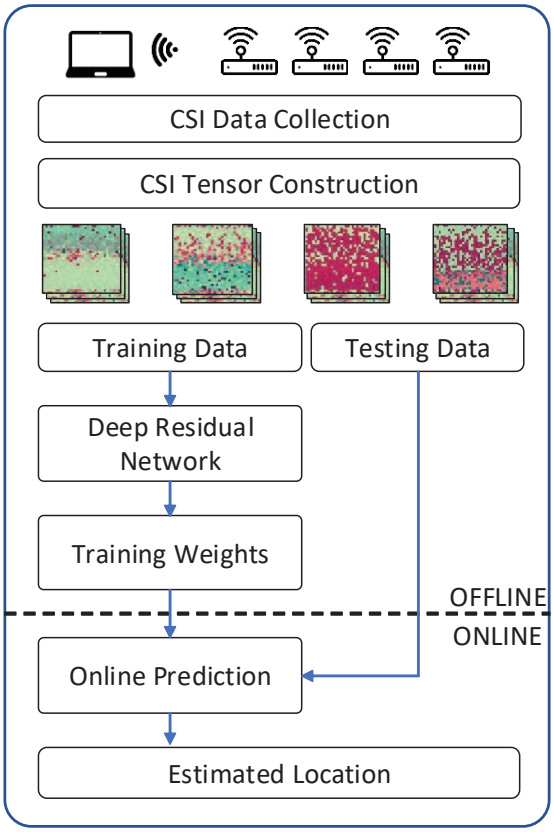


Figure 6.1: The ResLoc system architecture.

Input Block

For the input block, the bi-modal CSI tensor data can be trained by four different layers with the Convolution2D layer, batch normalization layer, activation layer and max pooling layer, respectively. It can obtain the local dependency and scale invariant feature from bi-modal CSI tensor. Furthermore, Input block can exploit more abstract representation of the input CSI tensor data from the lower layers to the higher layers, which can improve the feature extraction of CSI tensor data for indoor localization. We discuss four different layers for the input block.

The Convolution2D layer is to obtain feature maps within local regions in input CSI tensor or the previous layer's feature maps with several convolution kernels. In fact, each data of a feature map is connected with the local data in the previous layer. Moreover, by using different convolution kernels we can obtain all produced feature maps. Let θ_i^l denote as the ith feature map in layer l,

which is defined as

$$\theta_i^l = \sum_{m \in S_{l-1}} w_{im}^l * \theta_m^{l-1} + b_i^l, \tag{6.1}$$

where w_{im}^l is the convolutional kernel to generate the ith feature map in layer l, b_i^l is the bias of the ith feature map in layer l, S_{l-1} is the set of feature maps in layer $(l-1)$ connected to the current feature map, which is the same for different m due to local weights sharing. The convolution operation with weights sharing scheme can improve the efficiency for training deep network.

The batch normalization layer can adjust the input distribution for different layers and thus alleviate the problem of Internal Covariance Shift that is as the data flow propagates for different layers in deep network, the distribution of input will be shifted, thus reducing the learning capacity[103]. In batch normalization layer, the input data are normalized such that it can satisfy a zero mean and a unit standard deviation, where the estimation of mean and variance are obtained by each mini-batch. Then, to improve the representation ability in deep network, the normalized data is shifted and scaled. Thus, the batch normalization for the k_{th} input data x_k is formulated by

$$y_k = \gamma \frac{x_k - u_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{6.2}$$

where u_B and σ_B^2 are the the mean and variance of mini-batch, respectively, ϵ is the small constant value to avoid numerical problems in batch normalization, γ and β are the scaled and shifted parameters, which are learned from training. By using batch normalization, it can instead of Dropout for avoiding overfitting in training.

The activation layer can be employed to avoid obtaining trivial linear combinations of input data, which can detect nonlinear features. Traditional nonlinear activation functions mainly exploit sigmoid $\sigma(x) = \frac{1}{1+\exp^(-x)}$ and tanh $\tanh(x) = 2\sigma(x) - 1$ functions in neural networks. In ResLoc system, we leverage rectified linear unit (ReLU) as the activation function with the expression $ReLu(x) = max(x, 0)$, which can stay the positive part and suppress the total negative part

to zero [104]. For training in deep convolution neural networks, ReLU function has faster training than that for traditional sigmoid and tanh functions. Moreover, it can also exploit the sparse representations in the hidden units and can have effectively training without pre-training.

The max pooling layer can reduce the resolution of the feature maps by downsampling over a local neighborhood in the feature maps of the previous layer. It is invariant to distortions and small shifts on the inputs. Moreover, it also improves the robustness of the deep network. The feature maps in the previous layer are pooled over a local temporal neighborhood by the max pooling function, as

$$\theta_{ij}^{l+1} = \max_{k \in G_j^l} \theta_{ik}^{l}, \tag{6.3}$$

where G_j^l is the set of pooling region for the jth value in the ith feature map in layer l, θ_{ik}^l is the kth value of the ith feature map in layer l. Other methods such as the mean or sum pooling function can be also used in this stage for reducing the training time.

Residual Block

For the residual block, we propose a new deep residual sharing learning for improving the training capacity with two channels CSI tensor data. The proposed method is different from the original deep residual unit without sharing the residual function. Moreover, we can stack many residual blocks for adding the depth of the deep network, thus achieving higher learning and representation ability. For residual learning [10, 102], the idea is that instead of learning the underlying mapping $H(x)$ by using a few stacked layers, we can learn the residual function $F(x) = H(x) - x$. Thus, the original mapping can become $F(x) + x$, where x is implemented by identity mapping with the shortcut connection. Thus, it is easy for training very deep network by using residual learning. Moreover, we implement the proposed deep residual sharing learning by sharing the residual functions for two channels input data in Figure 6.2. On the other hand, the residual function includes two layers convolution operations, each of which includes the batch normalization layer, the activation layer, and the convolution2D layers. They are implemented as the same as the input block.

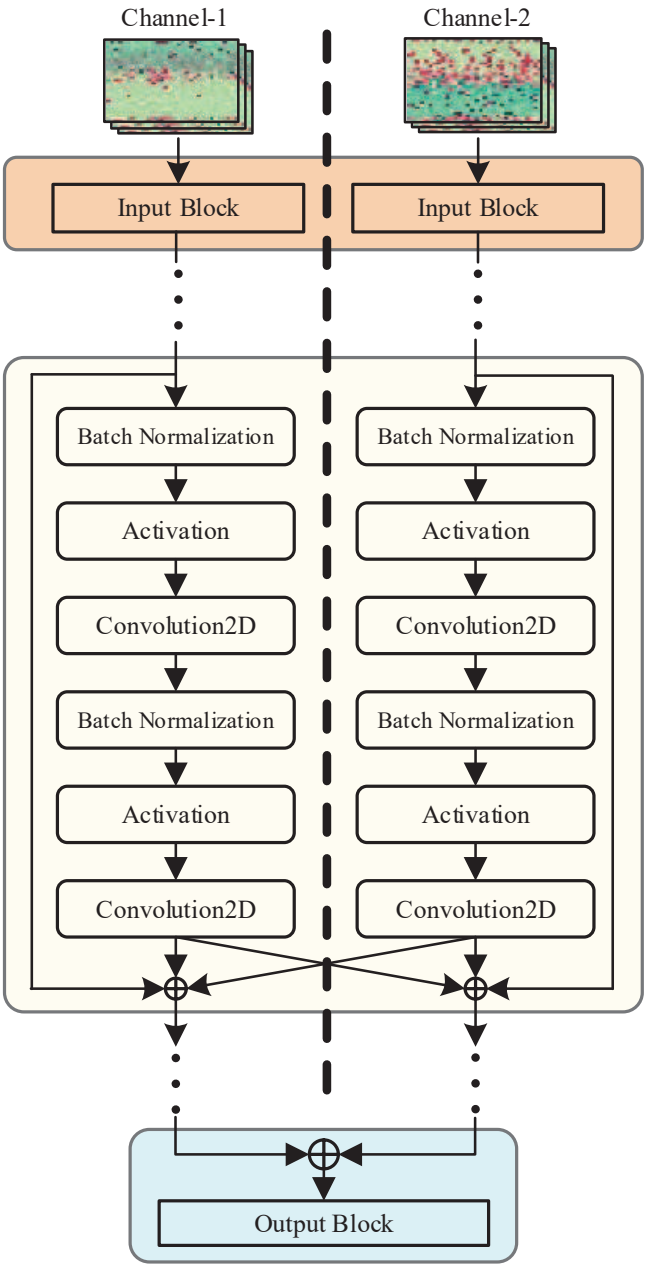


Figure 6.2: Deep residual sharing learning for offline training

For analyzing deep residual sharing learning for forwarding and backward propagation, we denote the x_k^1 and x_k^2 as the input data with channel 1 and channel 2 for the k_{th} residual block, respectively. Let R denote the residual function with two 3×3 convolution layers. Based on Figure 6.2, we have $x_{k+1}^1 = x_k^1 + R(x_k^1) + R(x_k^2)$ and $x_{k+1}^2 = x_k^2 + R(x_k^2) + R(x_k^1)$ for the k_{th} residual block at channel 1 and channel 2, respectively. Thus, we can recursively obtain the x_K^1

and x_K^2 for the K_{th} residual block at channel 1 and channel 2, which is formulated by

$$\begin{aligned} x_K^1 &= x_k^1 + \sum_{i=k}^{K-1} R(x_i^1) + \sum_{i=k}^{K-1} R(x_i^2) \\ x_K^2 &= x_k^2 + \sum_{i=k}^{K-1} R(x_i^2) + \sum_{i=k}^{K-1} R(x_i^1) \end{aligned} \tag{6.4}$$

Based on the above equations about forward propagation, we can find that the output x_K^1 and x_K^2 shares the same residual function, which can be represented by the summation of preceding residual functions adding input x_i^1 or x_i^2, respectively. This reduces the error of the gradient propagation. Moreover, it is easier to train the sharing residual functions, which are pushed into zeros when the identity mapping are optimal. On the other hand, we consider the loss function as L for the backward propagation. Based on the chain rule of backpropagation, we can obtain:

$$\begin{aligned} \frac{\partial L}{\partial x_k^1} &= \frac{\partial L}{\partial x_K^1}(1 + \frac{\partial}{\partial x_k^1}(\sum_{i=k}^{K-1}(R(x_i^1) + R(x_i^2)))) \\ \frac{\partial L}{\partial x_k^2} &= \frac{\partial L}{\partial x_K^2}(1 + \frac{\partial}{\partial x_k^2}(\sum_{i=k}^{K-1}(R(x_i^2) + R(x_i^1)))) \end{aligned} \tag{6.5}$$

By the the above equations about backward propagation, we can see that the gradients $\frac{\partial L}{\partial x_K^1}$ and $\frac{\partial L}{\partial x_K^2}$ are directly propagated back to the any shallower input x_k^1 and x_k^2, respectively. Moreover, because the gradients for the sharing residual functions $\frac{\partial}{\partial x_k^1}(\sum_{i=k}^{K-1}(R(x_i^1) + R(x_i^2)))$ and $\frac{\partial}{\partial x_k^2}(\sum_{i=k}^{K-1}(R(x_i^2) + R(x_i^1)))$ are not always -1, the gradients $\frac{\partial L}{\partial x_K^1}$ and $\frac{\partial L}{\partial x_K^2}$ cannot be canceled for the mini-batch with SGD to avoid the problem of the vanishing of gradient. Thus, the proposed deep residual sharing learning can increase the learning capacity and leverage two channels CSI tensor data.

Output Block

For the output block, we first merge two channel data into single channel. Then, we implement basic data operations for the merged data including batch normalization, activation with ReLU, and max pooling. Moreover, the main operation in the output block is the fully-connected layer, which employs a basic neural network with one hidden layer to train the output data based on softmax classifier. We consider the input data for the softmax function as the R dimensional vector

$z = [z_1, z_2, ..., z_R]$, where R is the number of clusters. Then, the softmax function can map the R dimensional vector to the normalized data $p = [p_1, p_2, ..., p_R]$, that is

$$p_i = \frac{e^{z_i}}{\sum_{r=1}^{R} e^{z_r}} \text{for } i = 1, 2, ...R. \tag{6.6}$$

In addition, we define the loss function as the cross-entropy to measure the difference between the output normalized data and the true label data, that is

$$E = -\sum_{r=1}^{R} y(r) log(p_r), \tag{6.7}$$

where $y(r)$ means the true label data for the r_{th} location. Then, we can train the parameters in deep network with the stochastic gradient descent method by minimizing the values of the loss function.

Weight Training with Deep Learning

The pseudocode for offline training with two input tensors is presented in Algorithm 7 and Algorithm 8. The inputs to the algorithm 7 are two bimodal CSI tensors, For one input tensor, it includes two phase difference slices and an amplitude slice. Each of slices has the same size with 30 times 30 based on the estimated AoA values and the CSI amplitude values for every 30 packets(x-axis) with 30 subcarriers (y-axis). The input datasets are spit into mini batches to train the network. First, batches are processed by the input block, which consists of a convolution layer, a batch normalization layer, an activation layer and a pooling layer. To obtain the output of the input block, batches are dealt by the layers sequentially (lines 10-18). Because of our two-channel framework, two input tensors pass two channels parallel based on tensorflow. The outputs of the input block are processed by residual blocks.Then the outputs of residual blocks are delivered into the output block. Similarly, the output block consists of a batch normalization layer, an activation layer, a convolution layer and a pooling layer as well. What the difference in the output block is two special layers, a merge layer and a fully-connected layer. Before the input tensors pass the merge layer, they are passed into two-channel framework parallelly. Namely, two inputs of the output

block are dealt by the batch normalization layer, the activation layer and the convolution layer parallel (line22-29). After the two-channel framework merges together at the merge layer, the output of the merge layer is processed by a batch normalization layer, an activation layer, a pooling layer and a fully connected layer sequentially (line 32-45). Once the output of the fully-connected layer is obtained, we could compute cross entropy between the prediction result of the network and the desired labels. Then, the weights and biases are updated using the error with back-propagation algorithm. Finally, we need to update all batches, which is implemented for 50 epochs in the offline training algorithm.

The pseudocode for residual blocks is given in Algorithm 8. The inputs to the algorithm are the number of repetitions for residual blocks $K$and two output tensors of input blocks I_1 and I_2. The repetition defines the number of residual blocks that are stacked to form a residual part. Typically, the repetition is a one-dimensional array, whose length is the number of residual blocks with different size convolution layers. And the element of the array defines the number of residual blocks with same size convolution layers. The total amount of residual blocks is defined by the sum of elements in the repetition vector. The basic residual block is composed by a two-channel framework, which includes two convolution layers, a batch normalization layer and an activation layer in each channel. The stacking sequence of these layers is as shown (line10-17). It is noteworthy that there is a sharing layer at the end of residual block. The output of previous layer and the residual output of the blocks are sum up in the sharing layer as the new output and the new residual output of the block.

6.4.3 Test Phase

For the test phase, a probabilistic method is leveraged to estimate the position of the mobile device by using the newly received CSI tensor data from the test points based on the trained deep network. Let T denote the number of CSI tensor from one position, and p_{ij} denote as the output result of the deep network with the jth CSI tensor for the ith location. The matrix $\boldsymbol{P}$ as the prediction output of the deep network with T CSI tensor data for R training locations, that is

Algorithm 7: Weights Training

Input: Input tensor dataset T_1 and input tensor dataset T_2, number of repetitions for residual blocks K ;
Output: : Trained weight W, b;
Divide input datasets T_1 and T_1 into a batches that contains q CSI tensors;
c denotes as channel index;
while $epoch < 50$ **do**
    for $d = 1 : a$ **do**
        $\theta_1 = M_1^a$;
        $\theta_2 = M_2^a$;
        //M denotes as CSI tensor batch;
        for $c = 1 : 2$ **do**
            $\theta_c = Convolution(\theta_c)$;
            //Calculate outputs of the convolution layer
            $\theta_c = \gamma_c \frac{\theta_c - u_{Bc}}{\sqrt{\sigma_{Bc}^2 + \epsilon_c}} + \beta_c$;
            //Calculate outputs of the batch normalization layer
            $\theta_c = ReLU(\theta_c)$;
            //Calculate outputs of the activation layer
            $\theta_c = pool(\theta_c)$;
            //Calculate outputs of the pooling layer
        end
        do Residual blocks;
        for $c = 1 : 2$ **do**
            $\theta_c = X_c$;
            //X_c is the output of the residual block
            $\theta_c = \gamma_c \frac{\theta_c - u_{Bc}}{\sqrt{\sigma_{Bc}^2 + \epsilon_c}} + \beta_c$;
            //Calculate outputs of the batch normalization layer
            $\theta_c = ReLU(\theta_c)$;
            //Calculate outputs of the activation layer
            $\theta_c = Convolution(\theta_c)$;
            //Calculate outputs of the convolution layer
        end
        $S = \theta_1 + \theta_2$; //Calculate outputs of the merge layer
        $S = \gamma \frac{S - u_{Bs}}{\sqrt{\sigma_{Bs}^2 + \epsilon_s}} + \beta_s$; //Calculate outputs of the batch normalization layer
        $S = ReLU(S)$; //Calculate outputs of the activation layer
        $S = pool(S)$; //Calculate outputs of the pooling layer
        $q = softmax(W * flattened(S) + b)$;
        //Fully-connected Layer
        Loss function $L = -\sum_r y_r \log(q_r)$;
        Update weights and bias using the error with back-propagation;
    end
end

Algorithm 8: Pseudocode for residual blocks

Input: two outputs of input blocks, I_1 and I_2, and number of repetitions for residual blocks K;
Output: : two outputs of residual blocks,X_1 and X_2;
c denotes as channel index;
for $k = 1 : K$ **do**
    if $k == 1$ **then**
        $X_1 = I_1$;
        $X_2 = I_2$;
    end
    for $c = 1 : 2$ **do**
        $\theta_c = Convolution(X_c)$;
        //Calculate outputs of the convolution layer
        $\theta_c = \gamma_c \frac{\theta_c - u_{Bc}}{\sqrt{\sigma_{Bc}^2 + \epsilon_c}} + \beta_c$;
        //Calculate outputs of the batch normalization layer
        $\theta_c = ReLU(\theta_c)$;
        //Calculate outputs of the activation layer
        $\theta_c = Convolution(\theta_c)$;
        //Calculate outputs of the convolution layer
    end
    $X_1 = \theta_1 + \theta_2 + X_1$;
    $X_2 = \theta_1 + \theta_2 + X_2$;
    //Calculate outputs of residual blocks
end

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1T} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{R1} & p_{R2} & p_{R3} & \cdots & p_{RT} \end{bmatrix}. \tag{6.8}$$

To reduce the variance of the output results, T output data for every location are averaged. Thus, we can obtain the vector $\bar{P} = [\bar{p}_1, \bar{p}_2, ..., \bar{p}_R]$, where $\bar{p}_i$ is the mean for the output vector $[p_{i1}, p_{i2}, ..., p_{iT}]$ in the i_{th} row.

Finally, we can compute the location of the mobile device as a weighted average of all R locations, that is

$$\hat{L} = \sum_{i=1}^{R} l_i \times \bar{p}_i, \tag{6.9}$$

where l_i is the ith training location.

6.5 Experimental Study

6.5.1 Experiment Configuration

To evaluate the performance of the ResLoc, we implement it with 5GHz WiFi devices. In order to collect CSI data, a desktop computer and a Dell laptop are used as access point and mobile device. Both computers are equipped with an Intel 5300 network card, running Ubuntu desktop 14.04 LTS system. To transmit the data to the desktop, the Dell laptop with one antenna works in the injection mode. The monitor mode is executed in the desktop to receive data. The distance between two adjacent antennas on the Desktop is set as 2.68 cm, which is a half wave length for 5.58GHz WiFi signal. Moreover, the PHY is the IEEE 802.11n OFDM system with QPSK modulation and 1/2 coding rate. To accelerate the training process, we employ the offline stage of the ResLoc in Keras with tensorflow backend on a PC with Intel(R) Core(TM) i7-6700K CPU, and an Nvidia GTX1070 GPU [105].

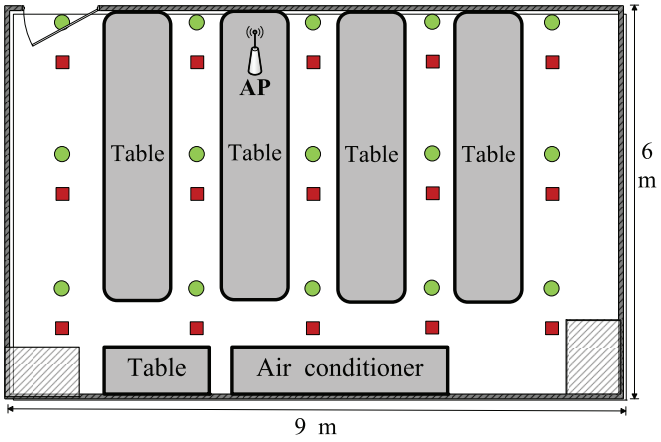


Figure 6.3: Layout of the computer laboratory: training locations are marked as red squares and testing locations are marked as green dots.

ResLoc are compared with two typical deep learning localization approaches, BiLoc [101] and DeepFi [3], to evaluate its performance. Moreover, we also consider the localization performance for ResLoc with the single channel. For the sake of fairness, the same CSI training dataset and testing dataset are leveraged in all four approaches. We examine them in two experimental environments including a computer laboratory and a corridor.

Computer Laboratory

Computer Laboratory: We set up the first testbed in a 6×9 m^2 computer laboratory in the Borun Hall in the Auburn University campus. This laboratory is a crammed environment. The furniture and appliances block the most of LOS paths. 15 training locations are shown as red squares in Figure 6.3, while the other 15 green dots are testing locations. The distance between two adjacent training locations is 1.8 m. Our receiver is fixed on the table. We collect 1000 CSI packets from every training location and testing location to accumulate CSI data. Moreover, we set the number of layers for the proposed deep network as 34, which has higher localization accuracy and smaller training time.

Corridor:

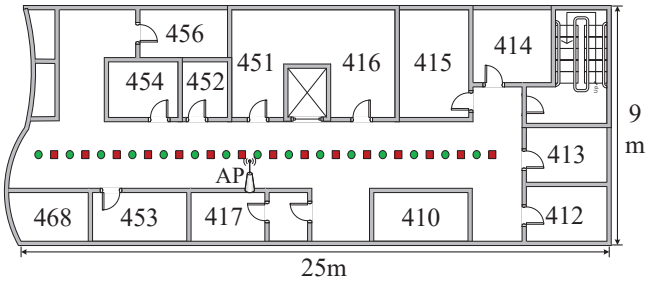


Figure 6.4: Layout of the corridor: training locations are marked as red squares and testing locations are marked as green dots.

Corridor in Broun Hall

We set up the second testbed in a long corridor in Borun hall, which is 9×25 m^2. filled with no furniture and appliance. In this scenario, LOS path is majority. We employ 15 training location and 15 testing location in a straight line. The distance between two adjacent training locations is 1 m. The red squares are training locations and the rest green dots are testing locations. We set the receiver in the middle of the corridor. 1000 packets are obtained from every training location and testing location to collect 5GHz CSI data. The number of layers in the deep network in the corridor is the same as that in the computer laboratory.

Accuracy of Location Estimation

Figure 6.5 depicts the training loss over epoches of the ResLoc for the laboratory and corridor scenarios. To prevent overfitting for the training CSI tensor data set and reduce training time, the epoch is set as 50. As illustrated in Fig.3, the train loss for the corridor curve reaches about 0.3 and the training loss for the lab scenario stops at about 0.5. Moreover, based on Nvidia GTX1070 GPU, we can obtain the smaller training time for the laboratory and corridor scenarios, which are 608.14 s and 619.35 s, respectively. Also, the test time for the laboratory and corridor scenarios are 0.587 s and 0.647 s, which can be accepted for indoor localization.

Figure 6.6 shows the CDF of distance error across the 15 positions in the laboratory. Unlike the corridor scenario that the LOS is majority, the furniture and appliances block most of LOS paths in this environment. As we can see, the maximum distance errors for ResLoc with two

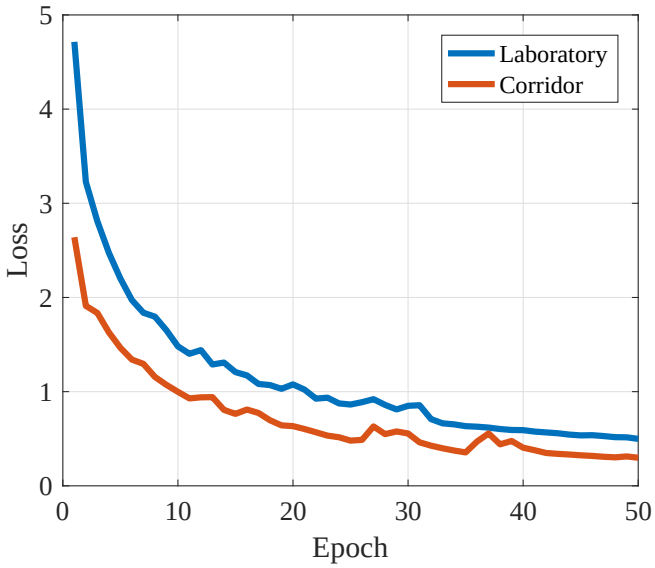


Figure 6.5: Training errors for the laboratory and corridor experiments.

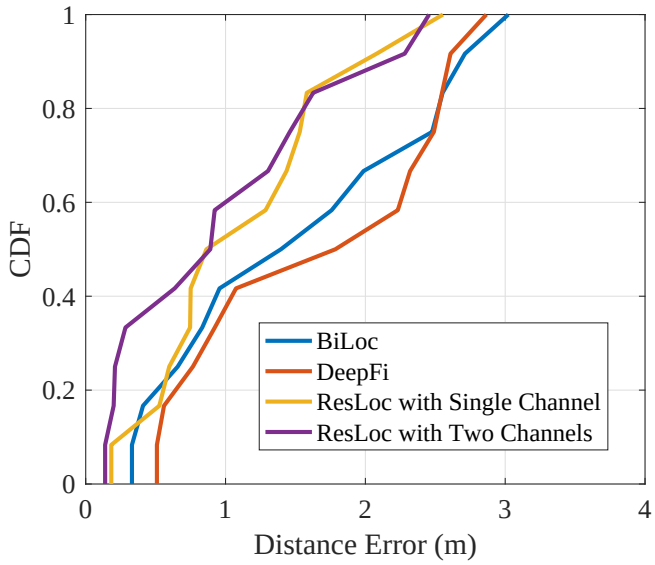


Figure 6.6: CDF of localization errors for the laboratory experiment.

channels and single channel are about 2.5 m, which is less than DeepFi and BiLoc. In addition, the median of distance errors for ResLoc with two channels and single channel are about 0.89 m, which also outperforms BiLoc and DeepFi by 0.51 m and 0.89 m, respectively. For ResLoc with two channels, the distance error of over 30% testing data is less than 0.3 m. However, there is no data falling within this error range for DeepFi and BiLoc. In summary, based on the proposed

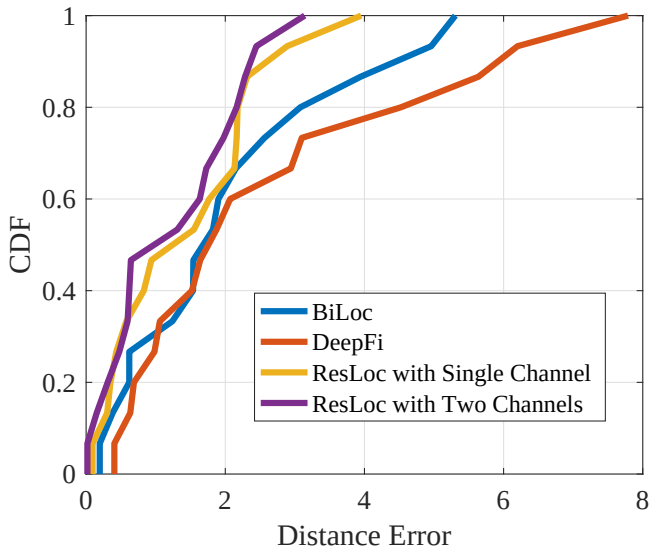


Figure 6.7: CDF of localization errors for the corridor experiment.

deep residual sharing learning, ResLoc with two channels exhibits the best performance in this rich multipath scenario.

Figure 6.7 plots the CDF of localization error in the corridor scenario. As shown in Figure 6.7, the maximum distance error for ResLoc with two channels and single channel are 3.14 m and 3.95 m, respectively, which are significantly less than that of other two schemes, DeepFi and BiLoc. It shows that the ResLoc has a better stability than DeepFi and BiLoc. In addition, the median of distance errors for ResLoc with two channels and single channel, BiLoc and DeepFi are about 0.98 m, 1.24 m, 1.68 m, and 1.75 m, respectively. Thus, ResLoc with two channels achieves the best performance in this scenario. Besides the better performance, the proposed ResLoc system only requires one set of weights for all training locations to achieve localization, which means that it is not necessary for ResLoc to store fingerprints for every training location like BiLoc and DeepFi. Furthermore, ResLoc does not need a ratio for the bi-modal data to obtain a better localization performance.

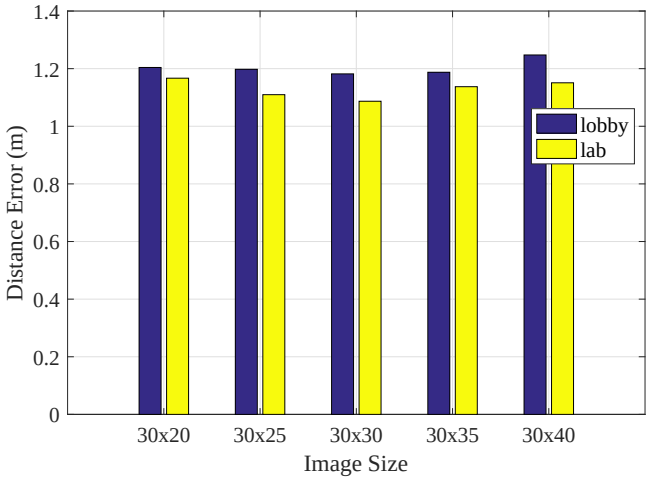


Figure 6.8: The average distance error for different size of pictures.

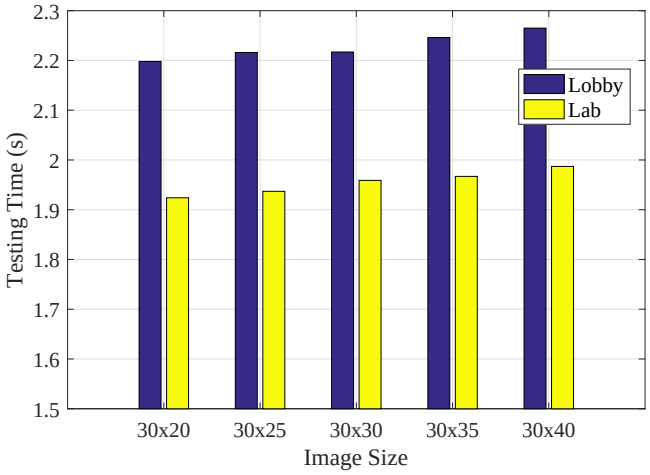


Figure 6.9: The average testing time for different size of pictures.

Effect of Different Parameters

To determine how image size impacts the accuracy of the indoor localization, we test ResLoc with images sized to 30x20, 30x25, 30x30, 30x35 and 30x40. For fairness, 50 images are constructed for every training position. Epoch and batch size are set to 50 and 10, respectively. As is shown in Figure 6.8, distance errors for both scenarios decrease slightly as the image size increases from 30x20 to 30x30, then rise as the image size increases from 30x30 to 30x40. However, the distance errors are stable in these two scenarios. The errors in the lobby and the lab are about 1.2 m and 1.13 m, respectively. This result indicates that the localization performance of Resloc is robust enough

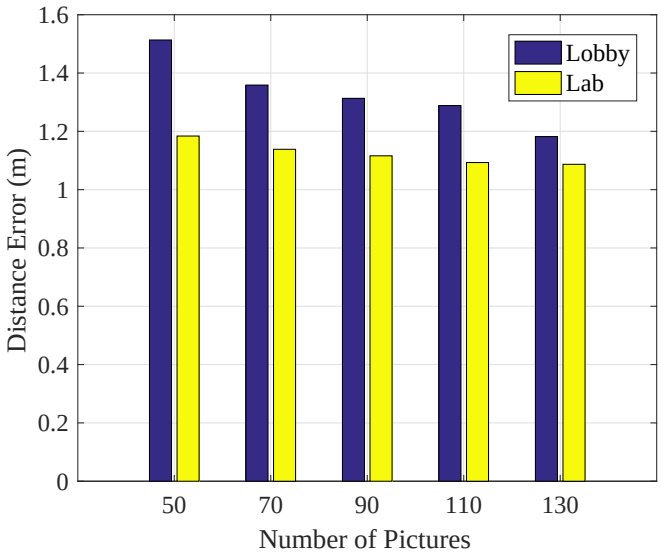


Figure 6.10: The average distance error for different number of pictures.

to the image size. Even though the size of images is changed, Resloc could achieve localization with high precision.

As discussed previously, the image size does not show a significant effect on localization accuracy. To select the best size for training and testing the network, we also compare the testing time with different image sizes.As is shown in Figure 6.9, it is apparent for both scenarios that the testing time rises with the increment of the image size. Theoretically, with the same image size, the testing time should be identical in these two scenarios. However, there is a 0.1 s gap between the lab and the lobby. We find that this gap is resulted from the computer performance. Considering the testing time and the distance error, the image size of 30x30 is the best choice for training and testing because of its lowest distance error and acceptable testing time in two scenarios.

To further explore how many the number of pictures affects the distance error, we build 5 datasets with different number of pictures in every position. As is shown in Figure 6.10, the distance error declines with the increase of the number of pictures. The lowest distance errors, 1.0869 m for the lab and 1.1819 m for the lobby, are obtained when the number of pictures is 130. This result indicates that the number of pictures is related to the localization accuracy positively. Furthermore, the distance error in the lobby is more sensitive to the number of pictures. We also

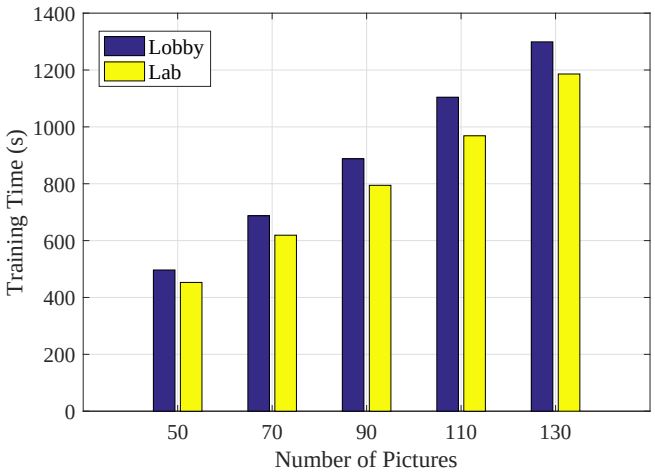


Figure 6.11: The average training time for different number of pictures.

notice that all distance errors for the lab are smaller than 1.2 m and distance errors for the lobby are lower than 1.3 m when the number of pictures is greater than 90. It also shows that the performance of ResLoc is robust to the change of the number of pictures.

Figure 6.11 shows the training time across all datasets with different number of pictures. It is intuitive to show that the training time is directly proportional to the number of pictures. For the same number of pictures, the training time for the lobby is slightly longer than the training time for the lab. Considering that the training process is a part of offline stage, namely the training time does not compromise user experience. Thus, we choose the dataset with 130 pictures in every training point as the input of ResLoc because of the lowest distance error.

Impact of Bimodality

To evaluate the performance of our proposed bimodal input, we also deploy our ResLoc model with different kind of input datasets, the amplitude dataset, the phase difference dataset and the bimodal dataset. We compare the performance of these three datasets in two indoor environments, a computer laboratory and a long corridor. We know that CSI amplitude values reflects channel frequency responses with abundant multipath components and channel fading. In other words, the performance of amplitude dataset is degraded by the indoor environments. The computer lab is a cluttered environment. The furniture, computers, and appliances block most of the LOS paths and

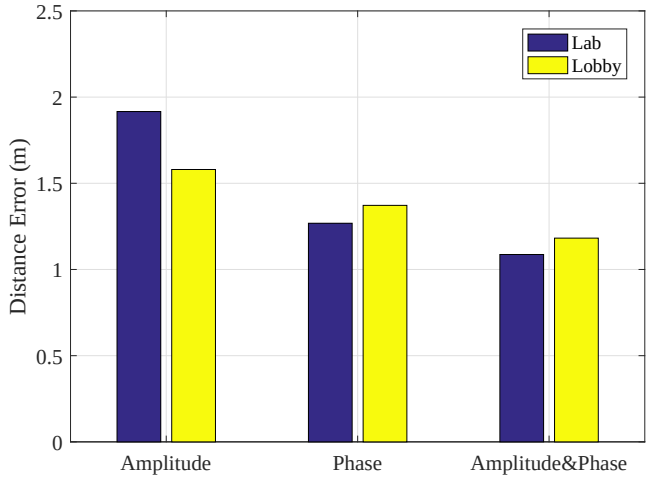


Figure 6.12: The average distance error for different input dataset with two channel model.

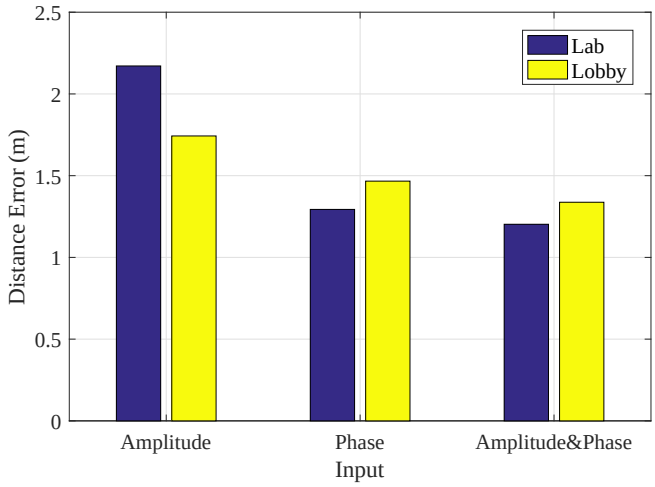


Figure 6.13: The average distance error for different input dataset with one channel model.

generate lots of multipath. As is shown in Figure 6.12, the worst performance is achieved by the amplitude dataset in the lab. Comparing with the amplitude values, the phase values of the signal with the periodical change over the propagation distance is relatively more robust. According to Figure 6.12, we have a lower distance error with the phase difference dataset. The bimodal tensor shows the lowest distance error among three datasets. Due to the use of bi-modal CSI tensor, the phase difference values can be utilized to mitigate the influence of the complex indoor environment. The mean distance errors are 1.0869 m and 1.819 m in the lab and the corridor, respectively.

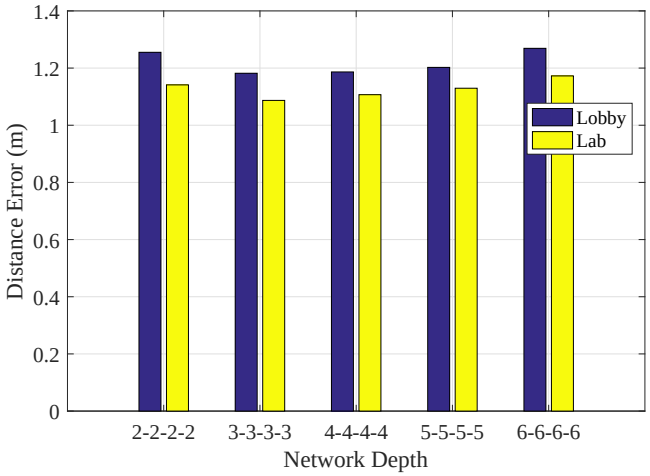


Figure 6.14: The average distance error for different network depth.

We also implement the single channel version ResLoc on these three datasets to investigate the performance difference between the two channel ResLoc and the single channel ResLoc. Figure 6.13 shows a similar trend to previous two channel ResLoc, which means that the complexity of the indoor environment is a dominant effect to the localization performance. However, it is noticed that all distance errors showed in Fig. 6.13 are larger than corresponding distance errors slightly. The distance error that is obtained by the amplitude dataset in the lab is over 2 m. For the single channel ResLoc, the lowest distance error, 1.2027 m, that is obtained by bimodal dataset is still higher than the corresponding distance error, 1.0869 m, which is the best result of the two channel ResLoc. According to Figure 6.12 and Figure 6.13, it is obvious to reveal that the two-channel architecture enhances the performance of the ResLoc system.

We now show the impact of different number of layers on the proposed ResLoc system. All convolutional kernels in residual blocks are sized to 3x3. There are four sizes of residual blocks. For each convolutional layer, the number of feature maps in the first block, second block, third block, forth block are 64, 128, 256 and 512 respectively. Moreover, two convolutional layers are stacked up to form a basic residual block. To evaluate how the depth of network affects the performance of the network, four basic residual blocks are repeated twice, three times, four times, five times and six times, respectively. Theoretically, increasing layers may reduce the distance error. However, Figure 6.14 shows that the distance error reaches lowest point when the network

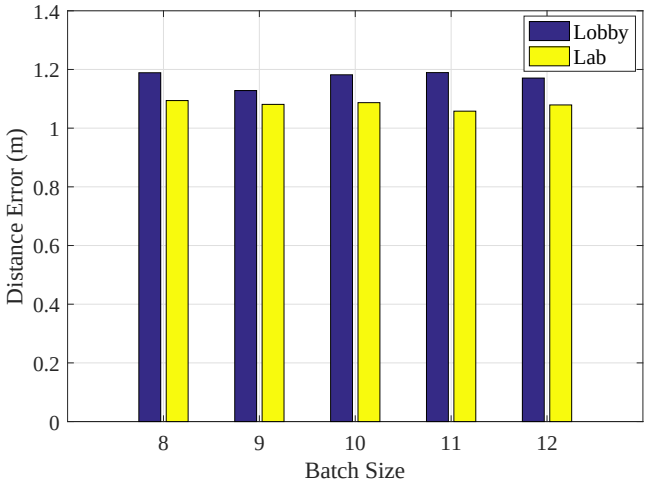


Figure 6.15: The average distance error for different batch size.

