PRECISE INDOOR POSITIONING USING BLUETOOTH LOW ENERGY (BLE) TECHNOLOGY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYBARS KEREM TAŞKAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

NOVEMBER 2019

Approval of the thesis:

PRECISE INDOOR POSITIONING USING BLUETOOTH LOW ENERGY (BLE) TECHNOLOGY

submitted by **AYBARS KEREM TAŞKAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Hande Alemdar
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Cem Ersoy
Computer Engineering, Boğaziçi University _____

Assist. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

Assist. Prof. Dr. Pelin Angın
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Aybars Kerem Taşkan

Signature :

ABSTRACT

PRECISE INDOOR POSITIONING USING BLUETOOTH LOW ENERGY (BLE) TECHNOLOGY

Taşkan, Aybars Kerem
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Hande Alemdar

November 2019, 121 pages

Using wireless technologies, locating objects with very precise measures is hard. It is even harder when we want to locate indoor positions due to surrounding materials preventing signals to be received. With the help of Bluetooth low energy technology, we can locate indoor objects/people with more precision than using previous Bluetooth technologies. In this thesis, methods will be improved and implemented to achieve cost-effective precise indoor location.

Keywords: Bluetooth Low Energy, Indoor Positioning, Particle Filter, Multilateration

ÖZ

BLUETOOTH DÜŞÜK ENERJİ TEKNOLOJİSİ (BDE) KULLANARAK YÜKSEK DOĞRULUKLU YER TESBİTİ

Taşkan, Aybars Kerem
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Hande Alemdar

Kasım 2019 , 121 sayfa

Kablosuz Ağ Teknolojileri kullanarak, nesnelerin yer tesbitini yüksek doğrulukla yapmak zor; kapalı alanlarda nesnelerin yer tesbiti, bu alanı çevreleyen nesnelerden dolayı, daha da zor bir problem. Bluetooth düşük enerji teknolojisi kullanarak, kapalı alanlarda nesnelerin/insanların yer tesbitini, daha önceki Bluetooth teknolojilerinde olmadığı kadar yüksek doğrulukla yapmak mümkün. Tezim kapsamında bu problemi çözmek için yöntemler geliştirilip, yüksek doğrulukla ve uygun bir maliyetle uygulanacak.

Anahtar Kelimeler: Bluetooth Düşük Enerji, Kapalı Alan Yer Tesbiti, Parçacık Filtresi, Multilaterasyon

This thesis is dedicated to my family for their love, endless support and to my beloved ones.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Assist. Prof. Dr. Hande ALEMDAR for her guidance, support , encouragements and toleration throughout the preparation of this thesis. I also would like to thank Prof. Dr. Cem ERSOY and Assist. Prof. Dr. Pelin ANGIN for their feedbacks and comments about this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

AP	Access point (as in Wi-Fi AP)
BLE	Bluetooth low energy
CoO	Cell of origin
csv	Comma separated values (a file extension)
dBm	Decibel-miliwatts, a unit used to measure electromagnetic wave signal powers. It represent decibels relative to one milliwatt.
DBSCAN	Density-based spatial clustering of applications with noise
DBSCAN-KRF	An algorithm merging DBSCAN, KNN and random forest algorithms.
DR	Dead reckoning
DRNN	Deep recurrent neural network
DTDOA	Differential time difference of arrival
EDR	Enhanced data rate (as in BLE v2.1 + EDR)
EID	Ephemeral identifier (as in Eddystone-EID)
EKF	Extended Kalman filter
EM	Electromagnetic
EMW	Electromagnetic wave
FCC	The Federal Communications Commission (based in the US)
FM	Frequency modulation
GAP	Generic access profile
GPS	Global positioning system
HCI	Host controller interface
HS	High speed (as in BLE v3.0 + HS)

IE	Indoor environment where we track the people in.
IEEE	Institute of Electrical and Electronics Engineers
Info	Information
INR	Interference to noise ratio
iOS	iPhone operating system
IoT	Internet of things
IPS	Indoor positioning system
ISM	Industrial, scientific and medical (as in ISM band)
JPA	Joint probability algorithm
KF	Kalman filter
KG	Kalman gain
KNN	K-nearest neighbor
KWNN	K-weighted nearest node
L2CAP	Logical link control and adaptation layer protocol
LL	Link layer
LSE	Least squared error
LSTM	Long short-term memory
MACID	Media access control identification
ML	Machine learning
mW	Milliwatt. A unit used to measure EMW signal power.
N_eff	Efficient number of samples. A ratio of most weighted particles over all particles.
NA	Not available
nD	n dimensional (as in 2D, 3D)
ODA	Offline date acquisition
ONM	Obstruction-aware no-signal multilateration
OP	Online positioning

PAN	Personal area networks
PDF	Probability distribution function
PF	Particle filter
PhD	Doctor of philosophy
PHY	Physical layer
POI	Person of interest, the human that we track on the map.
RAF	Running average filtering
RF	Radio frequency
RFID	Radio-frequency identification
RMSE	Root mean square error
RSSI	Received signal strength indicator
SIR	Sequential importance resampling
SIS	Sequential importance sampling
SVM	Support vector machine
TDOA	Time difference of the signal
TLM	Telemetry (as in Eddystone-TLM)
TWTF	Two-way time of flight
TX Power	Transmission power of the signal from the beacon
UID	Unique identifier
UKF	Unscented Kalman filter
URI	Universal resource identifier
URL	Uniform resource locator
US	United States
UUID	Universally unique identifier
UWB	Ultra wide band.
WKNN	Weighted k-nearest neighbor
WNN	Weighted nearest neighbor
WPAN	Wireless personal area network

CHAPTER 1

INTRODUCTION

As the *indoor environments* (IEs) get larger, it becomes harder to find our ways in them. For a visually impaired individual, finding ways is even harder. As the IEs like malls have more stores and the markets have more products to choose from, it can be really confusing what to do and which section to go without spending much time on looking for directions. For these kinds of situations, *indoor positioning systems* (IPSs) become really handy. An IPS can direct us and can tell us where we are. Besides, IPSs provide many business and marketing benefits. For instance, a store owner may wonder where their customers spend time and what their dwell times in the store are; a museum application can send informative push notifications to the museum visitors when they look at a particular painting or an antique piece by knowing the visitor positions via an IPS. However, for an IPS to work properly, the accuracy of the system should be high. Furthermore, this IPS should be cost-effective so that its usage becomes more broad and hence, more people can benefit from it.

1.1 Problem Definition

Our problem is an indoor tracking problem. For indoor tracking, we need some information about the possible location of the *person of interest* (POI), that is, person who we track. Depending on the problem, this information can be acquired using sensor devices like camera, gyroscope and accelerometer or using wireless signal antennas [1]. For tracking using signals, we need to eliminate the noise in the signals and extract the right information from these signals.

Indoor positioning can be an expensive problem either due to the methods used or due

to expensive hardware components. Using camera technologies for indoor tracking may not be preferable due to two main reasons:

- If we need to track people, we need to be able to recognize them using techniques like face recognition.
- Cameras are expensive as data acquisition devices.

Using wireless signal technologies for indoor positioning is a cheaper option than using cameras most of the time. However, some wireless technologies could be more preferable than others. For example, BLE gives us broader opportunities comparing to other wireless methods. BLE is a common wireless technology that is generally used in small proximities which is supported by Android, *iPhone operating system* (IOS) and many other devices. BLE uses very low energy which means long battery life. For example while ZigBee consumes low energy as BLE does, ZigBee is not as widely supported. Wi-Fi is widely supported; yet has higher energy requirement than BLE. GPS is a well-known technology for positioning, however it does not work well indoors. On a map, GPS could show the same spot for a point which is located vertically the same on all the floors of a building.

Solutions using BLE may be expensive in some cases due to expensive hardware requirements like high quality antennas, using many *beacons* (signal transmitters), using many receivers or using beacons with battery-draining parameters such as low advertisement interval and high *transmission* (TX) power. Therefore, we present a cost-effective solution which mitigates the problems we mention for BLE indoor tracking problem.

1.2 Aim

With this thesis, we aim to give a detailed analysis of the BLE beacons and how to use them for indoor positioning. Our main aim is to track a person who carry a beacon object in an indoor environment with low cost and high accuracy.

Moreover, we aim to give visual results for better understanding of the BLE technology and indoor positioning problem. To that end, we present a simulation for our

solution which demonstrates the theoretical part of our solution in a graphical environment. We also conduct some experiments in a real world environment to show how BLE signals behave in real world and how our solution performs in practice. Different test results for different hardware-related tools are mentioned in the scope of this thesis with the hope that this would give an idea about how hardware affects the signal propagation.

1.3 Contributions

Our contributions are as follows:

- We propose a wireless indoor positioning algorithm called IP-PPFONM, which is publicly available at *https://github.com/AKerem/IP-PPFONM*. In PPFONM algorithm,
  - We take into account the effect of obstructions in the indoor environment and make our positioning calculations accordingly.
  - We can position the POI with some uncertainty using only one *received signal strength indicator* (RSSI) value unlike the common requirement of three RSSIs for IPSs.
  - We follow a parametrized and flexible approach so that our solution is applicable for different environmental conditions.
- We propose ways to make indoor positioning more cost-effective:
  - We use experimentally calibrated beacon parameters to find the optimum parameters so that batteries of the beacons last as long as possible. The longer the battery life, the less amount of replacement of the beacons with the new ones.
  - We implement a way to place receivers in the indoor environment in the most efficient way possible.
- We implement a simulation tool to simulate and visualize the IP-PPFONM algorithm. This tool can simulate people trajectories in an IL and generate RSSIs at the receivers. This simulation tool visualizes:
  - the positioning result of the IP-PPFONM algorithm.

  - the uncertainty of our indoor positioning prediction.
  - the material and obstruction information in the indoor environment.
- We perform real-world experiments to show
  - how BLE signals behave in certain environmental conditions.
  - how the IP-PPFONM algorithm performs in real life.

1.4 Organization

In Chapter 2, we give background information about the concepts we discuss throughout this thesis and we explain currently used wireless indoor positioning technologies with their advantages and disadvantages. Then, we explain why we choose to proceed with BLE technology.

In Chapter 3, we mention related studies performed in the field of indoor positioning using wireless technologies. We explain various wireless indoor positioning methods, the environments where these methods are applied in and the accuracy of these methods.

In Chapter 4, we explain our *indoor positioning via prefiltering and particle filtering with obstruction-aware no-signal multilateration* (IP-PPFONM) algorithm and some of its use cases.

In Chapter 5, we perform some theoretical experiments using our simulation tool to show how IP-PPFONM would behave in different environmental conditions and different parameter settings **in theory**. Then, we analyze the results of these simulations and discuss if IP-PPFONM gives satisfactory results according to our expectations.

In Chapter 6, we perform two types of real-world experiments. The first type is performed to see how IP-PPFONM works **in practice** by tracking a person in an office environment where the second type is performed to understand the BLE signal behavior in different situations.

In Chapter 7, we summarize our experiments and the IP-PPFONM algorithm. Then, we conclude this thesis by mentioning some ideas to improve our work.

CHAPTER 2

BACKGROUND INFORMATION

In this chapter, we explain different terms related to *object tracking*, different *wireless signal technologies* that can be used for indoor positioning and *indoor positioning with wireless signals*. In Section 2.3, we explain *methods used* and *inherent problems of wireless signals* for indoor positioning.

2.1 Object Tracking

The main method for tracking is the Bayesian approach which is the conceptual basis for other tracking algorithms. The Bayesian approach is composed of three main parts:

- **Prior:** Initial guess of a position.
- **Likelihood:** Measurement for prior position.
- **Posterior:** Updated position.

Hence, according to the Bayesian approach, we make a *prediction*. Then, we make some measurements to find the real position. After that, we *update* our prediction according to measurement results. In the *prediction* part, *Chapman-Kolmogorov* equation is used as shown in Equation 2.1.