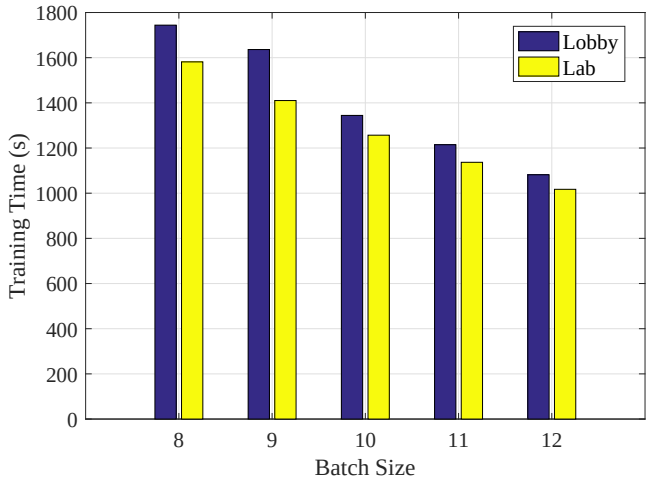


Figure 6.16: The average training time for different batch size.

scheme is 3-3-3-3. After that the distance error rises as the network goes deep. We believe that all schemes are deep enough to solve indoor localization problem. All distance errors are about 1.2 m, which means the distance error is robust when the network is designed as deep as we did. We choose 3-3-3-3 scheme as the best scheme to train the network, because of the lowest distance error and a relatively simple scheme.

Batch size defines number of samples that can be propagated through the network. We study the impact of batch size on localization accuracy under the two environments. Figure 6.15 illustrates the mean distance errors for increasing batch size in the lab and lobby scenarios. As we can see, there is no relation between the value of batch sizes and the mean distance error. For the lab

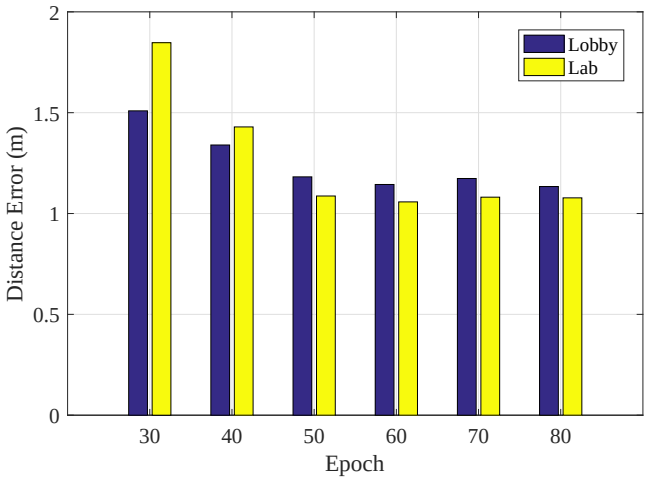


Figure 6.17: The average distance error for different epoch.

scenario, the highest mean error is 1.0869 m and the lowest mean error is 1.0579 m. The difference between the maximum and minimum of mean error is 0.029 m, which means the mean distance error is robust enough to the change of the batch size. Similarly, the difference between the maximum and minimum of the mean error is only 0.0613 m for the lobby scenario. It also shows that the mean distance error is independent to the value of batch size.

Figure 6.16 depicts the training time for different batch sizes. Typically, networks trains faster with mini-batches. We observe that the training time gets shorter with increasing batch size. According to the Figure 6.16, we know the longest training times are 1744 s and 1581 s in the lobby and the lab, and the shortest training times are 1081 s and 1017 s in the lobby and the lab.

To improve the accuracy of ResLoc, we adjust the value of epoch. The impact of epoch on localization accuracy is shown in Figure 6.17. In both indoor environments, the lab and the lobby, the highest distance error is obtained when the value of epoch is 30. Along with the growth of the value of epoch, the distance error keeps decreasing. And it maintains at about 1.1 m from 50 epochs. Intuitively, the network does not converge before 50 epochs. When the network reaches convergence, the distance error remains at same level. It is noticed that the lowest distance errors in the lab and the lobby are obtained at 50 epochs.

Figure 6.18 depicts the training time against the value of epoch. As is shown, the training time increases as the value of the epoch increases in both scenarios. It is consistent with our intuitive

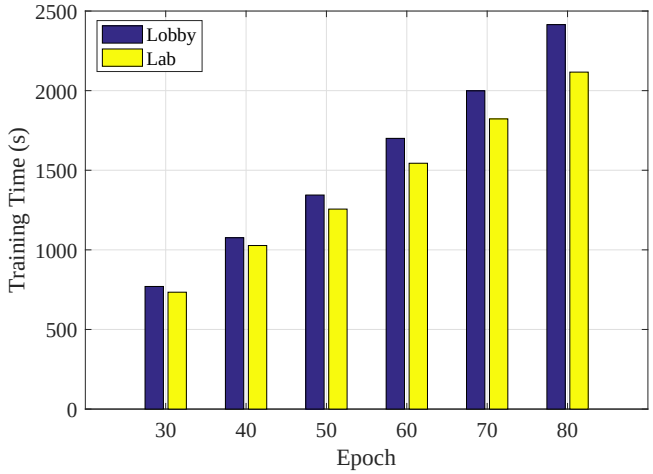


Figure 6.18: The average training time for different epoch.

result that the more epoch loops the more time is consumed. To reach the lowest distance error, we spend about 1344 s and 1256 s to train the network in the lobby and the lab respectively.

6.6 Conclusions and Future Work

In this chapter, we presented ResLoc, a deep residual sharing learning based system for indoor localization with two channels CSI tensor data. We introduced CSI in WiFi network with OFDM system and discussed how to build CSI tensor data for indoor localization. Then, we designed the ResLoc system, which leverages two channels CSI tensor data to train the deep network by using the proposed deep residual sharing learning. For online test, we used newly received CSI tensor data to compute the location of the mobile device based on the probabilistic method. Finally, the experimental results showed the superior performance of the proposed ResLoc system.

Chapter 7

PhaseBeat: Phase Information for Tracking Vital Signs with Commodity WiFi Device

7.1 Introduction

It is evaluated that 100 million Americans have chronic health conditions such as lung disorders and heart disease, which require three-fourths of total US healthcare costs to treat these conditions [106]. This causes an increasing demand for long-term health monitoring in indoor environments. Tracking vital signs such as breathing and heart rates can be leveraged to estimate the humans physical health and offer the important clues for medical problems. For example, monitoring vital signs can help a patient to find the sleep disorders or anomalies, and reduce sudden infant death syndrome (SIDS) for sleeping infants [107]. The traditional methods for vital monitoring are required a person to wear special devices such as a capnometer [108] to detect breath rate or a pulse oximeter [109] on the finger to measure the heart rate. These technologies are inconvenient and uncomfortable. Thus, an alternative solution is required to offer a contact-free and long-term vital signs monitoring.

Recently, RF based vital sings monitoring systems have attracted more attention, which employs wireless signal to extract the breathing-induced chest change of a person. Vital-Radio system leverages frequency modulated continuous wave (FMCW) radar to estimate breathing and heart rates, even for multiple person subjects in parallel [110]. But the system requires a custom hardware with a large bandwidth from 5.46 GHz to 7.25 GHz. Some techniques such as the Doppler radar [111, 112] and the ultra-wideband radar [113] are also used to monitor vital signs, which require the dedicated hardware with high frequency and the high cost. mmVital system [114] can use

60 GHz millimeter wave (mmWave) signal based on RSS for breathing and heart rates estimation with the larger bandwidth about 7GHz, which also requires the custom hardware with a mechanical rotator. Moreover, UbiBreathe system only monitors the breathing signal by leveraging WiFi RSS with the coarse channel information [115]. This system is required that the device is placed in the line of sight path between the transmitter and the receiver, which limits the RF monitoring range in the deployment environment.

CSI provides fine-grained channel information, which can now be obtained from several off-the-shelf WiFi NIC, e.g., Intel WiFi Link 5300 NIC [21], and the Atheros AR9580 chipset [81]. Also, CSI represents both amplitude and phase information of subcarrier-level measurements of OFDM channels. It is a more stable representation of channel characteristics than RSS. Recently, the authors leverage the amplitudes of CSI data of WiFi to track vital signs. This system is mainly to monitor the vital signs when a person is sleeping [116]. However, the collected phase information of CSI data is not directly usable for vital monitoring because of large phase fluctuation from the noise and the unsynchronized time and frequency of the transmitter and receiver.

In this chapter, we leverage CSI phase difference data between two antennas to monitor the breathing and heart rate. First, CSI phase difference data is stable because the randomness of raw CSI phase is removed at the subtraction in the WiFi NIC, which has the same sampling clock and the same down-converter frequency for each of three antennas. Moreover, for different distances and orientations between the transmitter and the receiver, CSI phase difference is more robust than the amplitude based method, which has large attenuation due to obstacles and long distances. On the other hand, our work is the first to prove that for indoor mutipaths environments with the small scale signal fading, the CSI phase difference data at the receiver is a periodic signal with the same frequency as the breathing signal when the wireless signal is reflected from the chest of one person. Moreover, we also prove that leveraging directional antenna to improve the power of the transmitter can boost the magnitude of CSI phase difference data, which can be exploited to monitor the minute heart beating signal.

In this chapter, we then design PhaseBeat, CSI phase difference data for monitoring breathing and heart beats with commodity WiFi device. First, PhaseBeat system employs CSI phase difference data to extract the periodic signal from the change in the chest of a person such as inhaling and exhaling. Then, we implement data preprocessing for the collected phase difference data, which includes environment detection, data calibration, subcarrier selection and discrete wavelet transform. For environment detection, we need to detect the stationary person such sitting, standing and sleeping with the threshold method. Then, effective phase difference data is calibrated by removing direct current (DC) component and high frequency noise, and by implementing the downsampling for the processed data. Due to frequency diversity, the most sensitized subcarrier is selected for implementing the discrete wavelet transform to obtain the denoised breathing signal and the reconstructed heart signal. Finally, we leverage the peak detection method for breathing signal detection and FFT based method for heart signal estimation.

We implement PhaseBeat on the commodity WiFi devices and evaluate its performance with four persons over three months in different indoor environments such as a computer laboratory, a through-wall scenario and a long corridor. The results demonstrate that our PhaseBeat system can achieve high estimated accuracy of breathing rate with the medium error about 0.25 bpm. Moreover, the medium error for the heart rate estimation is about 1 bpm by using directional antennas at the transmitter. We also extensively evaluate the robustness of PhaseBeat for breathing rate estimation under varying environmental parameters.

The main contributions of this chapter are summarized below.

- We theoretically and experimentally validate the feasibility of using CSI phase difference for vital signs monitoring. In particular, we deeply analyze the measured phase errors and prove the phase difference with the same frequency with breathing rate. To the best of our knowledge, we are the first to leverage CSI phase difference for breathing rate and heart rate estimation.

- We implement several signal processing algorithms in data preprocessing for the collected CSI phase difference data, which can obtain the denoised breathing signal and the restructured heart signal. Then, we leverage the peak detection method for breathing rate estimatation and FFT based method for heart signal estimation.

- We prototype the PhaseBeat system with commodity WiFi devices and validate its superior performance in different indoor environments with extensive experiments. Our experimental results demonstrate that our PhaseBeat system can obtain better performance than the amplitude based method for breathing rate estimation.

In the rest of this chapter, the preliminaries and phase difference information are provided in Section 7.2. We propose the PhaseBeat system in Section 7.3 and demonstrate its performance in Section 7.4. Section 7.5 reviews related work and Section 7.6 concludes this chapter.

7.2 Phase Difference Analysis

In this section, we show that the difference of CSI phase values between two antennas for continuous packets of the 5GHz OFDM channel is highly stable. Although the CSI phase information is also available from the Intel 5300 NIC, it is highly random and cannot be directly used for vital signs monitoring, due to noise and the unsynchronized time and frequency of the transmitter and receiver. Recently, two useful algorithms are used to remove the randomness in CSI phase. The first approach is to make a linear transform of the phase values measured from the 30 subcarriers [67, 79]. The other one is to exploit the phase difference between two antennas in 2.4GHz and then remove the measured average [66]. Although both methods can stabilize the CSI phase in consecutive packets, the average phase value they produce is always near zero, which is different from the real phase value of the received signal.

We provide an analysis to validate the stability from the measurement phase difference. Let $\angle\widehat{CSI}_i$ denote the measured phase of subcarrier i, which is given by [80, 81]

$$\angle\widehat{CSI}_i = \angle\text{CSI}_i + (\lambda_p + \lambda_s)m_i + \lambda_c + \beta + Z, \tag{7.1}$$

where $\angle\text{CSI}_i$ is the true phase from wireless propagation, Z is the measurement noise that is assumed as the AWGN of variance σ^2, β is the initial phase offset because of the phase-locked loop (PLL), m_i is the subcarrier index of subcarrier i, λ_p, λ_s and λ_c are phase errors from the packet boundary detection (PBD), the sampling frequency offset (SFO) and central frequency offset (CFO), respectively [80], which are expressed by

$$\begin{cases} \lambda_p = 2\pi\frac{\Delta t}{N} \\ \lambda_s = 2\pi(\frac{T'-T}{T})\frac{T_s}{T_u}n \\ \lambda_c = 2\pi\Delta f T_s n, \end{cases} \tag{7.2}$$

where Δt is the packet boundary detection delay, N is the FFT size, T' and T are the sampling periods from the receiver and the transmitter, respectively, T_u is the length of the data symbol, T_s is the total length of the data symbol and the guard interval, n is the sampling time offset for current packet, Δf is the center frequency difference between the transmitter and receiver. It is noticed that we cannot obtain the exact values about Δt, $\frac{T'-T}{T}$, n, Δf, and β in (7.1) and (7.2). Moreover, λ_p, λ_s and λ_c vary for different packets with different Δt and n. Thus, the true phase $\angle\text{CSI}_i$ cannot be derived from the measured phase value. Fortunately, the measured phase difference on subcarrier i can be leveraged as the following theorem.

Theorem 2. *The measured phase difference on subcarrier i between two antennas is stable, and its mean and variation are expressed by*

$$\mathbb{E}(\Delta\angle\widehat{CSI}_i) = \mathbb{E}(\Delta\angle CSI_i) + \Delta\beta, \tag{7.3}$$

$$Var(\Delta\angle\widehat{CSI}_i) = Var(\Delta\angle CSI_i) + 2\sigma^2. \tag{7.4}$$

Proof. Note that the three antennas of the Intel 5300 NIC use the same clock and the same down-converter frequency. Consequently, the measured phases of subcarrier i from two antennas have identical packet detection delay, sampling periods and frequency differences (and the same m_i) [76]. Thus the measured phase difference on subcarrier i between two antennas can be

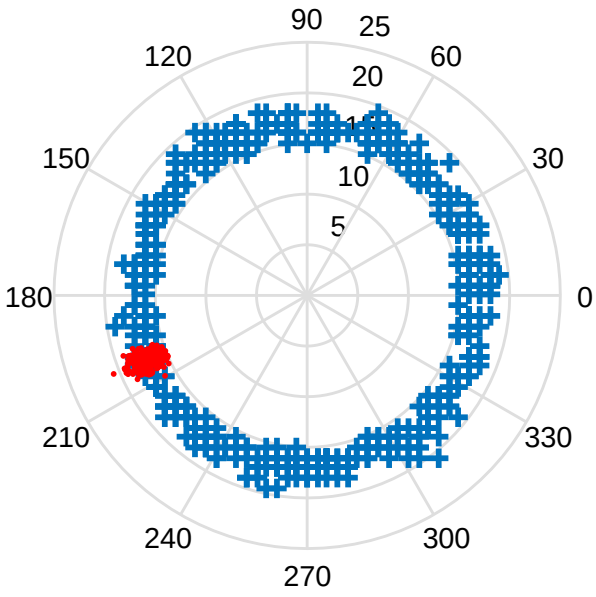


Figure 7.1: The comparison between the single antenna phases (as blue crosses) and the phase differences (as red dots) of the $5th$ subcarrier in the polar coordinate system for 600 continuing packets

approximated as

$$\Delta\angle\widehat{CSI}_i = \Delta\angle\text{CSI}_i + \Delta\beta + \Delta Z, \tag{7.5}$$

where $\Delta\angle\text{CSI}_i$ is the true phase difference of subcarrier i, $\Delta\beta$ is the unknown difference in phase offsets, which is in fact a constant [76], and ΔZ is the noise difference with the variance $2\sigma^2$. We can find that $\Delta\angle\widehat{CSI}_i$ is stable for different packets because of the above equation (7.5) without Δt, Δf and n. For the above equation (7.5), we have the results about the mean and variation of the measured phase difference on subcarrier i as $\mathbb{E}(\Delta\angle\widehat{CSI}_i) = \mathbb{E}(\Delta\angle\text{CSI}_i) + \Delta\beta$, $Var(\Delta\angle\widehat{CSI}_i) = Var(\Delta\angle\text{CSI}_i) + 2\sigma^2$. Thus, we proof the theorem 2. □

We can see that the difference between the mean of the measured and the mean of true phase differences on subcarrier i is constant, which would not change the estimation frequency of vital signals, although its variation becomes larger. Fig. 7.1 shows the comparison between the single

antenna phases (as blue crosses) and the phase differences (as red dots) of the $5th$ subcarrier in the polar coordinate system for 600 continuing packets. We can see that the single antenna phase of the $5th$ subcarrier is nearly uniform distribution between 0 and 360 degree, which is greatly unstable. However, all phase difference data of the $5th$ subcarrier concentrate into a sector between 190 and 210 degree, which supports the theorem 2. On the other hand, we provide the theorem of phase difference information with periodic as the following.

Lemma 1. *When the wireless signal is reflected from the chest of one person with the breathing frequency f_b, the true phase of the refection signal at any antenna of the receiver is a periodic signal with the frequency f_r, that is*

$$f_r = f_b. \tag{7.6}$$

Proof. Because the wireless signal for subcarrier i is a plane wave, its true phase at the receiver is related with the propagation distance with the signal, that is

$$\angle \text{CSI}_i = \frac{2\pi d(t)}{\lambda_i}. \tag{7.7}$$

where $d(t)$ is the propagation distance at the t time, λ_i is the wavelength of the subcarrier i. When the chest of a person periodically inhales and exhales with the frequency f_b, the propagation distance $d(t)$ for the reflection signal can be updated by

$$d(t) = D + A \cdot cos(2\pi f_b t) \tag{7.8}$$

where D is the constant distance for the reflection path, A is the amplitude of the periodic signal. Thus, its true phase of the reflection signal at the receiver is $\angle \text{CSI}_i = \frac{2\pi(D+A\cdot cos(2\pi f_b t))}{\lambda_i}$. It is noticed that the true phase at the receiver is a period signal with the same the frequency f_b. Thus, we prove the lemma with $f_r = f_b$. □

Theorem 3. *For indoor environments with mutipaths, when the wireless signal is reflected from the chest of one person with the breathing frequency f_b, the true phase at any antenna of the receiver is also a periodic signal with the frequency f_d as the following*

$$P\left(|f_d - f_b| < \epsilon\right) = 1, \text{ for any } \epsilon > 0. \tag{7.9}$$

Proof. Based on Lemma 1, we can find that the true phase of the reflection signal at the receiver with $\angle \text{CSI}_i = \frac{2\pi(D+A\cdot cos(2\pi f_b t))}{\lambda_i}$ is a periodic signal with the frequency f_b. We mark the reflection signal as the dynamic component, while LOS and other mutipath as static component. Then, we can update the equation 2.2 as

$$\begin{aligned}
\text{CSI}_i &= \sum_{k=0}^{K} r_k \cdot e^{-j2\pi f_i \tau_k} \\
&= \sum_{k=0,k\neq d}^{K} r_k \cdot e^{-j2\pi f_i \tau_k} + r_d \cdot e^{-j2\pi f_i \tau_d} \\
&= \text{CSI}_i^s + \text{CSI}_i^d. \\
&= |\text{CSI}_i^s| \exp\left(j\angle \text{CSI}_i^s\right) + |\text{CSI}_i^d| \exp\left(j\angle \text{CSI}_i^d\right)
\end{aligned} \tag{7.10}$$

where CSI_i^s is the static component that is represented as $\sum_{k=0,k\neq d}^{K} r_k \cdot e^{-j2\pi f_i \tau_k}$ and CSI_i^d is the dynamic component that is represented as $r_d \cdot e^{-j2\pi f_i \tau_d}$, $|\text{CSI}_i^s|$ and $\angle \text{CSI}_i^s$ is the amplitude and phase of CSI_i^s, $|\text{CSI}_i^d|$and $\angle \text{CSI}_i^d$ is the amplitude and phase of CSI_i^d. Moreover, $\angle \text{CSI}_i^d = \frac{2\pi(D+A\cdot cos(2\pi f_b t))}{\lambda_i}$ is a periodic signal with the frequency f_b. And $|\text{CSI}_i^s|$, $\angle \text{CSI}_i^s$ and $|\text{CSI}_i^d|$ are considered as constant.

To obtain the phase of CSI_i, we need to build the geometric relationship among the static component CSI_i^s, the dynamic component CSI_i^d and the total component CSI_i using an *in-phase-quadrature* (I-Q) plot in Fig. 7.2. Based the geometric relationship in Fig. 7.2, we can easily obtain the angle $\angle \text{DST} = \angle \text{CSI}_i^s - \angle \text{CSI}_i^d$ and the length $OT = |\text{CSI}_i^d| \cdot \cos(\angle \text{CSI}_i^s - \angle \text{CSI}_i^d) + |\text{CSI}_i^s|$, and the length $TD = |\text{CSI}_i^d| \cdot \sin(\angle \text{CSI}_i^s - \angle \text{CSI}_i^d)$. Thus, the phase of the total component CSI_i

is computed by

$$\angle \text{CSI}_i = \angle \text{CSI}_i^s - \arctan \frac{|\text{CSI}_i^d| \cdot \sin(\angle \text{CSI}_i^s - \angle \text{CSI}_i^d)}{|\text{CSI}_i^d| \cdot \cos(\angle \text{CSI}_i^s - \angle \text{CSI}_i^d) + |\text{CSI}_i^s|}. \tag{7.11}$$

Because the $\angle \text{CSI}_i^d = \frac{2\pi(D + A \cdot cos(2\pi f_b t))}{\lambda_i}$ is a periodic signal, the $\angle \text{CSI}_i$ is also a periodic signal. Then, to prove the frequency of the total phase $\angle \text{CSI}_i$ with f_b, we make the derivation of the equation 7.11 zero, that is $\frac{d\angle \text{CSI}_i}{d\angle \text{CSI}_i^d} = 0$. Then, we can obtain

$$\cos(\angle \text{CSI}_i^s - \angle \text{CSI}_i^d) = -\frac{|\text{CSI}_i^d|}{|\text{CSI}_i^s|}, \tag{7.12}$$

which has the solutions that are the $\angle \text{CSI}_i^d = \angle \text{CSI}_i^s + \pi - \arccos(|\text{CSI}_i^d|/|\text{CSI}_i^s|)$ with node D2 or $\angle \text{CSI}_i^d = \angle \text{CSI}_i^s + \pi + \arccos(|\text{CSI}_i^d|/|\text{CSI}_i^s|)$ with node D1 in Fig. 7.2. We can find, when the phase $\angle \text{CSI}_i^d \in (\angle \text{CSI}_i^s + \pi + \arccos(|\text{CSI}_i^d|/|\text{CSI}_i^s|) - 2\pi, \angle \text{CSI}_i^s + \pi - \arccos(|\text{CSI}_i^d|/|\text{CSI}_i^s|))$, the equation 7.11 is increasing function; otherwise, it is decreasing function. Thus, unless the only two nodes D1 and D2, the phase of the total component CSI_i is periodic signal with the frequency f_b. Also, because the true phase CSI_i is continuing, the probability of the true phase CSI_i at node D1 or node D2 equals zero. Thus, we proves the theorem 3. □

On the other hand, considering another antenna with the same analysis, we can obtain the similar expression (7.11), which is also a periodic signal with the frequency f_b . However, its the static component CSI_i^s and the dynamic component CSI_i^d for on subcarrier i are different, because different positions of two antennas produces different wireless channels. In fact, the phases of the total component CSI_i for any two antennas have different phase difference while the same frequency. Thus, we can obtain the true phase difference between any two antennas is also a periodic signal with the frequency f_b.

Based on theorem 2 and equation 7.5, we can easily find that the measured phase difference on subcarrier i between two antennas is also a stable periodic signal with the frequency f_b, although

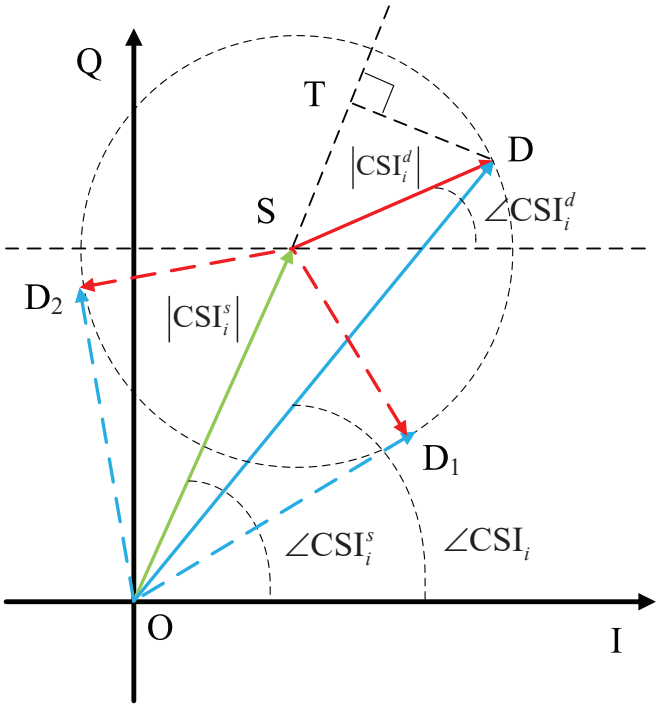


Figure 7.2: The geometric relationship among the static component CSI_i^s with the green vector $\vec{OS}$, the dynamic component CSI_i^d with the red vector $\vec{SD}$ and the total component CSI_i with the blue vector $\vec{OD}$ in I-Q plot

the waveform of the signal is attenuated due to the increasing noise. To improve the signal waveform, directional antennas is always used as the transmitter, which can strengthen the power of the reflection signal from the body of one person. In our system PhaseBeat, we use the directional antennas as the transmitter to estimate the heart rate, because of this greatly weak signal power. To see the function of directional antennas, we derive the following the corollary based on theorem 3 and equation 7.11.

Corollary 3.1. *When the ratio is $|CSI_i^d|/|CSI_i^s| \to \infty$, the true phase of subcarrier i at the one antennas is a periodic signal with the frequency f_b, and has the following result:*

$$\angle CSI_i = \angle CSI_i^d. \tag{7.13}$$

Proof. Base on the equation 7.11, we can easily obtain $\angle \text{CSI}_i = \angle \text{CSI}_i^d$ when the ratio is $|\text{CSI}_i^d|/|\text{CSI}_i^s| \rightarrow \infty$. Moreover, $\angle \text{CSI}_i^d$ is a periodic signal with the frequency f_b, and thus we can proof the corollary 3.1. □

Corollary 3.2. *When the ratio is $|CSI_i^d|/|CSI_i^s| \rightarrow 0$, the true phase of subcarrier i at the one antennas is not a periodic signal, and has the following result:*

$$\angle CSI_i = \angle CSI_i^s. \tag{7.14}$$

Proof. Base on the equation 7.11, we can also easily obtain $\angle \text{CSI}_i = \angle \text{CSI}_i^s$ when the ratio is $|\text{CSI}_i^d|/|\text{CSI}_i^s| \rightarrow 0$. Moreover, $\angle \text{CSI}_i^s$ is not a periodic signal, which proves the corollary 3.2. □

Based on corollary 3.1 and corollary 3.2, we can find that when the reflection from the chest of one person becomes strong, the waveform of the received signal is periodic with high signal-to-noise ratio (SNR); when the reflection from the chest of one person becomes weak, the waveform of the received signal is periodic with low SNR, even it is not periodic. Thus, to estimate the breathing rate and heart rate, it is still challenging because of wireless multipaths and low SNR of reflection signal such as the long distance between the person and the receiver or several obstacles. Thus, we design the PhaseBeat system to overcome the above challenge for estimating the breathing rate and heart rate, even for several persons.

7.3 The PhaseBeat System

7.3.1 PhaseBeat System Architecture

The core idea of our PhaseBeat system is to monitor the vital signs such as breathing and heart beating of a person by leveraging CSI phase difference data with the commodity WiFi device. In fact, PhaseBeat system employs CSI phase difference data to extract the periodic signal from the change in the chest of a person such as inhaling and exhaling. Based on the above theorem 2 and theorem 3, PhaseBeat system exploits CSI phase difference data to monitor the vital signs for

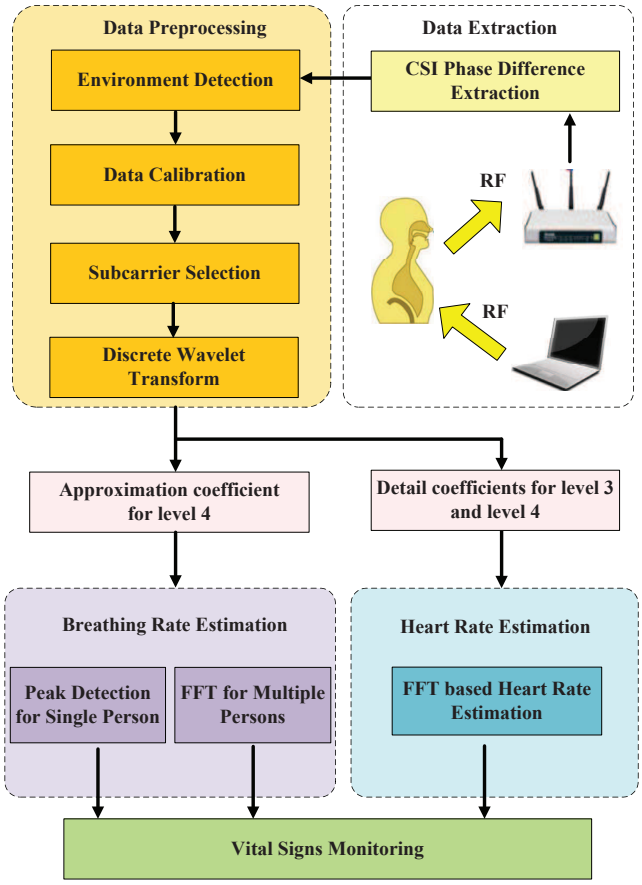


Figure 7.3: PhaseBeat system flow

three reasons. First, CSI phase difference data is relatively stable for continued packets at stationary environments such as sitting, standing or sleeping, which is thus effective for monitoring vital signs. Second, CSI phase difference data includes the periodic signal with the same frequency with breathing signal. Finally, the CSI phase difference data is robust, where the change of phase difference data is small for different distances or different orientations, compared with the amplitude based method for monitoring vital signs.

Fig. 7.3 shows PhaseBeat system flow. It includes four basic modules: Data Extraction, Data Preprocessing, Breathing Rate Estimation and Heart Rate Estimation. For Data Extraction module, the PhaseBeat system can extract CSI phase difference data between two antennas at the receiver with an off-the-shelf WiFi device. Then, Data Preprocessing module is implemented, which consists of environment detection, data calibration, subcarrier selection and discrete wavelet transform. For environment detection, we leverage a threshold method to determine the stationary situations

with a person such as sitting, standing or sleeping for tracking vital signs. For data calibration, we need to remove direct current component and high frequency noise, and to implement the down-sampling for the processed data. Then, subcarrier selection can be used to improve the reliability of CSI phase difference data. For discrete wavelet transform, It can obtain the denoised breathing signal with approximation coefficient for level 4 and the restructured heart signal with the sum of detail coefficients for level 3 and level 4. For Breathing Rate Estimation module, we leverage peak detection for single persons and FFT method for multiple persons. For Heart Rate Estimation module, we use FFT method based for heart signal estimation.

7.3.2 Data Preprocessing

Environment Detection

After CSI phase difference extraction based on equation (7.5), we need to determine whether one person is in stationary environment or not. If one person is moving such walking, running, jumping or gesture moving indoor environment, even there is no person, our PhaseBeat needs to continuously detect. Only if the person is determined at stationary such as sitting, standing, or sleeping, PhaseBeat system is leveraged to estimate the breathing rate and heart rate. In fact, a threshold-based method is used to identify whether a segment of CSI phase difference data is in stationary environment or not by computing the absolute deviation of the CSI phase difference data in a short moving window. We define V as the sum of absolute deviations of all CSI phase difference data in the moving window as the following

$$V = \frac{1}{|W|} \sum_{i=1}^{i=30} \sum_{k \in W} |\Delta\angle\widehat{CSI}_i(k) - \mathbb{E}(\Delta\angle\widehat{CSI}_i(k))|, \tag{7.15}$$

where $\Delta\angle\widehat{CSI}_i(k)$ is the measured phase difference in subcarrier i for the packet k, W is the index set of the packets in the moving window, $|W|$ is the length of the moving window. Because the other movement events lead to the larger change of CSI phase difference data than that from the minute movements of breathing and heart signals, the threshold-based approach by using the

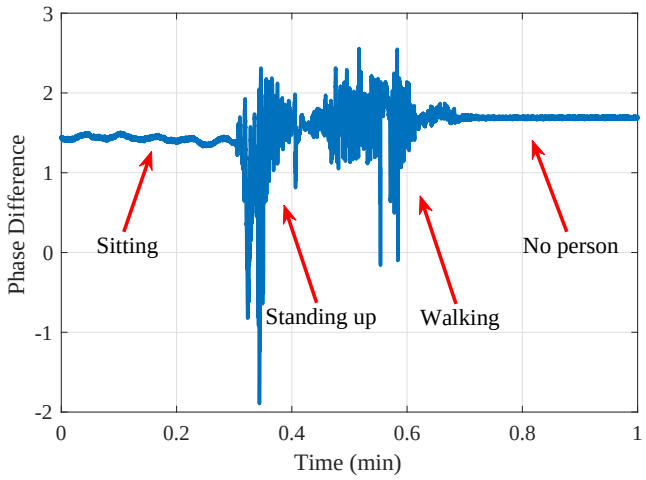


Figure 7.4: Environment detection

absolute deviation of CSI phase difference data in the moving window can be used to detect the large movements. In our PhaseBeat system, we set the threshold between 0.25 and 6 as the useful data for vital signs monitoring. Fig. 7.4 shows environment detection results for different situations. We can notice that in sitting situation the phase difference data presents a sinusoidal-like periodic signal over the time data; in no person situation the data seems a line with small fluctuation; in standing up and walking situations, the data has larger fluctuation. Thus, we can leverage the threshold to obtain the stationary situations with person such sitting, standing and sleeping.

Data Calibration

To obtain the robust CSI phase difference data, data calibration is leveraged to remove DC component and high frequency noise, and to implement the downsampling for the processed data. Firstly, because the DC component influences subcarrier selection, peak detection performance and FFT frequency estimation results, our PhaseBeat system needs to remove it based on the Hampel Filter. Different from traditional data calibration method that is to only remove the high noise, we firstly use the Hampel Filter for detrending of the original CSI phase difference data to remove DC component. The Hampel Filter is utilized to obtain the basic trend of original data, which is set as a large moving sliding window with 2000 samples and a small threshold with 0.01. Then, the

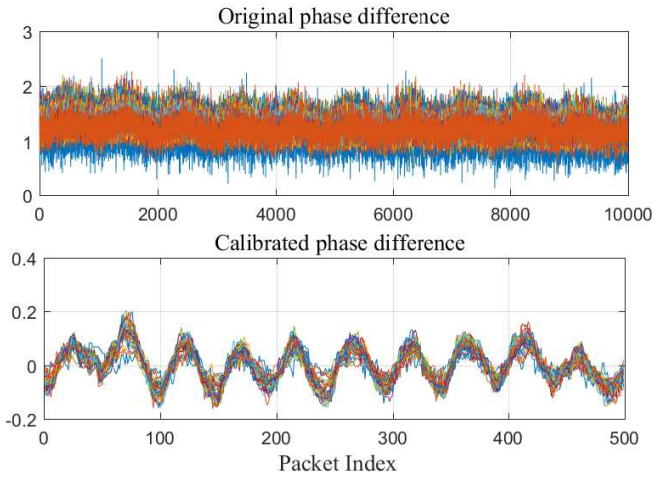


Figure 7.5: Data calibration

detrending data is obtained from the basic trend data subtracted from the original data. In addition, we also leverage the Hampel Filter to reduce the high frequency noise by using the sliding window with 50 samples and the same threshold with 0.01. On the other hand, because PhaseBeat system employs 400 Hz for data sampling, we need to implement downsampling for reducing the computation complexity of breathing rate and heart rate estimation. Thus, we use the 20 sampling interval to obtain the low frequency CSI phase difference data, that is identical to sampling with 20 Hz.

Fig. 7.5 shows data calibration for original phase difference. It is noticed that the original phase differences of all subcarriers have the DC component and high frequency noise. By implementing the our data calibration scheme, we can see that the DC component is removed and all subcarriers show a sinusoidal-like periodic signal over the packets with low noise as well as the number of packets is decreased from 10000 to 500, which is effectively leveraged for implementing other signal processing methods.

Subcarrier Selection

Subcarrier selection can be leveraged to boost the reliability of CSI phase difference data, because different subcarriers have different wavelengths, leading to the different sensitivity for breathing

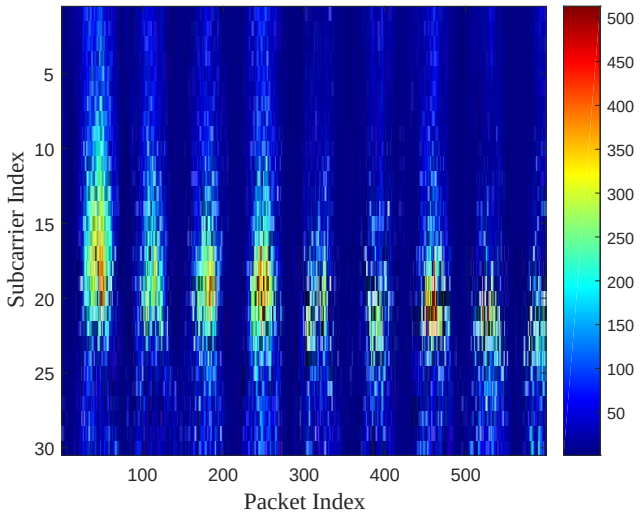


Figure 7.6: CSI phase difference series patterns after data calibration

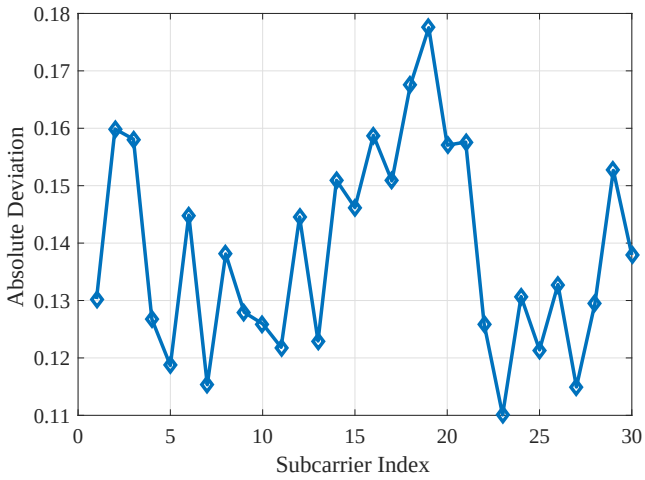


Figure 7.7: Absolute deviation of each subcarrier

and heart signals. We utilize the absolute deviation of CSI phase difference data for every subcarrier to measure its sensitivity. In fact, it is noticed that the larger absolute deviation, the higher sensitivity. Thus, we first choose k maximum absolute deviations of CSI phase difference data. To improve much more robustness of subcarriers to avoid the fact that some large absolute deviations are from higher noise. Then, we leverage the medium of k absolute deviations of CSI phase difference data to select the final subcarrier. Fig. 7.6 shows CSI phase difference series patterns after data calibration. We can see that the neighboring of subcarrier 20 have higher sensitivity for breathing signals. Then, as shown in Fig. 7.7, the absolute deviation of CSI phase difference data

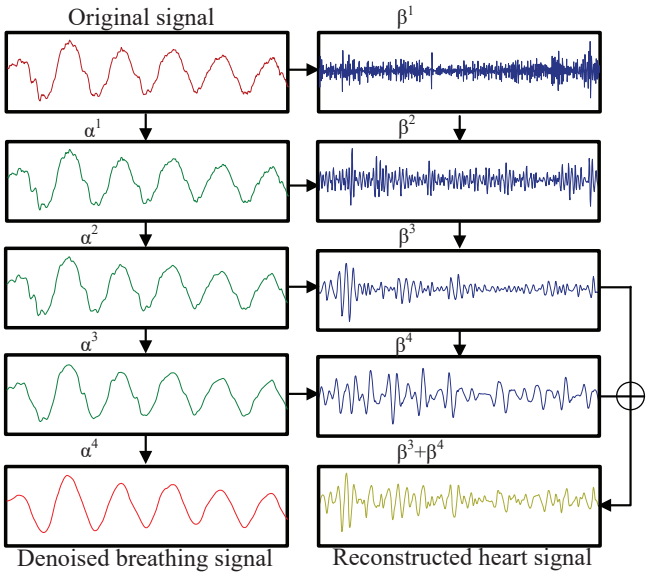


Figure 7.8: Discrete wavelet transform

of subcarrier 19 is the maximum. In our PhaseBeat system, we set the $k = 3$ as the default value, where subcarrier 19, 18 and 2 are thus selected. Based on our scheme, the subcarrier 18 is the final subcarrier, which can reflect the high sensitivity based on in Fig. 7.7.