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_{k-1} \tag{2.1}$$

In the *update* part, a posterior probability function is used as shown in Equation 2.2.

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)} \tag{2.2}$$

where the normalizing constant is [2]:

$$p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_k \tag{2.3}$$

Bayesian solution is an analytical solution which is hard to solve exactly, therefore there are optimum approximation methods called *Kalman filters*. For Kalman filters to work, our system should:

- be linear
- be unimodal
- have Gaussian noise
- have Gaussian posterior

If these conditions are met, the Kalman filter is an optimum algorithm for tracking. Kalman filter has a term called *Kalman gain* which gives us how much of the prediction should be used for our measurement. If we have 0 Kalman gain, we only trust the measurement; if we have 1, we only trust our prediction neglecting any measurements that we received to update our position information in the IE we are interested in.

However most of the time our system is not linear e.g. human motion is not linear considering direction and speed at each time step. To achieve non-linearity, *extended Kalman filter* (EKF) can be used. *Extended Kalman filter* is basically the Kalman filter method where non-linearites are approximated by linear functions. To approximate a non-linear motion graph in the shape of a curve, this curve is approximated by the first order Taylor expansion at the mean point. This means that we approximate this curve as the slope of this curve at the mean point of the curve which results in a linear line. For the rest, Kalman filter is used.

There is also *unscented Kalman filter* (UKF) which resembles the extended Kalman filter in terms of trying to approximate a non-linear curve as a line. The difference is, unscented Kalman filter uses multiple points to make the approximation unlike extended Kalman filter where we only use the mean point. However, these different Kalman filtering methods only applicable for unimodal systems which means we cannot track multiple objects using them. Furthermore, they require Gaussian noise and a Gaussian motion model which may not be the case in real life.

Hence, we need a filtering algorithm which can be used for multimodal, non-linear systems with non-Gaussian motion and noise models. Particle filter has the properties that we want. Hence, we use *particle filtering* for the algorithm we propose in Chapter 4 in this thesis.

There are different versions of particle filtering. The five main steps of a particle filtering method is as follows:

- **Initialization:** We initialize imaginary points called particles all around the IE map that we want to locate the *person of interest* (POI) in. These particles will accumulate towards the POI as time goes and we will know the POI position with some uncertainty. The initialization step is done once, but the following steps are done at each iteration.
- **Prediction:** We predict where our particles should be given the human motion model and update particle positions according to our prediction.
- **Weight update:** We update the weights given the measurements (e.g. the signal power information)
- **Resampling:** The prediction and update step purposes are the same as that of Bayesian approach. Yet, each filtering algorithm has its own parameters and update/prediction steps are used to change these parameters. In the particle filtering case, our parameters are particle positions and weights. In the resampling step, all particles are recreated around the region where the most weighted particles reside previously.
- **Positioning:** We make the positioning estimation according to particle weights and distribution.

Different versions of particle filtering algorithms may differ in the resampling step. We use a particle filtering algorithm which uses *sequential importance resampling* (SIR) for the resampling part of our algorithm. We do not use a method called *sequential important sampling* (SIS) because SIS suffers from the degeneracy problem. We explain the particle filtering steps that we use for our application in more detail in Chapter 4.

2.2 Wireless Signal Technologies

There are different wireless technologies that can be used for positioning. Systems like *global positioning system* (GPS) is only suitable for outdoor usage, but not for indoor usage. The reason is GPS signals are weak and severely degraded/scattered due to indoor environment's roof, walls, etc. which prevents them to reach the indoor environment. Even if they reach, the receiver devices should have a strong antenna to compensate for the weakness of the signal or there should be some repeater devices at the roof and some parts of the indoor environment depending on the size of the building.

Hence, GPS is not reliable indoors. To make this system a little bit more reliable, we need to have additional equipments which increase the cost. Therefore, we inspect some technologies like Bluetooth that can be used indoors and explain why we choose Bluetooth over the other wireless technologies for indoor positioning [3] [4]. We explain some wireless technologies that can be used for indoor positioning which are listed below:

- Bluetooth
- Wi-Fi
- ZigBee
- Ultra-Wide Band (UWB)
- Radio-Frequency Identification (RFID)
- Frequency Modulation (FM)

From all of these wireless technology signals, we can extract a value called the *received signal strength indicator* (RSSI) which is used to determine the distance of the transmitter of these signals. This makes these technologies suitable for positioning. [5]

2.2.1 Bluetooth:

Bluetooth is a wireless communication protocol standard developed for *personal area networks* (PAN). The key features of Bluetooth are its low cost, low energy consump-

tion and robustness. At first, this technology was standardized as IEEE 802.15.1. This standard specifies wireless *medium access control* (MAC), *physical layer* (PHY) specifications for *wireless personal area networks* (WPANs) and explains methods for communication devices in WPANs. [6] [7]. However, Bluetooth is currently maintained by the Bluetooth Special Interest Group (SIG). Bluetooth versions are published in a specification called *Bluetooth core specification*. These specifications are periodically revised and each revision is another Bluetooth version. Revision 1.0a is the first version of Bluetooth specification published on the public website. Starting with Revision v1.2, the SIG adopted the Bluetooth core specification.

Bluetooth operates in the unlicensed $2.4GHz$ *industrial, scientific and medical radio* (ISM) band. This frequency band contains frequencies between $2400MHz$ and $2483.5MHz$. Bluetooth is a constantly developing technology and each major development is specified as a new version. Bluetooth versions up to v2.0 is called as Bluetooth *basic rate* (BR) . Bluetooth v2.0 specifies *Bluetooth enhanced data rate* (v2.0 + EDR) where a new modulation scheme is added for Bluetooth to increase the bit rate, hence enabling higher data transfer speed. The Bluetooth technology combining the modulation scheme in BR and EDR is called as *Bluetooth BR/EDR*, which is also known as *Bluetooth classic*. Bluetooth v3.0 specifies *Bluetooth high speed* (v3.0 + HS) enabling much higher data rates compared to Bluetooth classic. With the emergence of Bluetooth v4.0, a technology called *Bluetooth low energy* (BLE) became available in addition to Bluetooth classic and Bluetooth high speed which is a Bluetooth technology lowering the power consumption considerably. Several new features were added to make BLE technology to consume lower energy when transitioning from v3.0 + HS to v4.0. These features include changes in several layers in which some of them are: [8].

- Physical Layer (PHY)
- Link Layer (LL)
- Host Controller Interface (HCI)
- Logical Link Control and Adaptation Layer Protocol (L2CAP)
- Generic Access Profile (GAP)

When transitioning from v4.2 to v5.0 on December 2016, the maximum achievable

BLE range is increased, which is stated as *LE Long Range* in the specifications [8]. Additional to *LE 1M* which is the PHY used in Bluetooth 4, Bluetooth 5 adds two new PHY variants to the PHY specification used in Bluetooth 4 which are *LE 2M* and *LE Coded*. With the help of *LE 2M*, the speed of Bluetooth 4 is doubled while *LE Coded* helped to quadruple the range of Bluetooth 4. New *LE 2M* PHY allows the PHY to operate at $2Ms/s$ and hence enables higher data rates than LE 1M [9]. At the time of writing, the latest Bluetooth version is v5.1. [8].

Bluetooth core specification revisions before and after the SIG adoption with their corresponding revision dates can be found in Table 2.1 and Table 2.2 respectively.

Table 2.1. Bluetooth Revisions after SIG

Revision	Date
v5.1	Jan 21 2019
v5.0	Dec 06 2016
v4.2	Dec 02 2014
v4.1	Dec 03 2013
v4.0	June 30 2010
v3.0 + HS	April 21 2009
v2.1 + EDR	July 26 2007
v2.0 + EDR	Aug 01 2004
v1.2	Nov 05 2003

Table 2.2. Bluetooth Revisions before SIG

Revision	Date
1.1	Feb 22 2001
1.0B	Dec 01 1999
1.0a	July 26 1999
1.0 draft	July 05 1999
0.9	April 30 1999
0.8	Jan 21 1999
0.7	Oct 19 1998

Devices supporting BLE are known as Bluetooth smart devices whereas devices supporting both BLE and Bluetooth classic are known as Bluetooth smart ready devices. If a device has Bluetooth v4.0 or higher in their specifications, this does not necessarily mean that this device support Bluetooth classic, BLE and Bluetooth high speed modes together. It just shows that this device supports at least one of these modes of the Bluetooth technology complying with the requirements of the corresponding Bluetooth version.

BLE is a tempting indoor positioning technology due to its low cost and ubiquity. A BLE receiver can measure a value named RSSI from the BLE signal at the time of receiving which can be used for distance calculations making this technology suitable

for indoor positioning. Moreover, BLE signals carry some data which could be useful for positioning or to identify the signal source. Hence, we explain the packet format of BLE where we extract these signal data that we use for indoor positioning.

The BLE packet format has four main sections[10]:

- **Preamble:** This value is an alternate sequence of zeros and ones. Using these sequences, receiver device synchronize its radio to the right frequency and do some more calculations to make sure the remaining part of the BLE packet is received correctly.
- **Access Address:** This address is used as a correlation code to ensure the transmission is indeed for the receiver which is receiving the packet. This address prevents unrelated BLE devices using the same RF channel simultaneously.
- **Protocol Data Unit(PDU)**: This section contains the main information of the BLE packet.
- **Cyclic Redundancy Check(CRC):** It is the 3-byte checksum calculated over the PDU.

PDUs split into two categories which are *advertising channel PDUs* and *data channel PDUs*. Yet, we will focus on the *advertising channel PDUs* (ACP). *Access address* value is the same for all ACP. ACP have three types:

- **Advertising PDUs:** These PDUs consist of the **advertisement packet**. There are different specifications like iBeacon, Eddystone and AltBeacon which try to form a standard for advertisement packet. We will only mention iBeacon and Eddystone.
- **Scanning PDUs:** These PDUs consists of the *Scan Response Packet*. There is no globally accepted standard for scan response packet. Hence, different beacon manufacturers may send different information in this packet like battery status and beacon name.
- **Initiating PDUs**: Link layer uses these PDUs to initiate a connection to the advertiser (beacon).

In the PDU payload, there is also BLE MAC address information of the beacon even though this MAC address might have been spoofed. If this MAC address does not change, this address can also be used to identify the beacon. Note that, RSSI value is not in the original BLE packet sent by the beacon, but it is the value sensed by the receiver (sniffer) device.

We mention two specifications iBeacon and Eddystone for the advertisement packet structure which reside in advertising PDUs.

2.2.1.1 iBeacon

It is the Apple's software protocol to transmit BLE signals. We use this protocol. Some of the fields used by this protocol are [11]:

- **Major:** Two byte data. It has a decimal value between 0 and 65535.
- **Minor:** Two byte data. It has a decimal value between 0 and 65535.
- **UUID:** Sixteen byte data (Thirty two hexadecimal digits).
- **RSSI at 1 meter:** This value is determined after calculating RSSI value at the receiver device 1 meter ($1m$) away from the beacon. It is used to predict the distance of the beacon to the receiver device.

UUID, major and minor fields are used to identify the beacon. So, for instance, we can make two groupings using these three fields by grouping beacons by their UUIDs first and then their major values. After all groupings, using the minor value, we can identify each individual beacon.

2.2.1.2 Eddystone

It is the Google's open software protocol which is designed to be robust and transparent [12]. Inside an Eddystone frame, different payload types can be included:

- **Eddystone-UID:** Sixteen byte data.
- **Eddystone-URL:** Data generally used by a service to redirect client (receiver) to the service's website.

- **Eddystone-TLM:** Beacon status data.
- **Eddystone-EID (Ephemeral Identifier):** An identifier field varies by time which can resolved to reach information shared by a service that sends information to a beacon by a key called *ephemeral identity key* (EID) [13].