Discrete Wavelet Transform

Different from FFT and short time Fourier transform (STFT), discrete wavelet transform (DWT) can implement a time-frequency representation of data, which not only provide the optimal resolution both the time and frequency domains but also obtain multiscale analysis of the data. Based on DWT, the phase difference data after the subcarrier selection can be decomposed into an approximation coefficient vector with low-pass filter and a detail coefficient vector with high-pass filter. In fact, the approximation coefficient vector represents the basic shape of the input signal with the large scale characteristic, while the detail coefficient vector describes the high frequency noise and the detail information with the small scale characteristic. In wavelet decomposition, the following steps recursively split the previous approximation coefficient and detail coefficient into two new coefficients based on the same scheme [117]. After L steps, the DWT can obtain an approximation coefficient α^L and a sequence of detail coefficients $\beta^1, \beta^2, ...\beta^L$. We can compute the DWT

coefficients as the following,

$$\alpha_k^{(L)} = \sum_{n \in Z} \Delta\angle\widetilde{CSI}(n)\phi_{n-2^L k}^{(L)} \qquad L \in Z \tag{7.16}$$

$$\beta_k^{(l)} = \sum_{n \in Z} \Delta\angle\widetilde{CSI}(n)\psi_{n-2^l k}^{l} \qquad l \in \{1, 2, ...L\} \tag{7.17}$$

where $\Delta\angle\widetilde{CSI}(n)$ is the phase difference data after the subcarrier selection, Z is the integer set, ϕ's and ψ's are the wavelet basis functions, which are traditionally orthogonal to each other. The phase difference data $\Delta\angle\widetilde{CSI}(n)$ can be approximated by using inverse DWT,

$$\Delta\angle\widetilde{CSI}(n) = \sum_{k \in Z} \alpha_k^{(L)}\phi_{n-2^L k}^{(L)} + \sum_{l=1}^{L}\sum_{k \in Z} \beta_k^{(l)}\psi_{n-2^l k}^{l}, \tag{7.18}$$

In our PhaseBeat system, the DWT is leveraged to remove high frequency noise from the collected CSI phase difference data. Moreover, the approximation coefficient α^L is used to extract the breathing rate, while the sum of detail coefficients $\beta^{L-1} + \beta^L$ is employed to estimate the heart rate, where L is set as 4 in our system. As shown in Fig. 7.8, for the original signal, we first implement the DWT based decomposition recursively by four levels with Daubechies(db) wavelet filter. Because we obtain 20 Hz sampling rate after the data calibration and the sampling rate is halved for every step decomposition, the detail coefficient β^1 and the approximation coefficient α^1 have the frequency range from 10 Hz to 5 Hz and 0 Hz to 5Hz. Then, the approximation coefficient α^4 has the frequency is 0 Hz to 0.625 Hz to obtain the denoised breathing signal, while the sum of detail coefficients $\beta^3 + \beta^4$ has the range from 0.625 Hz to 2.5 Hz to restructured the heart signal;

7.3.3 Breathing Rate Estimation

Peak Detection for Single Person

As breathing signal is a small periodic movement of inhaling and exhaling, the phase difference data can extract the periodic change. Although FFT based method can be used to effectively estimate the breathing rate, its peak cannot obtain the accurate frequency estimate, because the

frequency resolution depends on the window size of FFT. It is noticed that if the window size becomes larger, the estimated accuracy is improved, but it leads to a lower time domain resolution. Thus, our PhaseBeat system leverages the peak detection to estimate the breathing rate based on the approximation coefficient α^L for improving the estimated accuracy.

However, we find that the approximation coefficient α^L still includes the fake peak that is not the true peak but it has larger values than its two neighboring samples. To avoid the fake peak, we leverage the moving window method to obtain the true peak, where the window size is set as 51 samples based on humans maximum breathing periodic [116]. Then, we can find all peaks by identifying whether the medium of all samples in the window is the maximum value or not. After peak detection, all peak-to-peak intervals are averaged to obtain the final periodic of breathing signal this time, which is defined as P. Thus, the estimated breathing rate can be computed by $60/P$ bpm.

FFT for Multiple Persons

Breathing rate estimation for multiple persons becomes challenging based on the approximation coefficient α^L, because the reflection components of the received signal are from multiple independent movements of the breathing chests. Thus, the peak-to-peak detection method cannot be available for the approximation coefficient α^L, which is not a clear periodic signal. Thus, we can leverage FFT based method to transfer the approximate the approximation coefficient α^L in the time domain to the frequency domain to estimate the breathing frequencies from multiple persons. Fig. 7.9 shows breathing rate estimation for two persons. We can notice that two estimated frequencies are 0.2 Hz and 0.3 Hz, respectively, which are approximately the ground truths. It demonstrates our method for PhaseBeat system is effective for multiple estimation.

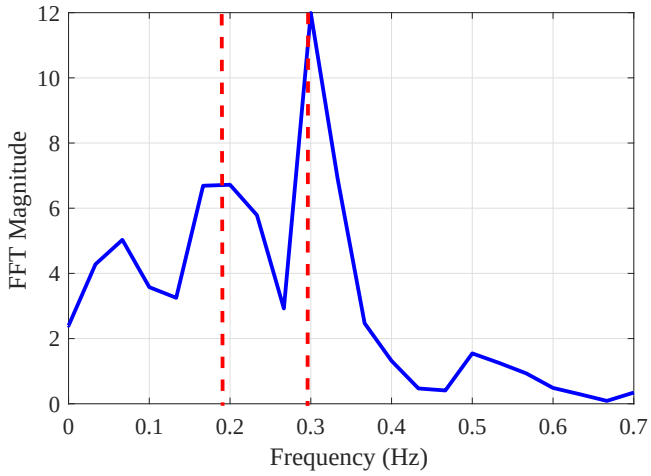


Figure 7.9: Breathing rate estimation for two persons

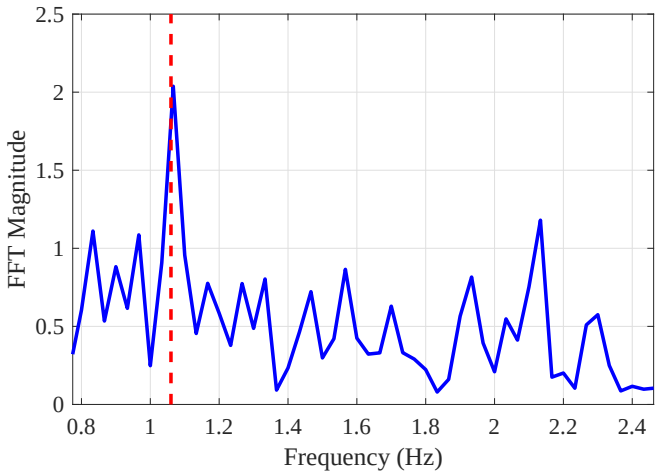


Figure 7.10: Heart rate estimation based on FFT

7.3.4 Heart Rate Estimation

FFT Scheme based Heart Rate Estimation

Heart rate can express the person's health condition and vital sign. Similar to breathing signal, the heart signal is also periodic, but its magnitude is greatly weak. Traditionally, breathing signal is orders of magnitude larger than the heart signal because of the small change of blood vessels from heart beating such as diastole and systole. Thus, the minute heart signal detection is challenging. Moreover, due to the stronger breathing signal, its frequency can leak to nearby frequencies, which

will mask the heart signal. Also, the harmonic wave of breathing signal will influence the heart rate detection.

In our PhaseBeat system, we need to leverage directional antenna as the transmitter to improve the power of the reflection signal, and then the sum of detail coefficients $\beta^{L-1} + \beta^{L}$ based on wavelet decomposition is employed to estimate the heart rate. When the level of decomposition is $L = 4$, the frequency range is between 0.625 Hz and 2.5 Hz, which filters out the breathing frequency range that is about between 0.17 Hz and 0.62 Hz and higher noise. Finally, we can leverage FFT based method to transform the sum of detail coefficients $\beta^{L-1} + \beta^{L}$ to the frequency domain for estimating the heart rate. To improve the frequency resolution, we leverage the method [110] to estimate the heart rate. After finding the peak of FFT, we use the three bins including the peak bin and its two adjacent bins, where an inverse FFT is leveraged to obtain a complex time-domain signal. The heart rate is estimated by computing the phase of the signal. Fig. 7.10 shows the heart rate estimation based on FFT. We can see that the estimated frequency is 1.07 Hz, while the ground truth is 1.06 Hz from the measurements of commercial fingertip pulse sensor for 30 s. The heart rate estimated error is 0.01 Hz, that is 0.6 bpm. It demonstrates that my method can obtain the higher accuracy of heart rate estimation.

7.4 Experimental Study

7.4.1 Test Configuration

In this section, we implement the experimental study with PhaseBeat in the 5GHz band. In the experiments, we leverage a desktop computer as an access point and a Lenovo laptop as a mobile device, both equipped with an Intel 5300 NIC. In fact, we employ the desktop computer instead of the commodity routers that are not equipped with the Intel 5300 NIC nowadays. Our PhaseBeat system is implemented on the Ubuntu desktop 14.04 LTS OS for both the access point and the mobile device. For the access point, it is set in monitor model and the distance between two adjacent antennas is d = 2.68 cm, which is half of a wavelength for the 5GHz band. For the mobile device, it is set in injection model, which transmits packets at 400 packets per second using only

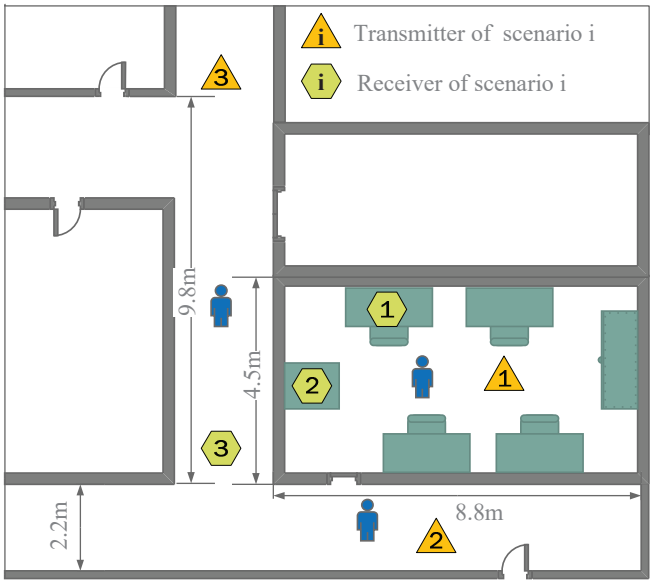


Figure 7.11: Experimental setup

one antenna. Then, we extract CSI phase difference data between two adjacent antennas at the receiver for vital signals estimation.

We implement our experiments with the total of four persons over three months. The test scenarios include a computer laboratory and a corridor in Fig. 7.11. We have three setups for the two environments. The first setup is within the laboratory with 4.5 $\times$ 8.8 m^2 room. Also, there are many tables and PCs crowded, which block parts of the LOS paths and form a complex radio propagation environment. The second setup is through-wall scenario, where the person is on the transmitter side. The final setup is the long corridor with the length 20 m, where we set the longest distance as 11 m for locating the the receiver and the transmitter. We use omnidirectional antennas for the receiver and the transmitter for breathing estimation at all setup scenarios. However, due to the minute signal change of heart signals, we leverage the directional antennas to increase the power of the transmitter for the first setup scenario to estimate the heart signal. Moreover, we leverage the NEULOG Respiration and a fingertip pulse oximeter to record the ground truths of the breathing and heart rates.

7.4.2 Performance of Breathing and Heart Rate Estimation

Fig. 7.12 shows the CDF of estimation error for the performance of breathing rate estimation. We use the amplitude based method [116] as the benchmark to compare with our PhaseBeat system. We can see that two systems have same medium estimate error about 0.25 bpm. However, we can see that for PhaseBeat, 90% of the test data have an estimated error under 0.5 bpm, while 70% of the test data for the amplitude based method have an estimated error under 0.5 bpm. Moreover, the maximum estimation errors for PhaseBeat and the amplitude based method are 0.85 bpm and 1.7 bpm, respectively. Therefore, our PhaseBeat system obtains much better performance than the amplitude based method for breathing rate estimation.

Fig. 7.13 shows the CDF of estimation error for the performance of heart rate estimation. For heart signal detection, we need to leverage the directional antenna at the transmitter to improve the power. Based on corollary 3.1, we can notice that the change of CSI phase difference data becomes larger, while the change of CSI amplitude data is small, even we cannot observe the periodic signal. Thus, we only show the phase difference data for heart signal estimation. In Fig. 7.13, we can find that PhaseBeat system has the medium estimate error about 1 bpm, while 80% of the test data have an estimated error under 2.5 bpm. Moreover, the maximum estimation errors for PhaseBeat is about 10 bpm. We notice that the estimated accuracy of heart rate is lower than the breathing rate estimation because of minute heart signal and the harmonic wave of breathing signal.

Fig. 7.14 shows the accuracy of breathing and heart rates estimation for different sampling frequencies. For data calibration part, we leverage 400 Hz sampling frequency to estimate the vital signs, which aims to accurately detect the heart signal. As is shown in the Fig. 7.14, we can find that the breathing rate estimation have the similar high accuracy about 98% for different sampling frequencies. However, the accuracy of the heart rate estimation is only 88% for sampling frequency with 20 Hz, while it can obtain 95% for 400 Hz sampling. Thus, we choose the 400 Hz sampling for PhaseBeat system, which is used for the following extensively experimental data with breathing rate estimation for different factors.

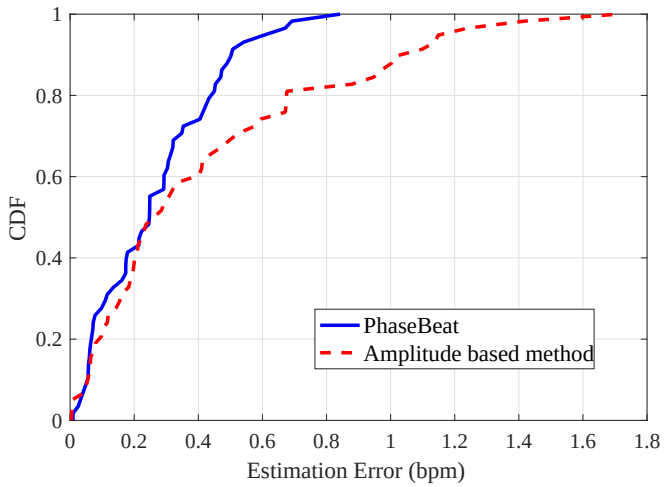


Figure 7.12: Performance of breathing rate estimation

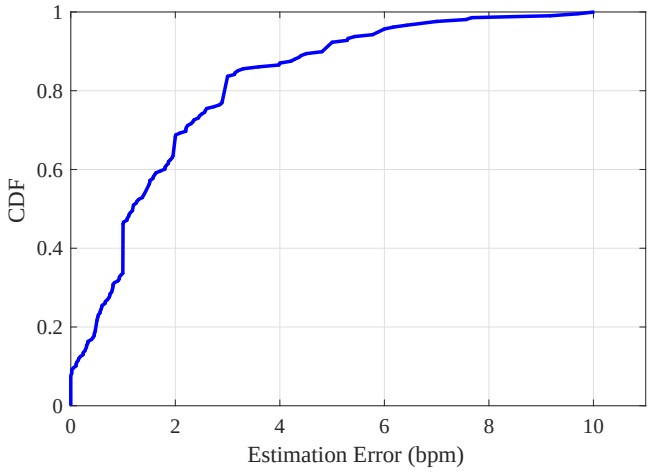


Figure 7.13: Performance of heart rate estimation

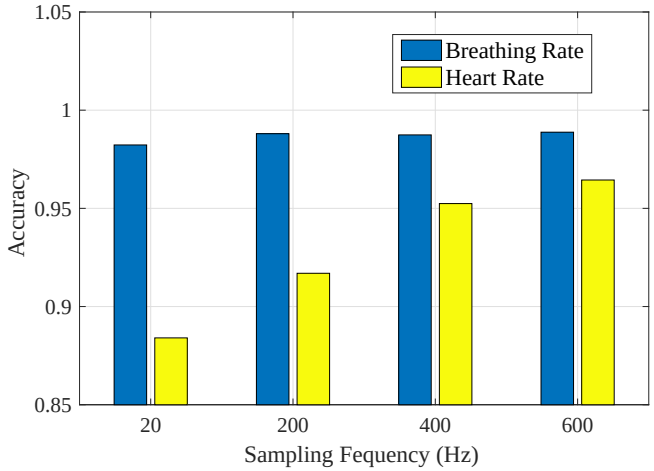


Figure 7.14: Accuracy of breathing and heart rates estimation for different sampling frequency

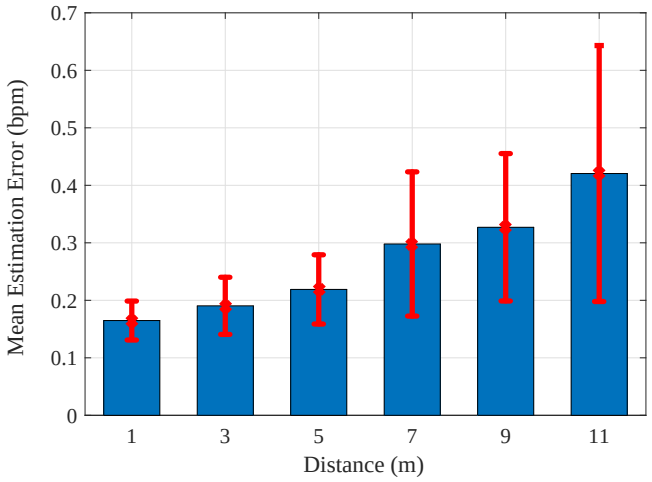


Figure 7.15: Impact of the distance between the transmitter and the receiver for the long corridor

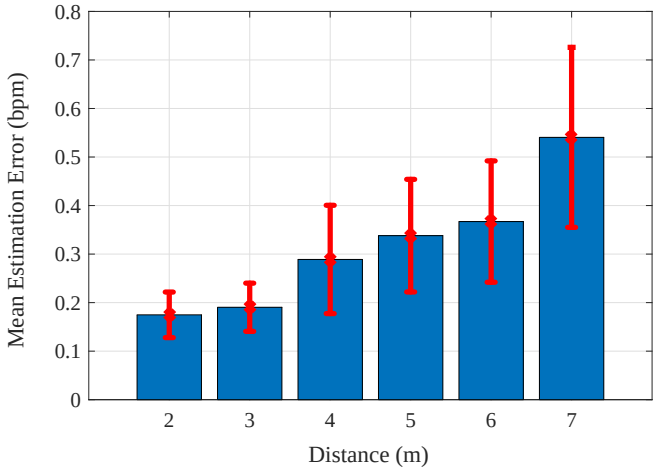


Figure 7.16: Impact of the distance between the transmitter and the receiver for through-wall scenario

7.4.3 Impact of Various Factors

Impact of the Distance between the Transmitter and the Receiver

Fig. 7.15 and Fig. 7.16 show the impact of the distance between the transmitter and the receiver for the long corridor and through-wall scenario, respectively. It is noticed that with the increase of the distance between the transmitter and the receiver, mean estimation error is also increased. This is because the reflection signal is reduced for long distance between the transmitter and the receiver, which influences the change of phase difference data. Moreover, we can see that the mean estimation error with the same distance for through-wall scenario is larger than that for the long

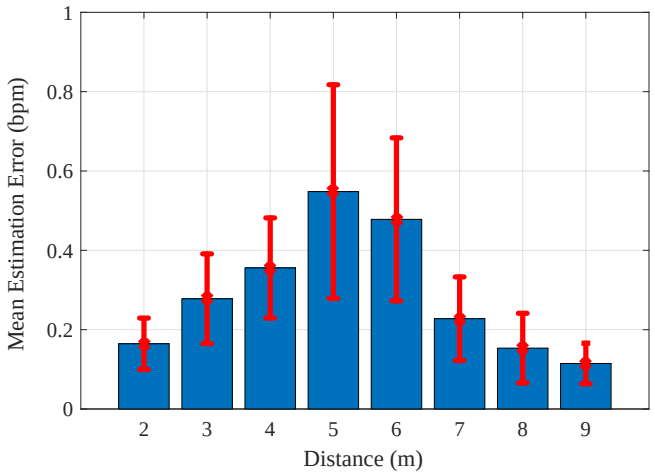


Figure 7.17: Impact of the distance between the user and the receiver

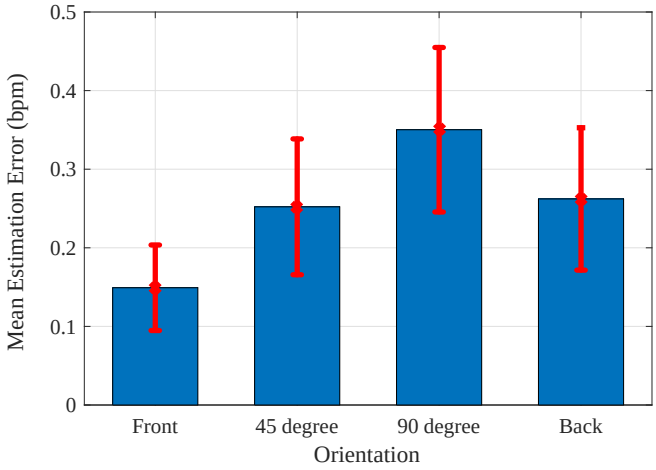


Figure 7.18: The impact of user orientation relative to the receiver

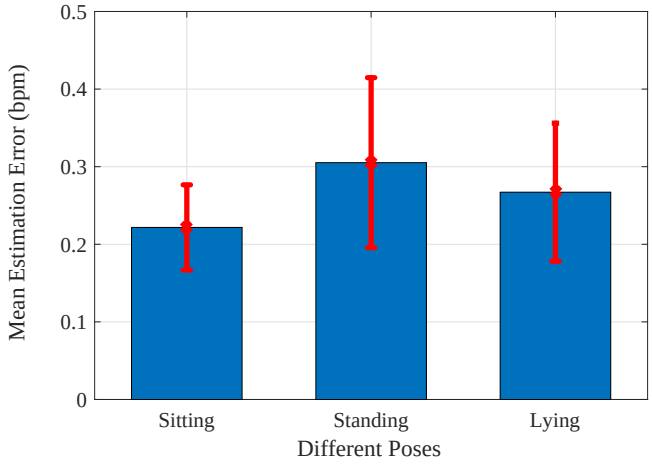


Figure 7.19: Impact of different poses

corridor. For example, when the distance is 7m, the mean estimation errors for the long corridor and the through-wall scenario are 0.3 bpm and 0.52 bpm, respectively. It is because the signal for the through-wall scenario has larger attenuation than that for the long corridor.

Impact of the Distance between User and the Receiver

Fig. 7.17 shows the distance between the user and the receiver for the long corridor. We can notice that when the user locates in the middle of the transmitter and the receiver, the mean estimation error is the maximum with about 0.52 bpm. In addition, when the user is in the side of transmitter or the receiver, the estimation error is the minimum values with about 0.1 bpm and 0.15 bpm in the sides of transmitter and the receiver, respectively. When the user is far away with WiFi devices, the reflection signal from the transmitter is greatly weaken, which influences the phase difference data.

Impact of User Orientation Relative to the Receiver

Fig. 7.18 shows the impact of user orientation relative to the receiver within the laboratory. We consider four cases including front (0 degree), 45 degree, 90 degree and back (180 degree). As is shown in the Fig. 7.14, we can see that for 90 degree, mean estimation error is the maximal value with 0.3 bpm, while for the front orientation relative to the receiver, we can obtain the minimum value with 0.14 bpm. When the user orientation relative to the receiver is the front or the back, the reflection component of the wireless signal can mainly capture the movement chest of body such as inhaling or exhaling. Thus, we can obtain low mean estimation errors.

Impact of Different Poses

Fig. 7.19 shows the impact of different poses within the laboratory. We consider three common poses for a stationary person such as sitting, standing and lying. As is shown in the Fig. 7.19, for standing pose, mean estimation error with 0.31 bpm is larger than other cases such as sitting with

0.22 bpm and lying with 0.26 bpm. This is because the chest of the person will have less reflection from the wireless signal for standing pose.

7.5 Related Work

This work is closely related to two categories of vital signs monitoring, i.e., sensor based and RF signal based, which are discussed in the following.

Sensor based systems for vital signs monitoring leverages the special hardware attached to the person body. Typically, the special devices, such as the capnometer that can measure carbon dioxide (CO2) concentrations in respired gases, are leveraged to monitor patients breathing rate in hospital [108]. However, they are uncomforted for the patient wearing them, which are mainly used in clinical environments. Photoplethysmography (PPG) is an optical technique to measure the blood volume changes in the tissues by detecting light absorption changes, which requires the sensors attached to persons finger such as pulse oximeters [109]. Moreover, smartphone can utilize the camera to measure light changes from the video frames. Then, the pixel of the frame is transformed into RGB components, which can extract the PPG signal to estimate the heart rate [118]. Recently, the smartphones can measure the breathing rate by leveraging the built-in accelerometer, gyroscope [119] and microphone [120], which require persons to place smartphones near-by and wear sensors in the monitoring environment. These techniques, however, leverage attached sensors, which cannot be applied for remote monitoring vital signs.

RF based systems for vital signs monitoring leverages wireless signal to extract the breathing-induced chest change of a person, which is mainly based on radar and WiFi techniques. For radar based vital signs monitoring, some techniques such as the Doppler radar [111, 112] and the ultra-wideband radar [113] are leveraged to monitor vital signs, which require the special hardware with high frequency and the high cost. Recent work leverages frequency modulated continuous wave (FMCW) radar to estimate breathing and heart rates, even for multiple person subjects in parallel [110]. But the system requires a custom hardware with a large bandwidth from 5.46 GHz to 7.25

GHz. For WiFi based vital signs monitoring, UbiBreathe system leverages WiFi RSS for breathing rate estimation, which, however, requires the device placed in the line of sight path between the transmitter and the receiver for monitoring the breathing signal [115]. Furthermore, based on RSS, mmVital can use 60 GHz millimeter wave (mmWave) signal for breathing and heart rates estimation with the larger bandwidth about 7GHz citeMillimeter. Its techniques cannot monitor the longer distance and require high gain directional antennas for the transmitter and the receiver. Recently, the authors leverage the amplitudes of CSI data of WiFi to track vital signs [116]. This work is mainly to monitor the vital signs when a person is sleeping.

The PhaseBeat system is motivated by these interesting prior works. To the best of our knowledge, it is the first to leverage CSI phase difference data to monitor the breathing and heart rates with commodity WiFi devices, which can obtain the higher estimated accuracy of vital signs. And this work analyzes and proves the phase difference data is periodic and has the same frequency as the the breathing signal.

7.6 Conclusions

In this chapter, we presented PhaseBeat, CSI phase difference data to monitor breathing and heart beats with commodity WiFi device. PhaseBeat system leveraged CSI phase difference data to extract the periodic signal from the change in the chest of a person such as inhaling and exhaling. Then, We implemented data preprocessing including environment detection, data calibration, subcarrier selection and discrete wavelet transform. Moreover, we employed the peak detection approach for breathing rate estimation and FFT based method for heart rate estimation. We conducted with the experiments with three setups such as the laboratory, through-wall scenario and the long corridor. The results showed that the PhaseBeat system can obtain better performance than the amplitude based method. In the future, we will research how to use the omnidirectional antennas as the transmitter to detect the minute heart signal by leveraging phase difference data.

Chapter 8

TensorBeat: Tensor Decomposition for Monitoring Multi-Person Breathing Beats with Commodity WiFi

8.1 Introduction

Recently, the authors in [116] use the CSI amplitude data to monitor breathing and heart signals, which requires the person to remain in the sleeping mode. However, the measured CSI phase data has not been fully exploited in prior works, largely due to random phase fluctuation resulting from asynchronous times and frequencies of the transmitter and receiver. For multiple person breathing monitoring, because the reflected components in the received signal are from the chests of multiple persons, each moves slightly due to breathing and the movements are independent. Thus, vital signs monitoring and estimation for multiple persons still remains a challenging and open problem.

In this chapter, we propose to utilize CSI phase difference data between antenna pairs to monitor the breathing rates of multiple persons. First, we show that when the person is in a stationary state, such as standing, sitting, or sleeping, the CSI phase difference data is highly stable in consecutively received packets, which can be leveraged for extracting the small, periodic breathing signal hidden in the received WiFi signal. In fact, phase difference is more robust than amplitude, which usually exhibits large fluctuations because of the attenuation over the link distance, obstacles, and the multipath effect. Moreover, the phase difference data captures and preserves the periodicity of breathing, when the wireless signal is reflected from the patients' chests. To extract the weak

breathing signal, and more important, to distinguish among multiple persons, we propose to employ a tensor decomposition method to handle the phase difference data [121, 122, 123]. We create the CSI tensor data by increasing the dimension of CSI data from one to three, which can be used to effectively separate different breathing signals in different clusters.

We present a system termed *TensorBeat*, *Tensor* decomposition for estimating multiple persons breathing *Beats*, by exploiting CSI phase difference data. TensorBeat operates as follows. First, it obtains 60 CSI phase difference data from antenna pairs 1 and 2, and 2 and 3, at the receiver. Next, a data preprocessing procedure is applied to the measured phase difference data, including data calibration and Hankelization. In the data calibration phase, the DC component and high frequency noises are removed. In the Hankelization phase, a two dimensional Hankel matrix is created based on the calibrated phase difference data from every subcarrier, and the rank of the Hankel matrix is analyzed. Then, we adopt Canonical Polyadic (CP) decomposition for estimating multiple persons breathing signs, and prove the uniqueness of the proposed CSI tensor. After CP decomposition, we obtain twice amount of breathing signals, which, however, are randomly indexed. We thus design a stable signal matching algorithm (for the stable roommate problem [124]) to identify the decomposed signal pairs for each person. Finally, we combine the decomposed signals in each pair and employ a peak detection method to estimate the breathing rate for each person.

We implement TensorBeat on commodity 5 GHz WiFi devices and verify its performance with five persons over six months in different indoor environments, such as a computer laboratory, a through-wall scenario, and a long corridor. The results show that the proposed TensorBeat system can achieve high accuracy and high success rates for multiple persons breathing rates estimation. Moreover, we demonstrate the robustness of the proposed TensorBeat system for monitoring multiple persons' breathing beats under a wide range of environmental parameters.

The main contributions of this chapter are summarized as follows.

1. We theoretically and experimentally verify the feasibility of leveraging CSI phase difference for breathing monitoring. In particular, we analyze the measured phase errors in detail and

demonstrate that phase difference data is stable and can be used to extract breathing signs. To the best of our knowledge, we are the first to leverage phase difference for multiple persons breathing rate estimation.

2. We are also the first to apply tensor decomposition for RF sensing based vital signs monitoring. We use the phase difference data to create a CSI tensor for all subcarrier at the three antennas of the WiFi receiver. We then incorporate CP decomposition to obtain the desired breathing signals. A stable signal matching algorithm is developed to match the decomposed signals for each person, while a peak detection method is used to estimate multiple persons' breathing rates.

3. We prototype the TensorBeat system with commodity 5 GHz WiFi devices and demonstrate its superior performance in different indoor environments with extensive experiments. The results show that the proposed TensorBeat system can achieve very high accuracy and high success rates for multiple persons breathing rate estimation.

The remainder of this chapter is organized as follows. The preliminaries and phase difference analysis are provided in Section 8.2. We present the TensorBeat system design and performance analysis in Section 8.3 and verify its performance with extensive experiments in Section 8.4. We provide the related work in Section 8.5. Section 8.6 concludes this chapter.

8.2 Preliminaries and Phase Difference Information

8.2.1 Tensor Decomposition Preliminaries

A tensor is considered as a multidimensional array [125]. The dimensions of the tensor are called as modes, and the order of the tensor is the number of the modes. For example, the N-order tensor is a N-mode tensor. Moreover, It is noticed that a first-order tensor is a vector, a second-order tensor is a matrix, and a third-order tensor is a cubic structure. Higher-order tensors with $(N \geq 3)$ have a wide range of applications such as data mining, brain data analysis, recommendation systems, wireless communications, computer vision, and healthcare and medical applications [121].

For higher-order tensors, they face various computational challenging because of the exponential increase in time and space complexity with the orders increase of tensors. This leads to the curse of dimensionality. Fortunately, tensor decomposition as one powerful tool is leveraged for alleviating the curve by decomposing high-order tensors into a limited number of factors. Also, it can obtain hidden feature components, thus extracting physical insight of higher-order tensors. Two main tensor decompositions are tucker decomposition and CP decomposition [125]. We consider CP decomposition for multiple persons breathing rate estimation because it can easily obtain the unique solution [125]. On the other hand, we will provide some necessary definitions and equations of tensor decomposition, which can be used for our proposed algorithm.

Definition 1. *(Frobenius Norm of a Tensor). The Frobenius norm of a tensor $\chi \in \mathbb{K}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all its elements, which is defined by*

$$\|\chi\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1,i_2 \cdots i_N}^2}. \tag{8.1}$$

where $\mathbb{K}$ stands for $\mathbb{R}$ or $\mathbb{C}$.

Definition 2. *(Kronecker Product). The Kronecker product of matrics $\boldsymbol{A} \in \mathbb{K}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{K}^{M \times N}$ is denoted as $\boldsymbol{A} \otimes \boldsymbol{B}$. The result is an $(IM) \times (JN)$ matrix, which is defined by*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}. \tag{8.2}$$

Definition 3. *(Khatri-Rao Product). The Khatri-Rao product of $A \in \mathbb{K}^{I \times J}$ and $B \in \mathbb{K}^{M \times J}$ is denoted as $A \odot B$. It is the column-wise Kronecker product with the size $(IM) \times J$, which is defined by*

$$\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1, \boldsymbol{a}_2 \otimes \boldsymbol{b}_2, \cdots, \boldsymbol{a}_J \otimes \boldsymbol{b}_J]. \tag{8.3}$$

Definition 4. *(Hadamard product). The Hadamard product of* $\boldsymbol{A} \in \mathbb{K}^{I \times J}$ *and* $\boldsymbol{B} \in \mathbb{K}^{I \times J}$ *is denoted as* $\boldsymbol{A} * \boldsymbol{B}$*. It is the elementwise matrix product with the size* $I \times J$*, which is defined by*

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & \vdots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{IJ} \end{bmatrix}. \tag{8.4}$$

8.2.2 Phase Difference Information for Multiple Breathing Monitoring

Breathing rate estimation for multiple persons is a challenging problem, because the reflected components in the received signal are from the chests of multiple persons, each moves slightly due to breathing and the movements are independent. Thus, the peak-to-peak detection method cannot be effective for detecting the multiple breathing signals from the received signal. The aggregated breathing signal from multiple persons is not a clearly periodic signal anymore. Fig. 8.1 shows the detected breathing signals for one person (the upper plot) and three persons (the lower plot). We can see that for one person, the breathing signal exhibits a noticeable periodicity. So the breathing rate can be estimated by peak detection after removing the noise. However, the aggregated breathing signal of three persons does not show noticeable periodicity for packet 400 to 600. Traditional FFT based methods can transform the received signal from the time domain to the frequency domain to estimate the breathing frequencies from multiple persons. Fig. 8.2 shows the breathing rate estimation for one person (the upper plot) and three persons (the lower plot) with the FFT method. We can see that the estimated frequency for one person is 0.2 Hz, which is almost the same as the true breathing rate. However, for three-person breathing rate estimation, the FFT curve only has two peaks, and the estimated breathing rates are much less accurate. In particular, the third peak cannot be estimated. This is because FFT based methods require a larger window size to improve the frequency resolution. We show that the proposed tensor decomposition based method is highly effective for multi-person breathing rate estimation in the following section.

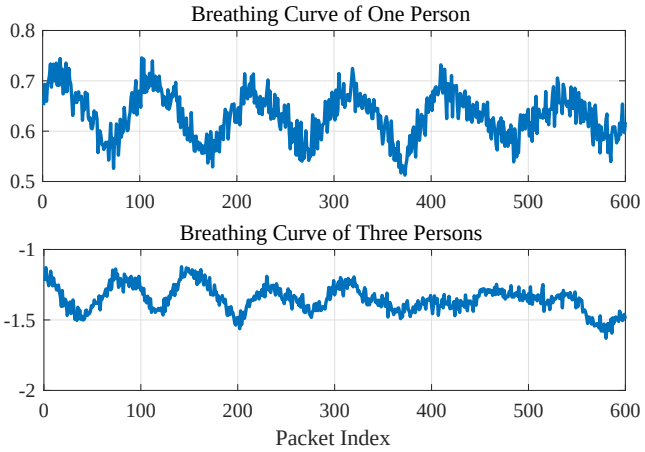


Figure 8.1: Detected breathing signals for one person (the upper plot) and three persons (the lower plot).

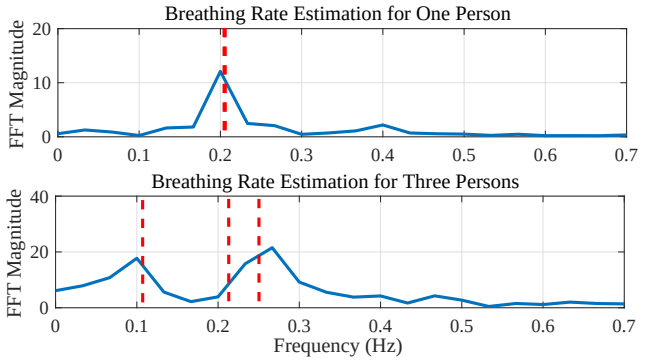


Figure 8.2: Breathing rate estimation for one person (the upper plot) and three persons (the lower plot) based on FFT.

8.3.1 TensorBeat System Architecture

The main idea of the proposed TensorBeat system is to estimate multi-person breathing rates by employing a tensor decomposition method. To obtain CSI tensor data, we first create a two dimensional Hankel matrix with phase difference data from back-to-back received packets extracted from each subcarrier at each antenna. Then, by leveraging the phase differences from the 60 subcarriers, i.e., that between antennas 1 and 2, and between antennas 2 and 3, we can construct the third dimension of the CSI tensor data. The TensorBeat system will then leverage the created CSI tensor to estimate multi-person breathing signs. Our approach is motivated by two observations. First, for stationary modes of a person, such as standing, sitting, or sleeping, CSI phase difference from consecutively received packets is highly stable. It can thus be useful for extracting the periodic breathing signals. Second, the tensor decomposition method can effectively estimate multi-person breathing beats. We create the CSI tensor data by increasing the dimension of CSI data, from one dimension to three dimensions. The higher dimension CSI data is helpful to effectively separate different breathing signals by forming different clusters. This strategy is similar to the kernel method in traditional machine learning, such as SVM [18] or multiple hidden layers in deep learning [91, 64, 3].

As shown in Fig. 8.3, the TensorBeat system consists of four main modules: Data Extraction, Data Preprocessing, CP Decomposition, Signal Matching, and Breathing Rate Estimation. For Data Extraction, TensorBeat obtains 60 CSI phase difference data, 30 between antennas 1 and 2, and 30 between antennas 2 and 3, at the receiver with an off-the-shelf WiFi device. The Data Preprocessing module includes data calibration and Hankelization. Data calibration is implemented to remove the DC component and high frequency noises. Hankelization is to create a two dimensional Hankel matrix with phase difference data from each subcarrier for back-to-back received packets. The rank of the constructed Hankel matrix is then analyzed. We next apply CP decomposition to estimate multiple persons' breathing signals, and prove the uniqueness of the proposed CSI tensor.

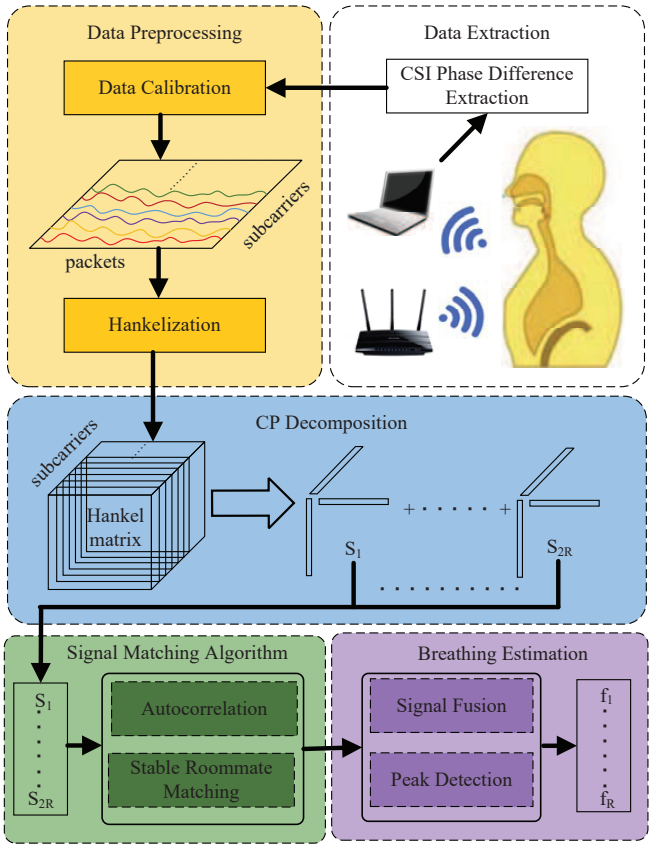


Figure 8.3: The TensorBeat system architecture.

For Signal Matching, we first compute the autocorrelation function of the decomposed signals, and incorporate a stable roommate matching algorithm to identify the decomposed signal pairs for each person, where a preference list is computed with the dynamic time warping (DTW) values of the autocorrelation signals. For Breathing Rate Estimation, we combine the decomposed signals in each pair and use the peak detection method to compute the breathing rate for each person.

In the remainder of this section, we present the design and analysis of each module of the TensorBeat system in detail.