2.2.2 Wi-Fi

Wi-Fi is the most commonly used wireless communications technology which is based on *Electrical and Electronics Engineers* (IEEE) wireless communication standard 802.11. This standard makes specifications about *wireless local area networks* (WLANs).

Wi-Fi Technology is a trade-mark of a global non-profit association formed by worldwide network of companies, named *Wi-Fi Alliance*. The trademark term "Wi-Fi" is adopted in 2000 [14]. The 802.11 standard adopted by Wi-Fi was initially released in 1997 before even Wi-Fi Alliance formed, where standard specifications made in 1997 is superseded by specifications in 1999, 2007, 2012 and 2016 respectively. [15] [16] [17] [18] [19]. Therefore, at the moment of writing, latest 802.11 standard depends on the specifications made in IEEE 802.11-2016. The usage of channels and frequency spectrum width for WLAN in $2.4GHz$ band may differ in different regions of the world as in the *United States* (US) and Europe.

Being the most common wireless technology, this technology is used in many indoor environments already. Moreover, Wi-Fi signal has signal strength value indicator which decreases as the distance traveled by this signal increases which makes this technology usable in *indoor positioning systems* (IPS).

2.2.3 ZigBee

ZigBee is a wireless communication protocol standard developed for Personal Area Networks and Local Area Networks. This technology is intended to require low energy and have low data rate. It is based on IEEE 802.15.4 standard. At the time of writing the newest ZigBee standard is ZigBee 3.0 which is built on ZigBee PRO.

It uses $2.4GHz$ ISM Band like Bluetooth globally and it also enables regional operation at $868Mhz$ for America and $915Mhz$ for Europe. It uses:

- 16 channels each of which $2MHz$ wide in $2.4GHz$ ISM Band with a data throughput of of $250Kbits/s$.
- 27 channels for $915 - 921MHz$ with a data throughput of of $10Kbits/s$.
- 63 channels for $868MHz$ with a data throughput of of $100Kbits/s$.

ZigBee 3.0 can reach a range of $300m$ outdoors, when transmitter and receiver devices are in line of sight and can reach $75m - 100m$ for indoor usage which makes its usage suitable for indoor positioning.

For positioning systems, wireless signal inteference is an important issue that should be handled and ZigBee mitigates this problem having 16 separate channels in $2.4GHz$ ISM band where several of these channels do not overlap with European and US versions of Wi-Fi.

2.2.4 Ultra Wide Band

Ultra wide band (UWB) is a wireless technology that is used to transfer data at high rates over short distances by generating very narrow electrical pulses. As the name of this tehcnology implies this technology covers a very wide bandwidth, which is at least $500MHz$ and this brings some problems like interference with other signals such as *universal mobile telecommunications service* (UMTS), *global system for mobile communications* GSM, *television* (TV) and GPS. To prevent UWB from interfering with other signals, regulatory actions are taken and thus, the radiated power of UWB is low [20].

The Federal Communications Commission (FCC) in the US defines UWB as the radio tecnology having bandwidth of at least $500MHz$ or a bandwidth occupying more than 20% of the center frequency. [20], FCC also regulates parameters like *interference to noise ratio* (INR) for the UWB signal powers [21].

UWB is suitable for indoor positioning since, owing to its high transmission bandwidth, UWB makes accurate positioning estimations and does not fade away easily

when propagating or penetrating through objects which is mostly the case for the other signals like Wi-Fi [22]. UWB may achieve as minimum as $10cm$ as ranging accuracy [23]

This technology is emerged as a result of the IEEE 802.15.3a Task Group's work [24]. However, later, this technology is adopted and promoted by a global non-profit organization called *WiMedia Alliance* [25] and an industry organization *UWB Forum*. Now it is promoted by a global non profit organization named *UWB Alliance*. UWB Alliance promotes 802.15.4 and other standards as base standards for UWB. This Alliance also promotes usage of frequencies ranges greater than $95GHz$ [26]. The delay of the standardization process of UWB might have delayed the ubiquity of this technology.

2.2.5 Radio-Frequency Identification

Radio-Frequency Identification (RFID) is a low cost, long-lasting and low power consumer wireless technology where the installation of it in an IE is easy. This technology can work at four different frequency bands which can be classified as: [27]

- **Low Frequency:** $125 - 134kHZ$.
- **High Frequency:** $13.56MHz$.
- **Ultra High Frequency:** $433 - 956MHz$.
- **Microwave frequency:** $2.45GHz$ and $5.8GHz$.

Normally RFID is transmitted through RFID tags according to their frequency range. But $2.4GHz$ and $5.8GHz$ RFID spectrum may be combined into one RFID tag to get the best of real time positioning info as well [28].

Systems using RFID can be categorized as passive and active. Passive RFID tags do not require batteries solving the problem of maintenance for the dead batteries. Passive tags have a longer time than active does, but their reading coverage distance is limited to where they are placed. Active tags have batteries and can measure various values like temperature, acceleration and can form a distributed wireless ad hoc network. [29]

2.2.6 Frequency Modulation

Frequency modulation (FM) is a technique that encodes the data on an alternating analog or digital signal wave by varying the instantaneous frequency of the wave. It is developed by Edwin Howard Armstrong [30]. This modulation scheme is equivalent to a phase modulation where the phase shift is inversely proportional to the audio frequency. [31]

2.3 Indoor Positioning with Wireless Signals

To use wireless signals for indoor positioning, there should be devices transmitting some wireless signals and devices receiving the transmitted signals to extract some meaningful information from these signals. The devices transmitting the signals are called *beacons* whereas the devices receiving the signals are called *receivers*. Sometimes, receivers are called *anchor nodes* or *reference nodes* in indoor positioning parlance. In this section, we explain methods used for wireless indoor positioning and problems inherent in some of the wireless signals which prevents precise indoor positioning.

2.3.1 Methods Used

In this section, we explain multilateration and fingerprinting methods which are used as a part of wireless indoor positioning algorithms. We also mention the relation of these terms to this thesis.

2.3.1.1 Fingerprinting

Fingerprinting is a method where positioning in a location is carried out using a previously constructed wireless signal map where the construction of this map is made over a time period [32]. There are different approaches and methods used for fingerprinting to make improvements [33].

We use the fingerprinting method since it captures environment information and helps

filtering the RSSI info in a natural way. Since making site survey, which is the acquisition fingerprinting procedure, is hard due to requiring a lot of labor and time, there are methods to fasten this process [34].

2.3.1.2 Multilateration

Multilateration is a positioning technique that is based on *time difference of arrival* (TDoA) of the signal to different reference stations. This technique is also known as *hyperbolic positioning*. Normally, a multilateration system requires four or more stations to be taken as a reference. The signals coming from two different reference stations forms the arrival time difference. For the time difference calculation to be accurate, high-precisioned synchronization is needed for the local clocks of the reference stations [35].

As opposed to normal multilateration methods, our multilateration algorithm can make estimates using less than three receivers, although the precision of the positioning using less then three receivers may be lower than using four or more. For distance calculations, three approaches are often used which are *time of arrival* (ToA), TDoA and RSSI. Therefore, multilateration can also be done using RSSI instead of TDoA. [36]. We use RSSI information for our multilateration algorithm.

2.3.2 Inherent Problems with Wireless Signals

There are some inherent problems in wireless signals like Bluetooth which prevents precise indoor positioning. For example, Bluetooth signal attenuates as it moves in the air and it is prone to the multipath effect. In this section, we explain wireless signal attenuation and multipath effect focusing on Bluetooth signal.

2.3.2.1 Wireless Signal Attenuation

Bluetooth is an *electromagnetic wave* (EMW) operating at $2.4GHz$ frequency. Signals attenuate as they move in the air or when they hit an object. Different materials

have different signal absorption and reflection coefficients and therefore, not all objects cause the same amount of attenuation for the signal.

There are more studies and tests made for signal attenuation for Wi-Fi signals than there are for Bluetooth signals as we observe during our research. However, since Wi-Fi and Bluetooth are both electromagnetic Waves, their waves transmit in the same way where they differ in the way they use $2.4GHz$ bands. So, we think Wi-Fi attenuation tests for the frequency of $2.4GHz$ gives an idea how Bluetooth signals would attenuate for the same materials.

There are different studies on $2.4GHz$ electromagnetic signal wave loss. [37], [38]. According to a white paper [37], non-tinted glass causes $2dBm$, human body $3dBm$, marble $5dBm$ and concrete wall causes $10 - 15dBm$ signal attenuation as everyday objects we see around us. Besides these materials, copper and aluminum are also known to block RF signals well [39] [40].

2.3.2.2 Multipath Effect

Multipath propagation is a phenomenon that results in EMW arriving the receiver device's antenna by two or more paths [41]. Multipath propagation can be caused by refraction, reflection, diffraction or scattering of EMW due to objects in the transmission path [41] [42]. Multipath effect can cause $-30dBm$ of signal strength loss for BLE signals and the signal strength attenuation amount varies without any pattern at any distance. [43]. There are studies which show that using multiple BLE frequency channels lead to more accurate results than using a single channel since using multiple channels help to handle different signal transmission paths. Moreover, BLE fingerprinting method mitigates the multipath effect by taking the mean of the RSSI of the signals over a time period [43].

CHAPTER 3

RELATED WORK

Indoor positioning can be done using different wireless signal technologies. In this chapter, we mention studies that use some non-Bluetooth wireless signals for indoor positioning. Then, we explain some indoor positioning studies performed using Bluetooth low energy signals in detail to show how our approach differs from the other approaches or how we contribute to these approaches.

3.1 Non-Bluetooth Solutions For Indoor Positioning

In this section, we mention studies performed using wireless signals which are Wi-Fi, ZigBee, *ultra wide band* (UWB), *radio frequency identification* (RFID) and frequency modulation (RF) radio for indoor positioning. We explain the environmental setup, applied methods and accuracy of these studies.

3.1.1 Wi-Fi

Ma et al. (2015) [44], propose an algorithm based on traditional location fingerprinting algorithms consisting *offline date acquisition* (ODA) and *online positioning* (OP) phases. The authors compare their method with *weighted k-nearest neighbor* (WKNN), WKNN with improved Euclidean distance, *joint probability algorithm* (JPA) and JPA with improved joint probability. They observe that the algorithm they propose is superior to the other algorithms with an accuracy of $1.54m$ in an area of $5m \times 10m$ which is a room in Beijing Institute of Technology.

Park and Rhee (2017) [45], mention that they use Wi-Fi fingerprint data using a cus-

tomized *k-nearest neighbor* (KNN) algorithm which outperforms classical KNN approaches. The authors make the experiments in an area of $5.4m \times 5.4m$ using 3 *access points* (AP). The experiment area is divided into 4×4 cells (16 cells) and accuracies are calculated according to the POI being inside a cell or not. The proposed algorithm firstly clusters 100 RSSI values collected for a while in each cell and then uses KNN for classification, which is whether the POI is inside the cell or not. The authors achieve an accuracy of 90% by using the right k value for the KNN algorithm.

In a study made in *Korea Advanced Institute of Science and Technology* (KAIST), Sahar and Han (2018) [46], propose an algorithm which uses *long short-term memory* (LSTM) with fingerprinting by saying the sequence of Wi-Fi fingerprints fit for *deep recurrent neural network* (DRNN) approaches and fits even better for handling sequential data. So, they try different deep and recurrent approaches, but mainly focus on LSTM to best make use of fingerprinting data. For the evaluation, the authors collected 860 training points and 240 testing points in a floor of a university building. According to tests, the authors observed that 2 hidden layers with 0.005 learning rate gave the best result for the LSTM. They got an average positioning accuracy of $1.98m$ on the floor of a building.