8.3.2 Data Preprocessing

Data Calibration

We use a 20 Hz sampling rate to obtain 60 CSI phase difference data, 30 between antennas 1 and 2, and 30 between antennas 2 and 3, at the receiver with an off-the-shelf WiFi device at 5 GHz for

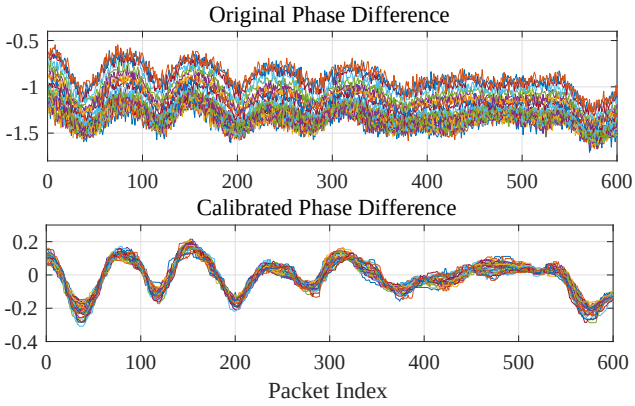


Figure 8.4: Data calibration: an example.

data extraction. Then, data calibration is applied to remove the DC component and high frequency noises. Because the DC component is also considered as a kind of signal, which may affect CSI tensor decomposition, TensorBeat adopts the Hampel filter to remove the DC component. Unlike traditional data calibration approaches that only remove the high frequency noise, we use the Hampel Filter for detrending the original CSI phase difference data to remove DC component. In fact, the Hampel Filter, which is set as a large sliding window with 150 samples wide and a small threshold of 0.001, is firstly used to extract the basic trend of the original data. Then, the detrended data is generated by subtracting the basic trend data from the original data. We also utilize the Hampel Filter to reduce the high frequency noise by using a sliding window of 6 samples wide and a threshold of 0.01.

Fig. 8.4 presents an example of data calibration. We can see that the original phase differences of all the subcarriers have both a DC component and high frequency noises. With the proposed data calibration approach, it can be seen that the DC components are readily removed and all the subcarriers demonstrate a similar calibrated signal over the 600 packet range with low noise. Such calibrated signal will then be used for estimating the breathing rates of multiple persons.

Hankelization

After data calibration, we obtain the CSI phase difference data matrix with a dimension of (number of packets $\times$ number of subcarriers). We then employ a Hankelization method to transform the

large CSI matrix into a CSI tensor by expanding the packets into an additional dimension [126]. Specifically, we rearrange the signals of each subcarrier into a 2-D Hankel matrix, so that the signals from all the 60 subcarriers can be considered as a 3-Dimensional tensor. Define H_r as the constructed Hankel matrix with the size $I \times J$ for subcarrier r, which is created by mapping N packets onto the Hankel matrix with $N = I + J - 1$. We consider the Hankel matrix with size $I = J = \frac{N+1}{2}$. We thus obtain the Hankel matrix H_r for subcarrier r, as

$$\boldsymbol{H}_r = \begin{bmatrix} h_r(0) & h_r(1) & \dots & h_r(\frac{N+1}{2}-1) \\ h_r(1) & h_r(2) & \dots & h_r(\frac{N+1}{2}) \\ \vdots & \vdots & \vdots & \vdots \\ h_r(\frac{N+1}{2}-1) & h_r(\frac{N+1}{2}) & \dots & h_r(N-1) \end{bmatrix}, \tag{8.5}$$

where $h_r(i)$ is the calibrated phase difference data from subcarrier r for packet i. In our experiments, we set $N = 599$ and $I = J = 300$. To determine the number of components needed for CSI tensor decomposition, we provide the following theorem for estimating R breathing signals.

Theorem 4. *If there are R breathing signals in an indoor monitoring environment, the constructed Hankel matrix $\boldsymbol{H}_r$ for subcarrier r has a rank of $2R$ when noise is negligible.*

Proof. When analyzing signal data structure, we assume the noise is negligible. Moreover, let the ith breathing signal be represented as $S_i(t) = A_i \cos(w_i t + \varphi_i)$. The observed signal from a subcarrier can be represented by [127]

$$Y(t) = \sum_{i=1}^{i=R} K_i S_i(t) = \sum_{i=1}^{i=R} \hat{K}_i \cos(w_i t + \varphi_i), \tag{8.6}$$

where K_i is the coefficient for breathing signal i and the new coefficient $\hat{K}_i = K_i A_i$. The ith component of $Y(t)$, $\hat{K}_i \cos(w_i t + \varphi_i)$, can be decomposed using Euler's formula. We have

$$\begin{aligned} \hat{K}_i \cos(w_i t + \varphi_i) &= \frac{\hat{K}_i}{2} \exp(j(w_i t + \varphi_i)) + \frac{\hat{K}_i}{2} \exp(j(-w_i t - \varphi_i)) \\ &= \frac{\hat{K}_i}{2} \exp(j\varphi_i) \exp(jw_i t) + \frac{\hat{K}_i}{2} \exp(-j\varphi_i) \exp(-jw_i t). \end{aligned} \tag{8.7}$$

Each breathing signal can be separated into two exponential signals with different coefficients. Combining all the R breathing signals, we have

$$\begin{aligned} Y(t) &= \sum_{i=1}^{R} \left(\frac{\hat{K}_i}{2} \exp(j\varphi_i) \exp(jw_i t) + \frac{\hat{K}_i}{2} \exp(-j\varphi_i) \exp(-jw_i t) \right) \\ &= \sum_{i=1}^{2R} \tilde{K}_i Z_i^t, \end{aligned} \tag{8.8}$$

where the updated signal Z_i^t is denoted as $Z_i^t = \exp(\pm jw_i t)$, and $\tilde{K}_i = \frac{\hat{K}_i}{2} \exp(\pm j\varphi_i)$ is its coefficient. For packets received at discrete times, we represent the received signal as $Y(n) = \sum_{i=1}^{2R} \tilde{K}_i Z_i^n$. Note that the combined signal can be considered as an exponential polynomial with $2R$ different exponential terms. Map signal $Y(n)$ for $n = 1, 2, \cdots, N$ into a Hankel matrix with size $I = J = \frac{N+1}{2}$, we have

$$\boldsymbol{H}_r = \begin{bmatrix} \sum_{i=1}^{2R} \tilde{K}_i Z_i^0 & \sum_{i=1}^{2R} \tilde{K}_i Z_i^1 & \cdots & \sum_{i=1}^{2R} \tilde{K}_i Z_i^{\frac{N+1}{2}-1} \\ \sum_{i=1}^{2R} \tilde{K}_i Z_i^1 & \sum_{i=1}^{2R} \tilde{K}_i Z_i^2 & \cdots & \sum_{i=1}^{2R} \tilde{K}_i Z_i^{\frac{N+1}{2}} \\ \vdots & \vdots & \cdots & \vdots \\ \sum_{i=1}^{2R} \tilde{K}_i Z_i^{\frac{N+1}{2}-1} & \sum_{i=1}^{2R} \tilde{K}_i Z_i^{\frac{N+1}{2}} & \cdots & \sum_{i=1}^{2R} \tilde{K}_i Z_i^{N-1} \end{bmatrix}. \tag{8.9}$$

We can see that the Hankel matrix can be decomposed with Vandermonde decomposition [126], as

$$\boldsymbol{H}_r = \boldsymbol{V}_r \cdot \text{diag}(\tilde{K}_1, \tilde{K}_1, \cdots, \tilde{K}_{2R}) \cdot \tilde{\boldsymbol{V}}_r^T, \tag{8.10}$$

where the Vandermode matrices $\boldsymbol{V}_r \in \mathbb{K}^{\frac{N+1}{2} \times 2R}$ and $\tilde{\boldsymbol{V}}_r \in \mathbb{K}^{\frac{N+1}{2} \times 2R}$ are given by

$$\boldsymbol{V}_r = \tilde{\boldsymbol{V}}_r = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ Z_1 & Z_2 & \cdots & Z_{2R} \\ \vdots & \vdots & \cdots & \vdots \\ Z_1^{\frac{N+1}{2}-1} & Z_2^{\frac{N+1}{2}-1} & \cdots & Z_{2R}^{\frac{N+1}{2}-1} \end{bmatrix}. \tag{8.11}$$

Because a Vandermode matrix is full rank, which is obtained by different poles, the rank of the Hankel matrix generated by R breathing signals is $2R$. □

According to Theorem 4, $2R$ signal components is required to separate the R breathing signals.

Next we consider the influence of measurement noise on the Hankel matrix $\boldsymbol{H}_r$. Because of noise, the Hankel matrix H_r is actually a full-rank matrix. However, Theorem 4 shows that the rank of the combined breathing signal is $2R$, meaning that the first $2R$ weighted decomposed components are much stronger than the remaining ones as long as the signal to noise ratio (SNR) is not very low. This shows that the Hankel matrix structure can be used to effectively separate breathing signals from white noise. Actually, the different signals will be well denoised and separated by using tensor decomposition, as to be discussed in Section 8.3.3.

8.3.3 Canonical Polyadic Decomposition

Once the CSI tensor is ready, we apply CP decomposition to estimate multiple persons' breathing signals. With CP decomposition, the CSI tensor data can be approximated as the sum of $2R$ rank-one tensors according to Theorem 4. Denote $\chi \in \mathbb{K}^{I \times J \times K}$ as a third-order CSI tensor, which can be obtained by the sum of three-way outer products as [125, 121]

$$\chi \approx \sum_{r=1}^{2R} a_r \circ b_r \circ c_r, \tag{8.12}$$

where a_r, b_r, c_r are the vectors at the rth position for the first, second, and third dimension, respectively, and $2R$ is the number of decomposition components, which is the approximation rank of the tensor based on CP decomposition [128, 129]. Their outer product is defined by

$$(a_r \circ b_r \circ c_r)(i,j,k) = a_r(i)b_r(j)c_r(k), \quad \text{for all} \quad i,j,k. \tag{8.13}$$

We consider factor matrices $\boldsymbol{A} = [a_1, a_2, \cdots, a_{2R}] \in \mathbb{K}^{I\times 2R}$, $\boldsymbol{B} = [b_1, b_2, \cdots, b_{2R}] \in \mathbb{K}^{J\times 2R}$, and $\boldsymbol{C} = [c_1, c_2, \cdots, c_{2R}] \in \mathbb{K}^{K\times 2R}$ as the combination of vectors from rank-one components. Moreover, define $\mathcal{X}_{(1)} \in \mathbb{K}^{I\times JK}$, $\mathcal{X}_{(2)} \in \mathbb{K}^{J\times IK}$, and $\mathcal{X}_{(3)} \in \mathbb{K}^{K\times IJ}$ as 1-mode, 2-mode, and 3-mode matricization of CSI tensor $\chi \in \mathbb{K}^{I\times J\times K}$, respectively, which are obtained by fixing one mode and arranging the slices of the rest of the modes into a long matrix [125]. Then, we can write the three matricized forms as

$$\mathcal{X}_{(1)} \approx \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T, \tag{8.14}$$

$$\mathcal{X}_{(2)} \approx \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^T, \tag{8.15}$$

$$\mathcal{X}_{(3)} \approx \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^T, \tag{8.16}$$

where $\odot$ denotes the Khatri-Rao product.

When the number of components $2R$ is given, we apply the Alternating Least Squares (ALS) algorithm, the most widely used algorithm for CP decomposition [125]. To decompose the CSI tensor, we minimize the square sum of the differences between the CSI tensor χ and the estimated tensor.

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \left\| \chi - \sum_{r=1}^{2R} a_r \circ b_r \circ c_r \right\|_F^2. \tag{8.17}$$

Note that (8.17) is not convex. However, the ALS algorithm can effectively solve the problem by fixing two of the factor matrices, to reduce the problem to a linear least squares problem with the

third factor matrix as variable. If we fix $\boldsymbol{B}$ and $\boldsymbol{C}$, we can rewrite problem (8.17) as

$$\min_{\boldsymbol{A}} \left\| \mathcal{X}_{(1)} - \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T \right\|_F^2 . \tag{8.18}$$

We can derive the optimal solution to problem (8.18) as $\boldsymbol{A} = \mathcal{X}_{(1)}[(\boldsymbol{C} \odot \boldsymbol{B})^T]^\dagger$. Applying the property of pseudoinverse of the Khatri-Rao product, it follows that

$$\boldsymbol{A} = \mathcal{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\boldsymbol{C}^T\boldsymbol{C} * \boldsymbol{B}^T\boldsymbol{B})^\dagger, \tag{8.19}$$

where $*$ denotes the Hadamard product. This equation only requires computing the pseudoinverse of a $2R \times 2R$ matrix rather than a $JK \times 2R$ matrix. Note that R is much smaller than J and K, thus the computing complexity can be greatly reduced. Similarly, we can obtain the optimal solutions for $\boldsymbol{B}$ and $\boldsymbol{C}$ as

$$\boldsymbol{B} = \mathcal{X}_{(2)}(\boldsymbol{C} \odot \boldsymbol{A})(\boldsymbol{C}^T\boldsymbol{C} * \boldsymbol{A}^T\boldsymbol{A})^\dagger \tag{8.20}$$

$$\boldsymbol{C} = \mathcal{X}_{(3)}(\boldsymbol{B} \odot \boldsymbol{A})(\boldsymbol{B}^T\boldsymbol{B} * \boldsymbol{A}^T\boldsymbol{A})^\dagger. \tag{8.21}$$

Applying ALS to CP decomposition, we obtain matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. To guarantee the effectiveness of the decomposed components, we next examine the uniqueness of CP decomposition. The basic theorem on the uniqueness of CP decomposition is given in [125], which is provided in the following.

Fact 1. *For tensor χ with rank L, if $k_A + k_B + k_C \geq 2L + 2$, then the CP decomposition of χ is unique, where $k_{\boldsymbol{A}}$, $k_{\boldsymbol{B}}$, and $k_{\boldsymbol{C}}$ denote the k-rank of matrix $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, respectively. Here k-rank means the maximum value k such that any k columns are linearly independent [125].*

Based on Fact 1, we have the following theorem for the CSI tensor.

Theorem 5. *For the proposed CSI tensor χ with rank $2R$, the CP decomposition of χ is unique.*

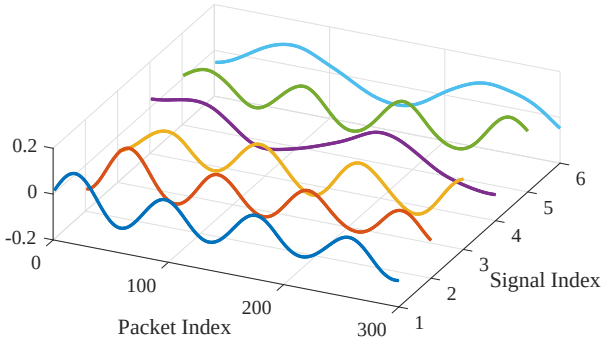


Figure 8.5: CP decomposition results for a CSI tensor of three persons.

Proof. The proposed CSI tensor χ is created by K Hankel matrix, where the rth Hankel matrix $\boldsymbol{H}_r$ is rank-2R according to Theorem 4. Thus, for the k-rank of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we have $k_{\boldsymbol{A}} = 2R$ and $k_{\boldsymbol{B}} = 2R$. On the other hand, because phase differences of subcarriers between antennas 1 and 2, and antennas 2 and 3 are independent, the k-rank of matrix $\boldsymbol{C}$ has $k_{\boldsymbol{C}} \geq 2$. Thus, the expression is $k_{\boldsymbol{A}} + k_{\boldsymbol{B}} + k_{\boldsymbol{C}} \geq 2R + 2R + 2 = 2(2R) + 2$, which satisfies the conditions in Theorem 1. This proofs the theorem. □

Theorem 5 indicates that the CP decomposition of the created CSI tensor is unique, which can be used to effectively estimate multiple breathing rates. In the proposed TensorBeat system, we leverages the matrix $\boldsymbol{A} = [a_1, a_2, \cdots, a_{2R}]$ as decomposed signals $S_1, S_2, \cdots, S_{2R}$. For example, Fig. 8.5 shows the results of CP decomposition for CSI tensor data from three persons ($R = 3$). We can see that there are six signals. Moreover, signals 1 and 2 are similar, signals 3 and 5 are similar, and signals 4 and 6 are similar. This is because CP decomposition cannot guarantee that similar signals are located in adjacent locations (i.e., the output signals are randomly indexed). Thus, we need to identify the signal pairs among the decomposed signals for each person, which will be addressed in Section 8.3.4.

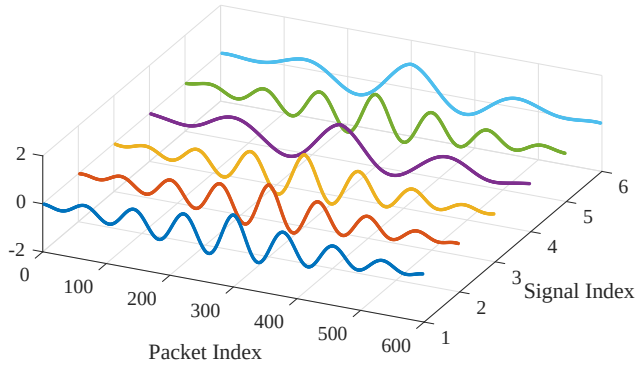


Figure 8.6: Autocorrelation of the decomposed breathing signals.

8.3.4 Signal Matching Algorithm

The CP decomposition of CSI tensor data yields $2R$ decomposed signals, i.e., $S_1, S_2, \cdots, S_{2R}$, which, however, are randomly indexed. In this section, we propose a signal matching algorithm to pair the two similar decomposed signals that belong to the same person. The main idea is to leverage the autocorrelation to strengthen the periodicity of decomposed signals and use the Dynamic Time Warping (DTW) method to compute the similarity value for any pair of signals. Finally, we apply the stable-roommate matching algorithm to pair the decomposed signals for each person, using the DTW values as the closeness metric. We introduce the proposed signal matching algorithm in the following.

Autocorrelation and Dynamic Time Warping

After CP decomposition of CSI tensor data, we first compute the autocorrelation function of the $2R$ decomposed signals to strengthen their periodicity. We evaluate the autocorrelation function of the decomposed signals for two reasons. The first is that the autocorrelation of a decomposed signal can increase the data length, which helps to improve the accuracy of the peak detection. Second, because the decomposed signals have phase shift and nonalignment, using the autocorrelation of decomposed signals can reduce such shifts and strengthen the periodicity of the decomposed signals. Fig. 8.6 shows the autocorrelation of the decomposed breathing signals produced by CP

decomposition. We can see that each autocorrelation signal exhibits a more obvious periodicity than that of the original decomposition signals. Moreover, the data length is increased from 300 to 600.

Furthermore, we employ the DTW approach to measure the distance between any pair of autocorrelation signals, which is different from the Euclidean distance method that computes the sum of distances from each value on one curve to the corresponding value on the other curve. Moreover, the Euclidean distance method believes that two autocorrelation signals with the same length are different as long as one of them has a small shift. However, DTW can automatically identify these shifts and provide the similar distance measurement between two autocorrelation signals by aligning the corresponding time series, thus overcoming the limitation of the Euclidean distance method.

With the autocorrelation signals, we design the DTW method for measuring their pairwise distance. Given two autocorrelation signals and a cost function, the DTW method seeks an alignment by matching each point in the first autocorrelation signal to one or more points in the second signal, thus minimizing the cost function for all points [57, 130, 131]. To reduce the computational complexity of DTW, we apply downsampling to the two autocorrelation signals, which leads to a reduced number of packets N'. Then, consider two downsampled autocorrelation signals $P_i = [P_i(0), P_i(1), \cdots, P_i(N'-1)]$ and $P_j = [P_j(0), P_j(1), \cdots, P_j(N'-1)]$, we need to find a warping path $W = [w_1, w_2, \cdots, w_L]$, where L is the length of the path, and the lth element of the warping path is $w_l = (m_l, n_l)$, where m and n are the packet index for the two downsampled autocorrelation signals. The objective is to minimize the total cost function by implementing the non-linear mapping between two downsampled autocorrelation signals P_i and P_j. The formulated

problem is given by

$$\min \quad \sum_{l=1}^{L} \|P_i(m_l) - P_j(n_l)\| \tag{8.22}$$

$$\text{s.t.} \quad (m_1, n_1) = (0, 0) \tag{8.23}$$

$$(m_L, n_L) = (N' - 1, N' - 1) \tag{8.24}$$

$$m_l \leq m_{l+1} \leq m_l + 1 \tag{8.25}$$

$$n_l \leq n_{l+1} \leq n_l + 1. \tag{8.26}$$

The objection function is to minimize the distance between two downsampled autocorrelation signals. The first and second constraints are boundary constraints, which require that the warping path starts at $P_i(0)$ and $P_j(0)$ and ends at $P_i(N' - 1)$ and $P_j(N' - 1)$. This can guarantee all points of the two downsampled autocorrelation signals are used for measuring their distance, thus avoiding to use only local data. Furthermore, the third and fourth constraints are monotonic and marching constraints, which require that there be no cycles for w_i and w_j in the warping path and the path is increased with the maximum 1 at each step.

We apply dynamic programming to solve problem (8.22), to obtain the minimum distance warping path between two downsampled autocorrelation signals. We consider a two-dimensional cost matrix $\mathcal{C}$ with size $N' \times N'$, whose element $\mathcal{C}(m_l, n_l)$ is the minimum distance warping path for two downsampled autocorrelation signals $P_i = [P_i(0), P_i(1), \cdots, P_i(m_l)]$ and $P_j = [P_j(0), P_j(1), \cdots, P_j(n_l)]$. We design the recurrence equation in dynamic programming as follows.

$$\mathcal{C}(m_l, n_l) = \|P_i(m_l) - P_j(n_l)\| + \min\left[\mathcal{C}(m_l - 1, n_l), \mathcal{C}(m_l, n_l - 1), \mathcal{C}(m_l - 1, n_l - 1)\right]. \tag{8.27}$$

By filling all elements of the cost matrix $\mathcal{C}$, the value $\mathcal{C}(N'-1, N'-1)$ can be computed as the DTW value between the two downsampled autocorrelation signals. The time complexity is $O(N'^2)$. Fig. 8.7 shows the DTW results for downsampled autocorrelation signals 4 and 6 (the upper plot),

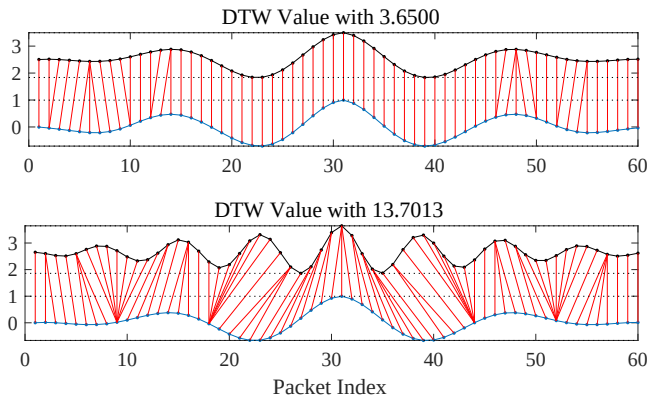


Figure 8.7: DTW results for downsampled autocorrelation signals 4 and 6 (the upper plot), and downsampled autocorrelation signal 4 and 3 (the lower plot), respectively.

and downsampled autocorrelation signals 4 and 3 (the lower plot), where we set the downsampling number of packets as $N' = \frac{N}{10} = 60$. It can be seen that downsampled autocorrelation signals 4 and 6 have a smaller DTW value (i.e., 3.65) than downsampled autocorrelation signals 4 and 3 (i.e., 13.7). That is, signals 4 and 6 are more similar, and more likely to belong to the same person. We also find that the downsampled autocorrelation signals have a high similarity in the center than that on the boundary, and it can reduce the phase shift values. Thus, the DTW value is a good measure of the distance between two downsampled autocorrelation signals. We need to compute the DTW values for all the downsampled autocorrelation signal pairs, which are then used in stable roommate matching.

Stable Roommate Matching

Since the CP decomposed signals are randomly indexed (see Fig. 8.6), we need to identify the pair for each person. With the DTW values for all downsampled autocorrelation signal pairs, we can model this problem as a stable roommate matching problem [124, 132, 133]. There are a group of $2R$ signals, and each signal maintains a preference list of all other signals in the group, where the preference value for another signal is the inverse of the corresponding DTW value (i.e., distance). The problem is to pair the signals, such that there is no such a pair of signals that both of them have

a more desired selection than their current selection, i.e., to find a stable matching [124, 132, 133]. The proposed signal matching algorithm is presented in Algorithm 9.

We first compute the autocorrelation of all decomposed signals. Then each autocorrelation signal populates its preference list with other autocorrelation signals according to the DTW values. The stable roommate matching algorithm is executed in two steps. In step 1, each signal proposes to other signals according to its preference list. If a signal m receives a proposal from another signal n, we implement the following strategy: (i) signal m rejects signal n if it has a better proposal from another signal; (ii) signal m accepts signal n's proposal if it is better than all other proposals that signal m currently holds. Moreover, signal n stops to propose when its proposal is accepted, while it needs to continue to propose to other signals if being rejected. This strategy is implemented in step 1 of the signal matching algorithm, where we use $finish\_flag$ to mark whether the current $signal\_num$ is accepted or not. Moreover, variables $accept\_num$ and $propose\_num$ are used to record the current signal's proposed number and proposing number, respectively. Also, variable $scan\_num$ is used to record the current scanning signal number. After completing step 1, every signal holds a proposal or one signal has been rejected by other signals (this case hardly happens in TensorBeat, because the CP decomposition produces two very similar signals with high probability for each person). Then, we need to delete some elements in all the preference lists based on the following method, which is that if signal m is the first on signal n's list , then signal n is the last on signal m's list. For the proposed algorithm, every signal can reject signals that have less than $accept\_num$ in its preference list symmetrically (reject each other).

An example is shown in Fig. 8.5. According to the DTW values, signals 1, 2, 3, 4, 5, and 6 have their preference lists as (2, 3, 5, 6, 4), (1, 3, 5, 6, 4), (5, 1, 2, 6, 4), (6, 5, 3, 2, 1), (3, 2, 1, 4, 6), and (4, 5, 3, 2, 1), respectively. When step 1 is executed, we have: Signal 1 proposes to 2, and signal 2 holds 1; Signal 2 proposes 1, and signal 1 holds; Signal 3 proposes to signal 5, and signal 5 holds; Signal 4 proposes to signal 6, and signal 6 holds; Signal 5 proposes to signal 3, and signal 3 holds; Signal 6 proposes to signal 4, and signal 4 holds. It is easy to find three pairs (1,2), (3,5), and (4,6).

Although most of decomposed signals are paired in step 1, step 2 will still be necessary for the more challenging cases of much more breathing signals and NLOS environments. In step 2, we consider the reduced preference lists, where some of the lists have more than one signals. By implementing step 2, we can reduce the preference lists such that each signal only holds one proposal. The main idea is that we need to find some all-or-nothing cycles and symmetrically delete signals in the cycle sequence by rejecting the first and last choice pairs. The signal in the cycle accepts the secondary choice, thus obtaining a stable roommate matching. To find all-or-nothing cycles, let p_1 be a signal with a preference list that contains more than one element, and generate the sequences such that q_i = the second preference of p_i's current list, and p_{i+1} = the last preference of q_i's current list. After the cycle sequence generation, denote p_s as the first element in the p sequence to be repeated. Then, we reject matching $(q_s + i - 2, p_s + i - 1)$ for $i = 1$ to r symmetrically, where r is the length of the cycle. Finally, we can obtain signal matching pairs based on all processed preference lists. The computational complexity of Algorithm 9 is $O(R^2)$, because steps 1 and 2 each has a complexity of $O(R^2)$, respectively.

8.3.5 Breathing Rate Estimation

Signal Fusion and Autocorrelation

After obtaining the outcomes of the signal matching algorithm, TensorBeat next applies peak detection to estimate the breathing rates for multiple persons. Comparing to the FFT method, a higher resolution in the time domain can be achieved. To implement peak detection, we first need to combine the decomposed signal pairs for each person into a single signal, by taking the average of the signal pairs. Averaging can decrease the variance of the decomposed signals while preserving the same period. For example, Fig. 8.8 shows the fusion results based on the outcome of the signal matching algorithm, where three smoothly decomposed signals with different periods are obtained. To strengthen the accuracy of peak detection, we compute the autocorrelation function again for every fused signal. Fig. 8.9 shows the autocorrelation of the three fused signals. It can be seen that

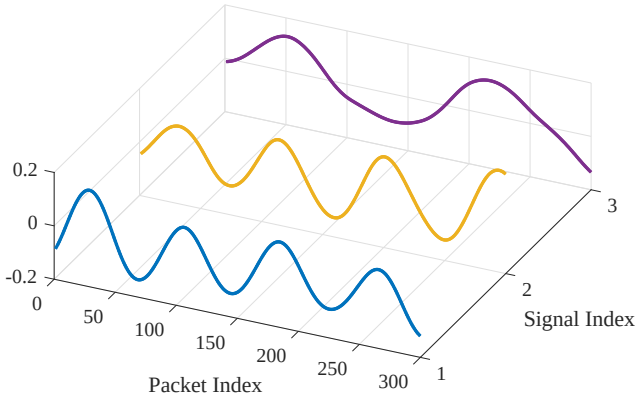


Figure 8.8: Fusion results based on the outcomes of the signal matching algorithm.

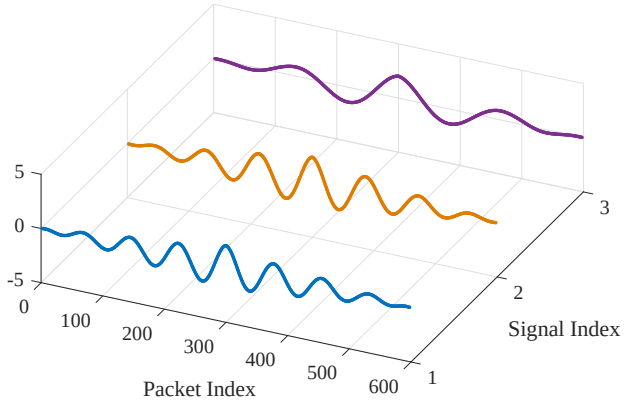


Figure 8.9: Autocorrelation of fused signals.

the length of data is increased from 300 to 600 and the number of peaks of every signal are also increased, which help to improve the estimation accuracy.

Peak Detection

Although breathing signal is generated by the small periodic chest movement of inhaling and exhaling, the phase difference data can effectively capture the breathing rate. Traditionally, estimation of breathing rates is achieved with FFT based methods. However, the FFT approach may have limited accuracy, because the frequency resolution of breathing signals is based on the window size of FFT. When the window size becomes larger, the accuracy will be higher, but the time domain resolution will be reduced. Also see Figs. 8.1 and 8.2 for the limitation of the FFT based approach

for the multi-person scenario. Therefore, we leverage peak detection instead in TensorBeat system to achieve accurate breathing rate estimation for each of autocorrelations of fused signals.

For peak detection, the traditional method based on amplitude needs to detect the fake peak, which is not a real peak but has larger values than its two immediate neighboring points. To avoid the fake peak, a large moving window can be used to identify the real peak based on the maximum breathing periodicity. This method is not robust, which requires adjusting the window size. In TensorBeat, we only consider a smaller moving window of 7 samples wide. This is because we leverage the Hankel matrix and CP decomposition to smooth out the breathing curves, which hardly contains any fake peaks. Then, for the ith autocorrelation curve of fused signal, we seek all the peaks by determining whether or not the medium of the 7 samples in the moving window is the maximum value. Finally, we consider the median of all peak-to-peak intervals as the final period of the ith breathing signal, which is denoted as T_i. Finally, the estimated breathing rates can be computed as $f_i = 60/T_i$, for $i = 1$ to R.

8.4 Experimental Study

8.4.1 Experiment Configuration

In this section, we validate the TensorBeat performance with an implementation with 5 GHz Wi-Fi devices. To obtain 5 GHz CSI data, we use a desktop computer and a Dell laptop as access point and mobile device, respectively, both of which are equipped with an Intel 5300 NIC. We use the desktop computer instead of the commodity routers, because none is equipped with the Intel 5300 NIC. The operating system is Ubuntu destop 14.04 LTS OS for both the access point and the mobile device. The PHY is the IEEE 802.11n OFDM system with QPSK modulation and 1/2 coding rate. Moreover, the access point is set in the monitor model and the distance between its two adjacent antennas is approximately 2.68 cm, which is half of the wavelength of 5GHz WiFi. Also, the mobile device is set in the injection model and uses one antenna to transmit data. Moreover, we use omnidirectional antennas for both the receiver and transmitter to estimate breathing signs

beats. With the packet injection technique with LORCON version 1, we can obtain 5 GHz CSI data from the three antennas of the receiver.

Our experimental study is with up to five persons over a period of six months. The experimental scenarios include a computer laboratory, a through-wall scenario, and a corridor, as shown in Fig. 8.10. The first scenario is within a 4.5 $\times$ 8.8 m^2 laboratory, where both single person and multi-person breathing rate estimation experiments are conducted. There are lots of tables and desktop computers crowded in the laboratory, which block parts of the LOS paths and form a complexity radio propagation environment. The second setup is a through-wall environment, where single person breathing rate estimation is tested due to the relatively weaker signal reception. The person is on the transmitter side, and the receiver is behind a wall in this experiment. The third scenario is a long corridor of 20 m, where the maximum distance between the receiver and transmitter is 11 m in the experiment. This scenario is still considered for single person breathing rate monitoring. We use a NEULOG Respiration to record the ground truths for single person breathing rates. The single person breathing rates estimation can be easily implemented by removing the signal matching algorithm, because there are only two decomposed signals after CP decomposition in this case. For muti-person breathing rate estimation in the first scenario, all persons participating in the experiment record their breathing rates by using a metronome smartphone application with 1 bpm accuracy at the same time. We consider five persons are stationary for LOS and NLOS environments for breathing monitoring. Moreover, there are no other persons in the breathing measurement area.

For multi-person breathing rate estimation, we need to define a proper metric for evaluating TensorBeat's performance. For R estimated breathing rates [$f_1, f_2, ...f_R$], the ith breathing rate estimation error, E_i, is defined as

$$E_i = \left| f_i - \hat{f}_i \right|, \quad \text{for } i = 1, 2, \cdots, R, \tag{8.28}$$

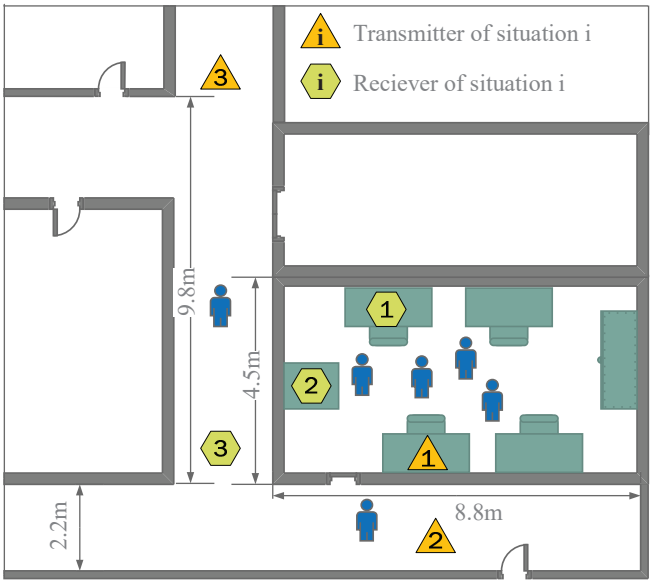


Figure 8.10: Experimental setup: computer laboratory, through-wall, and long corridor scenarios.

where $\hat{f}_i$ is the ground truth of the ith breathing rate. We also define a new metric termed success rate, denoted as SR, which is defined as

$$SR = \frac{N\{\max_i \{E_i\} < 2\text{bpm}\}}{N\{E\}} \times 100\%, \tag{8.29}$$

where $N\{\max_i \{E_i\} < 2\text{bpm}\}$ means the number of repeated experiments of the maximum breathing rate error less than 2 bpm, and $N\{E\}$ is the number of repeated experiments. We adopt the success rate metric because there are weak signals for multi-person experiments in indoor experiments at different locations, and sometimes a breathing signal may not be successfully detected [134].

8.4.2 Performance of Breathing Estimation

In Fig. 8.11, we present the CDF of the estimation errors for single person breathing rate detection for three different experiment scenarios. We can see that for TensorBeat, high estimation accuracy of breathing rates can be achieved in all the three scenarios. The maximum estimation error is less than 0.9 bpm. Moreover, it is noticed that 50% of the tests for the computer laboratory experiment

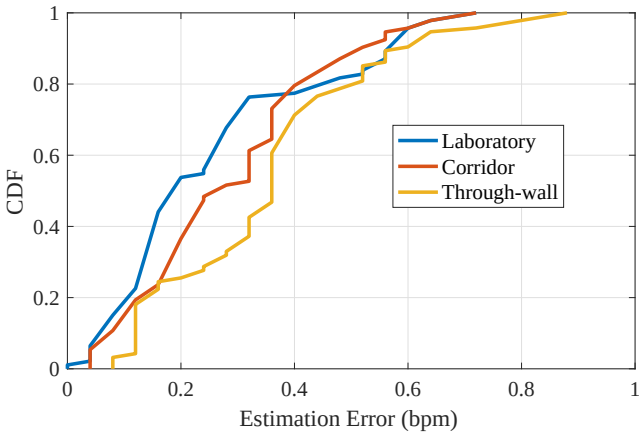


Figure 8.11: Performance of single person breathing rate estimation in the computer laboratory, through-wall, and long corridor scenarios.

have errors less than aboout 0.19 bpm, while the tests for the corridor and through-wall scenarios have errors less than approximately 0.25 bpm and 0.35 bpm, respectively. Thus, the performances in the laboratory setting is better than that in the corridor and through-wall scenarios. This is because the laboratory has a smaller space and the breathing signal is stronger than that of other two cases with larger attenuation due to the long distance and the wall.

Fig. 8.12 presents the performance of breathing rate estimation for different number of persons. It is noticed that higher accuracy is achieved for the single person test, where approximately 96% of the test data have an estimation error less than 0.5 bpm. The five-person test has the worse performance, where approximately 62% of the test data have an estimation error less than 0.5 bpm. Moreover, we fine that the performances of the two-person and three-person tests are similar, both of which can have an error smaller than 0.5 bpm for 93% of the test data. Generally, when the number of persons is increased, the performance of breathing rate estimation gets worse. In fact, when the number of breathing signals is increased, the distortion of the mixed received signal will become larger, thus leading to high estimation errors.

Fig. 8.13 plots the success rates for different number of persons. We find that although the success rate for one person is the highest, there are still few of test data that cannot obtain high accuracy breathing rates estimation. These test data should come from different locations in the indoor environments, where parts of the received signals are severely distorted. In fact, we find

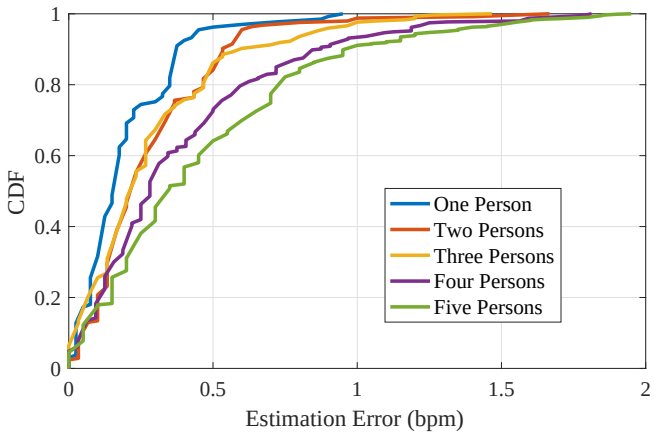


Figure 8.12: Performance of breathing rate estimation for different number of persons (computer laboratory).

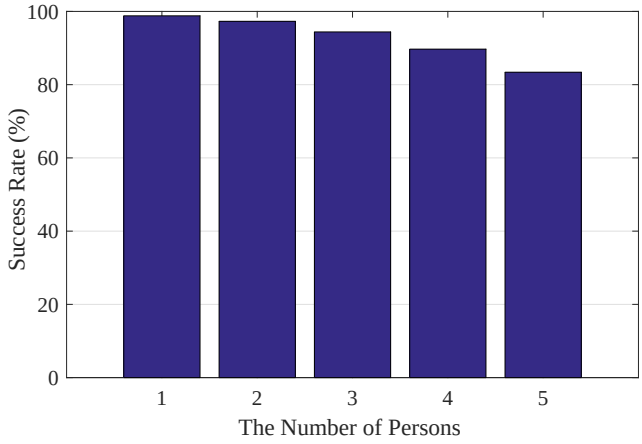


Figure 8.13: Success rates for different number of persons (computer laboratory).

that low phase difference usually occurs when the SNR is low. On the other hand, we can see that breathing rate estimation for two persons also has a high success rate, because the probability for two persons to have exactly the same breathing rate is very low. When the number of persons is increased, the chance of getting two close breathing rates becomes higher. Even in this case, TensorBeat can still effectively separate them with a high success probability. With the increase of the number of persons, the success rate for TensorBeat system decreases. The reason is that each breathing rate is more likely to cover each other and the strength of the received signal becomes lower. From Fig. 8.13, we can see that the success rate is about 82.4% when the number of persons is five.

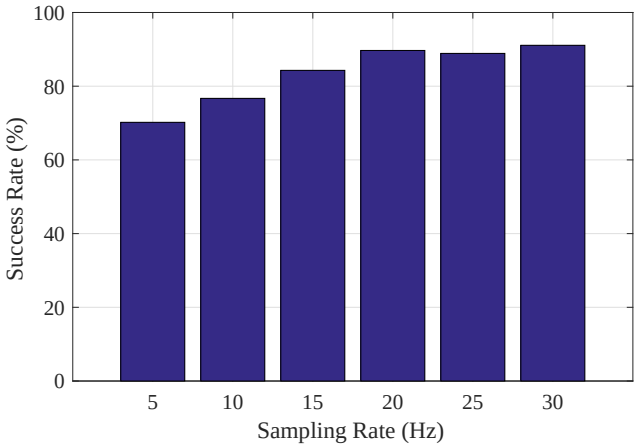


Figure 8.14: Success rates for different sampling rates (computer laboratory).