Wang et al. (2019) [47] apply an algorithm called DBSCAN-KRF with fingerprinting technique. DBSCAN-KRF merges *Density-Based Spatial Clustering of Applications* (DBSCAN), KNN and Random Forest algorithms to achieve a greater accuracy than using these algorithms alone. The authors perform the experiments in two offices and corridors of a floor in a building. Both of the offices have the same plan and size of $8.8m \times 5.6m$. Each room had many desks, cupboards and flow of people. The authors placed 12 reference points in each room and 7 reference points in corridors totaling 31 reference points. Most of the reference points are $1.5m$ away from each other. The authors achieve an accuracy of $1.5m$ and $4m$, 50% and 90% of the time respectively and shows that their algorithm is superior to using KNN, Random Forest or *k-weighted nearest node* (KWNN) algorithm [48] alone, in terms of accuracy.

In Table 3.1, we summarize the related indoor positioning studies that use Wi-Fi signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used. We do not always include all the studies if there are multiple

experiments and hence multiple accuracy.

Table 3.1. Wi-Fi Indoor Positioning Related Studies

Year	Accuracy	Test Area	Method and Reference
2015	$1.54m$	$5m \times 10m$	Wi-Fi fingerprinting, ODA with OP [44]
2017	Cellwise with 90%	$5.4m \times 5.4m$	Wi-Fi fingerprinting, KNN [45]
2018	$1.98m$	Floor of a building	Wi-Fi fingerprinting, LSTM, DRNN [46]
2019	$1.5m$ / $4m$ with 50% / 90%	$8.8m \times 5.6m$	Wi-Fi fingerprinting, DBSCAN-KRF, KWNN [47]

3.1.2 ZigBee

Uradzinski and Guo (2017) [49] state that they prefilter the fingerprinting data and apply different methods like Bayesian, nearest algorithm and weighted nearest neighbor (WNN) algorithm for positioning and compare the results. Experiments are made in an area of $42.5m \times 4.96m$ with 108 data sampling points (fingerprint points) where there is $1.6m$ between each data point. They place 4 receivers at the corners. They observe that filtering improves the accuracy. They observe that WNN algorithm is the best algorithm among the algorithms they apply and get an average positioning accuracy of $0.81m$.

Ou et al. (2017) [50] design their own algorithm based on triangulation where the minimum achievable error is $0.25m$. In the proposed algorithm, there are two types of reference nodes, decision nodes and non-decision nodes. In the proposed algorithm, two non-decision nodes are chosen as base nodes and the third node is used to find the intersection point using the triangulation algorithm. Then, triangulation result (intersection point of three points) is compared with the distance results calculated according to RSSI value at this third node (decision node). The closer the RSSI results to the intersection result, the more likely the intersect point is closer to the right position. For the experiments, the authors select 2 non-decision nodes as the base reference nodes and change the third reference node among the other decision nodes to find the most fit result. According to a figure the authors provide in their study, they make the experiments in an area of $14m \times 4m$ with 8 reference nodes where 2 of these nodes are decision nodes and the others are base nodes. The authors

get an average positioning accuracy of $0.42m$.

Bianchi et al. (2019) [51] propose an algorithm and simulate a real-world environment for the accuracy tests. The simulated experimentation area are two rooms (roomA and roomC) of size $40m^2$ each and a corridor (roomB) where the roomA and roomC are next to roomB. Five, twenty five and twenty five measurement points are determined in roomB, roomA and roomC respectively. Totally eight beacons are placed where four of them are in a room, two are in the other room and the other two in the corridor. Wearable beacons are used as the transmitter devices. The authors used a localization algorithm based on RSSI fingerprinting. Roomwise accuracy is considered where the accuracy formula is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

where TP is true positive, TN is true negative, FP is false positive and FN is false. negative values. If the POI is detected in roomA correctly, then it is true positive and if the POI is not detected in the roomA correctly, then it is true negative. So, for the accuracy test, roomB and roomC are considered as one place and roomA is considered as another place. Overall 98.2% accuracy is obtained.

In Table 3.2, we summarize the related indoor positioning studies that use ZigBee signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used.

Table 3.2. ZigBee Indoor Positioning Related Studies

Year	Accuracy	Test Area	Method and Reference
2017	$0.81m$	$42.5m \times 4.96m$	Prefiltering, WNN [49]
2017	$0.42m$	$14m \times 4m$	A customized triangulation method [50]
2019	Roomwise with 98.2%	$40m^2$	ZigBee fingerprinting [51]

3.1.3 Ultra-Wide Band

Yin et al. (2016) [52] propose an algorithm based on *unscented Kalman filtering* (UKF). The authors apply *two-way time of flight* (TWTF) and trilateration algorithms as well in addition to unscented Kalman filtering. A rectangular room of size $6m \times$

$7m$ is used for the experiments. The authors place 4 anchor nodes at the corners of the room. The authors obtain an average positioning accuracy of $10cm$ overall even though for some positions, average positioning accuracies are between $30 - 40cm$.

Dabove et al. (2018) [53] use a method called *two-way time of flight* (TWTF) to compute distances. Then, inputting these distances to their multilateration algorithm, position of the *transreceiver* (TAG). So, they analyze both range and positioning capabilites of system and test a commercial solution which leverages sensor information such as inertial chipsets, altimeter and magnetometers. For positioning estimates, the authors perform the experiments in two different environments:

- **Indoor Room:** 4 anchor (reference) nodes are placed at the corners in a room of size $6.44m \times 4.91m$. The average accuracy in 3D is $100 \pm 25mm$ which is quite accurate comparing to other wireless indoor positioning methods.
- **Indoor Corridor Environment:** 4 anchor nodes are placed at the corners and as another configuration 2 more additional nodes used at the middle of 2 of the edges in a corridor of size $1.8m \times 6.8m$. The average horizontal error is $87.4mm$.

Ling et al. (2018) [54] propose an algorithm using *differential time difference of arrival* (DTDOA) technique for positioning estimation. The experiments are performed in an area of $7m \times 7m$ where 4 anchors are placed at the corners of this area. During the experiments, the POI stays line-of-sight with the anchor (receiver) devices for UWB signals to be received well. As a result, the authors obtain an average accuracy of $30cm$.

In Table 3.3, we summarize the related indoor positioning studies that use *ultra-wide band* (UWB) signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used.

Table 3.3. UWB Indoor Positioning Related Studies

Year	Accuracy	Test Area	Method and Reference
2016	$30 - 40cm$	$6m \times 7m$	UKF, TWTF and trilateration [52]
2018	$100 \pm 25mm$ / $87.4mm$	$6.44m \times 4.91m$ / $1.8m \times 6.8m$	Multilateration, TWTF [53]
2018	$30cm$	$7m \times 7m$	DTDOA [54]

3.1.4 Radio Frequency Identification

In a highly cited study, Bahl and N. Padmanabhan (2000) [55] present RADAR, a *radio frequency* (RF) based user location and tracking system in an indoor environment. This system operates by recording and processing RSSI information at multiple base stations. These stations are located to provide overlapping coverage in the area where the positioning is done. They firstly summarize multiple RSSI measurement by averaging them, which means that they collect RFID fingerprinting data. As a second step they use two different approaches where one method is empirical and the other approach is signal propagation modeling. Lastly, they use their technique called *nearest neighbors in signal space* (NNSS) to compare multiple locations and pick the best fit according to RSSI measurements. The authors make the experiments on the second floor of a three-story building where the floor is of size $43.5m \times 22.5m$. The authors state that they observe an accuracy of $2 - 3m$ using RADAR.

Ni et al. (2003) [56] present a solution called LANDMARC which is a location sensing prototype system that uses RFID technology indoor positioning. This system use active RFID tags. They propose an approach which is a usable and cost-effective indoor positioning system to overcome the inaccurate indoor positioning capability of active RFID which provides an accuracy of $2.65m$ 50% of the time and $5.97m$ 90% of the time, for projects like RADAR at that time. The authors perform the experiments in an area of $4m \times 9m$. They place 4 RF readers and 16 reference tags. The authors follow a *k-weighted nearest neighbor* (KWNN) approach with a customized weight formula for the weights which gives the least error according to authors' measurements. They use the coordinates of k-nearest reference tags of an unknown tag to locate the unknown tag. They achieve an average positioning accuracy of $1m$ and $2m$, 50% and 90% of the time respectively.

Saab and Nakad (2011) [57] use an iterative approach and apply Kalman filter to preprocess their signals and then use RSSI values for positioning. They place tags with $1.2m$ spacing and the distance between tag-layout path and the reader path is $2m$. The authors conduct the experiments in a space with concrete columns and a roof surrounded with a concrete wall and two glass walls from three sides where the fourth side is open. The size of the test area is $6m \times 5m$ where the RFID tags are

placed at a height of $2m$. The averaged absolute position error is $0.1m$ with their approach.

Xu et al. (2017) [58] present BKNN, which is an indoor positioning solution based on KNN and Bayesian probability. The authors propose an approach that can reduce signal fluctuations and errors caused by multipath and environmental interference in LANDMARC, which is the approach that Ni et al. [56] propose. The authors conduct the experiments in an area of $3.6m \times 4.8m$ where they place 117 reference tags with $45cm$ space between each two and 5 target tags in total. They get an accuracy of $15cm$ as a result of their experiments.

In Table 3.4, we summarize the related indoor positioning studies that use *radio frequency identification* (RFID) signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used.

Table 3.4. RFID Indoor Positioning Related Studies

Year	Accuracy	Test Area	Method and Reference
2000	$2-3m$	$43.5m \times 22.5m$	RFID fingerprinting, NNSS [55]
2003	$2.65m$ / $5.97m$ with 50%/ 90%	$4m \times 9m.$	Cost effective active RFID usage, KWNN[56]
2011	$0.1m$	$6m \times 5m$	Kalman filter [57]
2017	$15cm$	$3.6m \times 4.8m$	KNN, Bayesian probability [58]

3.1.5 Frequency Modulation Radio

Chen et al. (2012) [59] propose an FM-based algorithm in which when combined with Wi-Fi, the accuracy is improved by upto 83% using Wi-Fi only when accounting for wireless signal temporal variation and up to 11% without accounting for temporal variation. Temporal variation stems from authors making some of the experiments in different days. The authors also propose an additional signal quality indicator at the *physical layer* (PHY) which improves localization accuracy by 5%. The authors utilize RSSI based fingerprinting in their algorithm as well. The authors observe that Wi-Fi and FM signals are complementary of each other for positioning estimations and hence, combining Wi-Fi and FM results in higher accuracies using these signals alone. The authors make experiments in 3 buildings across the US which are resi-

dential, office and mall buildings. We give results of residential and office buildings for this study. For RSSI fingerprinting data, they take measurements for more than 100 rooms. For example for office building, they collect the fingerprinting data in a hallway placing 100 measurement points along a straight line where there is $1.8cm$ between each point and then using only FM signals, the authors get an average accuracy of $0.3cm$ and $30cm$ for 50% and 90% of the time respectively. FM and FM with Wi-Fi accuracies are close for office building hallway. For residential building, roomwise accuracy is used and 100% accuracy is obtained using FM only and FM with Wi-Fi signals whereas 90% accuracy is obtained using Wi-Fi signals only.

Yoon et al. (2016) [60] propose ACMI, an FM-based indoor localization system that does not require a site survey. They establish a signal propagation model using the floor plan of a building without the need to place beacons inside that building, as a result of extensive field measurement study for FM signals. The authors construct a fingerprinting map which purely contains FM RSSI info where signals are transmitted from commercial FM radio stations. Then, with the fingerprinting map in hand, ACMI applies *parameter calibration* and *path matching* methods at runtime. The authors make the experiments in seven campus locations and three downtown buildings where they achieve an average accuracy of $6m$ and $10m$ respectively.