Fig. 8.14 shows the success rate for different sampling rates. In this experiment, there are four persons and the window size is set to 30 s. From Fig. 8.14, we can see that with the increase of the number of sampling rates, the success rate is also increased. It is noticed that the success rates for 5 Hz and 30 Hz are approximately 70% and 90%, respectively. As the sampling rate is increased, the length of the data for CP decomposition is increased for the 30 s window size case, which helps to improve the estimation accuracy. Furthermore, we find that the performance becomes stable when the sampling rate exceeds 20 Hz, indicating that a sampling rate of 20 Hz is sufficient for CP decomposition. Thus, we set the sampling rate to 20 Hz for the TensorBeat experiments.

Fig. 8.15 plots the success rates for different window sizes. This experiment is for the computer laboratory scenario with four persons and the sampling rate is set to 20 Hz. From Fig. 8.15, we can see that the success rate is greatly increased by increasing the window size of the Hankel matrix from 15 s to 30 s. This is because Hankelizaiton will take half of the data to smooth the phase difference signal, which reduces the resolution in the time domain. Thus, we need to increase the window size to improve the estimation accuracy. Furthermore, the change of success rate is small for window sizes from 30 s to 45 s. Thus, we select the window size of 30 s for the TensorBeat experiments.

Finally we examine the impact of LOS and NLOS scenarios. The success rates are plotted in Fig. 8.16. In this experiment, we consider the challenge condition of the NLOS scenario, where

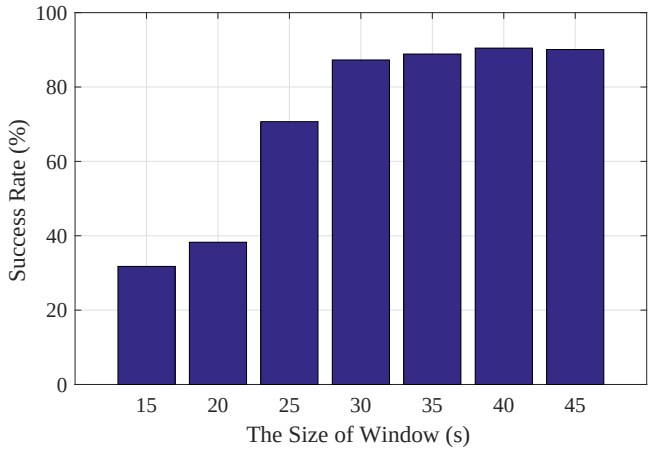


Figure 8.15: Success rates for different window sizes (computer laboratory).

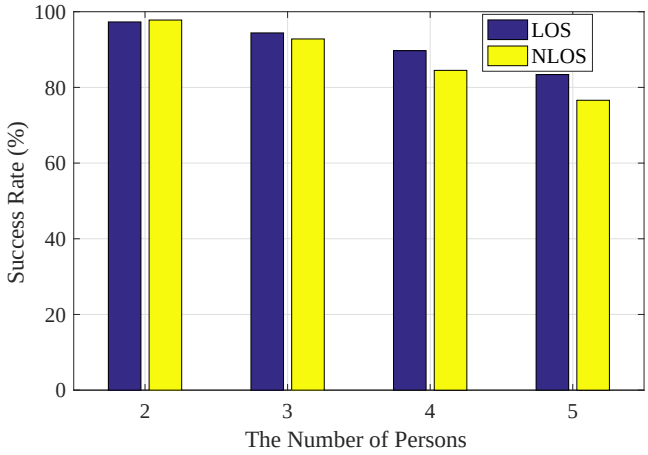


Figure 8.16: Success rates for (i) when multiple persons form a line in the LOS path between the transmitter and receiver; (ii) when multiple persons are in scattered around (computer laboratory).

all the persons stay on the LOS path between the transmitter and receiver, i.e., they form a straight line and block each other. From Fig. 8.14, we find that the performances for LOS and NLOS are nearly the same for the cases of two or three persons, where high estimation accuracy can be achieved. This is due to the WiFi multipath effect, which is regarded harmful in general but becomes helpful in breathing rate estimation when tensor decomposition is used. The breathing signal of every person can still be captured at the receiver from the phase difference data. However, when the number of the persons is further increased, the success rate will decrease quickly. In fact, the strength of the breathing signals for some persons will become too weak to be detected when there are too many people blocking each other.

8.5 Related Work

This work is closely related to RF signal based vital signs monitoring, as well as CSI based indoor localization and human activity recognition, which are discussed in the following.

RF based systems for vital signs monitoring use wireless signals to track the breathing-induced chest change of a person, which are mainly based on radar and WiFi techniques. For radar based vital signals monitoring, Vital-Radio employs FMCW radar to estimate breathing and heart rates, even for two person subjects in parallel [110]. But the system requires a custom hardware with a large bandwidth from 5.46 GHz to 7.25 GHz. For WiFi based vital signs monitoring, UbiBreathe system employ WiFi RSS for breathing rate monitoring, which, however, requires the device placed in the line of sight path between the transmitter and the receiver for estimating the breathing rate [115]. Moreover mmVital based on RSS can use 60 GHz millimeter wave (mmWave) signal for breathing and heart rates monitoring with the larger bandwidth about 7GHz, which cannot monitor the longer distance and require high gain directional antennas for the transmitter and the receiver [114][135]. Recently, the authors leverage the amplitudes of CSI data to monitoring vital signs [116]. This work is mainly to track the vital signs when a person is sleeping, which is limited for monitoring a maximum of two persons at the same time.

In additional to vital signs monitoring, recently, CSI based sensing systems have also been used for indoor localization and human activity recognition [136]. CSI-based fingerprinting systems have been proposed to obtain high localization accuracy. FIFS is the first work to uses the weighted average of CSI amplitude values over multiple antennas for indoor localization [22]. To exploit the diversity among the multiple antennas and subcarriers, DeepFi leverage 90 CSI amplitude data from the three antennas with a deep autoencoder network for indoor localization [64]. Also, PhaseFi leverages calibrated CSI phase data for indoor localization based on deep learning [79]. Different from CSI-based fingerprinting techniques, SpotFi system leverages a super-resolution algorithm to estimate the AOA of multipath components for indoor localization based

on CSI data from three antennas [137]. On the other hand, E-eyes system leverages CSI amplitude values for recognizing household activities such as washing dishes and taking a shower [75]. WiHear system employs specialized directional antennas to measure CSI changes from lip movement for determining spoken words [138]. CARM system considers a CSI based speed model and a CSI based activity model to build the correlation between CSI data dynamics and a given human activity [139]. Although CSI based sensing are effective for indoor localization and activity recognitions, there are few works for using CSI phase difference data to detect multiple persons behaviors at the same time.

The TensorBeat system is motivated by these interesting prior works. To the best of our knowledge, we are the first to leverage CSI phase difference data for multiple persons breathing rate estimation. We are also the first to employ tensor decomposition for RF sensing based vital signs monitoring, which can be also employed for indoor localization and human activity recognition.

8.6 Conclusions

In this chapter, we proposed TensorBeat, tensor decomposition for estimating multiple persons breathing beats with commodity WiFi. The proposed TensorBeat system employed CSI phase difference data to obtain the periodic signals from the movements of multiple breathing chests by leveraging tensor decomposition. We implemented several signal processing methods including data preprocessing, CP decomposition, signal matching algorithm, and peak detection in TensorBeat. We validate the performance of TensorBeat with extensive experiments under three indoor environments. Our analysis and experimental study demonstrated that the proposed TensorBeat system can achieve satisfactory performance for multiple persons breathing estimation.

Algorithm 9: Signal Matching Algorithm

Input: Decomposed signals: $S_1, S_2, \cdots, S_{2R}$.
Output: Matched signal pairs.

Compute autocorrelation of all decomposed signals;
Compute the DTW values for every pair of autocorrelation signals;
Each autocorrelation signal sets its preference list using the DTW values;
for $signal\_num = 1 : 2R$ **do**
    Set $finish\_flag = 0$;
    Set $scan\_num = signal\_num$;
    while $finish\_flag = 0$ **do**
        if *the proposal is the first one* **then**
            Proposing signal's $propose\_num$=the current choice;
            Set $finish\_flag = 1$;
            Proposed signal's $accept\_num = scan\_num$;
        else
            if *the signal prefers the former proposal* **then**
                Reject the current proposal symmetrically;
                Propose to the next choice;
            else
                Accept the current proposal;
                Reject the former proposal symmetrically;
                $scan\_num =$ proposed signal's $accept\_num$;
                Propose to the next choice;
            end
        end
    end
end
for $signal\_num = 1 : 2R$ **do**
    Reject signals that have less than $accept\_num$ in every preference list symmetrically;
end
$signal\_num = 1$;
while $signal\_num < 2R + 1$ **do**
    if $propose\_num = accept\_num$ **then**
        $signal\_num$=$signal\_num$+1;
    else
        Let p_1 be a signal whose preference list contains more than one element;
        while p *sequence is not cyclic* **do**
            q_i = the second preference of p_i's current list;
            p_{i+1} = the last preference of q_i's current list;
        end
        Denote p_s as the first element in the p sequence to be repeated and r as the length of the circle;
        for $i = 1 : r$ **do**
            Reject matching (q_{s+i-2}, p_{s+i-1}) symmetrically;
        end
        $signal\_num = 1$;
    end
end
Obtain signal matching pairs based on all processed preference lists;

Chapter 9

ResBeat: Resilient Breathing Beats Monitoring with Realtime Bimodal CSI Data

9.1 Introduction

Vital signs can provide useful clues to many diseases such as heart disease, lung disorder, and diabetes, which cost considerable expenses for treatment [140, 106]. Effective solutions are in great demand for realtime, long-term, and contact-free breathing signal monitoring [141, 142]. Recently, CSI amplitude based method is proposed for monitoring breathing and heart beats when a person is sleeping [116]. In addition, our recent works PhaseBeat [143] and TensorBeat [144] leverage CSI phase difference data to monitor a single person's vital signals and multiple persons' breathing signals, respectively. However, these works are not effective in detecting the weak breathing signals at some special locations [134], which motivates us to use bimodal CSI data for resilient breathing monitoring.

In this chapter, we employ CSI amplitude and phase difference bimodal data to detect and monitor breathing beats with commodity 5GHz WiFi devices. For indoor environments under small-scale fading, we consider the chest reflected signal as a *dynamic component*, while lumping the LOS and all other multipath signals together as a *static component*. We model the amplitude and phase response of the CSI subcarriers with the dynamic and static component approach, and prove that CSI amplitude and phase information carry the breathing information with the same rate. Moreover, we present an analysis of breathing signal anomaly with CSI amplitude and phase information, and show that the breathing signals can be weak at some monitoring locations. Thus, we propose to use bimodal CSI data for resilient breathing monitoring, due to the fact that CSI

amplitude and phase difference data in consecutively received packets are stable, and they are complementary to each other with respect to mitigating the anomalous breathing signals at some bad locations. The bimodal data is also robust to environment interferences and body movements for breathing signal monitoring.

We present the design of ResBeat, i.e., **Res**ilient realtime breathing **Beat** monitoring with bimodal CSI amplitude and phase difference data with commodity WiFi devices. The ResBeat system consists of data preprocessing, adaptive signal selection, and breathing signal monitoring modules. For data preprocessing, we calibrate CSI data with an *exponentially weighted moving average* (EWMA) method to extract the static, or environment, component and the dynamic, or breathing, component. For adaptive signal selection, we propose a signal selection algorithm based on signal energy detection and movement detection, to select the most sensitive signal group, which can successfully mitigate the effect of anomalous breathing signals. Finally, the peak detection method is used to estimate breathing rates in realtime CSI data.

We implement ResBeat with commodity 5GHz WiFi devices and evaluate its performance with four persons over three months in different indoor environments, such as a computer laboratory, a through-wall scenario, and a long corridor. The results validate that the ResBeat system can achieve high estimation accuracy of breathing rate with a median error of 0.25 bpm (beats per minute), and has a higher success rate of 90% for breathing rate detection at different locations.

The main contributions of this chapter are summarized below.

- We theoretically demonstrate the feasibility of using bimodal CSI data for breathing beats monitoring. In particular, we provide the breathing signal anomaly analysis for CSI amplitude and phase information. To the best of our knowledge, we are the first to leverage online CSI amplitude and phase difference data for breathing rate estimation.

- We implement data preprocessing, adaptive signal selection and breathing signal monitoring for the collected bimodal CSI data in ResBeat system. We employ the EWMA method for obtain environment component and breathing component in data calibration. Moreover, we utilize the peak detection method for breathing rate estimation.

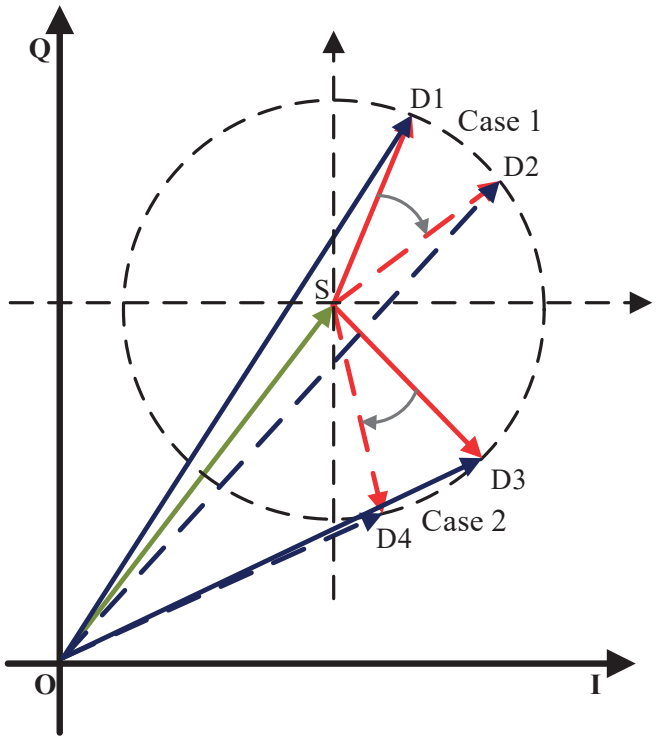


Figure 9.1: The geometric relationship of the static and dynamic components for breathing signal anomaly analysis.

- We prototype the ResBeat system with commodity 5 GHz WiFi devices and validate its superior performance in three different indoor environments with extensive experiments. The results show that our ResBeat system can achieve higher success rates than the amplitude based method and phase difference based method for breathing detection.

In the rest of this chapter, the preliminaries and breathing signal anomaly analysis are introduced in Section 9.2. We design the ResBeat system in Section 9.3 and validate its performance in Section 9.4. Section 9.5 summaries this chapter.

9.2 Breathing Signal Anomaly Analysis

We consider indoor environments with NLOS components [100], where the chest reflected signal is regarded as the *dynamic component*, and the sum of the LOS and all other mutipath signals is regarded as a *static component*. Thus, the channel frequency response of subcarrier i, denoted by

H_i, can be written as

$$\begin{aligned}\mathrm{H}_i &= \sum_{k=0,k\neq d}^{K} r_k \cdot e^{-j2\pi f_i \tau_k} + r_d \cdot e^{-j2\pi f_i \tau_d} = \mathrm{H}_i^s + \mathrm{H}_i^d \\ &= |\mathrm{H}_i^s| \exp\left(j\angle \mathrm{H}_i^s\right) + |\mathrm{H}_i^d| \exp\left(j\angle \mathrm{H}_i^d\right), \end{aligned} \tag{9.1}$$

where K is the number of multipaths, r_k and τ_k are the attenuation and propagation delay of the k_{th} path, respectively; $\mathrm{H}_i^s = \sum_{k=0,k\neq d}^{K} r_k \cdot e^{-j2\pi f_i \tau_k}$ is the static component and $\mathrm{H}_i^d = r_d \cdot e^{-j2\pi f_i \tau_d}$ is the dynamic component, $|\mathrm{H}_i^s|$ and $\angle \mathrm{H}_i^s$ are the amplitude and phase of H_i^s, respectively, and $|\mathrm{H}_i^d|$ and $\angle \mathrm{H}_i^d$ are the amplitude and phase of H_i^d, respectively.

The amplitude response of subcarrier i can be computed as

$$|\mathrm{H}_i| = \sqrt{|\mathrm{H}_i^s|^2 + |\mathrm{H}_i^d|^2 + 2|\mathrm{H}_i^s||\mathrm{H}_i^d| \cos\left(\angle \mathrm{H}_i^s - \angle \mathrm{H}_i^d\right)}. \tag{9.2}$$

In (9.2), the amplitude and phase of the static component H_i^s are regarded as constants, and the amplitude of the dynamic component H_i^d is also assumed to be constant. Moreover, the phase of the dynamic component H_i^d can be modeled as $\angle \mathrm{H}_i^d = 2\pi L/\lambda_i$, where λ_i is the wavelength of subcarrier i and L is the distance of the dynamic path going through the chest. Note that the phase of the dynamic component H_i^d is periodic because the dynamic path distance L is periodic due to chest movements (i.e., it gets slightly longer when exhaling and shorter when inhaling). Thus, the amplitude response of subcarrier i, $|\mathrm{H}_i|$, is also periodic. In most cases, the CSI amplitude can effectively capture the breathing signal. However, at some monitoring locations, when the phase difference between the static component H_i^s and the dynamic component H_i^d is nearly zero, the variations of the CSI amplitude will be small, leading to high monitoring errors.

This is illustrated in Fig 9.1, where the geometric relationship of the static and dynamic components is presented. The dynamic components are $\overrightarrow{SD}_i$, $i = 1, 2, 3, 4$, and the static component

is $\overrightarrow{OS}$. We can see that when the dynamic vector oscillates between $\overrightarrow{SD}_1$ and $\overrightarrow{SD}_2$, the CSI amplitude varies between $|\overrightarrow{OD}_1|$ and $|\overrightarrow{OD}_2|$, with very small variations. Such small variations in the CSI amplitude leads to a weak breathing signal and high detection error.

On the other hand, let's consider the phase response of subcarrier i, $\angle \mathrm{H}_i$, which can be written as [143]

$$\angle \mathrm{H}_i = \angle \mathrm{H}_i^s - \arctan\left\{\frac{|\mathrm{H}_i^d| \cdot \sin(\angle \mathrm{H}_i^s - \angle \mathrm{H}_i^d)}{|\mathrm{H}_i^d| \cdot \cos(\angle \mathrm{H}_i^s - \angle \mathrm{H}_i^d) + |\mathrm{H}_i^s|}\right\}. \tag{9.3}$$

Note that the phase response of subcarrier i is also periodic, which can be used to detect breathing rate. However, there are also some cases where the phase information is not effective for monitoring breathing signals. As shown in Fig 9.1, when the dynamic vector oscillates between $\overrightarrow{SD}_3$ and $\overrightarrow{SD}_4$, the phase value changes slightly between $\angle\overrightarrow{OD}_3$ and $\angle\overrightarrow{OD}_4$. Such negligible variations in phase value makes it hard to detect the breathing rate with phase information in this situation.

Fortunately, we can see that in the first example, when the variation in CSI amplitude is small, the change in CSI phase is between $\angle\overrightarrow{OD}_1$ and $\angle\overrightarrow{OD}_2$, which is quite large. On the other hand, in the second example when the variation in CSI phase is small, the change in CSI amplitude is between $|\overrightarrow{OD}_3|$ and $|\overrightarrow{OD}_4|$, which is also quite large. This observation motivates us to leverage bimodal CSI data, including CSI amplitude and phase difference, for resilient breathing monitoring.

Fig. 9.2 and Fig. 9.3 show the all calculated amplitude and phase difference from CSI values for position 1 and 2, respectively. In our experiment, there are three antennas used for receiving data, and each antenna can collect CSI values from all 30 subcarriers using Intel 5300 NIC. Thus, we can totally get 90 amplitudes and 90 phases from a single packet. In the experiment, we can consider 30 rows data as a group. For example, the first group of CSI amplitude values represents first 30 rows data, which are collected by antenna 1 from all 30 subcarriers, while the second and third group data are collected from antenna 2 and antenna 3, respectively. When it comes to phase data, the first group of CSI phase difference is collected from antenna 1 and antenna 2, while the second group and the third group of data are collected from antenna 2 and 3, antenna 1 and 3,

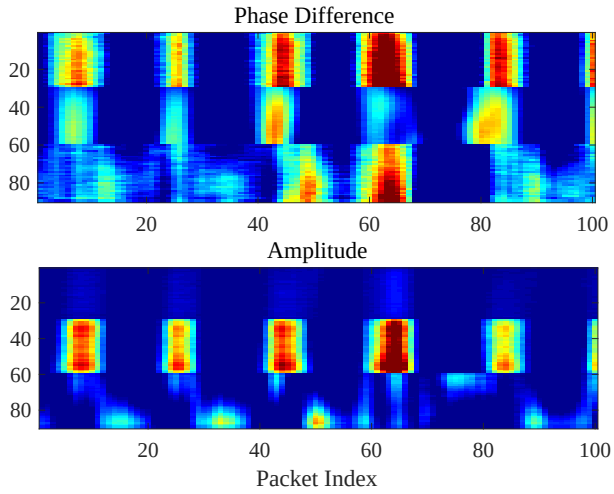


Figure 9.2: Colormap for breathing signals using CSI amplitude and phase difference in position 1.

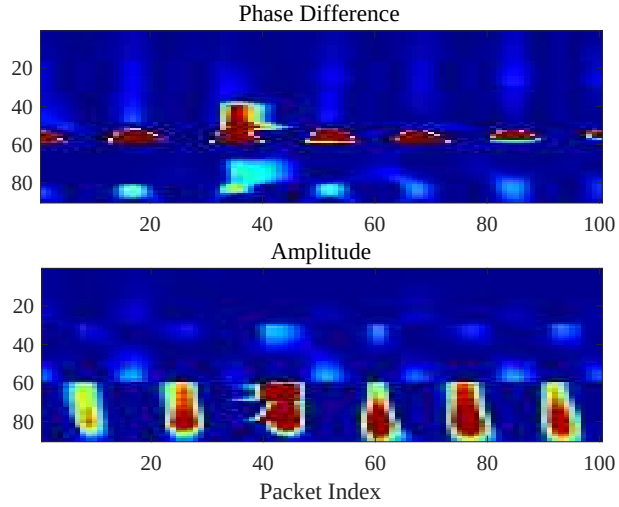


Figure 9.3: Colormap for breathing signals using CSI amplitude and phase difference in position 2.

respectively. The color of the data means the strength of the breathing signal, where red means the signal is strong, while blue means the signal is weak. Based on the colormap, we can easily find out some periodic signals from Fig. 9.2 and Fig. 9.3, which can stand for the breathing signals. As illustrated in Fig. 9.2, the first and the second group of phase difference can effectively capture the breathing signal, as well as the second group of the amplitude data. However, when the user moves to another position, the signal strength is totally different. Fig. 9.3 shows that the phase difference can barely capture the breathing signal effectively except some subcarriers in the second group, and

the second group of the amplitude values become weak as well. In order to continuously monitor human breathing, we should better leverage the third group of amplitude data and the second group of phase difference for breathing monitoring, thus avoiding the weak breathing signals because of the diversity of locations.

9.3 The ResBeat System

9.3.1 ResBeat System Architecture

The main idea of the proposed ResBeat system is to monitor breathing signals using realtime bimodal CSI data from 5GHz WiFi devices. We propose an adaptive signal selection method to select the most sensitive CSI data, i.e., amplitude or phase difference, to mitigate the anomalous breathing signals, as shown in Fig. 9.1. The ResBeat system can effectively exploit realtime bimodal CSI data to monitor breathing beats for three reasons. First, CSI amplitude and phase difference data are quite stable for consecutive packets in a stationary environment, both of which can effectively capture the breathing beats. Second, CSI amplitude and phase difference data are complementary to each other with respect to their resilience to the two anomalous cases shown in Fig. 9.1. Using the bimodal data can effectively deal with anomalous breathing signals. Third, ResBeat is robust to environment interference and body movements by using the bimodal CSI data.

Fig. 9.4 shows ResBeat system flow, which consists of three main modules: Data Preprocessing, Adaptive Signal Selection, and Breathing Signal Monitoring. Data Preprocessing module mainly includes CSI data extraction and data calibration, respectively. For CSI data extraction, we can obtain 90 CSI amplitude values and 90 phase difference values from three antennas for each received packet by using the modified Intel 5300 NIC driver. For data calibration, we employ the EWMA method to obtain the environment component, and the breathing component can be extracted by subtracting the environment component from the denoised CSI data. Adaptive Signal Selection module includes signal energy detection, movement detection and signal selection. We can use the normalized breathing component to compute signal energy for 30 subcarriers data from CSI amplitude or phase difference data. Then, we implement the counting method for three

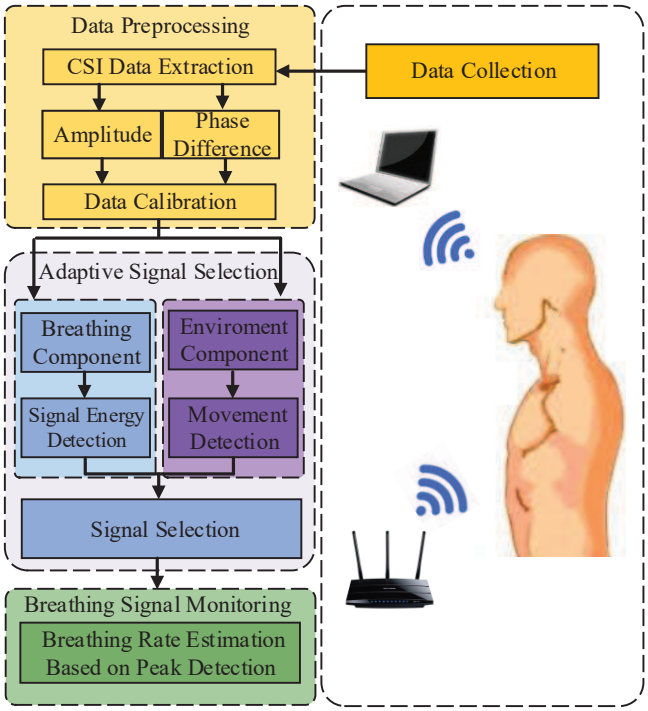


Figure 9.4: The ResBeat system architecture.

CSI phase difference groups for movement detection. Moreover, we develop the signal selection algorithm to boost the reliability of bimodal CSI data for breathing signal monitoring by selecting the most sensitive signal group from CSI amplitude and phase difference data groups. For Breathing Signal Monitoring module, peak detection is employed for breathing beat monitoring with real-time CSI data.

9.3.2 Data Preprocessing

CSI Data Extraction

We collect 90 CSI values for every received packet from the three antennas of the IEEE 802.11n NIC, each of which provide CSI values from 30 subcarriers (i.e., 90 CSI amplitude and phase values). We employ three groups of CSI amplitude values from antennas 1, 2, and 3, respectively. We also use (7.5) to obtain three groups of CSI phase difference values from antennas 1 and 2, antennas 2 and 3, and antennas 3 and 1, respectively. Each group includes 30 values, which will be processed in the next step.

Data Calibration

Data calibration is to partition the original CSI amplitude and phase difference data into the static component (or, the *environment component*) and the dynamic component (or, the *breathing component*). The environment component can represent the change of wireless channel from the surrounding environment such as reflection from walls, desks and stationary body of a person. On the other hand, the breathing component can represent the change of wireless signal due to chest movements when inhaling and exhaling.

In ResBeat, the environment component can be extracted using the EWMA method, which is based on a first-order autoregressive model [145]. Let Y_t be the realtime CSI data (i.e., CSI amplitude or phase difference), and M_t be its local mean. We have

$$M_t = \rho \cdot Y_t + (1 - \rho) \cdot M_{t-1}, \text{ for } t = 1, 2, ..., n, \tag{9.4}$$

where ρ is a parameter that determines the relative weights of the recent sample value and historical values, and n is the number of observed samples. In our experiments, we set $\rho = 0.1$ to obtain the environment component.

We propose the EWMA method for extracting the environment component for three reasons. First, the EWMA method does not require a large data buffer and is suitable for realtime breathing monitoring. Second, as the moving average (MA) method, the EWMA method can effectively extract the outline of realtime data, from which the breathing signal can be filtered. Last but not least, compared with MA, EWMA is more sensitive to the more recent sample data. When the body moves, EWMA will capture a large change in M_t and thus can have a more rapid response, which is beneficial for detecting environmental changes.

After obtaining the environment component, we first apply the MA method to the original CSI data to remove high frequency noises, where the window size is set to 3. Then, the breathing component can be extracted by subtracting the environment component from the denoised CSI data. Fig. 9.5 illustrates calibration of the original phase difference. We can see in the top plot that the

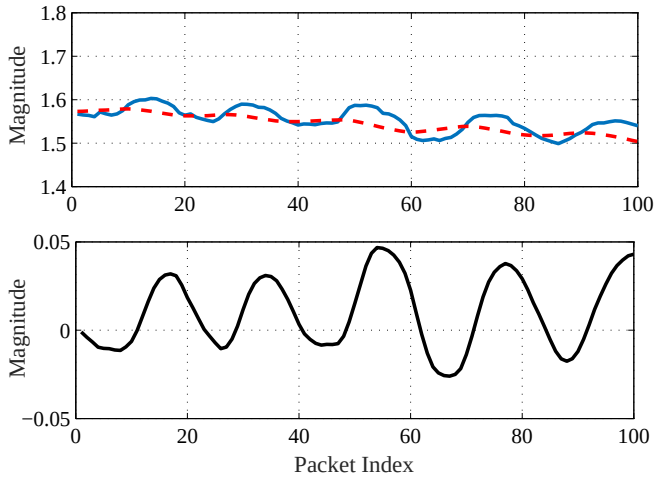


Figure 9.5: Data calibration results.

original phase difference values, for subcarrier 5 between antennas 1 and 2, have a DC component as well as high frequency noises. Applying the EWMA method, we can obtain the environment component, which is the outline of the original phase difference. Then, the breathing component can be extracted by de-noising the CSI phase difference data and removing the environment component, which exhibits a sinusoidal-like periodicity over the received packets with low noise (see the bottom plot).

9.3.3 Adaptive Signal Selection

For adaptive signal selection module, we will implement signal energy detection and movement detection to select the most sensitive signal group from three CSI amplitude groups and three CSI phase difference groups, because there are different sensitivities for CSI amplitude and phase difference information.

Signal Energy Detection

For signal energy detection module, we consider the energy values of breathing components to measure the sensitivity of CSI amplitude and phase difference information, where one CSI data group includes 30 values with the every received packet for CSI amplitude and phase difference information. Moreover, we consider the window size of online CSI data as 20 to compute the local

Algorithm 10: Movement Detection

Input: current average of the environment components of all phase difference (A_n), and the averages in the 2 last states (A_{n-1}) and (A_{n-2});
Output: move_flag;
//Initialize ;
set $move\_flag = 0$;
set $count\_num = 0$;
//Movement detection;
if $count\_num < 9$ **then**
    if *(*$A_n > 1.05A_{n-1}$ *and* $A_{n-1} > 1.05A_{n-2}$*) or (*$A_n < 0.95A_{n-1}$ *and* $A_{n-1} < 0.95A_{n-2}$*)* **then**
        $count\_num = count\_num + 1$;
    end
    else
        $count\_num = 0$
    end
end
else
    set $move\_flag = 1$;
    set $count\_num = 0$;
    return $move\_flag$;
end

energy value of the j_{th} CSI data group, $E_j(t)$ which is formulated by

$$E_j(t) = \sum_{i=1}^{i=30} \sum_{t=k-19}^{t=k} \left| Y_{ij}^{d}(t) \right|^2 \tag{9.5}$$

where k is the current data point, $Y_{ij}^{d}(t)$ is the normalized breathing component data for the i_{th} subcarrier of the j_{th} CSI data group at the t_{th} time. Compared with other advanced method such as FFT based method, the signal energy detection method can leverage smaller data points to measure the sensitivity of CSI amplitude and phase difference information, which can reach the real-time processing requirement. Signal energy values for six CSI groups are leveraged for signal selection module.

Movement Detection

In this module, we make use of the environment component in the phase difference data to detect body movement. We choose phase difference data because it has a fixed range between $-\pi$ and π, while CSI amplitude data has a variable range. We first compute the average of the environment components of all the phase difference data over the 30 subcarriers, which can be used to detect body movement by comparing with adjacent average values. In fact, the phase difference data can increase or decrease due to body movement. Thus, a body movement is detected if the current average value is lower than 0.95 times or larger than 1.05 times of the previous average value.

Furthermore, we find that small movements of the body or movements from nearby persons may also cause large changes in the average of the environment component. We propose a counting method to deal with such small body movements and environmental interference. The counting method is designed in algorithm 10 by the following steps. First, the count is initialized to 0. Then, the count will be increased by 1 when the current average value of phase difference data is lower than 0.95 times or larger than 1.05 times of the last average value. However, once the above condition is violated, the count will be reset to 0. A movement is detected if and only if the count reaches 10. In ResBeat, we implement the counting method that uses the three CSI phase difference data groups for movement detection. If a movement is detected in one of the CSI phase difference data groups, ResBeat will execute the following signal selection algorithm.

Signal Selection

This module is to select the most sensitive group from the six bimodal CSI data groups, for accurate breathing signal monitoring. The procedure is presented in Algorithm 12. The input parameters include the movement detection flag (mov_flag) and the local energy of each signal group (E_1, E_2, ..., E_6) computed as in (9.5). The output is the index of the most sensitive CSI data group.

In the beginning, we use 30 phase difference data from antennas 1 and 2 to implement the signal selection no matter whether a movement is detected or not. After the initialization, we use the above movement detection module based on the counting method to determine whether the

environment has a large change. When a movement is detected, we run the signal selection again. First, the system will wait for 2 s so that the environment components based on EWMA can return to stable values (recovers from the movement). Then, the system sorts the signal groups (i.e., 1–6) in descending order of the local energy of the CSI data groups (i.e., E_1, E_2, ..., E_6). If a group is ranked top 1 for three consecutive times, it will be selected as the most sensitive data group. If the top-ranked group is changed before the $select\_count$ reaches 3, the counting number will be reset to zero. Moreover, if there is a new movement detected, the signal selection will be restarted. ResBeat can keep on using the previously selected data group to monitor breathing beats until the next most sensitive data group is selected. The proposed signal selection algorithm is robust to environment interference and small movements of the body. Moreover, with the signal selection module, ResBeat can rapidly recover for large body movements.

9.3.4 Breathing Signal Monitoring

Although both of CSI amplitude and phase difference data can extract the breathing beats with small movement such as inhaling and exhaling, the most sensitive signal group we select can be effectively to avoid the anonymous breathing signal in some locations. Traditionally, FFT based method can estimate the breathing frequency with the larger moving window size to improve the estimated accuracy. However, it cannot achieve real-time breathing rates estimation for ResBeat system. Thus, the proposed ResBeat system employs the peak detection to compute the breathing rate online.

We leverage peak detection to estimate the breathing rates based on the breathing components in the selected signal group. However, the breathing components still include the fake peak, which is not the true peak but its value is larger than two neighboring points. Fig. 9.6 shows the calibrated breathing signal for 30 s, which contains five times respiration. We can see that, there is a fake peak located inside the second respiration. Even though the phase difference of the fake peak is higher than its two adjacent values, it should not be considered as a breathing signal peak. To avoid the fake peak, we employ a moving window approach with the window size as 6 samples to get the all

Algorithm 11: Signal Selection Algorithm

Input: movement detection flag (mov_flag) and the local energy of each data group (E_1, E_2, ..., E_6);
Output: index of the most sensitive data group;
Set $initialize\_flag = 1$;
Set $select\_num = 1$;
if $mov\_flag == 1$ **then**
    Wait for 2 seconds ;
    Set $mov\_flag = 0$;
    Set $select\_count = 0$;
    while $mov\_flag == 0$ *or* $initialize\_flag == 1$ **do**
        Sort the data groups (1-6) in descending order of (E_1, E_2, ..., E_6);
        if *New first group index equals to the former first group index* **then**
            $select\_count++$;
            if $select\_count == 3$ **then**
                Choose the first data group;
                Set $initialize\_flag = 0$;
                Set $select\_count = 0$;
                Break ;
            end
        end
        else
            Set $select\_count = 0$;
        end
    end
end
Return the index of the most sensitive data group ;

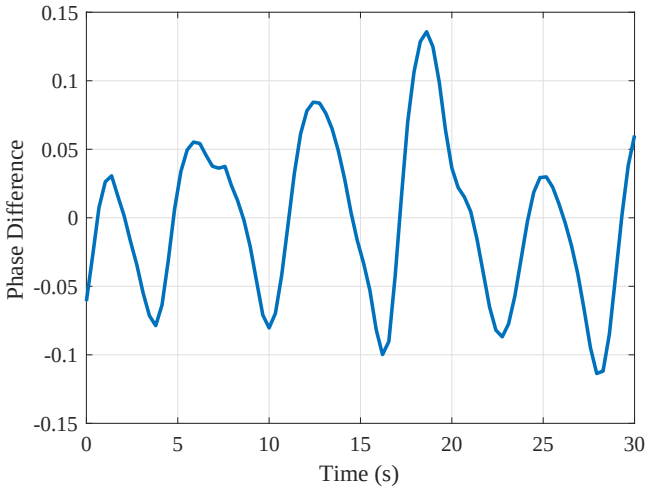


Figure 9.6: Calibrated breathing signal with the fake peak.

true peaks in a buffer with 100 points, which can be detected by determining whether the median of all points in the window is the maximum value or not. Then, we can compute the breathing period t by averaging all peak-to-peak intervals. Because the buffer data can be updated for real-time, we build a small buffer with 20 points to store 20 estimated breathing periodic values. Then, the final period of breathing signal can be computed by $T = \frac{1}{20}\sum_{i=1}^{i=20} t_i$. Therefore, we can obtain the estimated breathing frequency with $60/T$ bpm.

9.4 Experimental Study

9.4.1 Test Configuration

In our experiments, ResBeat operates in the 5GHz ISM band. Our ResBeat system is executed on the Ubuntu Desktop 14.04 LTS OS for both the transmitter and receiver, each of which is equipped with an Intel 5300 NIC. The transmitter is a Lenovo laptop set in the injection model, which transmits 10 packets per second using one antenna. The receiver is an Acer laptop working in the monitoring mode for collecting CSI data, where the three antennas are placed in a row with an interval of 2.68 cm. The ResBeat system collects realtime CSI amplitude and phase difference data from the three antennas for breathing beats monitoring.

We implement our Resbeat system in three different environments with the same floor in Fig. 9.7. In the first environment, both the transmitter and the receiver are placed in a Laboratory with the area of 4.5 $\times$ 8.8 m^2, where the person can stay at anywhere in the office. The second scenario is the through-wall scenario to test the performance of the ResBeat system for the received weak wireless signal because of large signal attenuation. The third scenario is implemented in the corridor to validate the influence of the long distance between the transmitter and the receiver for breathing rates estimation. On the other hand, we leverage omnidirectional antennas for the transmitter and the receiver at all three scenarios. In addition, NEULOG Respiration is used to measure the ground truths of the breathing rates.

The breathing signal could be extremely weak at certain measurement locations, which is hard to detect [134]. To measure the resilience of breathing rate monitoring methods, we define a

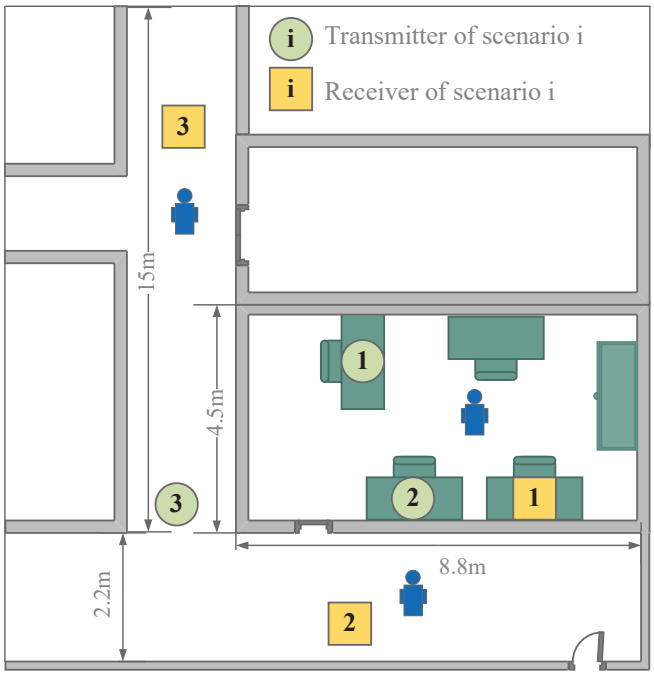


Figure 9.7: Configuration of the ResBeat experiments.

success rate performance metric, denoted by η, as

$$\eta = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\text{sgn}(2-e_i)+1}{2}\right), \tag{9.6}$$

where e_i is the breathing estimation error in bpm for the i_{th} location, sgn$(\cdot)$ is the `sign` function, and N is the total number of different locations tested in the experiment. The success rate represents the ratio of the number of locations having an error less than 2 bpm to the total number of locations.

9.4.2 Performance of Breathing Rate Estimation

Fig. 9.8 plots the CDF of estimation errors of breathing rate in the computer laboratory, through-wall, and long corridor scenarios. We find that ResBeat achieves lower breathing beats estimation errors, where the maximum error is less than 1.75 bpm. Moreover, it can be seen that for ResBeat, the median errors are 0.25 bpm, 0.25 bpm, and 0.3 bpm for the computer laboratory, long corridor, and through-wall cases, respectively. The breathing estimation errors in the laboratory and long

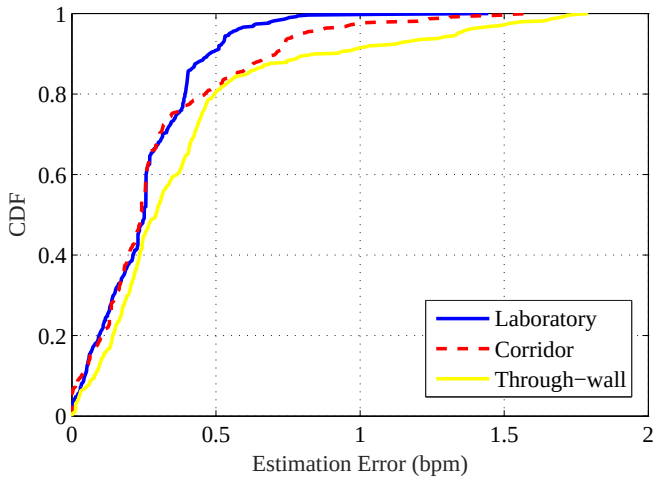


Figure 9.8: Performance of breathing rate estimation in the computer laboratory, through-wall, and long corridor scenarios.

corridor scenarios are lower than that in the through-wall scenario. This is because the breathing signal in the through-wall scenario is much weaker. In fact, the accuracy for the through-wall scenario is still high and acceptable. We conclude that the proposed ResBeat system is robust in different scenarios.