Popleteev (2017) [61] performs a study to investigate the long-term behavior of FM signals indoor positioning methods. Since the author aims to investigate the FM signal stability for indoor positioning for a longer period of time than the most studies do as the author mentions, he collects FM RSSI samples for large-scale and small-scale areas for at least 3 months via *software-defined radio* (SDR) receivers. During the sample collection period, the author makes biweekly experiments to analyze the results where these experiments are called *sessions*. The author makes the experiments in three different areas:

- floor -2, 0 and 1 of a *office building* of size $100m \times 50m$ with a 9-month sampling period. 16, 36 and 33 tests points are used for floors -2, 0 and 1 respectively. 17 sessions are conducted.
- floor 0 and 1 of a *campus building* of size $80m \times 80m$ with a 9-month sampling period. 13 tests points are used for both of the floors. 16 sessions are conducted.

- floor 3 of a *apartment building* of size $14m \times 7m$ with a 3-month sampling period. 37 test point is used. 6 sessions are conducted.

The author chooses *support vector machine* (SVM) for the tests since it outperforms kNN and random forest classifiers according to his results in terms of classification accuracy. For evaluation, the author uses leave-one-session-out approach where a measurement session is chosen to test and the other sessions are used to train the proposed indoor positioning system. The author calculates localization classification errors for every fingerprint position in each test. For *apartment building* test, 57.4% of the test locations classified correctly where 90% of the estimates are within $4.1m$ from the actual position of the POI. For campus test, 94.3% and 92.0% classification accuracy is obtained for floor 0 and 1 respectively. For office building test, 60.4%, 79.3% and 89.6% classification accuracies are obtained for floor -2, 0 and 1 respectively. As to time duration effect on the localization error; for instance, for the campus test, the author observes that localization error increases from 70% to 96% for results after two sessions and eight sessions respectively which shows that FM signals can provide robust results for months after installation.

In Table 3.5, we summarize the related indoor positioning studies that use *frequency modulation* (FM) radio signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used.

Table 3.5. FM Indoor Positioning Related Studies

Year	Accuracy	Test Area	Method and Reference
2012	$0.3cm$ / $30cm$ with 50% / 90%	Office building	PHY modification, FM fingerprinting [59]
2016	$6m$ / $10m$	7 campus point / 3 buildings	FM fingerprinting, parameter matching [60]
2017	Cellwise with 57.4%	$14m \times 7m$	FM fingerprinting, SVM [61]

3.2 Bluetooth Solutions For Indoor Positioning

In this section, we mention studies using Bluetooth low energy signals for indoor positioning. We explain the environmental setup, applied methods and accuracy of these studies. We investigate the studies chronologically by starting with older studies

and concluding with recent study explanations. Having known that BLE is released as a part of Bluetooth technology on June 2010 , we only look at studies made after 2010, mostly focusing on the studies published after 2016 to compare our solution with the recent or state-of-the-art solutions.

Subhan et al. (2011) [62] follow an RSSI based approach and estimate the position by trilateration methods. To improve the accuracy, they use a method called the **gradient filter**. Using the gradient filter, the authors observe that the average error drops from $5.87m$ to $2.67m$ overall and 45% of this drop comes from the gradient filter method they applied. But, since this study is made in 2011, having the advantage of improved BLE technology, now we should get better accuracies.

Khan (2014) [63] makes a comparative study of existing indoor positioning methods using BLE. The aim of the author is to find an efficient and low computationally cost solution for positioning in an indoor environment, which is also the main aim of ours for writing this thesis. As a result of comparing lateration based indoor positioning approaches like *trilateration*, *least square* based approach and *min-max* based positioning algorithm; the author finds out that *min-max* based approach is better for accuracy and its low cost. According to the article, *min-max* algorithm functions by constructing a bounding box for each anchor node based on the distance. The authors mean signal receiving devices by the term **Anchor node**. Then three boxes are constructed according to distance information and the intersection of these boxes determine the resulting position.

Kalbandhe and Patil (2016) [64] indicate that positioning with BLE is more accurate than positioning with Wi-Fi. They use mobile phones as the receiver devices and use a mobile application called *IPSAPP*. They indicate that they use $-76dBm$ TX power for experiments; however they actually use $-76dBm$ as the *RSSI at 1 meter* instead of TX Power. But, we sometimes see that, in literature, *RSSI at 1 meter* may be used as a synonym to TX Power. They use two different beacons and obtain fourteen *distance vs RSSI* results for both of the beacons where all of the distances used in measurements are between $0m$ and $3m$. The authors indicate that they are able to classify signal sources correctly for *near*, *immediate* and *far* positions where these positions represent $0 - 0.5m$, $0.5 - 3m$ and above $3m$ respectively.

Kriz et al. (2016) [65] follow a hybrid approach and share their results . Using only four Wi-Fi access points, they get $1m$ accuracy in average and adding BLE beacons up to seventeen increases the accuracy from $1m$ to $0.77m$ in an area of $52m \times 43m$. The authors indicate that they exclude outliers from the results. Using only Wi-Fi beacons, the max error that they get is $4.27m$ and this result improves to be $2.82m$ using seventeen BLE beacons. However, BLE beacons that they use, transmit at a frequency of $100ms$ which is pretty frequent which make this solution only suitable where BLE beacon batteries can be replaced often.

Ozer and John (2016) [66] try to improve the accuracy of BLE solutions using Kalman filtering. They indicate that their resulting accuracy is in the order of meters. They use 12 BLE beacons each having $4dBm$ of TX power which is one of the highest TX power levels we have seen for a BLE beacon, in general. They indicate that the scan period of the Android application that they used is $600ms$. If the transmission period of their beacons is also $600ms$, then this solution would not be applicable for solutions where frequent battery replacement is not preferable. The test area is a section of the second floor of Applied Engineering and Technology Building at the University of Texas at San Antonio Main Campus.

Memon et al. (2017) [67] implement a mobile indoor positioning system to track staff members of a university department. The authors make the experiments in a lab environment and split the area in blocks where they do not indicate the size of the blocks. The accuracy in terms of blocks is 94% when the transmission period of the BLE beacons is $1000ms$ and is 100% when transmission period is $500ms$. The range of the beacons are $15m$ according to the paper. Hence, we guess TX power of the beacons is something around $-8dBm$. If each block is $1m^2$ in the figure they have, then the experiments are done in an area of $30m^2$ with a corridor partially spllitting the room and 4 BLE beacons are used where there is a clean line of sight among only one group of three BLE beacons.

Sie and Kuo (2017) [68] use BLE beacons transmitting with a period of $100ms$. The authors perform different experiments with two TX power levels, $-30dBm$ and $-42dBm$. They collect RSSI values every $50cm$. According to results they get, $-42dBm$ can only be used within $0-50cm$ but with low accuracy and with $-30dBm$

they can get accurate measurements upto $1.5m$. They indicate that with BLE beacon accuracies smaller than $1m$ is enough using small TX powers. However, we think that making measurements every $50cm$ is not affordable and a good solution in a real world environment.

Nguyen et al. (2017) [69] use BLE beacons using the iBeacon protocol with TX power $4dBm$ and advertising interval (transmission period) of $400ms$. They use several filtering methods for preprocessing the RSSI values they collect, where some of these filtering methods are Gaussian filter, Kalman filter and simply averaging RSSI values collected. They use an improved least square method for positioning and compare this method with other techniques like *least square estimation* (LSE) and trilateration-centroid in 2D. They perform the experiments in a square area of $5x5 = 25m^2$ where 4 BLE beacons are placed at the corners at a height of $1.2m$. The authors get an accuracy of $0.192m$ using their improved LSE methods where classic LSE methods result in $0.333m$ and trilateration-weighted centroid method result in $0.375m$ error with higher maximum error values.

Radoi et al. (2017) [70] perform some experiments in an office environment. The authors analyze two methods: fingerprinting and particle filtering. According to the experiment results, particle filtering performs better than the fingerprinting method for a corridor of size $1.5m \times 12m$, with an accuracy of less than $2m$ with 80% confidence interval. For a room of size $8m \times 6m$, for errors lower than $2m$; fingerprinting method performs better. For errors higher than $2m$, particle filter performs better than fingerprinting. Particle filtering get an error of $4m$ at maximum, 97.5% of the time, for the room case. Hence, we can say that particle filtering performs better in general, for this study.

Teran et al. (2017) [71] implement a solution based on *internet of things* (IoT). The experimentation area is $5 \times 5 = 25m^2$ which is further divided by $1 \times 1m^2$ blocks for accuracy estimations. They get an accuracy of $2m$ using two machine learning classifiers.

Paterna et al. (2017) [72] use different channels of BLE technology to lower the effect of signal degrading phenomena like reflection, refraction and loss of BLE signals due to external objects or signals. The authors make the experiments in three different

areas with sizes $4.8m \times 6m$, $9.19m \times 6.18m$ and $17.6m \times 16.5m$ respectively and get accuracies of $2m$, $1.82m$ and $4.6m$ with 90% confidence interval. According to layout of the area with size $9.19m \times 6.18m$, this area is indeed a L-shaped room with size $2.79m \times 3m + 8.72m \times 6.18m$. This article does a very good job of summarizing the related works in BLE indoor positioning field in the first table of the article [72]. In this table, the authors summarize 20 different studies by giving information about precision of the previous studies, indoor size where the solution works and the methods used. The best accuracy in this table belongs to a Japanese study by Kajioka et al. (2014) [73]. In this Japanese study, $0.8m$ accuracy was obtained in an indoor environment of size $10.5m \times 15.6m$. However, among these 20 different studies, for large-size environments; even to obtain accuracies worse than $1.5m$, at least 9 receivers are used. This shows that sub-meter accuracies or accuracies close to a meter are unlikely without using several receivers.

Zue et al. (2018) [74] propose a graph optimization based approach which combines fingerprinting-based methods and range-based methods. The authors perform the test in an area of $90m \times 37m$ with two different number of beacons which are 24 beacons (sparse) and 48 beacons (dense) to see the effect of beacon density on positioning estimations. With sparse beacon environment and dense beacon environment, they get accuracies of $2.26m$ and $1.27m$ respectively.

Yanagaimoto et al. (2018) [75] propose four unsupervised methods for positioning with two assumptions. These assumptions are low RSSI values are not reliable and human motion is slow enough for tracking. They exclude the signal values lower than $-90dBm$ to have relatively more reliable signals only and they use a voting based algorithm to make an assumption for the missing values in the dataset they use. This dataset consists of RSSI data collected in an academic international conference. However, since they do not have any mechanism to know where a participant really is, they do not have the ground truth position values to make error calculations. Therefore this study is more of a theoretical study than a practical one.

Teran and Carrillo (2018) [76] implemented a hybrid system for *wireless local area network* (WLAN) and BLE positioning using ML cloud services. The authors compare the hybrid method with Wi-Fi only and BLE only. For positioning SVM and

KNN classifiers are used. They get better accuries using Wi-Fi and BLE together, 75% of the time, for sub-meter ranges using KNN. However using only BLE, they get a sub-meter accuracy, 68% of the time using KNN and 60% of the time using SVM. They indicate that average accuracy is slightly above $2m$ using the hybrid approach. They make the experiments in a $8m \times 8m$ area where 4 beacons are placed at the corners, $1m$ above the ground with a transmission period of $1000ms$. They use 5 strongest Wi-Fi signals along with BLE signals at the campus (in measurement area) where the higher the *signal-to-noise ratio* (SNR) of a Wi-Fi signal, the stronger the signal is.

Li and Ma (2018) [77] use BLE tags and BLE/Wi-Fi repeaters. Their system makes positioning calculations using two values: ***RSSI fingerprint*** and *cell of origin* (CoO). They perform experiments in a rest area and an office area. They get accuracies of $1.2m$ for the resting area and $1.37m$ for the office area. The transmission period of the beacons and TX power are $400ms$ and $4dBm$ respectively. The rest area is of size $4.8m \times 14.4m$ and covered by four *access points* (APs) placed $2.4m$ above the ground. The office is of size $8.4m \times 15m$ and covered by four APs.

Huang et al. (2019) [78] implement a hybrid method for a dense BLE environment. The authors use sliding window filtering, trilateration, *dead reckoning* (DR) and ***Kalman filter*** (KF) for an improved performance of BLE positioning. They try to come up with a solution for environments where a lot of Bluetooth sources exist where these Bluetooth sources could be *Bluetooth classic* devices instead of BLE. Since other Bluetooth devices would cause interference, they overcome these interferences with their methods. They call their method as hybrid due to fusing dead reckoning and trilateration methods. The experiments are performed in an area of $5.6m \times 8.8m$. Yet, they place the beacons in a rectangular area of $5.6m \times 4.8m$ according to the layout given in the paper. In this test area, there are eighteen Bluetooth lights incessantly transmitting Bluetooth signals and eight BLE beacons at the ceiling at a height of $2.7m$. The authors apply three methods: trilateration, dead reckoning and hybrid method. Trilateration, dead reckoning and hybrid methods has *root mean squared errors* (RMSE) of $2.33m$, $0.82m$ and $0.76m$ respectively. So, Hybrid method outperforms the other two methods.

Mekki et al. (2019) [79] implement an IOT based solution. The authors give some path loss exponents for different environments like free space and obstructions in buildings and factories since in some environments signal strength loss is higher than the others. They use beacons with different TX powers ranging from $-30dBm$ to $4dBm$. They perform the experiments in an office area of $6 \times 6m^2$ where obstacles like cubicles, bookcases and computers exist. They get an error of $1.5m$.

In Table 3.6, we summarize the related indoor positioning studies that use *Bluetooth low energy* (BLE) signals by giving information about year, accuracy, test area size ($width \times height$) and the methods used. This table is sorted by the accuracies of the studies. In this table green color represents our algorithm whereas red color represents the studies where the test area is not well-defined. Our solution seems to be close to the middle. Hence, we can say that, our solution provides similar accuracies compared to the studies mentioned in the literature. In Table 3.6 and 3.7, **NA** values mean not available. **1D** refers to distance information indicating how many meters the authors are away from the receivers at maximum.

Table 3.6. BLE Indoor Positioning Related Studies

Year	Accuracy (m)	Test Area (m^2)	Method and Reference
2017	0.192	5×5	Kalman / Gaussian filter, trilateration-centroid [69]
2019	0.76	5.6×8.8	Hybrid, sliding window + KF, trilateration, DR [78]
2014	0.8	10.5×15.6	BLE fingerprinting [73]
2018	1.0 with 75%	8×8	Hybrid (Wi-Fi+BLE), SVM, KNN, ML cloud service [76]
2018	1.27	90×37	Graph optimization, fingerprinting + range based [74]
2018	1.37	8.4×15	Hybrid (Wi-Fi + BLE), fingerprinting, CoO [77]
2017	$\sqrt{2}$ with 96%	$30m^2$	Selection of nearest BLE tag [67]
2019	1.5	6×6	Trilateration, RSSI filtering [79]
2019	Mean: 2.29, Std: 1.67	15×16	IP-PPFONM
2011	2.67	10×12	Trilateration, gradient filter [62]
2017	$2\sqrt{2}$ with 70.2%	5×5	Two machine learning classifiers [71]
2014	2.89	12×14	Trilateration, min-max, least square [63]
2017	4.0 with 97.5%	8×6	Particle filter, BLE fingerprinting [70]
2017	4.6 with 90%	16.5×17.6	Channel diversity, weighted trilateration and KF [72]
2016	A few meters	University room	Kalman filter [66]
2017	1.5 with 91.4%	3 (1D)	Low TX powered beacon utilization [68]
2016	0.5	3 (1D)	IPSAPP (A mobile application) [73]
2018	NA	Conference room	Voting based algo [75]

Table 3.7. BLE Indoor Positioning Related Studies Deployment Requirements

Year	Deployment Requirements	Extra Information
2017	3 beacons with 4dBm,400ms	4 beacons with 4 receivers fingerprinting | Static
2019	8 fixed beacons	NA
2014	22 fixed beacons	NA
2018	4 fixed beacons, 5 Wi-Fi AP	NA
2018	48 fixed beacons	Dynamic
2018	Some BLE tags and 4 BLE/Wi-Fi Repeater	$400ms$ period, $4dBm$ TX power | Static
2017	4 fixed beacons	Areawise accuracy ($1m^2$)
2019	9 fixed beacons	Dynamic (move with a mobile device)
2019	1 beacon, 3 receivers	Obstructions considered | Dynamic
2011	NA	NA
2017	4 fixed beacons, 1 receiver	Areawise accuracy ($4m^2$) | Fingerprinting
2014	4 fixed beacons	4 fixed target node position
2017	5 beacons	$300ms$ period, $-4dBm$ TX power
2017	4 receivers, 1 beacon	$100ms$ period | Cost-effective
2016	12 fixed beacons	Android app used to track
2017	1 beacon	$100ms$ period | $-30dBm/-42dBm$ TX power
2016	1 beacon	-76dBm TX Power at 1 meter
2018	NA	NA

Table 3.7 has the same entries in the same order as in Table 3.6 where last two columns are replaced with different ones and the *Accuracy* column is removed. In Table 3.7, hardwares that the other studies use are placed in the column named *Deployment Requirements*. We can see that our solution uses less equipment than most of the other approaches and our test area is larger than half of the other approaches. Moreover, our method is unique in the way it handles the obstructions in the indoor environment compared to the other methods.

In Table 3.6, there are cellwise accuracies besides positioning accuracies. To include cellwise accuracies in the table, we consider the diagonal lengths of the cells as the accuracies since the diagonal length represents the longest distance between two points in the corresponding cell.

In Table 3.7, in *Extra Information* column, static tracking means this study positions a static object whereas dynamic tracking means this study tracks a moving object or person. We also added the beacon related parameters like TX power and period in this column.

CHAPTER 4

INDOOR POSITIONING VIA PREFILTERING AND PARTICLE FILTERING WITH OBSTRUCTION-AWARE NO-SIGNAL MULTILATERATION

In this chapter, we make an overview of how *indoor positioning via prefiltering and particle filtering with obstruction-aware no-signal multilateration* (IP-PPFONM) algorithm works and then explain IP-PPFONM steps in detail with the help of pseudocodes and some formulas.

The organization of this chapter can be summarized as follows:

- In Section 4.1, we explain the main algorithm with the pseudocodes. At first, we define some constants and variables to be used in different parts of the algorithm.
- In Section 4.2 section, we explain how to place receivers in the IE so that as much coverage area as possible is achieved.
- In Section 4.3, we explain how BLE fingerprinting map is constructed prior to execution of the IP-PPFONM algorithm. We use this fingerprinting map data in our multilateration algorithm to locate the POI.
- In Section 4.4, we explain how we prefilter the *received signal strength indicator* (RSSI) values received by the receivers to make sure the signals we receive are strong enough for positioning calculations. This step is necessary since low RSSI values are not reliable for positioning estimations.
- In Section 4.5; based on the prefiltered RSSI values, the indoor environment layout and obstruction information, we calculate the POI position, which we call the *measured position* of the POI.
- In Section 4.6, particle filtering steps are explained. This step calculates the

predicted position of the POI using *measured position*, for the current time step and provides the smooth transitions between each positioning estimation. If we used only multilateration step for positioning, due to fluctuating nature of the RSSI values, we would get positioning estimations apart from each other for consecutive time steps as if the *person of interest* (POI) teleports from one place to another inside the IE. This section and the previous sections explain and conclude the main steps of the IP-PPFONM algorithm.

- In Section 4.7, we explain some common functions used by different components of the IP-PPFONM algorithm.
- In Section 4.8 section, we explain some of the use cases of the IP-PPFONM algorithm.

4.1 Main Algorithm

IP-PPFONM is a real-time algorithm and therefore, we make calculations periodically. The period of the calculations are chosen to be 1 second ($1s$) since our beacons transmit a signal per second. We call each step we make calculations a *time step*, each of which has a time difference of $1s$ in between. We make some distance calculations for positioning and all of the distances in IP-PPFONM are calculated according to *Euclidean distance* metric.

In IP-PPFONM, we firstly create a fingerprinting map for the IE by keeping the beacons stay in the *indoor environment* (IE) for a while. Then, we **prefilter** each signal arriving at the receivers via *running average filter* (RAF) algorithm. After that, we use *obstruction-aware no-signal multilateration* (ONM) algorithm to find the measured position of the POI. Finally, we apply *particle filtering* on the arriving signals to locate the POI with some confidence value.

To be able to adjust different environmental conditions and different situations that may be encountered in an IE, we follow a flexible approach for the IP-PPFONM algorithm by having parameters that can be changed for various purposes. The main constant parameters are:

- **minUsefulSignal:** This parameter represents the minimum RSSI value to be

used for our particle filtering algorithm calculations.

- **minSignalValue:** This parameter represents the minimum RSSI value that can be caught by a receiver device. We do not prefilter RSSI values lower than *minSignalValue* and directly reject it since we do not expect such a low signal to be transmitted from our signal transmitting devices (beacons). *minSignalValue* is always less than or equal to *minUsefulSignal*.

- **maxSignalError:** This parameter represents the maximum signal noise possible in the IE. Hence, the signal noise in the indoor environment for algorithm is always between $0dBm$ and *maxSignalError*.

- **sensitivityOfResult:** This parameter represents what should be the distance between each checkpoint for the minimization function used in our multilateration algorithm.

- **numberOfBlocks:** This parameter represents the number of objects that may affect the BLE signals in the IE. We know that objects causes some signal loss when an EM signal hits them. Since there could be many obstructions in an indoor environment, we only consider the large objects that my affect the BLE signals as a simplification.

- **numberOfRooms:** This parameter represents the number of rooms in the IE. We know if rooms exist in an IE, due to having walls around them, they cause some signal loss when an EM signal hits them.

- **blockWidths, blockLengths, roomWidths, roomLengths**: *blockWidths* and *blockLengths* parameters represent the widths and lengths of the blocks in the IE in meters, respectively whereas *roomWidths* and *roomLengths* parameters represent the widths and lengths of the rooms in the IE in meters, respectively.

- **blockMaterials, roomMaterials:** Besides the width and length of the blocks and rooms, the material used to make these blocks or rooms should be taken into account since different materials have different *electromagnetic* (EM) signal transmission, absorption and reflection values [38]. Therefore, we include these parameters in the IP-PPFONM algorithm. *blockMaterials* and *roomMaterials* represent the material of the blocks and rooms respectively. Each of

these parameter can contain multiple materials to indicate different materials for each block or room.

- **materialSignalDisturbanceCoefficients:** This parameter represents the signal disturbance coefficient of the materials as key and value pairs like a hash table data structure. Key and value pairs are represented as $key : value$ where keys are materials and values are the signal disturbances for the corresponding material. Signal disturbance here represents how much dBm of the signal is lost for the corresponding material of $1m$ width.

- **pastCoeff:** The name of this parameter is abbreviation for *past coefficient*. This parameter represents how much of the previous velocity should be used for determining the current predicted velocity of the POI . We use this coefficient parameter to take human motion into account. Since humans go along in a direction for some time, they make a similar movement for most of the time. For the other times, humans stop and turn into another direction. Hence, thinking that POI will make the same movement as the previous motion would be more probable than not.

  Past coefficent value changes between 0 and 1. 0 means only consider the current motion. 1 means only consider the previous motion. The values between 0 and 1 indicate how much of the previous motion we should take into account for the magnitude and direction of the current POI motion. This past coefficient value is ignored for the first motion since there is no previous motion before the first one.

- **NumberOfParticles:** This parameter represents how many particles we should use for the particle filtering algorithm.

- **xmin, xmax, ymin, ymax:** *xmin* and *xmax* represent minimum and maximum x coordinates in the IE whereas *ymin* and *ymax* represent minimum and maximum y coordinates in the IE. Hence, the indoor environment is a map whose coordinates are defined by $(xmin,\ xmax) \times (ymin,\ ymax)$.