Fig. 9.9 presents the success rates for three different schemes in the computer laboratory, through-wall, and long corridor scenarios. In this experiment, We use the amplitude based method [116] and PhaseBeat [143] as benchmarks for success rate comparison. We find that the proposed ResBeat system achieves high success rates about 90% in the laboratory and long corridor scenarios, and about 86% in the through-wall scenario. This means that stronger breathing signals can help to achieve higher breathing beat estimation accuracy. On the other hand, the success rate of the proposed ResBeat is higher than that of the other two schemes in all the three environments. This is because ResBeat employs bimodal CSI data and the adaptive signal selection method to select the most sensitive data group, which can effectively mitigate the effect of anomalous breathing signals at certain locations.

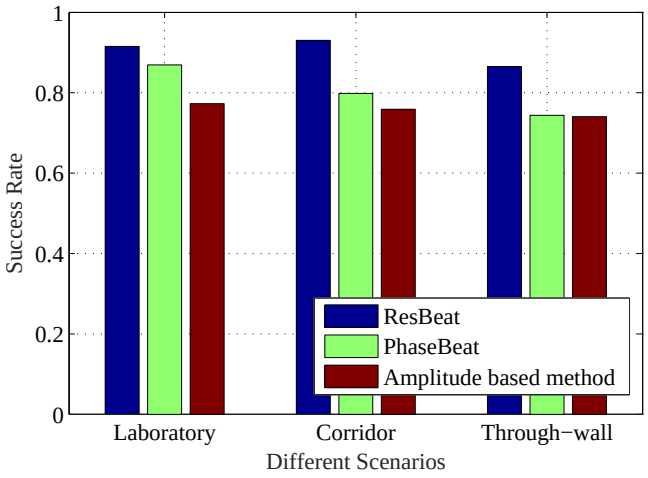


Figure 9.9: Success rates for three different schemes in the computer laboratory, through-wall, and long corridor scenarios.

9.4.3 Impact of Environments and Parameters

Fig. 9.12 shows the success rates for different orientations. The transmitter and the receiver are placed besides each other in this experiment in A part of Fig. 9.10. 0 degree means the user faces directly to the devices, and 180 degree means the back is directly to the devices. The results show that when the user directly faces to the devices, the success rate achieves the maximum, which is about 95%, while the minimum success is about 80% with the human back is directly to the antennas. We can also find out that the success rate decreases as the user turns back to the devices. It is because the reflected signal strength is the highest when the user faces the transmitter. As the user turns around, the wireless signal can be rarely transmitted to the human chest directly. Although the signal can still reach the human chest by reflecting from surroundings, the strength of the signal is much more weaker than the direct transmitted signal.

Fig. 9.11 shows the success rates for different distances between the user and devices. In this scenario, the transmitter and the receiver are all placed close to each other in the laboratory in B part of Fig. 9.10. As Fig. 9.11 shows, when the distance between the user and devices increases, the success rate of the ResBeat system is reduced because of the loss of reflected signal strength. Moreover, we can see that the success rate is higher than 90% when the distance increases from 1 m to 3 m, while the success rate drops significantly as the distance is larger than 4 m. This is because

when the user is not so far from the transmitter, the most of reflected signal from surroundings can reach human chest. These signals can also help to increase the strength of the reflected signal from the human chest. However, when the user is too far away from the transmitter, only a few wireless signals can be reflected by the human chest. Thus, the success rate of the system decreases significantly with long distance from the user to the devices.

To prove the robustness of the ResBeat system, we evaluate our system in five different positions in the laboratory in Fig. 9.13. Different positions mean different deployments for the transmitter, the receiver, and the user. The wireless channel between the transmitter and the receiver is independent for each position, so the success rates for these five locations are statically independent. We notice that the minimum success rate is achieved in position 4 which is about 85%, while the highest success rate achieved in position 1 is about 95%. Fig. 9.13 shows that success rates for all five different positions are all higher than 85%, and three of them are higher than 90%. The results demonstrate that the the proposed ResBeat system is robust for monitoring human breathing rate in different environments.

We also test the impact of different EWMA parameters on the success rate of the ResBeat system in Fig. 9.14. As we mention, EWMA algorithm is used to extract environment component from the raw received signal. From the equation 9.4, we find out that if the parameter ρ is too small, the EWMA will hardly capture the real-time change of human body. However, if the parameter ρ is too large, the EWMA result will be too sensitive to the instant movement so the environment component is hardly to be extracted. To find out the appropriate parameter ρ for EWMA, we test the success rates for different ρ. Fig. 9.14 represents the success rate with different parameter ρ in EWMA algorithm. we can see that, the success rate achieves the maximum when the ρ equals to 0.1. The success rate decreases significantly when the parameter ρ is smaller than 0.05 or larger than 0.25. As a result, we choose 0.1 as the parameter used in the EWMA algorithm.

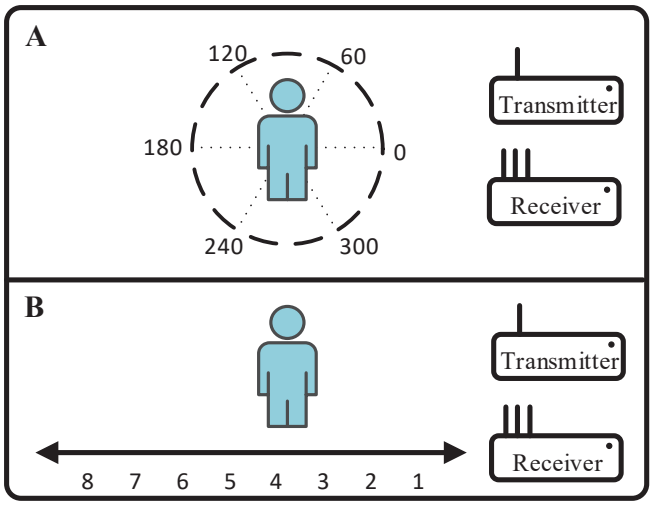


Figure 9.10: Deployment for different orientations and distances.

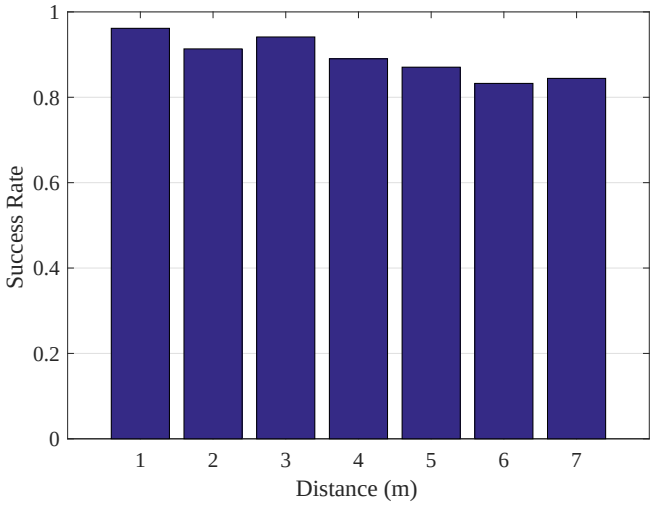


Figure 9.11: Success rates for different distances.

9.5 Conclusions

In this chapter, we proposed ResBeat system, resilient breathing beats monitoring by using online CSI amplitude and phase difference data in the commodity WiFi devices. We developed the ResBeat system including data preprocessing, adaptive signal selection and breathing signal monitoring. For data preprocessing, we implemented the data extraction and calibration to obtain the breathing component and environment component. For the adaptive signal selection, we proposed the signal selection algorithm based on signal energy detection and movement detection for selecting the most sensitive signal group. Then, we leveraged peak detection to estimate breathing

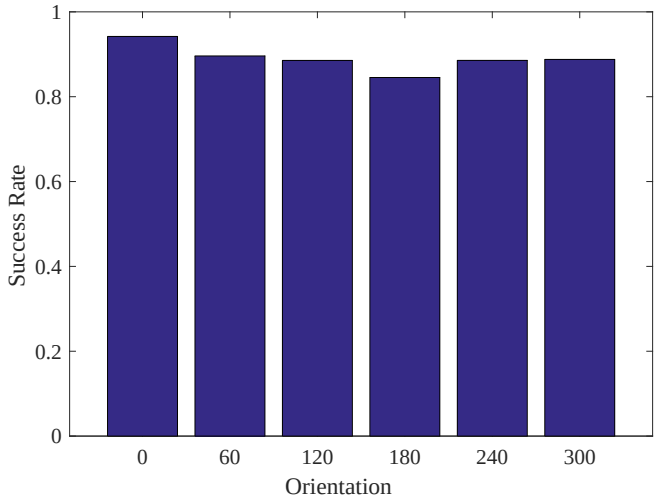


Figure 9.12: Success rates for different orientations.

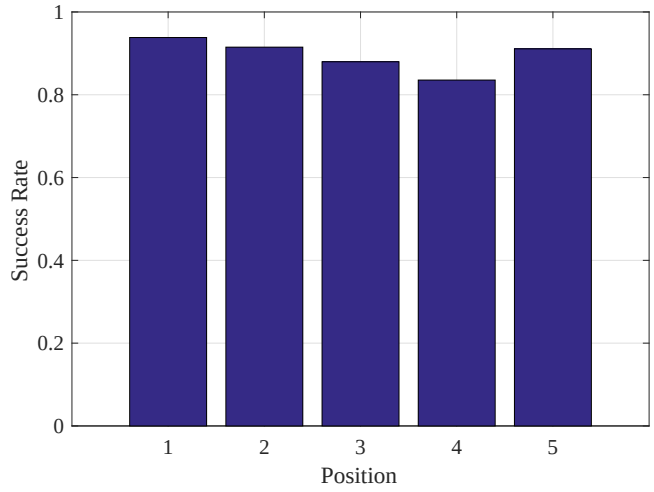


Figure 9.13: Success rates for different positions in the laboratory.

rates. We implemented with the experiments with three different scenarios. The results validated the effectiveness of the proposed ResBeat system.

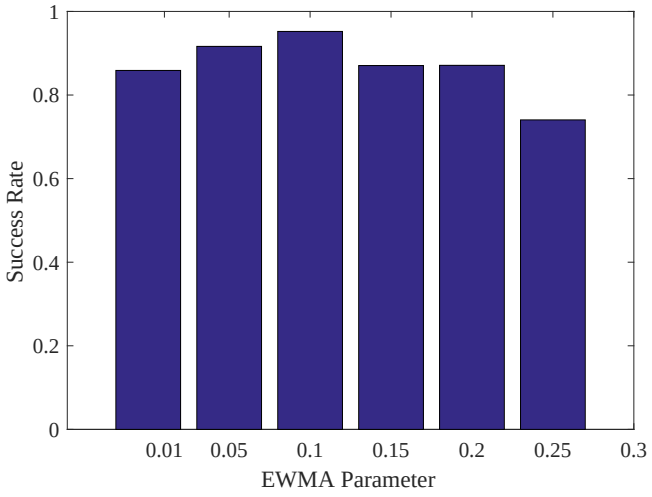


Figure 9.14: Success rates for different EWMA parameters.

Chapter 10

SonarBeat: Sonar Phase for Contactless Breathing Beats Monitoring with Smartphones

10.1 Introduction

With the rapid development of mobile techniques and the growth in the living standard, healthcare has become one of the main application areas for IoT [146, 2, 147]. The healthcare IoT architecture mainly consists of three layers: (i) the sensing layer for monitoring vital signals, such as body temperature, heart rate, respiration rate, and blood pressure; (ii) the gateway layer for collecting data from the sensing layer, and transmitting them to the third layer, the cloud layer [148]; (iii) the cloud layer consisting of data centers in the cloud to store, process, and analyze multi-modal medical datasets [149, 150, 151, 152, 153], and deliver the analysis results to medical centers. Particularly, the respiration signal is one of the key vital signs to be collected in the first layer, which is indispensable for physical health monitoring, since such vital signals can offer important information for personal health problems such as SIDS [107]. Traditional systems in the sensing layer require a person to wear special devices, such as a pulse oximeter [109] or a capnometer [108] to monitor breathing rates, which are not convenient for monitoring vital signals for the elders and infants, and are hard to be used for an extended period of time. Thus, technologies that can enable contact-free, easy deployment, and long-term vital sign monitoring are highly desirable for healthcare provisioning.

Existing vital signal monitoring systems are mainly focused on radio frequency (RF) based techniques, which leverage RF signals to capture breathing and heart movements. The existing techniques can be classified into (i) radar based and (ii) WiFi based approaches. Examples of radar

based vital sign monitoring include Doppler radar [111, 112], ultra-wideband radar [113], FMCW radar [110], all of which require a piece of customized hardware working on high frequency. With the development of wireless techniques, WiFi based methods include UbiBreathe [115] and mmVital [114], which exploit the RSS of 2.4 GHz WiFi and 60 GHz mmWave signals (i.e., 802.11ad), respectively. Recently, the authors in [116] employ the amplitude of WiFi CSI data to track vital signs for a sleeping person, while our prior works PhaseBeat [143] and TensorBeat [144] exploit the CSI phase difference data for vital sign monitoring for single and multiple persons, respectively. Although RF based techniques work over a relatively long distance, they could be susceptible to environmental change, such as the movements of other persons nearby.

To this end, the smartphone can serve as an excellent platform for vial sign monitoring, by exploiting its built-in sensors, such as accelerometer, gyroscope [119], and microphone [120]. Usually the smartphone should be placed near the body, or the person needs to wear special types of sensors that connect to the smartphone. The device-free and contact-free monitoring techniques aim to relieve the burden of attached sensors. In a recent work [154], the authors propose to use an active sonar built in the smartphone by leveraging the FMCW technique for respiration monitoring. The scheme is shown to work well, but the FMCW based technique requires an accurate estimation of the distance between the smartphone and the chest. When the body suddenly moves (e.g., rolling over in bed), the system needs to detect the new smartphone-chest distance, thus leading to a large time complexity. Alternatively, the Low-Latency Acoustic Phase (LLAP) system employs a continuous-wave (CW) radar to measure distance and achieves device-free hand tracking using sonar phase information [155]; while [156] uses the phase of acoustic OFDM signals for finger tracking.

Motivated by these interesting studies, we employ sonar phase data with a smartphone implementation to monitor the periodic signal caused by the rises and falls of the chest (i.e., inhaling and exhaling). We find that the sonar phase information can effectively track the periodic signal of breathing rate with a high accuracy. Compared with other existing systems such as Doppler shift and FMCW [154], the sonar phase based scheme has a lower latency and complexity. In addition,

the sonar phase data is highly robust to different orientations, distances, and respiration rates of different persons.

Specifically, we first present a rigorous sonar phase analysis, which proves that the sonar phase information can accurately capture the breathing rate with the same frequency. Built upon analysis, we design SonarBeat, an active **Sonar** phase for **Breat**hing rate monitoring system with a smartphone. The SonarBeat system consists of four modules, including signal generation, data extraction, received signal preprocessing, and breathing rate estimation. First, it transmits an inaudible sound signal in the frequency range of 18-22 KHz from the smartphone speaker, which serves as a CW radar. Then, the reflected signal from the chest of the monitored subject is received by the microphone of the same smartphone. The received signal is then calibrated and the breathing signal will be recovered. We implement SonarBeat with a Android smartphone and validate its performance with extensive experiments that involve five persons over a period of three months in three different environments, including an office scenario, a bedroom scenario, and a movie theater scenario. The experimental results show that SonarBeat can achieve a low estimation error for breathing rate estimation, with a medium error of 0.2 bpm in most experiments. We also find that SonarBeat is highly robust to different experimental parameters and settings.

The main contributions of this chapter include the follows.

- Through analysis and experiments, we validate the feasibility of leveraging the active sonar phase information for breathing rate estimation. To the best of our knowledge, this is the first work to employ active sonar phase information for breathing monitoring with smartphones.
- We design SonarBeat based on the analysis and address the technical challenges on using active sonar phase. We implement several signal processing algorithms, including signal generation, data extraction, received signal preprocessing, and breathing rate estimation. Specially, we propose an adaptive median filter approach to remove the static vector in the received signal, which allows to effectively extract the inaudible phase information.

- We prototype the SonarBeat system with Android smartphones and validate its superior performance with comparison with an existing scheme in three different indoor scenarios. Our extensive experimental results demonstrate the superior performance of SonarBeat under different environment factors and different experimental parameters.

In the remainder of this chapter, Section 10.2 reviews related work. Then, we present the sonar phase analysis and technical challenges in Section 10.3. We describe the SonarBeat design in Section 10.4 and validate its performance in Section 10.5. Section 11 concludes this chapter.

10.2 Related Work

The work is related to the prior works on sensing systems and mobile health systems based on audible signal with smartphones. We review the key related works in this section.

Mobile sensing systems with audible signals have attracted great attention [157, 158]. Such systems are increasingly convenient for people's life and healthcare. In the meantime, by using mobile audible sensing systems with smartphones, people do not need to pay extra money for new devices. Traditionally, mobile audible sensing systems can be classified into two categories, including passive and active sensing systems. First, the passive audible sensing systems mainly focus on how to leverage the microphone to sense and recognize the surround audible signal [159]. The recent work AAmouse leverages an inaudible sound pulse at different frequencies to transform a mobile device into a mouse by exploiting the Doppler shift, speed, and distance estimation [160]. Moreover, the CAT system implements a distributed FMCW for tracking devices, such as VR/AR headsets. This work mainly is focused on synchronizing two smartphones and using the microphone as a mouse, which can interact with VR/AR headsets for more accurate localization [161]. In addition, audible based sensing technique is also used for wireless virtual keyboard with smartphones. Keystroke snooping [162] and Ubik systems [163] can obtain the sound signal with a smartphone's single or dual microphones, and leverage the time-difference-of-arrival (TDOA) measurements to monitor finger stroke on the table. Then, the strokes are transformed into related alphabets in the same position as a computer keyboard. SilentWhistle is a light-weight

indoor localization system using acoustic sensing for obtaining users' locations [164]. Another work Dhwani builds an acoustics-based NFC system with smartphones, using a technology JamSecure. It can provide a secure communication channel between devices, which is an OFDM channel for audible signals [165].

On the other hand, active inaudible sensing systems transfer a smartphone to an active sonar using ultrasonic sound waves at 18 KHz to 22 KHz, which is closely related to the proposed SonarBeat system [166]. OFDM based sensing systems such as FingerIO can track the finger movement in a 2-D domain through tracking echoes from the finger that are received by the microphone, to measure the finger position [156]. BatMapper employs an acoustic sensing based system for fast and accurate floor plan construction using commodity smartphones [167]. Moreover, LLAP leverages the principal of CW radar to measure distance and implement device-free hand tracking [155]. This work is closely related to SonarBeat, because both system use the phased based CW signal to sense movements. The difference between the two systems is that SonarBeat is more robust to different environments with the adaptive median filter technique. On the other hand, the AudioGest system employs a pair of built-in speaker and microphone to send inaudible sound and leverage the echos to sense the hand movement [168]. Our SonarBeat system is motivated by the active inaudible sensing systems to transform a smartphone into an active sonar with ultrasonic sound waves.

Mobile health applications and research have become an important part of the IoT [169]. Smartphones and other wearable devices can provide people with a more convenient way to monitor their health conditions without the need of professional equipment [170]. The recently work Burnout leverages accelerometers to sense skeletal muscle vibrations, which does not require to wear a suit embedded with sensors [171]. The authors in [172] propose to capture the depth video of a human subject with Kinect 2.0 to monitor heart rate and rhythm. Moreover, wearable devices for monitoring exercise and body are widely available. For example, the FEMO system achieves an integrated free-weight exercise monitoring service with RFID tags on the dumbbells and leverages the Doppler shift for recognition and assessment of free-weight exercise [171]. For vital signal

monitoring, fine-grained sleep monitoring using a microphone in the earphone can record human breathing sound to monitor people's heath signal when they are sleeping [120], which adopts a passive audible method. The Apnea system uses an active sonar with smartphone to monitor the breathing signal [154]. This work leverages the FMCW technique for breathing monitoring, which requires the system to seek the distance between the smartphone and the chest of the person.

10.3 Sonar Phase Analysis and Technical Challenges

10.3.1 Sonar Phase Analysis

We propose to use smartphones to monitor respiration signals by utilizing an inaudible sound signal, where the speaker and microphone of the smartphone emulate an active sonar system. In particular, the speaker transmits an inaudible sound signal in the frequency range of 18–22 KHz, in the form of a CW signal as $C(t) = A\cos(2\pi ft)$, where A is the amplitude and f is the frequency of the sound. Then the signal is reflected by the chest of test subject and received by the microphone. One unique advantage of the smartphone based design is that, because the speaker and microphone use the same frequency, there is no carrier frequency offset (CFO) errors between the sender and receiver. Thus, we can exploit the phase of the received inaudible signal to accurately estimate the vital sign.

To extract the phase of the CW signal, we need to design a coherent detector to down-convert the received sound signal $R(t)$ to I-component and Q-component of a baseband signal in Fig. 10.1. I-component and Q-component of the baseband signal can represent a complex vector, which can be used to obtain the amplitude and phase information of the baseband signal. For SonarBeat system, we mainly exploit the phase information extracted from I-component and Q-component of the baseband signal, thus capturing the period signal caused by the movement of the chest such as inhaling and exhaling. The SonarBeat design is to first split the received sound signal into two identical copies. Then, these two copies are multiplied with the transmitted signal $C(t) =$

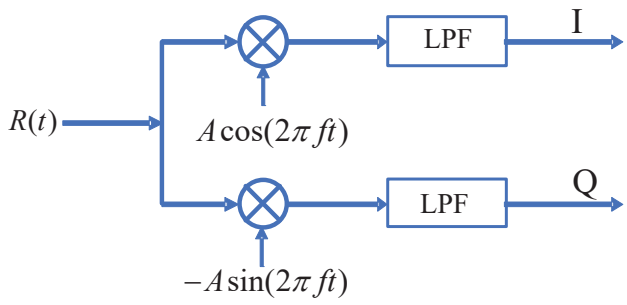


Figure 10.1: I/Q demodulation for the received signal.

$A\cos(2\pi ft)$ and its phase shifted version $C'(t) = -A\sin(2\pi ft)$. Finally, the corresponding In-phase and Quadrature signals are obtained by using a low-pass filter (LPF) to remove the high frequency components.

We first present a simple analysis for the ideal case that there is no multipath effect (or, for the high SNR regime, where the LOS component is the dominant part of the received signal). Under the assumption, the inaudible signal travels through a single path (i.e., from the speaker to the chest, and then back to the microphone) and the propagation delay can be modeled as

$$d(t) = (D_0 + D\cos(2\pi f_b t))/c,$$

where D_0 is the constant distance of the reflected path, D and f_b are the amplitude and frequency of the chest movements, respectively, and c is the speed of sound. The received inaudible signal from this path can be modeled as $R(t) = A_r\cos(2\pi ft - 2\pi fd(t) - \theta)$, where A_r is the amplitude of the received inaudible signal and θ is a constant phase offset due to the delay in audio recording and playing. To estimate the phase of the inaudible signal, we need to remove the high frequency components. Multiplying the received signal with $C(t) = A\cos(2\pi ft)$, we have

$$\begin{aligned} & A_r\cos(2\pi ft - 2\pi fd(t) - \theta) \times A\cos(2\pi ft) \\ = & \frac{A_r A}{2}(\cos(4\pi ft - 2\pi fd(t) - \theta) + \cos(-2\pi fd(t) - \theta)). \end{aligned} \tag{10.1}$$

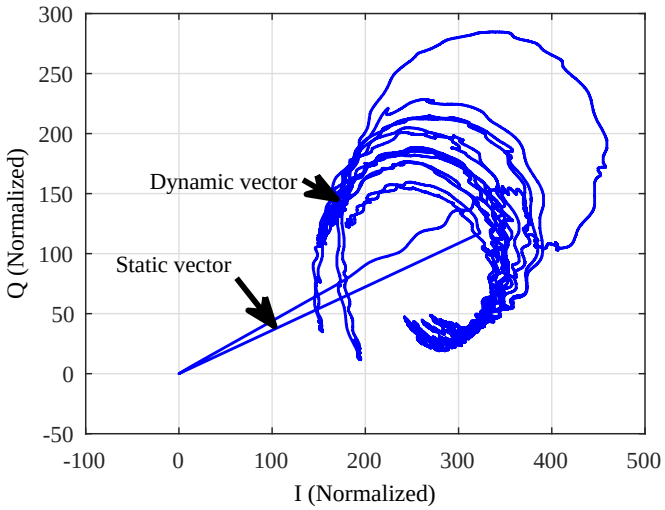


Figure 10.2: Complex I/Q traces of the received audio signal.

The first term in (10.1) has a high frequency of $2f$, which can be removed with a properly designed low-pass filter. Thus, the I-component of the baseband is extracted as $I = \frac{A_r A}{2} \cos\left(-2\pi f d(t) - \theta\right)$. With a similar approach (i.e., multiplying by $C'(t)$ and removing the high frequency component), we can estimate the Q-component of the baseband signal as $Q = \frac{A_r A}{2} \sin\left(-2\pi f d(t) - \theta\right)$. We then demodulate the phase of the inaudible signal data as

$$\begin{aligned} \varphi(t) &= \arctan(Q/I) = -2\pi f d(t) - \theta \\ &= -2\pi f (D_0 + D\cos(2\pi f_b t))/c - \theta. \end{aligned} \tag{10.2}$$

Note the phase signal $\varphi(t)$ has the same frequency as the respiration signal. In the general case, the received inaudible signal is a complex signal, which includes a *static component* and a *dynamic component* due to the multipath effect in indoor environments. For example, Fig. 10.2 shows complex I/Q traces for the received audio signal, with the static vector and dynamic vector in the I-Q plane. To track the breathing rates, we need to demodulate the phase from the I/Q components by removing the static vector. Fig. 10.3 shows the complex I/Q traces of the received audio signal after removing static vector. It is noticed that the demodulated phase is a good indicator of the breathing caused chest movements.

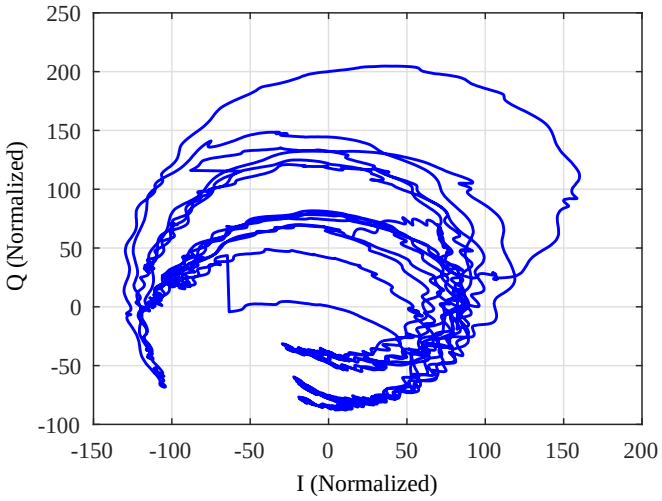


Figure 10.3: Complex I/Q traces of the received audio signal after removing the static vector effect.

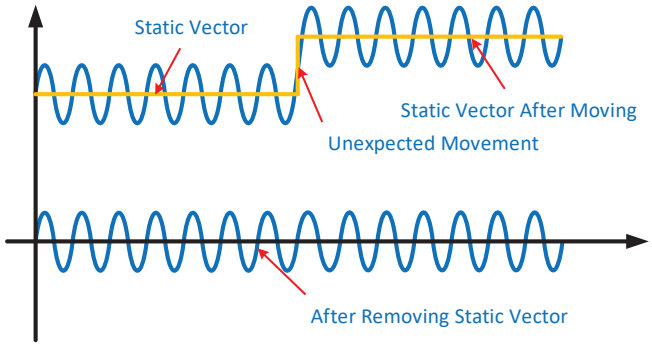


Figure 10.4: Illustration of adapting to body movements by eliminating the static vector.

10.3.2 Technical Challenges

Mitigate the Static Vector Effect

The main challenge for respiration monitoring with phase modulated data is to mitigate static vector effect, which directly influences the sensitivity and correctness of the phase data. The larger the stationary component, the larger the error in the extracted phase data. This is because the SNR at the receiver will become low when there is a large static component, making it hard to demodulate the phase data. In [155], the authors adopt the local extreme value detection (LEVD) to remove the stationary components for hand tracking. However, this method may not be effective for respiration monitoring, because the LEVD method needs to set an empirical threshold for each different environment. In this chapter, we propose an adaptive median filter method to deal

with this challenge, which is shown effectively for removing the stationary component in different scenarios.

Adapt to Body Movements and Environment Noise

The second challenge for tracking breathing signals is adapt to body movements and environment noise. Body movement is unavoidable in the monitoring phase, e.g., during sleep, and its impact should be mitigated. The FMCW based scheme in [154] requires to estimate the distance between the smartphone and chest before respiration monitoring. When the body suddenly moves, the system needs to seek the new distance, thus leading to large time complexity. The proposed sonar phase based approach is effective on adapting to body movements. For example, Fig. 10.4 illustrates the idea of adapting to body movements by eliminating the static vector in SonarBeat. When there is an unexpected small body movement, the magnitude of the breathing signal becomes larger, leading to a smaller SNR. After eliminating the static vector, we can mitigate the effect of body movement, and still obtain a neat respiration signal, as shown in the lower part of Fig. 10.4.

Furthermore, consider the case of multiple persons in the testing environment. Not only their movements cause interference to the reflected respiration signal, but also the background noise could be high (e.g., when they are talking). We employ coherent I/Q demodulation in SonarBeat to remove the environmental noise from external audio sources.

Realtime Monitoring with Lower Delay

For a vital sign monitoring system to be really useful, it should work in realtime, with good interactions with the user. Realtime monitoring is challenging since most smartphones have a high sampling rate of 48 KHz, which leads to 96,000 multiplication operations per second for the coherent detector to down convert the received sound signal to the base band. To address this challenge, we perform down-sampling for I/Q demodulation, which can reduce the computation complexity while still capturing the breathing rate.

For better interaction with the user, SonarBeat operates in three stages. In the first stage of 15 s, it performs respiration monitoring in realtime without FFT based breathing rate estimation. In the second stage of 15 s, SonarBeat analyzes the data collected in a 15-second sliding window to extract the respiration signal, and plots the respiration signal on the screen to give the user some preliminary testing results. In the final stage, after 30 s, SonarBeat applies FFT to all the captured phase data to achieve an accurate breathing rate estimation.

10.4 The SonarBeat System

10.4.1 SonarBeat System Architecture

According to the sonar phase analysis, SonarBeat can effectively exploit sonar phase information to monitor respiration signals. First, the phase information can track the periodic breathing rates with a high accuracy, and the phase information is sensitive to the small breathing induced chest movements. Second, compared with other traditional methods, such as Doppler shift and FMCW [110], the phase based approach has a lower latency and complexity. Finally, the sonar phase data is robust to different orientations, different distances, different cloth thickness, and different breathing rates of different persons. It is also robust to large body movements, which only leads to a change of the stationary component of the phase data, which can be effectively removed with the proposed adaptive median filter method.

Fig. 10.5 presents the SonarBeat system architecture, which includes four basic modules: (i) Signal Generation, (ii) Data Extraction, (iii) Received Signal Preprocessing, and (iv) Breathing Rate Estimation. The *Signal Generation* module mainly implements a Pulse-code Modulation (PCM) of the inaudible signal, where a CW signal at 18 KHz to 22 KHz is generated and modulated with the PCM technique. The *Data Extraction* module is to detect the audio signal, which employs a Short-Time Fourier transform (STFT) for audio signal detection. A threshold based method is proposed for detecting the beginning part of the received inaudible signal. The *Received Signal Preprocessing* module consists of (i) I/Q demodulation, which implements downsampling followed by a coherent phase detector; (ii) static vector effect reduction, which implements the proposed

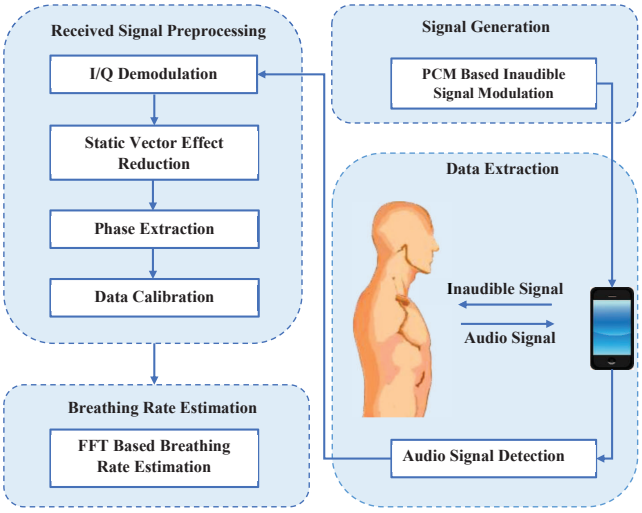


Figure 10.5: The SonarBeat system architecture.

adaptive median filter method to remove the static vector of the In-phase and Quadrature signals; (iii) phase extraction, where the sonar phase information is extracted and calibrated; and (iv) data calibration, where a median filter is applied as a simple Low Pass finite impulse response (FIR) filter to remove noise. The *Breathing Rate Estimation* module employs an FFT based method to estimate the breathing rate.

10.4.2 Signal Generation

The signal generation module uses one speaker of the smartphone as transmitter, to produce the inaudible signal. We implement the signal generation module as a PCM based modulator on the Android platform. Specifically, the speaker generates an inaudible sound signal in the frequency range of 18–22 KHz in the form of a CW signal, i.e., $C(t) = A\cos(2\pi ft)$. We produce the sampled analogy signal and then use PCM to digitally represent the sampled CW signal. To generate a PCM stream, the amplitude of the analog CW signal is sampled at uniform intervals, where each sampled value is quantized. The PCM based inaudible signal modulation is implemented with the AudioTrack class.

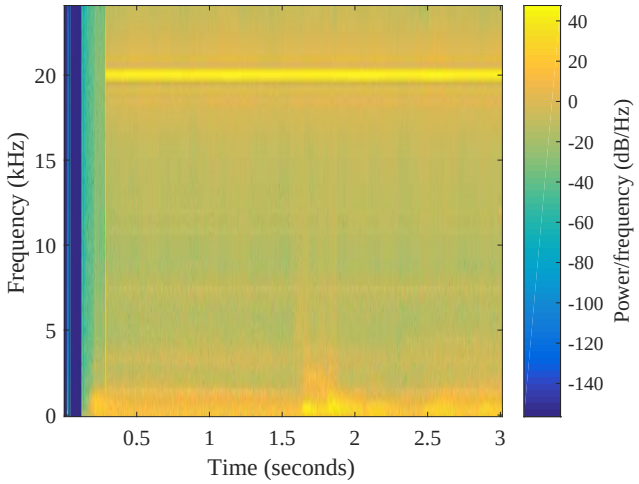


Figure 10.6: STFT based method for audio signal detection.

10.4.3 Data Extraction

We use the microphone of the smartphone to receive the inaudible signal reflected from the chest with a sampling rate of 48 KHz. The microphone will record other sound signals with different frequencies from the surrounding environment as well. We implement an audio signal detection method to identify the beginning of the desired signal as follows.

The proposed audio signal detection method is based on STFT. At the beginning of of the signal, there will be a drastic of power at the carrier frequency. A threshold based method is used to detect the beginning of the inaudible signal. Fig. 10.6 illustrates the STFT based method for audio signal detection, where the carrier frequency is 20 KHz. We can see that before 0.25 s, the microphone only receives audio frequencies from the surrounding environment. After 0.25 s, the microphone detects the inaudible signal, since the magnitude of the 20 KHz spectrum becomes much stronger than other audio frequencies (see the bright yellow, horizontal strip at 20 KHz). We adopt a window size of 512 in STFT for estimating the spectrum. Moreover, we set a threshold of 200 for the power change to detect the beginning of the inaudible signal. In fact, if we detect the power change with the threshold method, the beginning of the inaudible signal can be set as the end of the STFT chirp.

10.4.4 Received Signal Preprocessing

We describe the four components of the Received Signal Preprocessing module in this section, including I/Q Demodulation, Static Vector Effect Reduction, Phase Extraction, and Data Calibration.

I/Q Demodulation

Before I/Q Demodulation, we need to down-sample the received signal $R(t) = A_r \cos(2\pi ft - 2\pi fd(t) - \theta)$ for reducing computation complexity, which is necessary for realtime monitoring. The original system with sampling frequency of 48 KHz is reduce to 480 Hz with a down-sampling ratio of 100. Then, we implement the I/Q demodulation to obtain the I-component and Q-component of the baseband signal using a coherent detector. The design is to split the received audio signal into two identical copies. Due to the down-sampling ratio of 100, these two copies should be multiplied with the signal $A\cos(2\pi\frac{f}{100}t)$ and its phase shifted version $-A\sin(2\pi\frac{f}{100}t)$ to obtain the I-component and Q-component of the baseband signal, respectively. Because we use phase modulation for breathing monitoring, down-sampling only reduce the number of samples of the amplitude of breathing signal, rather than the phase information.

Finally, a LPF is employed to obtain the corresponding In-phase and Quadrature signals, which has a cutoff frequency of 1 Hz, a sampling rate of 480 Hz, and a resonance of 2. This setting has been shown to be effective for removing the high frequency components and environment noises. In Figs. 10.7 and 10.8, we plot the raw I-component and Q-component of the baseband signal, respectively, after the LPF (the dashed curves), which, however, still include their static vectors.

Static Vector Effect Reduction

As discussed, the performance of SonarBeat largely depends on how the effect of the static vector is mitigated in multipath environments. This is because usually the static vector is much stronger than the dynamic vector that representing the small chest movements. It is difficult to detect the weak breathing signal if we directly use the received sound signal. Recently, there are two methods

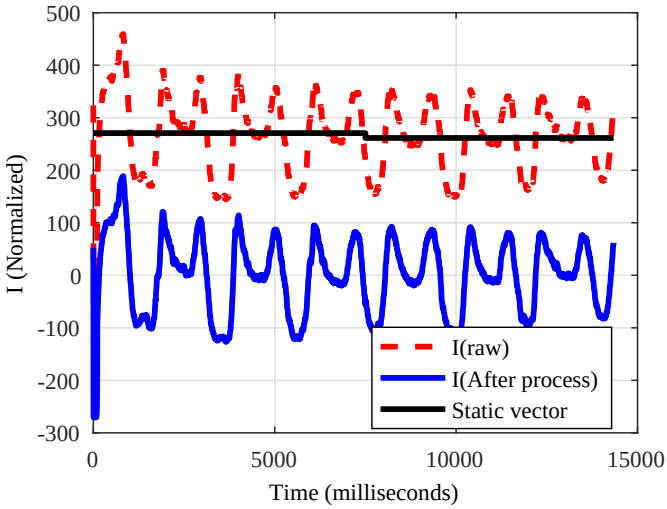


Figure 10.7: The adaptive median filter for removing the static vector in the baseband I-component.

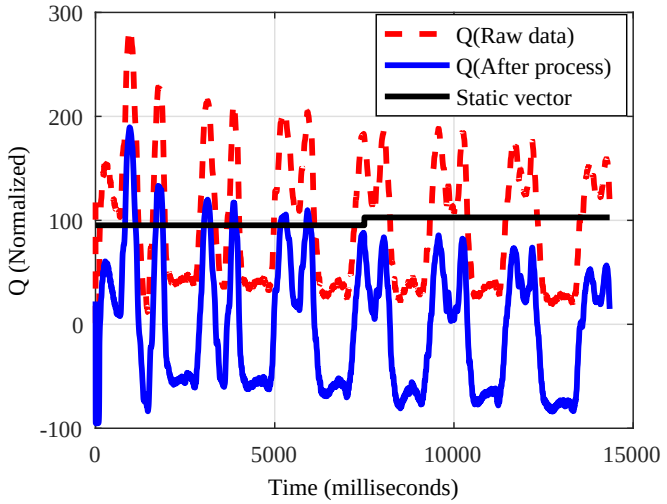


Figure 10.8: The adaptive median filter for removing the static vector in the baseband Q-component.

proposed for static vector effect mitigation. The Dual-Differential Background Removal approach is used for hand tracking with 60 GHz mmWave signals [173]. The method is susceptible to environment noise and has large latency, which is not effective for realtime respiration monitoring. The second scheme, termed LEVD [155], is also developed for tracking hand movements. The method requires an empirical threshold for detecting the static vector, which is not robust for different environments and different test subjects.

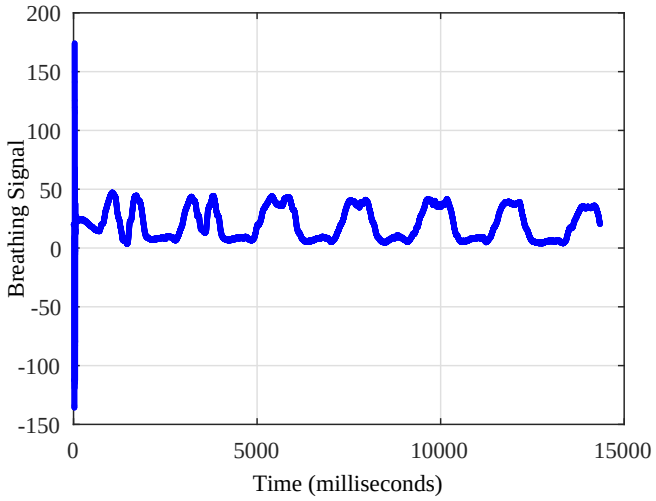


Figure 10.9: Respiration curve for phase data without removing static vector effect.