- **movingLimit:** This parameter represents how many meters at most should the POI be able to move for x and y dimensions, at each time step. In this thesis, since each time step is assumed to be 1 second, this parameter represents the

maximum velocity for the POI in m/s. For example, to track people driving electric cars in an airport, this parameter can be chosen as $20m/s$.

- **FPbeaconPos**: This parameter is an abbreviation for *fingerprinting beacon positions*. It represents the positions of the beacons we keep in the IE for a while to collect RSSI values for fingerprinting.

- **FPcoeff:** This parameter is an abbreviation for *fingerprinting coefficient*. It represents a multiplier determining the effect of fingerprinting map in the IPPPFONM. This parameter can take any positive value. The higher this parameter is the larger the fingerprinting map affects the multilateration algorithm results. If the value of this parameter is greater than 1, then fingerprinting results would suppress the other calculations we do for minimization.

- **numberOfReceivers:** This parameter represents how many receiver devices should be placed in the map.

- **receiverPositions:** This parameter represents the 2D positions of the receivers installed in the IE.

- **TXPower:** This parameter is related to beacons. Beacons emit signal with a certain power. This power is represented as *milliwatt* (mW) normally; yet, in distance calculations, usage of *decibel-milliwatts* (dBm) is more common. Therefore this parameter represent how much power the beacon transmits the signal in terms of dBm.

- **RSSIatOneMeter:** This beacon-related parameter is usually stated in the website of the firm where beacons are bought from. If not stated, one can measure the RSSI value at a receiver which is 1 meter away from a beacon as this value. We choose *RSSI at 1 Meter* value equal to $TXPower - 65$ since we generally see that this value is $(TXPower - 65) \pm 3$ according to different beacon firms. For TX Power values lower than $-20dBm$, like $-30dBm$ and $-40dBm$, *RSSI at 1 Meter* is generally less than $(TXPower - 65) \pm 3$. Since it is relatively easy to measure signal power at $1m$, this value is used as a reference to make calculations further than $1m$. For distances closer than $1m$, we calculate the RSSI as an exponential function whose resulting value change between *TX Power* and *RSSI at 1 meter* between distances $0m$ and $1m$ respectively.

Having finished the explanation of the parameters that we use, for better understanding of the IP-PPFONM algorithm, we present some pseudocodes. In Algorithm 1, we firstly define some global constant and variables which can be accessed by all of our functions if necessary. Then, we use our optimum receiver positioning algorithm if we do not prefer to determine receiver positions manually by ourselves. After that, our algorithm starts its dynamic phase which is a set of procedures and algorithms for positioning of the POI according to the RSSI values received at each time step.

Algorithm 1 IP-PPFONM Main Procedure

```
1:  procedure LocatePOI
2:      DefineGlobalConstants()
3:      DefineGlobalVariables()
4:      if receiverPositions is not determined manually then
5:          receiverPositions ← GetReceiverPositionsToInstall()
6:      end if
7:      for each Time Step do                                   ▷ Main Loop
8:          RSSIatEachRec ← GetRSSIatEachReceiver()
9:          prefilteredRSSIs ← Prefilter(RSSIatEachRec)
10:         MeasuredPOIPos ← Multilateration(prefilteredRSSIs)
11:         PredictedPosOfThePOI ← ParticleFilter()
12:     end for
13: end procedure
```

Explanations and Comments For Algorithm 1:

- *DefineGlobalConstants* function initializes the constants mentioned in Table 4.1.
- *DefineGlobalVariables* function initializes the variables to be used throghout the IP-PPFONM algorithm.
- *Main Loop*, refers to the dynamic phase of the IP-PPFONM algorithm which is executed for each time step.
- RSSIatEachRec is a list of raw RSSIs caught by each receiver device. To access the RSSI at $receiver_i$, we use the access operator ([]) like this: *RSSIatEachRec[i]*.
- For each time step, using *Prefilter*,we filter out the incoming signals. Then, using *Multilateration*, we measure the POI position. After that, we apply *ParticleFilter* on the measured position to locate the POI in a confidence region

calculated using a probability distribution of the particles.

- *PredictedPosOfThePOI* represents the positioning result of the IP-PPFONM algorithm for the current time step.

Table 4.1. Global Constant Parameter Definitions

Constant Name	Default Value	Unit, if available
minUsefulSignal	-90	dBm
minSignalValue	-100	dBm
maxSignalError	5	dBm
sensitivityOfResult	0.5	meter
numberOfBlocks	1	NA
numberOfRooms	3	NA
blockWidths	0.5	meter
blockLengths	0.5	meter
roomWidths	[5.3, 4,5]	x, y coordinate (meter)
roomLengths	[11, 3,3]	x, y coordinate (meter)
blockMaterials	plastic	NA
roomMaterials	concrete	NA
materialSignalDisturbanceCoefficients	$[concrete : 16dBm, glass : 6dBm]$	Material and dBm pairs.
pastCoeff	0.2	NA
NumberOfParticles	300	NA
xmin	0	x coordinate (meter)
xmax	15	x coordinate (meter)
ymin	0	y coordinate (meter)
ymax	16	y coordinate (meter)
movingLimit	5	meter
FPbeaconPos	[(0.25, 2.25),(5, 6), (11.5, 3.5), (12.5, 9)]	x,y coordinates (meter)
FPcoeff	0.2	NA
numberOfReceivers	3	NA
receiverPositions	[(5,7),(10.3,8),(7.4,2)]	x,y coordinates (meters)
TXPower	0	dBm
RSSIatOneMeter	-65	dBm

We also have variables whose values change at each iteration like *ParticleWeights*, *ParticlePositions*, *MeasuredPOIPos*, *prefilteredRSSIs* and *slidingWindows* and values initialized at the beginning like *xElems*, *yElems* and *RSSIinFP* as shown in Algorithm 2.

Algorithm 2 Variables Used In Algorithms

procedure DEFINEGLOBALVARIABLES
    $ParticleWeights \leftarrow EmptyQueue$ (of size NumberOfParticles)
    $ParticlePositions \leftarrow EmptyQueue$ (of size NumberOfParticles)
    $RSSIinFP \leftarrow$ CREATEFPMAP()
    $xElems \leftarrow EmptyQueue$
    $yElems \leftarrow EmptyQueue$
    for $tmpElem \leftarrow xmin; tmpElem < xmax + sensitivityOfResult, tmpElem \leftarrow tmpElem + sensitivityOfResult$ **do**
        Add tmpElem into xElems
    end for
    for $tmpElem \leftarrow ymin; tmpElem < ymax + sensitivityOfResult, tmpElem \leftarrow tmpElem + sensitivityOfResult$ **do**
        Add tmpElem into yElems
    end for
    $MeasuredPOIPos \leftarrow None$ ▷ Initially no measurement is made
    ▷ Variables below are related to the receivers. Each index is for a receiver
    $prefilteredRSSIs \leftarrow EmptyQueue$
    ▷ Initialize parameters for each receiver
    for each $i \leftarrow 0; i < numberOfReceivers; i \leftarrow i + 1$ **do**
        $slidingWindows[i] \leftarrow EmptyQueue$(of size 7) ▷ Each sliding window is a queue
    end for
end procedure

Explanations and Comments For Algorithm 2:

- *xElems* contains x coordinate of each 2D position in the IE which is separated by sensitivityOfResult meter from the closest x coordinate in *xElems*.
- *yElems* contains y coordinate of each 2D position in the IE which is separated by sensitivityOfResult meter from the closest y coordinate in *yElems*.
- Values inside the brackets are independent from each other and each entry inside brackets can be indexed by [] operator. Pairs inside parantheses like $(a,\ b)$ are x and y coordinates of a 2D position and to reach first (a) and second (b) coordinates , we can say $(a,\ b).x$ and $(a,\ b).y$ respectively.
- *RSSIinFP* is going to be filled in *createFPMap* function defined in Section 4.3.

4.2 Efficient Receiver Placement

We have to install our receiver devices as much efficient as possible given the indoor map. Efficiency here means positioning the least amount of receiver devices by still managing to receive signals regardless of where the POI is inside the IE. Placing lots of receiver devices does not contribute to locating the POI more accurately if all devices installed close to each other. We apply our receiver positioning algorithm on rectangular shaped indoor environments as it is the most commonly found shape for an indoor store. For combined or complex shapes, human eye would be more accurate for efficient positioning.

We consider the positioning problem as a clustering problem where each cluster consists of points formed around a receiver. Therefore, finding cluster centers is equal to finding the receiver devices. We need evenly spaced points spread across the whole indoor environment map and get cluster centers in the map almost evenly positioned. Therefore, we create evenly-spaced initial points spread across the map which are to be used by our *k-means* algorithm as the initial data points to be clustered.

In our k-means algorithm, *k* is the number of receivers we want to place in the IE. We run *k-means* algorithm with different initialization. Using a hundred different runs, *k-means* chooses the best clusters and their centers. As a result, we use the centers of these clusters as the receiver positions. We can see the receiver installation implementation in Algorithm 3.

Algorithm 3 Placement of the Receivers in the IE in the most efficient way

```
1: procedure GETRECEIVERPOSITIONSTOINSTALL
2:     initialPoints ← EmptyQueue
3:     for xIndex ← 0 ; xIndex < 1 + stepSize; xIndex ← xIndex + stepSize do
4:         for yIndex ← 0 ; yIndex < 1 + stepSize; yIndex ← yIndex + stepSize do
5:             Add (xIndex,yIndex) pair into initialPoints queue
6:         end for
7:     end for
8:     stepSize ← 1/(ROUNDUP(SQRT(numberOfReceivers * 1000)))
9:     receiverPositions ← KMEANS(numberOfClusters, NumberOfRuns, initialPoints)
```

10: **for each** $receiverPosition \in receiverPositions$ **do**
11: $receiverPosition.x \leftarrow xmin + receiverPosition.x * (xmax - xmin)$
12: $receiverPosition.y \leftarrow ymin + receiverPosition.y * (ymax - ymin)$
13: **end for**
14: **return** receiverPositions
15: **end procedure**

Explanations and Comments For Algorithm 3:

- *roundUp* function rounds the resulting floating point number to the closest integer larger than itself.
- *sqrt* function takes the square root of the given parameter.
- *kMeans* returns *numberOfClusters* of cluster centers given points to be clustered where these points are represented by *initialPoints* variable.
- *NumberOfRuns* represents how many times *k-means* algorithm should run to choose the best result among these runs.
- *initialPoints* represents the initial points in a unit square that we want to cluster. These points are evenly spaced in both x and y dimensions. The space between each initial point is *stepSize* unit where *stepSize* is a value between 0 and 1.
- Since our initial points in in a unit square, the k-means algorithm gives result in a unit square. Hence, after *k-means* algorithm is completed, we map receiver positions found in $(0,\ 1) \times (0,\ 1)$ coordinates to indoor environment's coordinates, that is, $(xmin,\ xmax) \times (ymin,\ ymax)$.

4.3 BLE Fingerprinting

First of all, we create a BLE *fingerprint* (FP) map by interpolating the signal values at the positions where we place our fingerprint beacons for a while. For each position $(x,\ y)$, we look at the closest beacon to $(x,\ y)$. We calculate the distance of the closest beacon to all receivers and the distance of $(x,\ y)$ to all receivers. Then, using the ratio between these distances, we calculate a signal strength value at position $(x,\ y)$. We expect that if $(x,\ y)$ is closer to receiver than its closest beacon is, then $(x,\ y)$ should have a higher RSSI value compared to the RSSI fingerprint value at closest beacon's position and vice versa. For distances between 0 and 1 and corresponding

RSSI values for these distances - RSSI values higher than *RSSIatOneMeter*- instead of making calculations according to closest beacon, we calculate the fingerprinting values according to the distance of $(x,\ y)$ to the closest receiver. In Algorithm 4 and 5, we can see the implementation of how we construct the *fingerprinting* (FP) map.