In Algorithm 12, we present an adaptive median filter method for removing the static vector, which has a low latency and is robust to different environments. The idea is to use a window to obtain the median for estimating the static vector. The only parameter is window size w, which is robust for different environments. For baseband signal component $I(n)$ or $Q(n)$, $n = 0, 1, ..., N-1$, we partition it into multiple non-overlapping sublists, each denoted by $W[1, 2, ..., w]$) with window size w, and a single sublist $R[1, 2, ..., r]$ with window size $r < w$, where r is the number of remaining elements of $I(n)$ or $Q(n)$ not included in the previous W sublists. The sublists $W[1, 2, ..., w]$ and sublist $R[1, 2, ..., r]$ are used to estimate the medians for the first $n_w - 1$ windows and the last window, respectively, where $n_w = \lfloor N/w \rfloor$. Finally, the output $O(n)$, $n = 0, 1, ..., N-1$, can be obtained as in Steps 14 and 21 by removing the static vector. The proposed method is simple and robust for realtime processing of received data in different environments with a low delay.

Fig. 10.7 and Fig. 10.8 illustrate how the adaptive median filter method removes the static vectors in the baseband signal components. We can see that the estimated static vector can represent well the average amplitude information of the baseband signal components. After the adaptive median filter, the components I and Q are roughly centered at zero; the improved SNR makes it easier for extracting the breathing signal they carry.

Algorithm 12: The Adaptive Median Filter Method

Input: One baseband signal component: $X(n)$, $n = 0, 1, ..., N-1$, and the window size w ;
Output: The baseband signal component with static vector removed: $O(n)$, $n = 0, 1, ..., N-1$;
//Initialization
n_w: number of windows ;
r: number of remaining elements of $X(n)$, which cannot form a full window of size w ;
$W[1, 2, ..., w]$: sublists with window size w ;
$R[1, 2, ..., r]$: sublist with the remaining r elements ;
//Find the median for each window
for $i = 0 : n_w$ **do**
    if $((i+1) * w) <= N$ **then**
        $W[1, 2, ..., w] \leftarrow X((w * i)$ to $((i+1) * w - 1))$;
        $M \leftarrow$ the median of $W[1, 2, ..., w]$;
        for $j = w * i : (i+1) * w$ **do**
            $O(j) = X(j) - M$;
        end
    end
    else if $r \neq 0$ **then**
        $R[1, 2, ..., r] \leftarrow X((w * i)$ to $(w * i + r - 1))$;
        $M \leftarrow$ the median of $R[1, 2, ..., r]$;
        for $j = w * i : (i+1) * w$ **do**
            $O(j) = R(j) - M$;
        end
    end
end
Return $O(n)$, $n = 0, 1, ..., N-1$;

Phase Extraction

After removing the static vector, we next extract the phase data in the I-Q plane, which only includes the dynamic breathing component. Let $O_I(t)$ and $O_Q(t)$ denote the outputs of Algorithm 12. The phase of the inaudible signal can be computed with (10.2), that is

$$\varphi(t) = \arctan\left(\frac{O_Q(t)}{O_I(t)}\right). \tag{10.3}$$

With (10.2) and (10.3), we find the phase value $\varphi(t)$ for the respiration signal reflected from the chest. Although the reflected respiration signal may still have multipath components, these

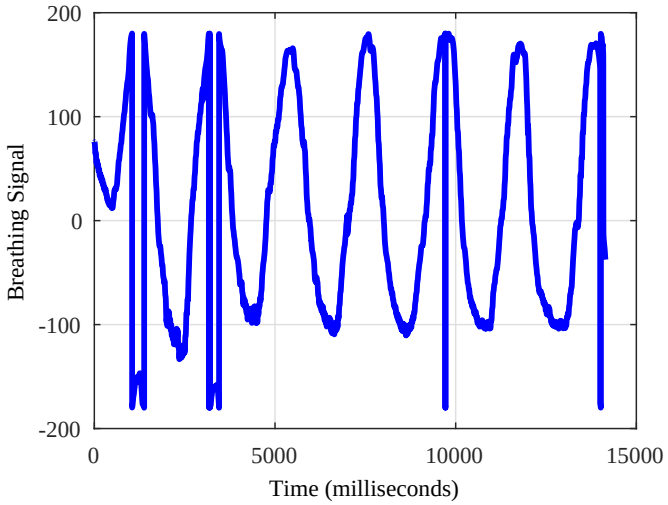


Figure 10.10: Respiration curve for phase data with removing static vector effect.

multipath signals have the same breathing frequency but with different phase shifts, each of which is a constant. Thus, the breathing rate will not be affected by the dynamic multipath effect. This is different from hand tracking, which requires only one path from the smartphone and the hand. Thus, our SonarBeat can estimate the breathing rate using a single subcarrier rather than multiple subcarriers.

Fig. 10.10 presents the respiration curve obtained from the phase data with removing static vectors. It is noticed that the magnitude of the breathing signal is large, which is periodic if we can remove the sudden phase changes. Thus, we need to implement a data calibration scheme for the demodulated phase data with better periodicity.

Data Calibration

-

We implement a phase unwrapping scheme for recovering the correct phase values, as well as a median filter for reducing the environment noise. To estimate breathing rates, we need to obtain the right breathing curve for phase data. Because the phase value will have a change of 2π for every wavelength distance, we implement a phase unwrapping scheme to process the demodulated phase data.

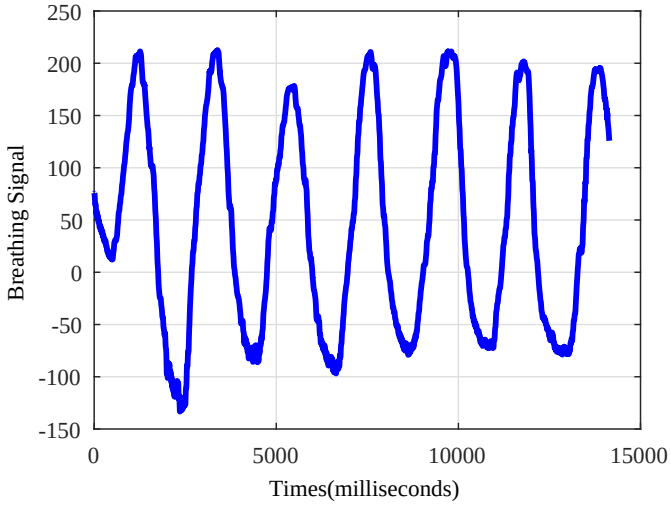


Figure 10.11: Respiration curve for phase data after unwrapping the phase data.

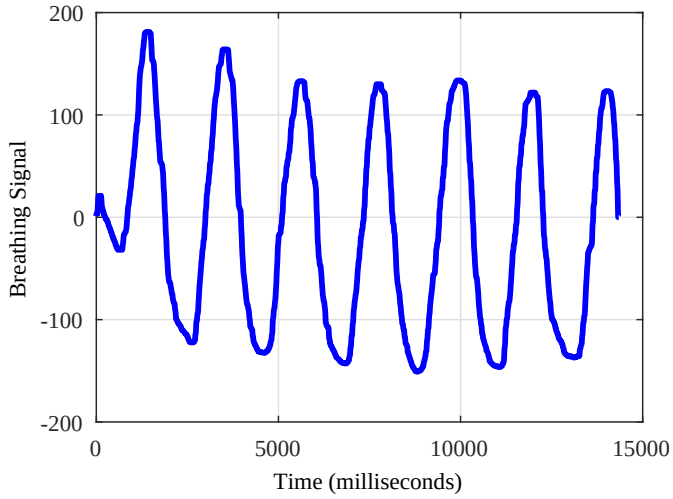


Figure 10.12: Respiration curve for unwrapped phase data after the median filter method.

Fig. 10.11 shows the respiration curve obtained from the phase data after phase unwrapping. It is a clear breathing signal, but still with smaller environment noises. Moreover, we adopt the median filter method to remove the environment noise, where the filter window size is set to 300. Fig. 10.12 presents the respiration curve for the unwrapped phase data after the median filter, which is next used for accurate breathing rate estimation.

10.4.5 Breathing Rate Estimation

SonarBeat operates in three stages for breathing monitoring, for better interactions with the user. In the first stage of 15 s, we cannot effectively estimate the breathing rate but just record the

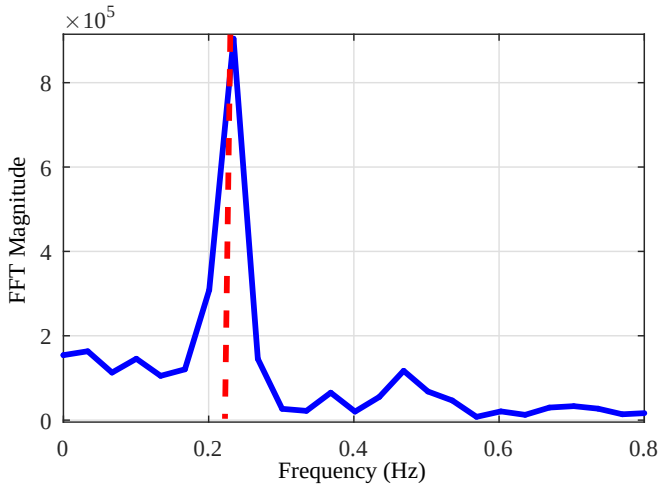


Figure 10.13: Respiration rate estimation based on FFT.

extracted respiration signal, because the phase data in a window can continually change when new data arrive. In the second stage of 15 s, we analyze the collected phase data in a 15-second sliding window to extract the breathing signal and plot it on the smartphone screen. In the final stage, after 30 s, we use all the collected phase data for breathing rate estimation with FFT, which can achieve a higher estimation accuracy (since more data is available now). In fact, the frequency resolution depends on the window size of FFT. If the window size becomes larger, the estimation accuracy will be higher, but a larger window size also leads to a lower time domain resolution. Thus, for online breathing rate estimation, we use the same window size as that of the STFT based method. It balances the tradeoff between the frequency domain resolution and the time domain resolution.

Fig. 10.13 illustrates the FFT based respiration rate estimation. We can see that the estimated frequency is 0.23 Hz, which is approximately the same as the true breathing rate measured by the NEULOG Respiration Monitor Belt Logger Sensor during the experiment.

10.5 Experimental Study

10.5.1 Experiment Configuration

We prototype the SonarBeat system on the Android platform in Java and the Android SDK, as an smartphone App. The first edition of SonarBeat is implemented with the minimum version of

Android 5.1.1 OS (API 21). So it works with all the more recent Android systems such as Android 6.0 and Android 7.0. The App is evaluated with Samsung Galaxy S6 and Samsung Galaxy S7 Edge smartphones. For respiration monitoring, we use one speaker and one microphone to transmit and receive the inaudible audio data, respectively, while the microphone and speaker are fixed at the bottom of the smartphone. Furthermore, we use the AudioTrack class to play inaudible sound and the AudioRecord class to record sound. The buffer of the recording thread is set to 1920 points with a sampling rate of 48 KHz. Therefore, we set the realtime signal processing unit to 1920 points, which is about 40 ms.

We conduct extensive experiments with SonarBeat with five persons over three months. The test scenarios include an office, a bedroom and a movie theater. The *office* is a 4.5×8.8 m^2 room. The room is crowed with tables and PCs, which form a complex propagation environment. In this office environment, we test SonarBeat under different parameters settings. The second environment is a *bedroom* of 3.9×6 m^2, where we test breathing monitoring for a single person. The third setup is a *movie theater* of a large 27×40 m^2 area, where many peoples are watching a movie, with strong audio interference from the movie and other people. The office and bedroom scenarios are implemented over longer periods with multiple times in different days as compared to the movie cinema that is tested in one hour for watching a movie. Moreover, the proposed system mainly focuses on estimating the breathing rate for a person, where we consider other persons have above 60 cm distance away from the smartphone. For comparison purpose, we use the NEULOG Respiration Monitor Belt Logger Sensor to record the ground truth of the breathing rate (see Fig. 10.17).

For breathing rate estimation each time, we use all the collected data for breathing rate estimation with FFT for 30 s. Moreover, we use CDF of breathing errors as the measurement metric, which can be employed to evaluate the total performance of the proposed system. Moreover, we also consider the mean estimation error as another evaluation metric for measuring the impact of various environmental factors and the impact of various system parameter.

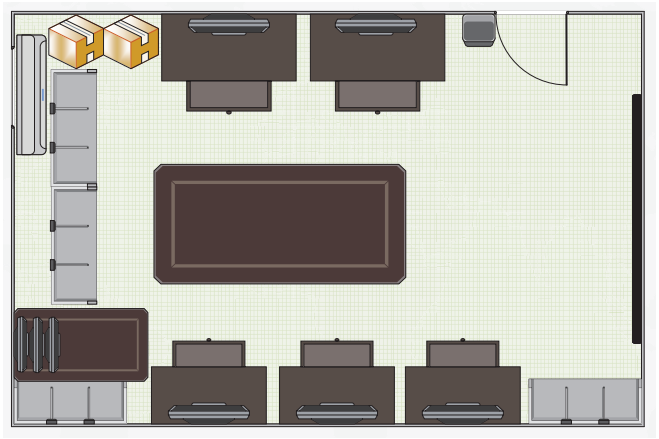


Figure 10.14: Experimental setup in the *office* scenario.

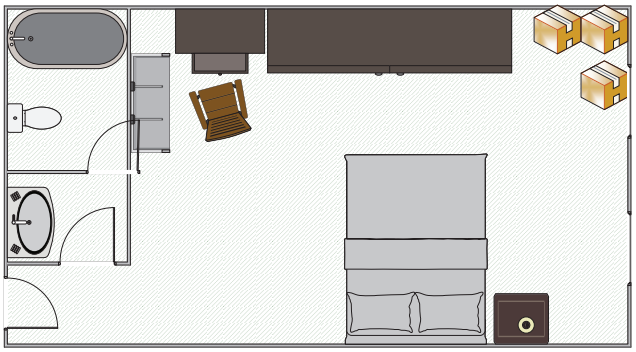


Figure 10.15: Experimental setup in the *bedroom* scenario.

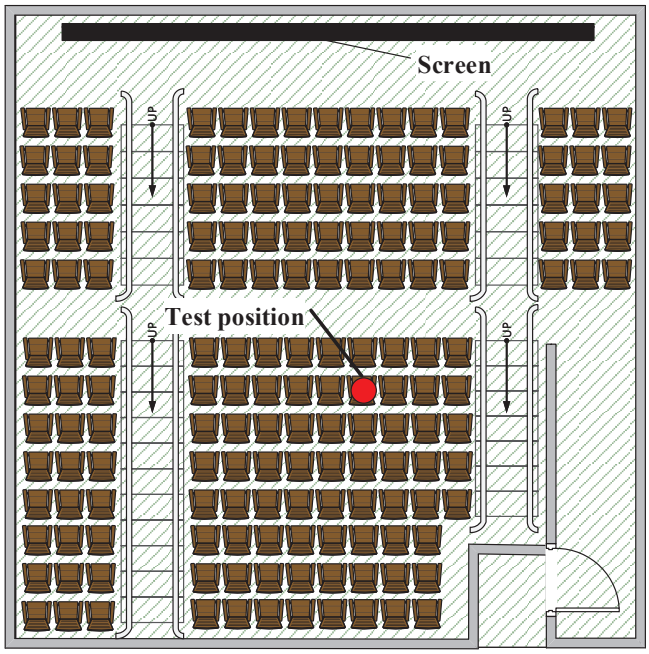


Figure 10.16: Experimental setup in the *movie theater* scenario.

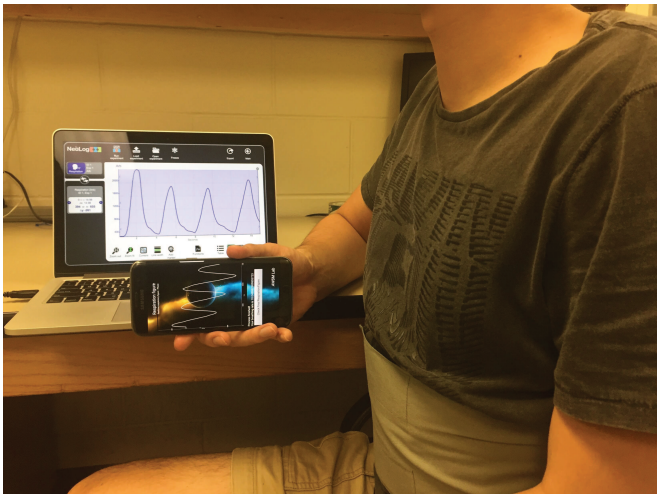


Figure 10.17: The office experiment, where the NEULOG Respiration Monitor Belt Logger Sensor records the ground truth (shown on the laptop screen).

10.5.2 Performance of Breathing Rate Estimation

Fig. 10.18 presents the CDFs of estimation errors in breathing rate estimation with SonarBeat. For comparison purpose, we also develop an LEVD based system [155], where the LEVD method is used for estimating the static vector and all other signal processing methods are the same as in SonarBeat. We find that SonarBeat and the LEVD based method achieve a median error of 0.2 bpm and 0.3 bpm, respectively. This illustrates that both systems can effectively estimate breathing rates. However, it is worth noting that for SonarBeat, 95% of the test results have an estimated error under 0.5 bpm, while only 60% of the test results with the LEVD based method have an estimated error under 0.5 bpm. Moreover, the maximum estimation error of SonarBeat and the LEVD based method are 2.4 bpm and 5 bpm, respectively. This is because the LEVD based method requires setting the empirical threshold based on the standard deviation of the baseband signal in a static environment. It is not robust in varying environments where the same threshold will not work. However, SonarBeat leverages the adaptive median filter method, and is thus more robust to changes in the environment. Thus SonarBeat can achieve a higher and more stable breathing rate estimation accuracy than LEVD.

Fig. 10.19 presents the mean estimation errors for the three different scenarios, which are 0.22 bpm, 0.11 bpm, and 0.33 bpm for the office, bedroom and movie theater scenarios, respectively.

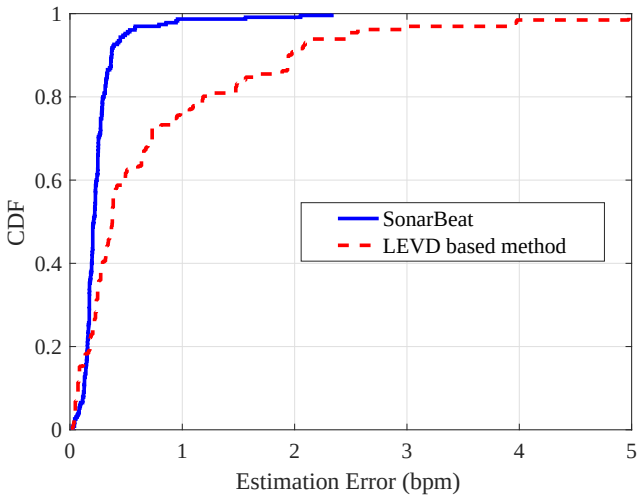


Figure 10.18: CDFs of estimation errors in breathing rate estimation.

We plot the 95% confidence intervals as error bars. The mean estimation error of the bedroom case is the minimum, because the bedroom is a better environment: with smaller noise and no sound interference from other persons. This shows that SonarBeat is suitable for breathing monitoring during sleeping, which helps to detect apnea or other sleeping problems. For breathing monitoring in the office, the performance is worse than the bedroom. This is because the propagation environment is more complex and there is interference from other people. Furthermore, higher noises from computers, air conditioner, and other equipment in the lab also influence the received inaudible signal. The movie theater test has the largest mean error and variance because of the more complex environment and stronger noises. In fact, breathing monitoring in the theater is still quite accurate given the extremely adverse environment. These experiments validate that SonarBeat is highly accurate and robust in different scenarios.

10.5.3 Impact of Various Environmental Factors

Fig. 10.20 shows the impact of different persons in the office scenario. In the experiment, we test five persons including three men and two women. Every volunteer wears the NEULOG Respiration unit to record the ground truth for breathing data. From Fig. 10.20, we can see that Persons 3 and 5 have a relatively lower mean error. This is because they work out quite often and have a stronger respiration, leading to stronger breathing signals. On the other hand, the other three

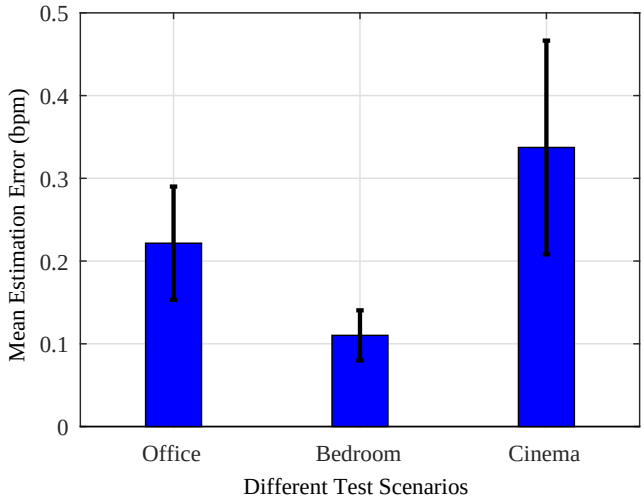


Figure 10.19: Mean estimation error for three different scenarios: office, bedroom, and movie theater.

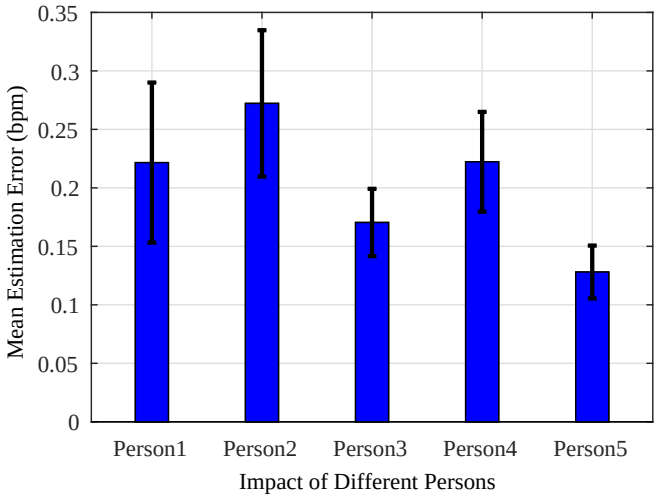


Figure 10.20: Breathing rate results for five different persons.

persons have weaker breathing magnitudes, but their estimation errors are still under 0.5 bpm, which is acceptable. Thus, we can see that SonarBeat is adaptive for different persons.

Fig. 10.21 shows the impact of different breathing rates in the office scenario, where the test subject controls his/her breathing at a slow, normal, and fast breathing rates, which are in the rangs from 6 bpm to 10 bpm, 13 bpm to 18 bpm, and above 30 bpm, respectively. It is noticed that with the increase of breathing rate, the mean estimation error will be increased. The reason is that with higher breathing rate, the stability of the breathing signal becomes weaker. In other words, with fast breathing, the chest movements are more irregular, thus leading to large variations in the captured

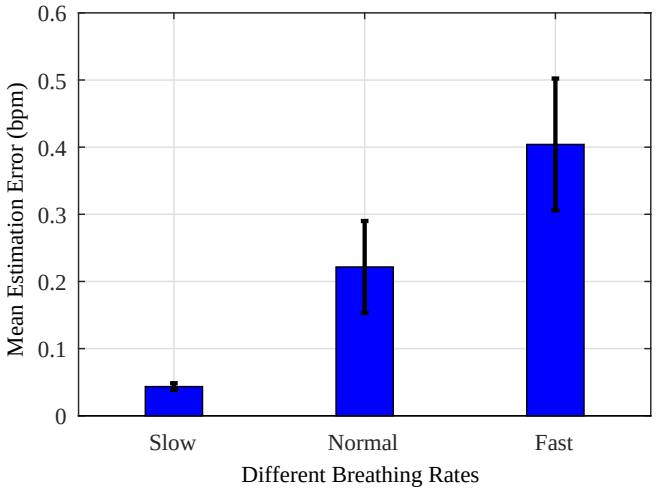


Figure 10.21: Impact of different breathing rates.

breathing curve. Moreover, we adopt the FFT based breathing estimation. When there are multiple breathing frequencies embedded in the captured breathing signal, FFT does not produced good frequency estimation. Nevertheless, SonarBeat can still effectively capture the different breathing rates with a mean error a little over 0.4 bpm in the fast breathing case.

Fig. 10.22 shows the impact of the distance between the chest and the smartphone. When the distance is increased, the accuracy of breathing estimation becomes lower. Particularly, we can see that for a distance of 55 cm, the mean estimation error becomes 1 bpm with a large variance. In this experiment, we find that the ultrasound wave in 18 KHz to 22 KHz experiences a large attenuation, and the microphone will receive a lower power from the chest reflection if the distance is increased. Moreover, breathing rate estimation with SonarBeat depends on I/Q demodulation. The magnitude of the I/Q components becomes weaker when the distance is increased, leading to higher errors. To improve the measurement distance, we leverage the parameter resonance of the low-pass filter to strengthen the amplitude of the inaudible signal near the cutoff frequency, thus improving the magnitudes of the I/Q components. In the experiment, we set the cutoff frequency to 40 Hz for the sampling rate of 48 KHz. We can see that under 50 cm, the proposed system can achieve very good accuracy.

Fig. 10.23 presents the breathing rate errors when the smartphone is held in hand or put on a desk. We find the error is low in both cases, which are 0.22 bpm when the smartphone is held in

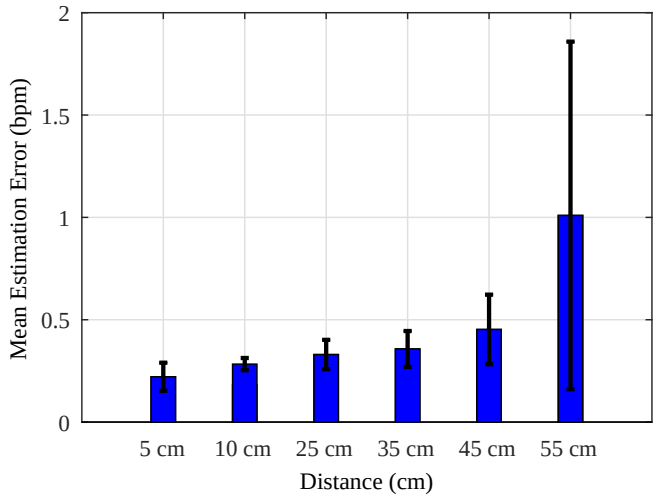


Figure 10.22: Impact of the distance between the test subject and the smartphone.

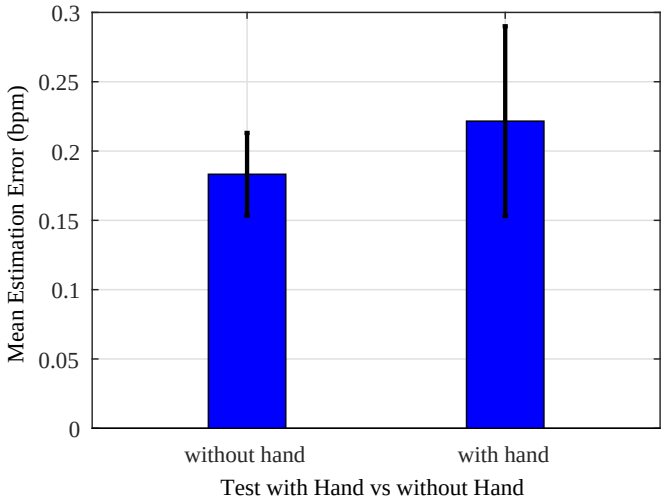


Figure 10.23: Breathing rate results when the smartphone is held in hand or put on a desk.

hand and 0.16 bpm when it is put on a desk. Moreover, the variance of breathing rate estimation errors with a hand held smartphone is larger. This is because the small hand movements will affect the reflected signal. Although the small hand movements do not influence the basic estimation accuracy, it still causes a larger variance of the estimation error. On the other hand, because we use the adaptive median filter to effectively remove the static vector, the mean breathing rate estimation results can be guaranteed for both cases.

Fig. 10.24 shows the impact of cloth thickness. In the experiment, the test subject wears clothes of different types and thickness. The distance between the user and the smartphone is kept between 10 cm to 15 cm. It is noticed that, with the increase of cloth thickness, the mean error

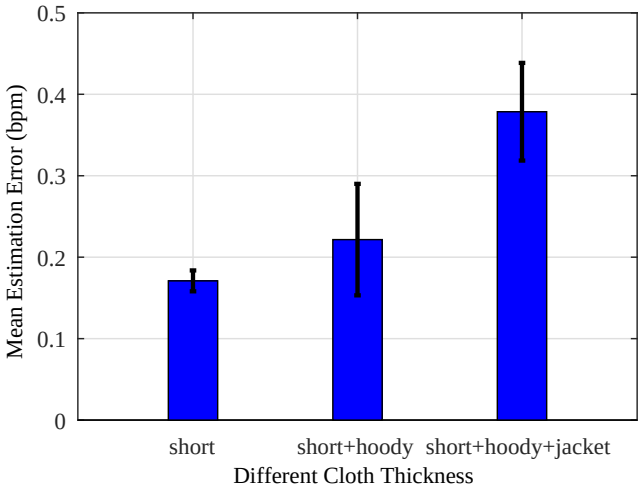


Figure 10.24: Impact of cloth thickness.

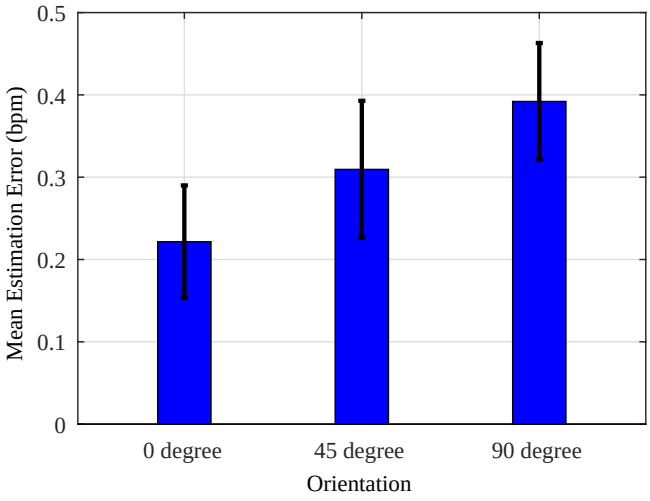


Figure 10.25: Impact of user orientation relative to the smartphone.

becomes larger. This is because the ultrasound wave at 18 KHz to 22 KHz experiences larger attenuation when the clothes gets thicker, which leads to smaller received breathing signal and a lower SNR. In fact, the maximum breathing estimation error is about 0.37 bpm in this experiment, which is still acceptable.

Fig. 10.25 shows the impact of chest orientation relative to the smartphone in the office scenario, where we consider three cases of 0°, 45° and 90°. It is noticed that at 0° direction with the front orientation relative to the smartphone, we can obtain the minimum mean estimation error, which is about 0.22 bpm. At the 90° direction, the maximum mean estimation error becomes 0.39 bpm. The received inaudible signal is the strongest when the person faces the smartphone speaker.

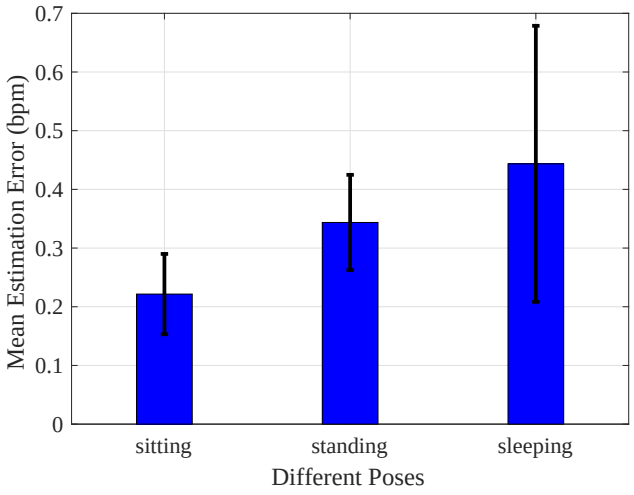


Figure 10.26: Impact of different poses.

Fig. 10.26 shows the impact of different poses including sitting, standing, and sleeping in the bedroom scenario. For sitting case, the smartphone is put on a desk and the distance between the smartphone and the person is under 15 cm. For standing case, the person holds the smartphone with the same distance. For sleeping case, the person wearing night-clothes is laying in the bed, who faces the smartphone on the desk with the same distance. We can see that for poses with the sitting and standing, the mean estimation errors are smaller than that with the sleeping. This is because the strength of the received signal in the sleeping scenario is smaller than the other two cases.

10.5.4 Impact of Various System Parameters

We evaluate the impact of various system parameters in this sections. Fig. 10.27 presents the estimation errors with a Samsung Galaxy S6 with Android 6.0 and a Samsung Galaxy S7 Edge working with the lasted Version Android 7.0. The speaker and microphone are on the bottom of both smartphones. We can see that Samsung Galaxy S7 Edge has a similar performance as Samsung Galaxy S6, with mean error of 0.21 bpm and 0.22 bpm, respectively. We find that Samsung Galaxy S7 Edge has stronger processing power, and thus it can obtain better realtime performance than Samsung Galaxy S6.

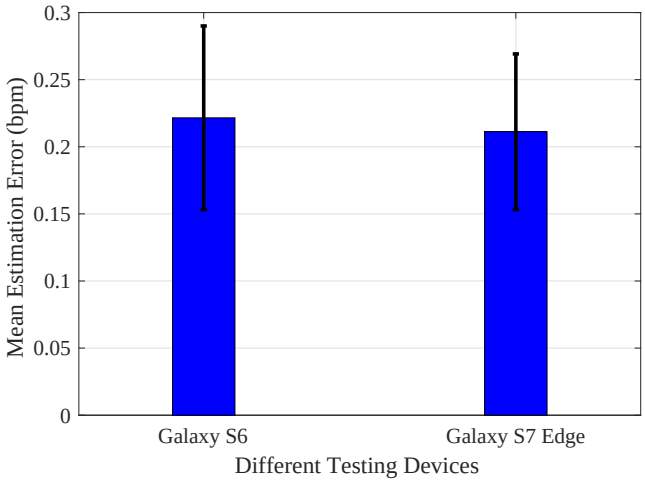


Figure 10.27: Estimation error results with two different smartphone platforms.

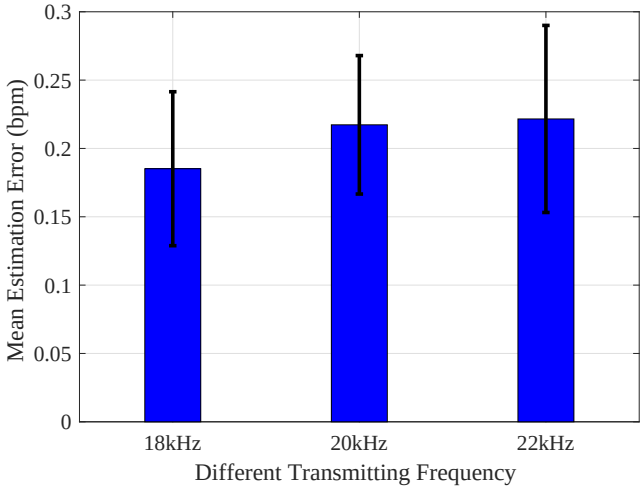


Figure 10.28: Impact of the frequency of the inaudible acoustic signal.

Fig. 10.28 shows the impact of different frequencies for the CW signal, including 18 KHz, 20 KHz, and 22 KHz. With the increase of frequency, the mean error also gets slightly larger. The maximum mean estimation error is 0.22 bpm, while the minimum mean estimation error is 0.17 bpm. This shows that SonarBeat is robust to different frequencies. On the other hand, we also know that for ultrasound signals propagating in the same transmission medium, the power attenuate becomes larger for higher signal frequencies, resulting in smaller SNR for the received breathing signal. Thus, the higher the frequency, the larger the mean breathing rate estimation error.

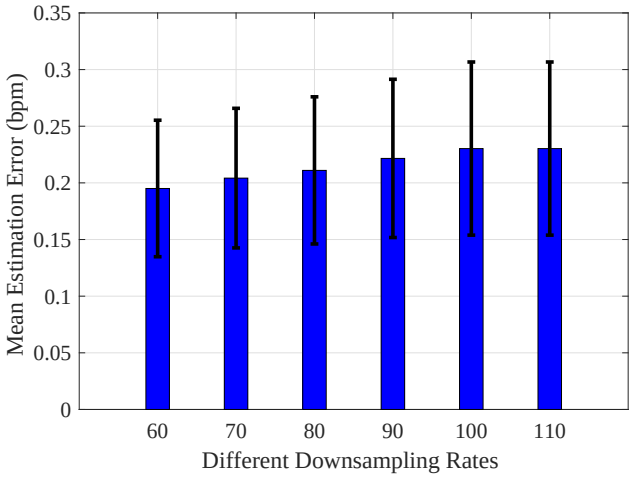


Figure 10.29: Impact of different downsampling rates.

Fig. 10.29 shows the impact of different down-sampling rates. When the down-sampling rate is increased from 60 to 100, the mean breathing rate estimation error will be increased from 0.19 bpm to 0.23 bpm, a small increase. Thus, we choose a down-sampling rate of 100 for SonarBeat. Because SonarBeat uses a sampling frequency of 48 KHz, we can reduce it to 480 Hz by down-sampling rate of 100. As shown before, the down-sampling operation does not affect the breathing rate embedded in the phase modulated signal. Thus, down-sampling the I and Q components with a rate of 100 can not only reduce the computational complexity for realtime breathing monitoring with the smartphones, but also achieve a high accuracy.

Fig. 10.30 presents the results with different window sizes for the adaptive median filter. Recall that the window size is used in the static vector effect mitigation stage. Breathing rate estimation with SonarBeat mainly depends on reduction of the static vector effect. The only parameter of the proposed adaptive median filter approach is the window size, which should be robust for different tests. From Fig. 10.30, we can see that, when the window size is increased from 7400 to 7700, the mean breathing rate estimation error is only increased slightly from 0.21 bpm to 0.26 bpm. Moreover, the larger window sizes from 7400 to 7700 are almost half of the signal points, which is effective for removing the static vector. Thus, we choose a window size of 7500 for SonarBeat, which can achieve the best breathing estimation accuracy.

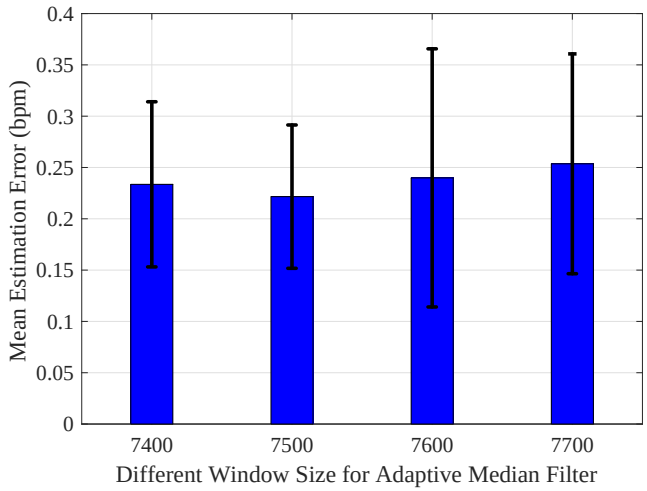


Figure 10.30: Impact of different window sizes of the adaptive median filter.

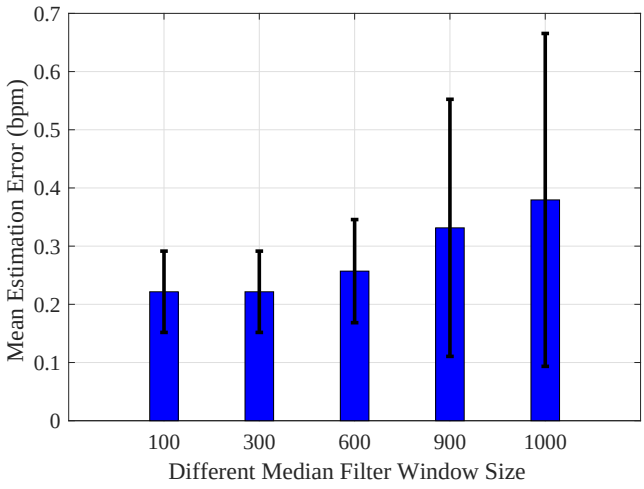


Figure 10.31: Impact of different sizes of the median filter window.

Fig. 10.31 shows the impact of different median filter window sizes on the mean error, which is used in the data calibration stage of SonarBeat. We find that when the median filter window size is set to 100 or 300, similar mean error of 0.21 bpm are achieved. On the other hand, when the median filter window size is increased from 600 to 1000, the mean estimation error is increased from 0.25 bpm to 0.38 bpm. This is because a smaller window size can process the local breathing curve, to effectively remove the environment noise. On the other hand, a larger median filter window size omits the local breathing noise, thus causing a higher error. Based on this experiment, we set the median filter window size to 300, which can not only achieve a higher breathing estimation accuracy, but also lead to the better breathing curves for realtime breathing monitoring.

10.6 Conclusion

In this chapter, we presented SonarBeat, a system that exploits phase based sonar to monitor breathing rates with smartphones. We first provided a rigorous sonar phase analysis and proved the sonar phase based method can obtain breathing signals. Also, we discussed the technical challenges for breathing estimation based on active sonar signal, including removing the static vector effect, adaption for body movements and environment noise, and on-line breathing monitoring with lower delay. We then described the SonarBeat design in detail, including signal generation, data extraction, received signal preprocessing, and breathing rate estimation. Finally, we implemented SonarBeat with two different smartphones, and conducted an extensive experimental study with three setups. The experimental results validated that SonarBeat can achieve superior performance on breathing rate estimation for different factors and parameters.

Chapter 11

Conclusions and Future Work

11.1 Conclusion

In this dissertation work, we investigate the problem of RF sensing for IoT using CSI and machine learning techniques. In particular, our work mainly focuses on indoor localization and vital sign monitoring for RF sensing. For indoor localization, we consider different CSI features as fingerprints for indoor localization using different deep learning methods. On the other hand, for vital sign monitoring, we consider breathing and heart rates monitoring for single person and multiple persons using CSI phase differences. Moreover, we study the resilient breathing beats monitoring for bad locations. In addition, we also exploit the acoustic signal for breathing rate estimation with smartphones.

In chapter 2, we proposed DeepFi, a deep learning based indoor localization using CSI amplitude information. In DeepFi, CSI information for all the subcarriers and all the antennas are collected through the device driver and analyzed with a deep network with four hidden layers. Based on the three hypotheses on CSI, we considered the weights in the deep network to represent fingerprints, and incorporated a greedy learning algorithm for weight training to reduce complexity. Moreover, a probabilistic data fusion method based on the RBF was exploited for online location estimation. The proposed DeepFi scheme was validated in two representative indoor environments, and was found to outperform several existing RSS and CSI based methods in both experiments.

In chapter 3, we proposed PhaseFi, a deep learning based indoor localization using CSI phase information. In PhaseFi, the phase information was first extracted and calibrated from the three

antennas of the Intel WiFi Link 5300 NIC by accessing the modified device driver. In the offline stage, we considered a deep network with three hidden layers to train the calibrated phase data, and used weights to represent fingerprints. To reduce complexity, a greedy learning algorithm was incorporated to train the weights layer-by-layer, where a sub-network between two consecutive layers formed an RBM approximately and solved by a CD-1 algorithm. In the online stage, a Bayes method based on RBF was used for location estimation. The proposed PhaseFi scheme was validated in two representative indoor environments, and was shown to outperform three benchmark schemes based on either CSI or RSS in both scenarios.

In chapter 4, We presented BiLoc, a bi-modal deep learning system for fingerprinting-based indoor localization with 5GHz commodity WiFi NICs. In BiLoc, we first extracted and calibrated CSI data to obtain bi-modal CSI data, including average amplitudes and estimated AOAs, which were used in both the offline and online stages. In the training phase, we leveraged a deep autoencoder network to train the bi-modal data, and the weights were used to represent the bi-modal fingerprints. In the test phase, a Bayesian approach based probability model was employed for estimating position with bi-model test data. We evaluated the performance of BiLoc with extensive experiments under three representative indoor environments. The experimental results validated the superior performance of BiLoc over several benchmark schemes.

In chapter 5, we proposed CiFi, a DCNN based fingerprinting system for indoor localization with 5 GHz Wi-Fi. We theoretically and experimentally verified the feasibility of using AOA values for indoor localization. We then presented the CiFi system, which first formed AOA images to train the DCNN, and then used newly received AOA images to estimate the location of the mobile device. Through extensive experiments, we demonstrated the superior performance of the proposed CiFi system under two representative indoor environments.

In chapter 6, we presented ResLoc, a deep residual sharing learning based system for indoor localization with two channels CSI tensor data. We discussed how to build CSI tensor data for indoor localization. Then, we designed the ResLoc system, which leverages two channels CSI tensor data to train the deep network by using the proposed deep residual sharing learning. For online

test, we used newly received CSI tensor data to compute the location of the mobile device based on the probabilistic method. Finally, the experimental results showed the superior performance of the proposed ResLoc system.

In chapter 7, we proposed PhaseBeat, CSI phase difference data to monitor breathing and heart beats with commodity WiFi device. PhaseBeat system leveraged CSI phase difference data to extract the periodic signal from the change in the chest of a person such as inhaling and exhaling. Then, We implemented data preprocessing including environment detection, data calibration, subcarrier selection and discrete wavelet transform. Moreover, we employed the peak detection approach for breathing rate estimation and FFT based method for heart rate estimation. We conducted with the experiments with three setups such as the laboratory, through-wall scenario and the long corridor. The results showed that the PhaseBeat system can obtain better performance than the amplitude based method.

In chapter 8, we presented TensorBeat, tensor decomposition for estimating multiple persons breathing beats with commodity WiFi. The proposed TensorBeat system employed CSI phase difference data to obtain the periodic signals from the movements of multiple breathing chests by leveraging tensor decomposition. We implemented several signal processing methods including data preprocessing, CP decomposition, signal matching algorithm, and peak detection in TensorBeat. We validate the performance of TensorBeat with extensive experiments under three indoor environments. Our analysis and experimental study demonstrated that the proposed TensorBeat system can achieve satisfactory performance for multiple persons breathing estimation.

In chapter 9, we proposed ResBeat, resilient breathing beats monitoring by using online CSI amplitude and phase difference data in the commodity WiFi devices. We developed the ResBeat system including data preprocessing, adaptive signal selection and breathing signal monitoring. For data preprocessing, we implemented the data extraction and calibration to obtain the breathing component and environment component. For the adaptive signal selection, we proposed the signal selection algorithm based on signal energy detection and movement detection for selecting the most sensitive signal group. Then, we leveraged peak detection to estimate breathing rates.

We implemented with the experiments with three different scenarios. The results validated the effectiveness of the proposed ResBeat system.

In chapter 10, we presented SonarBeat, a system that exploits phase based sonar to monitor breathing rates with smartphones. We first provided a rigorous sonar phase analysis and proved the sonar phase based method can obtain breathing signals. Also, we discussed the technical challenges for breathing estimation based on active sonar signal, including removing the static vector effect, adaption for body movements and environment noise, and on-line breathing monitoring with lower delay. We then described the SonarBeat design in detail, including signal generation, data extraction, received signal preprocessing, and breathing rate estimation. Finally, we implemented SonarBeat with two different smartphones, and conducted an extensive experimental study with three setups. The experimental results validated that SonarBeat can achieve superior performance on breathing rate estimation for different factors and parameters.

11.2 Future work

In the future, we will focus on four research directions as the following [2].

11.2.1 Fusion of Multiple Data Sources

Bimodal or even multimodal data can be exploited for better RF sensing performance. For example, WiFi and light sensors are both available on smartphones and can be integrated for indoor localization, where WiFi and light signals are complementary to each other. For example, the RSS of WiFi signals do not perform well at close locations, while the light intensities at such locations could be quite different. Using bimodal data of WiFi RSS and light intensity can increase data diversity, which results in higher location accuracy.

The key to exploit multimodal data is how to effectively fuse various data. The deep learning framework can be trained with different data sizes with the same deep network structure. One solution to train multimodal data is to adopt a multi-channel deep network architecture, one for each data source [4]. Signals from different channels can be fused at intermediate layers and/or

at the output layer [4]. Other deep networks such as deep reinforcement learning and generative adversarial networks can also be incorporated for fusion of multiple data sources to improve sensing accuracy or reduce cost with small training data. For effective data fusion, the input data from different sources should be normalized, and data samples from different sources should be aligned.

11.2.2 Exploring New Spectrum for RF Sensing

With the fast growth of 5G technologies, signals from new spectra, such as the low-bands (below 1 GHz), mid-bands (1 GHz to 6 GHz), and high-bands (above 24 GHz, i.e., mmWave), should be leveraged for RF sensing in the IoT. Specifically, the low-bands spectrum can be utilized for massive IoT and mobile broadband; the mid-band spectrum provides wider bandwidths and can be employed for mission-critical applications and enhanced Mobile Broadband (eMBB); the high-band spectrum provides a huge amount of bandwidth and is usually used for high throughput communications. In the literature, mmWave massive MIMO has been applied for fingerprinting with a deep learning approach. Moreover, narrow band (NB) IoT technologies, such as LoRaWAN and SIGFOX with low power and long range, can also be leveraged for detecting multiple objects.

It is expected that channel estimation based on deep learning could become an interesting research topic, where deep learning can be used to learn CSI information. Then, some key parameters, such as amplitude, AOA, and TOA from the multipaths can be predicted from training data with deep learning techniques for RF sensing. By applying deep learning techniques to new signals from 5G spectra, RF sensing could be greatly enhanced with a stronger data representation ability, not only for personal IoT applications such as indoor localization, activity recognition, and healthcare, but also for other IoT applications such as smart city, manufacturing, supply chain management, precision agriculture, and animal tracking.

11.2.3 From Cloud to Edge and Mobile Devices

Although deep learning models have achieved superior performance for recognition tasks, deep learning models are usually computation intensive and require large storage space. For image and

speech recognition applications, usually the programs are executed at a server or in the cloud. For RF sensing applications, it would be more appealing to execute the deep learning models at the edge or mobile devices to avoid large delay for better user experience. It is thus important to move deep learning models from the cloud to the edge or mobile devices, for reduced cost and delay and enhanced privacy [174].

The challenge is how to execute deep learning models at the relatively more resource constrained edge or mobile devices; How to reduce deep model parameters and accelerate computations. To this end, compressed deep network can be utilized for RF sensing on edge devices. Methods for compressing weights by utilizing the sparse features of data can be developed, to reduce storage demand of weights. Moreover, parallel and distributed deep learning are suitable for execution on edge and mobile devices to reduce the training time, to jointly learn the parameters for RF recognition tasks. Finally, GPU and FPGA-accelerated hardware can be used at the edge or mobile devices to greatly accelerate the computation of deep learning models for RF sensing applications.

11.2.4 Security and Privacy Preserving

Deep Learning can learn the features of RF signs, which is useful user information for security and privacy protection. By leveraging features of multi-path RF signals, deep learning can be used to classify eavesdropping, DoS attack, and bad data injection. The proposed deep learning based RF sensing frame work can be used for intrusion detection in smart homes. Specifically, deep LSTM networks can be used for realtime intrusion detection with commodity wireless devices. Moreover, RF sensing can be incorporated for user authentication with different RF signals such as WiFi, RFID, acoustics, and UWB, where implicit authentication can be used.

Deep learning security has become a hot research topic recently. The main challenge is how to recognize adversarial data and clean data; deep learning could perform poorly with adversarial data, which can be created by introducing small noises to clean data. In fact, an attacker can easily inject noise or jamming signals to RF sensing signals. Such adversarial data should be recognized

in the beginning stage for guaranteeing the recognition performance of deep learning. Another challenge is how to preserve user privacy in deep learning based RF sensing applications. While RF signals mostly propagate in all directions, it is important to prevent an illegitimate user from detecting a user's location or monitoring a patient's vital signs.

References

[1] G. Muhammad, S. M. M. Rahman, A. Alelaiwi, and A. Alamri, “Smart health solution integrating IoT and cloud: A case study of voice pathology monitoring,” *IEEE Commun.*, vol. 55, no. 1, pp. 69–73, 2017.

[2] X. Wang, X. Wang, and S. Mao, “RF sensing for Internet of Things: A general deep learning framework,” *IEEE Communications*, Aug. 2018.

[3] X. Wang, L. Gao, S. Mao, and S. Pandey, “CSI-based fingerprinting for indoor localization: A deep learning approach,” *IEEE Trans Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[4] X. Wang, X. Wang, and S. Mao, “Resloc: Deep residual sharing learning for indoor localization with csi tensors,” in *Proc. IEEE PIMRC 2017*, Montreal, Canada, Oct. 2017.

[5] M. Gong, B. Hart, and S. Mao, “Advanced wireless LAN technologies: IEEE 802.11ac and beyond,” *ACM Mobile Comput. Commun. Rev. (MC2R)*, vol. 18, no. 4, pp. 48–52, Oct. 2014.

[6] D. M. Dobkin, *The RF in RFID: UHF RFID in Practice*, 2nd ed. Boston, MA: Newnes, 2012.

[7] I. Oppermann, M. Hämäläinen, and J. Iinatti, *UWB: Theory and applications*. Hoboken, NJ: John Wiley & Sons, 2005.

[8] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[12] H. Liu, H. Darabi, P. Banerjee, and L. Jing, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[13] X. Wang, S. Mao, S. Pandey, and P. Agrawal, "CA2T: Cooperative antenna arrays technique for pinpoint indoor localization," in *Proc. MobiSPC 2014*, Niagara Falls, Canada, Aug. 2014, pp. 392–399.

[14] Y. Wang, S. Mao, and T. Rappaport, "On directional neighbor discovery in mmwave networks," in *Proc. IEEE ICDCS 2017*, Atlanta, GA, June 2017, pp. 1704–1713.

[15] Z. Jiang and S. Mao, "Opportunistic spectrum sharing in lte-unlicensed with lyapunov optimization based auction," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5217–5228, June 2017.

[16] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000, pp. 775–784.

[17] S. Dayekh, "Cooperative localization in mines using fingerprinting and neural networks," in *Proc. IEEE WCNC'10*, Sydney, Australia, Apr. 2010, pp. 1–6.

[18] Z. Wu, C. Li, J. Ng, and K. Leung, "Location estimation via support vector regression," *IEEE Trans. Mobile Comput.*, vol. 6, no. 3, pp. 311–321, Mar. 2007.

[19] M. Youssef and A. Agrawala, “The Horus WLAN location determination system,” in *Proc. ACM MobiSys’05*, Seattle, WA, June 2005, pp. 205–218.

[20] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. Ni, “CSI-based indoor localization,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, July 2013.

[21] D. Halperin., W. J. Hu., A. Sheth., and D. Wetherall., “Predictable 802.11 packet delivery from wireless channel measurements,” in *Proc. ACM SIGCOMM’10*, New Delhi, India, Sept. 2010, pp. 159–170.

[22] J. Xiao, K. Wu., Y. Yi, and L. Ni, “FIFS: Fine-grained indoor fingerprinting system,” in *Proc. IEEE ICCCN’ 12*, Munich, Germany, Aug. 2012, pp. 1–7.

[23] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, “You are facing the Mona Lisa: Spot localization using PHY layer information,” in *Proc. ACM MobiSys’12*, Low Wood Bay, Lake District, United Kingdom, June 2012, pp. 183–196.

[24] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Neural Information and Processing Systems 2012*, Lake Tahoe, NV, Dec. 2012, pp. 1106–1114.

[25] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[26] Y. Gu, A. Lo, and I. Niemegeers, “A survey of indoor positioning systems for wireless personal networks,” *IEEE communications surveys and tutorials*, vol. 11, no. 1, pp. 13–32, Jan. 2009.

[27] R. Want, A. Hopper, V. Falco, and J. Gibbons, “The active badge location system,” *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, Jan. 1992.

[28] A. Ward, A. Jones, and A. Hopper, “A new location technique for the active office,” *IEEE Personal Communications*, vol. 4, no. 5, pp. 42–47, Oct. 1997.

[29] N. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket location-support system,” in *Proc. ACM Mobicom’00*, Boston, MA, Aug. 2000, pp. 32–43.

[30] M. M. Atia, A. Noureldin, IEEE, and M. J. Korenberg, “Dynamic online-calibrated radio maps for indoor positioning in wireless local area networks,” *IEEE Trans. Mobile Comput.*, vol. 12, no. 9, pp. 1774–1787, Sept. 2013.

[31] W. Au, C. Feng, S. Valaee, S. Reyes, S. Sorour, S. N. Markowitz, D. Gold, K. Gordon, and M. Eizenman, “Indoor tracking and navigation using received signal strength and compressive sensing on a mobile device,” *IEEE Trans. Mobile Comput.*, vol. 12, no. 10, pp. 2050–2062, Oct. 2013.

[32] S. Yoon, K. Lee, and I. Rhee, “FM-based indoor localization via automatic ngerprint DB construction and matching,” in *Proc. ACM MobiSys’13*, Taipei, Taiwan, June 2013, pp. 207–220.

[33] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, “LANDMARC: indoor location sensing using active RFID,” *Wireless networks*, vol. 10, no. 6, pp. 701–710, 2004.

[34] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, “Indoor localization without infrastructure using the acoustic background spectrum,” in *Proc. ACM MobiSys’11*, Washington, DC, June 2011, pp. 155–168.

[35] A. Varshavsky, E. Lara, J. Hightower, A. LaMarca, and V. Otsasonl, “GSM indoor localization,” *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 698–720, Dec. 2007.

[36] M. Azizyan, I. Constandache, and R. R. Choudhury, “Surroundsense: mobile phone localization via ambience ngerprinting,” in *Proc. ACM Mobicom’09*, Beijing, China, Sept. 2009, pp. 261–272.

[37] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proc. ACM MobiSys'11*, Washington, DC, Jun. 2011, pp. 141–154.

[38] C. Chen, Y. Chen, Y. Han, H.-Q. Lai, and K. R. Liu, "Achieving centimeter-accuracy indoor localization on wifi platforms: A frequency hopping approach," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 111–121, 2017.

[39] C. Chen, Y. Chen, Y. Han, H.-Q. Lai, F. Zhang, and K. R. Liu, "Achieving centimeter-accuracy indoor localization on wifi platforms: A multi-antenna approach," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 122–134, 2017.

[40] X. Zhang, C. Wu, and Y. Liu, "Robust trajectory estimation for crowdsourcing-based mobile applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1876–1885, July 2014.

[41] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee:Zero-effort crowdsourcing for indoor localization," in *Proc. ACM Mobicom'12*, Istanbul, Turkey, Aug. 2012, pp. 293–304.

[42] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proc. ACM Mobicom'12*, Istanbul, Turkey, Aug. 2012, pp. 269–280.

[43] M. Alzantot and M. Youssef, "Crowdinside: Automatic construction of indoor floorplans," in *Proc. ACM SIGSPATIAL GIS'12*, Redondo Beach, CA, Nov. 2012, pp. 99–108.

[44] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: Indoor pathway mapping made easy," in *Proc. USENIX NSDI'13*, Seattle, WA, Apr. 2013, pp. 269–280.

[45] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *Proc. ACM Mobicom'14*, Maui, Hawaii, Sept. 2014, pp. 249–260.

[46] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-navi: Self-deployable indoor navigation system," in *Proc. ACM Mobicom'14*, Maui, Hawaii, Sept. 2014, pp. 471–482.

[47] H. Lim, L. C. Kung, J. C. Hou, and H. Luo, "Zero-conguration indoor localization over IEEE 802.11 wireless infrastructure," *Wireless Networks*, vol. 16, no. 2, pp. 405–420, Feb. 2010.

[48] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. ACM Mobicom'10*, Chicago, Illinois, Sept. 2010, pp. 173–184.

[49] K. Wu, J. Xiao, M. G. Y. Yi, and L. M. Ni, "Fila: Fine-grained indoor localization," in *Proc. ACM Infocom'12*, Orlando, Florida, Mar. 2012, pp. 2210–2218.

[50] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of WiFi based localization for smartphones," in *Proc. ACM Mobicom'12*, Istanbul, Turkey, Aug. 2012, pp. 305–316.

[51] R. Nandakumar, K. K. Chintalapud, and V. N. Padmanabhan, "Centaur: Locating devices in an ofce environment," in *Proc. ACM Mobicom'12*, Istanbul, Turkey, Aug. 2012, pp. 281–292.

[52] K. Liu, X. Liu, and X. Li, "Guoguo: Enabling fine-grained indoor localization via smartphone," in *Proc. ACM MobiSys'13*, Taipei, Taiwan, Jun. 2013, pp. 32–43.

[53] S. Sen, K. K. J. Lee, and P. Congdon, "Avoiding multipath to revive inbuilding WiFi localization," in *Proc. ACM MobiSys'13*, Taipei, Taiwan, Jun. 2013, pp. 249–262.

[54] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Proc. ACM NSDI'13*, Lombard, IL, Apr. 2013, pp. 71–84.

[55] S. Kumar, E. Hamed, D. Katabi, and L. Li, "LTE radio analytics made easy and accessible," in *Proc. ACM SIGCOMM'14*, Chicago, Illinois, Aug. 2014, pp. 211–222.

[56] S. Kumar, S. Gil, D. Katabi, and D. Rus, “Accurate indoor localization with zero startup cost,” in *Proc. ACM Mobicom’14*, Maui, Hawaii, Sept. 2014, pp. 483–494.

[57] J. Wang and D. Katabi, “Dude, where’s my card?: RFID positioning that works with multipath and non-line of sight,” in *Proc. ACM SIGCOMM’13*, Hong Kong, China, Aug. 2013, pp. 51–62.

[58] F. Fadib and D. Katabi, “Seeing through walls using WiFi!” in *Proc. ACM NSDI’13*, Lombard, IL, Apr. 2013, pp. 75–86.

[59] M. Brunato and R. Battiti, “Statistical learning theory for location fingerprinting in wireless LANs,” *Elsevier Computer Networks*, vol. 47, no. 6, pp. 825–845, Apr. 2005.

[60] X. Wang, “Deployment of high altitude platforms in heterogeneous wireless sensor network via mrf-map and potential games,” in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*. IEEE, 2013, pp. 1446–1451.

[61] K. Xiao, S. Mao, and J. Tugnait, “MAQ: A multiple model predictive congestion control scheme for cognitive radio networks,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2614–2626, Apr. 2017.

[62] ——, “Congestion control for infrastructure-based CRNs: A multiple model predictive control approach,” in *Proc. IEEE GLOBECOM 2016*, Washington DC, Dec. 2016, pp. 1–6.

[63] ——, “QoE-driven resource allocation for DASH over OFDMA networks,” in *Proc. IEEE GLOBECOM 2016*, Washington, DC, Dec. 2016, pp. 1–6.

[64] X. Wang, L. Gao, S. Mao, and S. Pandey, “DeepFi: Deep learning for indoor fingerprinting using channel state information,” in *Proc. WCNC’15*, New Orleans, LA, Mar. 2015, pp. 1666–1671.

[65] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, “CSI-MIMO indoor WiFi fingerprinting system,” in *Proc. IEEE LCN 2014*, Edmonton, Canada, Sep. 2014, pp. 202–209.

[66] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Liu, and M. Liu, "PhaseU: Real-time LOS identification with WiFi," in *Proc. IEEE INFOCOM'15*, Hong Kong, China, Apr. 2015, pp. 2038–2046.

[67] K. Qian, C. Wu, Z. Yang, Y. Liu, and Z. Zhou, "PADS: Passive detection of moving targets with dynamic speed using PHY layer information," in *Proc. IEEE ICPADS'14*, Hsinchu, Taiwan, Dec. 2014, pp. 1–8.

[68] D. Zhang, S. Zhao, L. T. Yang, M. Chen, Y. Wang, and H. Liu, "NextMe: Localization using cellular traces in internet of things," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 302–312, April 2015.

[69] K. Derr and M. Manic, "Wireless sensor networks node localization for various industry problems," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 752–762, Jun. 2015.

[70] A. Abu-Mahfouz and G. P. Hancke, "Distance bounding: a practical security solution for real-time location systems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 16–27, Feb. 2013.

[71] S. Ivanov and E. Nett, "Localization-based radio model calibration for fault-tolerant wireless mesh networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 246–253, Feb. 2013.

[72] J. Pak, C. Ahn, Y. Shmaliy, and M. Lim, "Improving reliability of particle filter-based localization in wireless sensor networks via hybrid particle/FIR filtering," *IEEE Trans. Ind. Informat.*, vol. 11, no. 5, pp. 1089–1098, Oct. 2015.

[73] B. Wu and C. Jen, "Particle filter based radio localization for mobile robots in the environments with low-density wlan aps," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6860–6870, Sep. 2014.

[74] S. Lee, B. Kim, H. Kim, R. Ha, and H. Cha, "Inertial sensor-based indoor pedestrian localization with minimum 802.15.4a configuration," *IEEE Trans. Ind. Informat.*, vol. 7, no. 3, pp. 455–466, Aug. 2011.

[75] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: Device-free location-oriented activity identification using fine-grained wifi signatures," in *Proc. ACM Mobicom'14*, Maui, HI, Sept. 2014, pp. 617–628.

[76] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson, "Phaser: Enabling phased array signal processing on commodity WiFi access points," in *Proc. ACM Mobicom'14*, Maui, HI, Sept. 2014, pp. 153–164.

[77] M. Akbar, D. Taylor, and G. Durgin, "Amplitude and phase difference estimation bounds for multisensor based tracking of rfid tags," in *Proc. IEEE RFID'15*, San Diego, CA, Apr. 2015, pp. 105–112.

[78] K. Kleisouris, Y. Chen, J. Yang, and R. P. Martin, "The impact of using multiple antennas on wireless localization," in *Proc. IEEE SECON'08*, San Francisco, CA, June 2008, pp. 55–63.

[79] X. Wang, L. Gao, and S. Mao, "PhaseFi: Phase fingerprinting for indoor localization with a deep learning approach," in *Proc. IEEE GLOBECOM'15*, San Diego, CA, Dec. 2015, pp. 1–6.

[80] M. Speth, S. Fechtel, G. Fock, and H. Meyr, "Optimum receiver design for wireless broadband systems using OFDM–Part I," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1668–1677, Nov. 1999.

[81] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity WiFi," in *Proc. ACM Mobicom'15*, Paris, France, Sept. 2015, pp. 53–64.

[82] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. 34, no. 3, pp. 276–280, Mar. 1986.

[83] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, "Sail: Single access point-based indoor localization," in *Proc. ACM MobiSys'14*, Bretton Woods, New Hampshire, USA, Jun. 2014, pp. 315–328.

[84] D. Vasisht, S. Kumar, and D. Katabi, “Decimeter level localization with a single wifi access point,” in *Proc. ACM NSDI’16*, Santa Clara, CA, March 2016, pp. 165–178.

[85] X. Wang, L. Gao, and S. Mao, “CSI phase fingerprinting for indoor localization with a deep learning approach,” *IEEE Internet of Things J.*, vol. 3, no. 6, pp. 1113–1123, Dec. 2016.

[86] X. Wang, X. Wang, and S. Mao, “Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi,” in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[87] Z. He, S. Mao, and S. Kompella, “Quality of experience driven multi-user video streaming in cellular cognitive radio networks with single channel access,” *IEEE Transactions on Multimedia*, vol. 18, no. 7, pp. 1401–1413, July 2016.

[88] Z. He, S. Mao, and T. Rappaport, “On link scheduling under blockage and interference in 60 ghz ad hoc networks,” *IEEE Access Journal*, vol. 3, pp. 1437–1449, Sept. 2015.

[89] Y. Wang and S. Mao, “On distributed power control in full duplex wireless networks,” *Elsevier Digital Communications and Networks Journal*, vol. 3, no. 1, pp. 1–10, Feb. 2017.

[90] N. Tang, S. Mao, Y. Wang, and R. Nelms, “LASSO-based single index model for solar power generation forecasting,” in *Proc. IEEE GLOBECOM 2017*, Singapore, Dec. 2017.

[91] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[92] F. J. Ordonez and D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” *MDPI Sensors*, p. 115, Jan. 2016.

[93] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, “Convolutional neural networks for human activity recognition using mobile sensors,” in *Proc. MobiCASE’14*, Austin, TX, Nov. 2014, pp. 197–205.

[94] B. Mei, Y. Xiao, R. Li, H. Li, X. Cheng, and Y. Sun, "Image and attribute based convolutional neural network inference attacks in social networks," *IEEE Transactions on Network Science and Engineering*, 2018.

[95] D. Hu, S. Mao, and J. H. Reed, "On video multicast in cognitive radio networks," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2222–2230.

[96] Y. Xu and S. Mao, "User association in massive mimo hetnets," *IEEE Systems Journal*, vol. 11, no. 1, pp. 7–19, 2017.

[97] M. Feng, S. Mao, and T. Jiang, "Base station ON-OFF switching in 5G wireless systems: Approaches and challenges," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 46–54, Aug. 2017.

[98] M. Feng, T. Jiang, D. Chen, and S. Mao, "Cooperative small cell networks: High capacity for hotspots with interference mitigation," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 108–116, Dec. 2014.

[99] M. Feng, S. Mao, and T. Jiang, "Boost: Base station on-off switching strategy for energy efficient massive mimo hetnets," in *Proc. IEEE INFOCOM 2016*, San Francisco, CA, Apr. 2016, pp. 1395–1403.

[100] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor localization via channel response," *ACM Computing Surveys*, vol. 46, no. 2, pp. 25:1–25:32, Nov. 2013.

[101] X. Wang, L. Gao, and S. Mao, "Biloc: Bi-modal deep learning for indoor localization with commodity 5ghz wifi," *IEEE Access*, vol. 5, pp. 4209–4220, 2017.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

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

[104] 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)*, 2010, pp. 807–814.

[105] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[106] O. Boric-Lubeke and V. Lubecke, "Wireless house calls: Using communications technology for health care and monitoring," *IEEE Microwave Mag.*, vol. 3, no. 3, pp. 43–48, Apr. 2002.

[107] C. Hunt and F. Hauck, "Sudden infant death syndrome," *Can. Med. Assoc. J*, vol. 174, no. 13, pp. 1309–1310, Apr. 2006.

[108] M. L. R. Mogue and B. Rantala, "Capnometers," *Journal of clinical monitoring*, vol. 4, no. 2, pp. 115–121, Apr. 1988.

[109] N. H. Shariati and E. Zahedi, "Comparison of selected parametric models for analysis of the photoplethysmographic signal," in *Proc. 1st IEEE Conf. Comput., Commun. Signal Process.*, Kuala Lumpur, Malaysia, Nov. 2005, pp. 169–172.

[110] F. Adib, H. Mao, Z. Kabelac, D. Katabi, and R. Miller, "Smart homes that monitor breathing and heart rate," in *Proc. ACM CHI'15*, Seoul, Korea, April 2015, pp. 837–846.

[111] A. Droitcour, O. Boric-Lubecke, and G. Kovacs, "Signal-to-noise ratio in Doppler radar system for heart and respiratory rate measurements," *IEEE Trans. Microw. Theory Technol.*, vol. 57, no. 10, pp. 2498–2507, Oct. 2009.

[112] P. Nguyen, X. Zhang, A. Halbower, and T. Vu, "Continuous and fine-grained breathing volume monitoring from afar using wireless signals," in *Proc. IEEE INFOCOM'16*, San Francisco, CA, Apr. 2016, pp. 1–9.

[113] J. Salmi and A. F. Molisch, "Propagation parameter estimation, modeling and measurements for ultrawideband mimo radar," *IEEE Trans. Microw. Theory Technol.*, vol. 59, no. 11, pp. 4257–4267, Nov. 2011.

[114] Z. Yang, P. Pathak, Y. Zeng, X. Liran, and P. Mohapatra, "Monitoring vital signs using millimeter wave," in *Proc. IEEE MobiHoc'16*, Paderborn, Germany, July 2016, pp. 211–220.

[115] H. Abdelnasser, K. A. Harras, and M. Youssef, "Ubibreathe: A ubiquitous non-invasive wifi-based breathing estimator," in *Proc. IEEE MobiHoc'15*, Hangzhou, China, June 2015, pp. 277–286.

[116] J. Liu, Y. Wang, Y. Chen, J. Yang, X. Chen, and J. Cheng, "Tracking vital signs during sleep leveraging off-the-shelf WiFi," in *Proc. ACM Mobihoc'15*, Hangzhou, China, June 2015, pp. 267–276.

[117] S. Sardy, P. Tseng, and A. Brace, "Robust wavelet denoising," *IEEE Trans. Signal Process.*, vol. 49, no. 6, pp. 1146–1152, June 2001.

[118] C. G. Scully, J. Lee, J. Meyer, A. M. Gorbach, D. Granquist-Fraser, Y. Mendelson, and K. H. Chon, "Physiological parameter monitoring from optical recordings with a mobile phone," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 2, pp. 303–306, Feb. 2010.

[119] H. Aly and M. Youssef, "Zephyr: Ubiquitous accurate multi-sensor fusion-based respiratory rate estimation using smartphones," in *Proc. IEEE INFOCOM'16*, San Francisco, CA, Apr. 2016, pp. 1–9.

[120] Y. Ren, C. Wang, J. Yang, and Y. Chen, "Fine-grained sleep monitoring: Hearing your breathing with smartphones," in *Proc. IEEE INFOCOM'15*, Hong Kong, China, Apr. 2015, pp. 1194–1202.

[121] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 2, pp. 16:1–16:44, Nov. 2016.

[122] Y. Luo, D. Tao, Y. Wen, R. Kotagiri, and C. Xu, "Tensor canonical correlation analysis for multi-view dimension reduction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3111–3124, Nov. 2015.

[123] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Towards scalable systems for big data analytics: A technology tutorial," *IEEE Access Journal*, vol. 2, pp. 652–687, July 2014.

[124] R. Irving, "An efficient algorithm for the 'stable roommates' problem," *J. Algorithms*, vol. 6, no. 6, pp. 577–595, 1985.

[125] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[126] L. D. Lathauwer, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-(Lr, Lr, 1) terms," *SIAM J. Matrix Anal. Appl*, vol. 32, no. 4, pp. 1451–1474, Dec. 2011.

[127] A. Cichocki, D. P. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. F. Caiafa, and A. H. Phan, "Tensor decompositions for signal processing applications: From two-way to multi-way component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[128] Y. Sun and M. Kumar, "Numerical solution of high dimensional stationary fokker-planck equations via tensor decomposition and chebyshev spectral differentiation," *Computers & Mathematics with Applications*, vol. 67, no. 10, pp. 1960–1977, 2014.

[129] ——, "A numerical solver for high dimensional transient fokker-planck equation in modeling polymeric fluids," *Journal of Computational Physics*, vol. 289, no. 10, pp. 149–168, 2015.

[130] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *Proc. KDD Workshop Mining Temporal Sequential Data*. Seattle, WA: ACM, Aug. 2004, pp. 70–80.

[131] ——, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis Journal*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[132] M. Feng, S. Mao, and T. Jiang, "Joint duplex mode selection, channel allocation, and power control for full-duplex cognitive femtocell networks," *Elsevier Digit. Commun. Netw*, vol. 1, no. 1, pp. 30–44, Feb. 2015.

[133] ——, "Duplex mode selection and channel allocation for full-duplex cognitive femtocell networks," in *Proc. IEEE WCNC 2015*. New Orleans, LA: IEEE, Mar. 2015, pp. 1900–1905.

[134] H. Wang, D. Zhang, J. Ma, Y. Wang, Y. Wang, D. Wu, T. Gu, and B. Xie, "Human respiration detection with commodity wifi devices: Do user location and body orientation matter?" in *Proc. Ubicomp'16*. Heidelberg, Germany: ACM, Sept. 2016, pp. 25–36.

[135] M. X. Gong, R. J. Stacey, D. Akhmetov, and S. Mao, "A directional CSMA/CA protocol for mmWave wireless PANs," in *Proc. IEEE WCNC 2010*, Sydney, Australia, Apr. 2010, pp. 1–6.

[136] I. Cushman, D. B. Rawat, A. Bhimraj, and M. Fraser, "Experimental approach for seeing through walls using wi-fi enabled software defined radio technology," *Elsevier Digit. Commun. Netw*, vol. 2, no. 4, pp. 245–255, Nov. 2016.

[137] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti., "Spotfi: Decimeter level localization using wif," in *Proc. ACM SIGCOMM'15*. London, United Kingdom: ACM, Aug. 2015, pp. 269–282.

[138] G. Wang, Y. Zou, Z. Zhou, K. wu, and L. Ni, "We can hear you with wi-fi!" in *Proc. ACM Mobicom'14*. Maui, Hawaii: ACM, Sep. 2014, pp. 593–604.

[139] W. Wang, A. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," in *Proc. ACM Mobicom'15*. Paris, France: ACM, Sept. 2015, pp. 65–76.

[140] X. Wang, C. Yang, and S. Mao, "Resbeat: Resilient breathing beats monitoring with realtime bimodal csi data," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. Singapore: IEEE, 2017, pp. 1–6.

[141] X. Wang, R. Huang, and S. Mao, "Sonarbeat: Sonar phase for breathing beat monitoring with smartphones," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017, pp. 1–8.

[142] ——, "Demo abstract: Sonarbeat: Sonar phase for breathing beat monitoring with smartphones," in *Sensing, Communication, and Networking (SECON), 2017 14th Annual IEEE International Conference on*. San Diego, CA: IEEE, 2017, pp. 1–2.

[143] X. Wang, C. Yang, and S. Mao, "PhaseBeat: Exploiting CSI phase data for vital sign monitoring with commodity WiFi devices," in *Proc. IEEE ICDCS 2017*, Atlanta, GA, June 2017, pp. 1–10.

[144] ——, "Tensorbeat: Tensor decomposition for monitoring multi-person breathing beats with commodity WiFi," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 1, pp. 8:1–8:27, Sept. 2017.

[145] S. Roberts, “Control chart tests based on geometric moving averages,” *Technometrics*, vol. 42, no. 1, pp. 97–101, 2000.

[146] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, “The internet of things for health care: a comprehensive survey,” *IEEE Access*, vol. 3, pp. 678–708, 2015.

[147] X. Wang, S. Mao, and M. X. Gong, “An overview of 3gpp cellular vehicle-to-everything standards,” *GetMobile: Mobile Computing and Communications*, vol. 21, no. 3, pp. 19–25, 2017.

[148] S. Cicalo, M. Mazzotti, S. Moretti, V. Tralli, and M. Chiani, “Multiple video delivery in m-health emergency applications,” *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 1988–2001, 2016.

[149] S.-E. Moon and J.-S. Lee, “Implicit analysis of perceptual multimedia experience based on physiological response: A review,” *IEEE Transactions on Multimedia*, vol. 19, no. 2, pp. 340–353, 2017.

[150] H. Müeller and D. Unay, “Retrieval from and understanding of large-scale multi-modal medical datasets: A review,” *IEEE Transactions on Multimedia*, 2017.

[151] I. M. Baytas, K. Lin, F. Wang, A. K. Jain, and J. Zhou, “Phenotree: Interactive visual analytics for hierarchical phenotyping from large-scale electronic health records,” *IEEE Transactions on Multimedia*, vol. 18, no. 11, pp. 2257–2270, 2016.

[152] P. Henriquez, B. J. Matuszewski, Y. Andreu-Cabedo, L. Bastiani, S. Colantonio, G. Coppini, M. D’Acunto, R. Favilla, D. Germanese, D. Giorgi *et al.*, “Mirror mirror on the wall... an unobtrusive intelligent multisensory mirror for well-being status self-assessment and visualization,” *IEEE Transactions on Multimedia*, 2017.

[153] X. Yuan, X. Wang, C. Wang, J. Weng, and K. Ren, "Enabling secure and fast indexing for privacy-assured healthcare monitoring via compressive sensing," *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 2002–2014, 2016.

[154] R. Nandakumar, S. Gollakota, and N. Watson, "Contactless sleep apnea detection on smartphones," in *Proc. ACM MobiSys'15*. Florence, Italy: ACM, May 2015, pp. 45–57.

[155] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *Proc. ACM MobiCom'16*. New York City, NY: ACM, Oct. 2016, pp. 82–94.

[156] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *Proc. ACM CHI'16*. Santa Clara, CA: ACM, June 2016, pp. 1515–1525.

[157] F. Li, H. Chen, X. Song, Q. Zhang, Y. Li, and Y. Wang, "CondioSense: High-quality context-aware service for audio sensing system via active sonar," *Personal and Ubiquitous Computing J.*, vol. 22, no. 1, pp. 17–29, Feb. 2017.

[158] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[159] B. Fu, J. Karolus, T. Grosse-Puppendahl, J. Hermann, and A. Kuijper, "Opportunities for activity recognition using ultrasound doppler sensing on unmodified mobile phones," in *Proc. 2nd international Workshop on Sensor-based Activity Recognition and Interaction*. Rostock, Germany: ACM, June 2015, p. 8.

[160] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *Proc. ACM MobiSys'15*. ACM, May 2015, pp. 15–29.

[161] W. Mao, J. He, and L. Qiu, "CAT: High-precision acoustic motion tracking," in *Proc. ACM Mobicom'16*. New York City, NY: ACM, Oct. 2016, pp. 491–492.

[162] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser, "Snooping keystrokes with mm-level audio ranging on a single phone," in *Proc. ACM Mobicom'15*. Paris, France: ACM, Sept. 2015, pp. 142–154.

[163] J. Wang, K. Zhao, X. Zhang, and C. Peng, "Ubiquitous keyboard for small mobile devices: Harnessing multipath fading for fine-grained keystroke localization," in *Proc. ACM Mobisys'14*. Bretton Woods, NH: ACM, June 2014, pp. 14–27.

[164] C. Qiu and M. W. Mutka, "Silent whistle: Effective indoor positioning with assistance from acoustic sensing on smartphones," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*. IEEE, 2017, pp. 1–6.

[165] R. Nandakumar, K. K. Chintalapudi, V. Padmanabhan, and R. Venkatesan, "Dhwani: secure peer-to-peer acoustic NFC," vol. 43, no. 4, pp. 63–74, 2013.

[166] R. Nandakumar and S. Gollakota, "Unleashing the power of active sonar," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 11–15, 2017.

[167] B. Zhou, M. Elbadry, R. Gao, and F. Ye, "Batmapper: Acoustic sensing based indoor floor plan construction using smartphones," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2017, pp. 42–55.

[168] W. Ruan, Q. Z. Sheng, L. Yang, T. Gu, P. Xu, and L. Shangguan, "AudioGest: Enabling fine-grained hand gesture detection by decoding echo signal," in *Proc. ACM UBICOMP'16*. Heidelberg, Germany: ACM, Sept. 2016, pp. 474–485.

[169] C. A. Dimoulas, "Audiovisual spatial-audio analysis by means of sound localization and imaging: A multimedia healthcare framework in abdominal sound mapping," *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 1969–1976, 2016.

[170] E. A. Bernal, X. Yang, Q. Li, J. Kumar, S. Madhvanath, P. Ramesh, and R. Bala, "Deep temporal multimodal fusion for medical procedure monitoring using wearable sensors," *IEEE Transactions on Multimedia*, 2017.

[171] F. Mokaya, R. Lucas, H. Y. Noh, and P. Zhang, "Burnout: A wearable system for unobtrusive skeletal muscle fatigue estimation," in *Proc. IEEE/ACM ISPN'16*. Kobe, Japan: IEEE, Oct. 2016, pp. 1–12.

[172] C. Yang, G. Cheung, and V. Stankovic, "Estimating heart rate and rhythm via 3d motion tracking in depth video," *IEEE Transactions on Multimedia*, 2017.

[173] T. Wei and X. Zhang, "mTrack: High-precision passive tracking using millimeter wave radios," in *Proc. ACM Mobicom'15*. Paris, France: ACM, Sept. 2015, pp. 117–129.

[174] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Comput.*, vol. 16, no. 3, pp. 82–88, July 2017.