Algorithm 4 Preparation of the BLE Fingerprinting Map

```
1:  procedure CREATEFPMAP
2:      RSSIinFP ← EmptyHashTable
3:      allPosDistancesToReceivers ← EmptyHashTable
4:      for i ← 0; i < numberOfReceivers; i ← i + 1 do
5:          for each x in xElems do
6:              for each y in yElems do
7:                  allPosDistancesToReceivers[i, x, y] ← EUCLIDEANDIST(receiverPositions[i], (x, y))
8:              end for
9:          end for
10:     end for
11:     allPosDistancesToBeacons ← EmptyHashTable
12:     for k ← 0; k < numberOfBeacons; k ← k + 1 do
13:         for each x in xElems do
14:             for each y in yElems do
15:                 allPosDistancesToBeacons[k, x, y] ← EUCLIDEANDIST(FPbeaconPos[k],(x, y))
16:             end for
17:         end for
18:     end for
19:     for i ← 0; i < numberOfReceivers; i ← i + 1 do
20:         for each x in xElems do
21:             for each y in yElems do
22:                 minDist ← ∞
23:                 mink ← 0
24:                 for k ← 0; k < numberOfBeacons; k ← k + 1 do
25:                     if allPosDistancesToBeacons[k, x, y] < minDist then
26:                         mink ← k
27:                         minDist ← allPosDistancesToBeacons[k, x, y]
28:                     end if
29:                 end for
30:                 baseDist ← EUCLIDEANDIST(FPbeaconPos[mink], receiverPositions[i])
31:                 targetDist ← allPosDistancesToReceivers[i, x, y]
32:                 baseRSSI ← PositionedBeaconRSSIinFP[i][mink]
33:                 RSSIinFP[i, x, y] ← CALCRELATIVERSSI(baseDist, targetDist, baseRSSI)
34:             end for
```

```
        end for
    end for
    return RSSIinFP
end procedure
```

Algorithm 5 Interpolation of Signal Strengths According to Relative Distances

```
function CALCRELATIVERSSI(baseDist,targetDist,baseRSSI)
    if targetDist >= 1 then
        return baseRSSI + −20 ∗ log10 (targetDist)/(baseDist)
    else
        return ZEROONEMETERDISTTORSSI(targetDist)
    end if
end function
```

Explanations and Comments For Algorithm 4 and Algorithm 5:

- *allPosDistancesToReceivers[i,x,y]* represents the distance of $(x,\ y)$ to $receiver_i$ (ith receiver).
- *allPosDistancesToBeacons[k,x,y]* represents distance of $(x,\ y)$ to fingerprinting $beacon_k$ (kth beacon).
- *PositionedBeaconRSSIinFP[i][k]* is a hash table representing the averaged RSSI value at $receiver_i$ where the RSSI is transmitted by $beacon_k$. This hash table consist of the values given in Table 6.3.
- *mink* represent the index of the fingerprinting beacon which is closest to $(x,\ y)$.
- *calcRelativeRSSI* function calculates an RSSI value for position $(x,\ y)$ using relative distance of $(x,\ y)$ to $receiver_i$. Relativeness is according to the closest beacon position and RSSI value at that position.
- *baseRSSI* represents the RSSI at the point we take as reference to interpolate other points.
- *baseDist* represents the distance to the related receiver device of the point we take as reference.
- *targetDist* represents the distance to the point we want to interpolate.
- *RSSIinFP[i,x,y]* represents the interpolated RSSI for (x, y) coordinate of the IE where this RSSI belongs to $receiver_i$.

4.4 Prefiltering

We use an algorithm called *running average filtering* (RAF) for RSSI prefiltering. In this algorithm, each receiver device has its own sliding window storing upto seven most recent incoming signal strengths (RSSIs) and signal arriving times. Therefore, for prefiltering algorithm, we use at most seven RSSI information. For each receiver, we need at least three signals in our sliding window, so we wait for at least three signals to arrive at the receiver.

Upon the arrival of the third signal, we sort the signals in our sliding window and discard the signals having the minimum and maximum strength. Then, we take the mean of the rest of the signals in the sliding window. For a window of size three, this means that we take the median signal strength valued signal in the sliding window.

After all, we check if the mean of the RSSIs is higher than a given threshold. If the mean is lower than this threshold, even though this signal still resides in the sliding windows for further prefiltering calculations, we do not use this signal value for positioning estimations since we decide that this signal is not reliable. In Algorithm 6 and Algorithm 7, we can see the implementation for the prefiltering part.

Algorithm 6 Prefiltering Step

```
1:  procedure PREFILTER(RSSIatEachRec)
2:      for each i ← 0; i < numberOfReceivers; i ← i + 1 do
3:          if There are 7 signals in the slidingWindows[i] queue then
4:              Remove the oldest RSSI from slidingWindows[i]
5:              Add the new RSSI (RSSIatEachRec[i]) into slidingWindows[i]
6:          else
7:              Add the new RSSI (RSSIatEachRec[i]) into slidingWindows[i]
8:          end if
9:          if FILTERANDCHECKSIGNAL(slidingWindows[i]) = True then
10:             Add new RSSI into prefilteredRSSIs[i]
11:         else                          ▷ Add None as if the signal is not received at all
12:             Add None value into prefilteredRSSIs[i]
13:         end if
14:     end for
15: end procedure
```

Algorithm 7 Filter Out Signal

```
1: function FILTERANDCHECKSIGNAL(slidingWindow)
2:     mean ← 0
3:     sum ← 0
4:     if SIZE(slidingWindow) < 3 then
5:         return False
6:     else
7:         sortedWindow ← SORT(slidingWindow)
8:         sortedWindow ← REMOVEFIRSTANDLASTELEMENTS(sortedWindow)
9:     end if
10:    for each signalValue ∈ sortedWindow do
11:        sum ← sum + signalValue
12:    end for
13:    mean ← sum/ SIZE(sortedWindow)
14:    return mean >= minUsefulSignal
15: end function
```

Explanations and Comments For Algorithm 6 and Algorithm 7:

- $size(x)$ returns number elements in x.
- $sort(x)$ sorts elements of x in descending or ascending order.
- $RemoveFirstAndLastElements(x)$ remove the minimum and the maximum elements from x to remove outlier signals.

4.5 Obstruction-Aware No-Signal Multilateration

Using the prefiltered RSSIs, positions of the receivers, obstruction information and a loss function, we make some error minimization measurements about where the POI is. If no RSSI value is obtained for the current time step or RSSI of the current time step gets filtered out, we look at the RSSI values at time steps which are at most some predefined number of before and ahead of the current time step until we find a RSSI value. Since this is a real-time system, to obtain the next time step RSSI value, we wait and do not make any calculations for the current time step to locate the POI.

To be able to locate the POI with high accuracy, we need to know at least three

distance values of the POI to three different points in the map. If the signals are not noisy, using only three reference points locates the POI with 100% correctness using an algorithm called *trilateration*.

However, in real life, the signal values are noisy and the distance values to reference points are not known 100%. Hence, if available, using more than three reference points result in more accurate results since we would have more distance information in the IE map.

For obstruction-aware no-signal multilateration (ONM), the values we know are receiver positions and the RSSI values arriving at these receivers. Therefore, using RSSIs and the position of the receivers, we have to be able to predict the object location. When the POI moves in the IE, each receiver catches some signal where the RSSI value may differ in each of the receivers. Due to different RSSI values, we get different distance information to different receivers. We approximately know what the distance should be given the RSSI value. Due to the noise in the signal, in reality , the real distance values might be different from what the RSSI value tells us. Hence, we need to correct this RSSI value to extract the right distance information of the POI to the receiver device having this RSSI.

Since our problem is indoor positioning, the area we investigate is limited and therefore, exhaustive search for discrete amount of points separated by some value is applicable in terms of time. Therefore, we follow an exhaustive search approach where we test discrete amount of positions in the IE for the best guess. The positions to be tested are determined by a modifiable sensitivity parameter called *sensitivityOfResult* where this parameter represents the distance between each point we test for positioning. Due to this sensitivity parameter, our multilateration positioning results are discrete points in the IE. If we want more accurate results, we set this sensitivity parameter to a lower value to decrease the distance between each test point. However, there is a trade-off between more accurate results and the time.

To find the measured position given the prefiltered RSSIs, we check every discrete $(x,\ y)$ point separated by a distance of *sensitivityOfResult* meters in the IE and calculate a minimization function for these points. After checking all $(x,\ y)$ points, we choose the point minimizing the error function as our measured position. For ex-

ample, for an IE of size $1m \times 2m$ in x and y dimensions respectively; if we have *sensitivityOfResult* of $0.5m$. The points that we check would be: $[0, 0]$, $[0, 0.5]$, $[0, 1]$, $[0, 1.5]$, $[0, 2]$, $[1, 0]$, $[1, 0.5]$, $[1, 1]$, $[1, 1.5]$, $[1, 2]$ and we would choose a point among these points as the resulting measured position.

For each $(x,\ y)$ point, we act like the POI is on this point and **postfilter** the RSSI value accordingly. For postfiltering; if there is an obstruction between point $(x,\ y)$ and a receiver, we correct the RSSI value by increasing the RSSI by how much RSSI is lost due to this obstruction. We cannot know the exact signal loss amount caused by the obstruction; however we calculate some loss using the obstruction material and thickness information where the thickness is chosen as the thickness of the part of the obstruction which stays in the way between the receiver and point $(x,\ y)$. To determine the part of the obstruction in the way, we consider a linear signal motion from point $(x,\ y)$ to the receiver as a simplification and construct a imaginary line segment between point $(x,\ y)$ and the receiver. Then, we take the intersection amount between this line segment and the obstruction as the thickness amount of the object we intersect and calculate the signal loss due to this obstruction accordingly. We make this calculation for each obstruction in the way of $(x,\ y)$ and the receiver.

For a point $(x,\ y)$ and the receiver position $(rx,\ ry)$ in 2D coordinates, we call the distance between this point and the receiver *xyDistToRec* as an abbreviation for *xy (2D) distance to receiver*. For each time step, we get a RSSI value at receiver $(rx,\ ry)$. After applying prefiltering and postfiltering on this RSSI, we measure a distance value from the receiver to the POI where we call this distance *distToReceiverGivenRSSI*. Moreover, we know the fingerprinting RSSI on point $(x,\ y)$ since we create the fingerprinting map before multilateration algorithm. Using this fingerprinting RSSI, which we call *RSSIinFP*, we calculate a distance called *xyDistToRecInFP*. To choose the point $(x,\ y)$ as the point minimizing the loss function, we want *xyDistToRec* and *xyDistToRecInFP* to be as close as possible to the *distToReceiverGivenRSSI* distance value. In our minimization algorithm, we have two cases depending on the signal receiving situation of the receivers.

- **CASE1: Receiver catches a signal for the current time step**

  If *distToReceiverGivenRSSI* is small, then the POI must be close to the re-

ceiver since strong signals generally convey right information about the distance. Hence, we expect the *xyDistToRec* to be small as well if we are to choose (x, y) as the predicted position over the other checkpoints in multilateration.

If *distToReceiverGivenRSSI* is large, it can be either since the POI is distant to the receiver or there is an obstruction between the receiver and the POI causing RSSI value to be low. Since we do not rely on low RSSI signals, we do not punish as much as we do when *distToReceiverGivenRSSI* is small.

The punishment in this case is proportional to the absolute difference between *distToReceiverGivenRSSI* and *xyDistToRec* and disproportional to *distToReceiverGivenRSSI*. Being disproportional to *distToReceiverGivenRSSI* is what provides punishing more when *distToReceiverGivenRSSI* is small than it is large. Lastly, we take the square of the punishment term so that we punish large errors more.

To use the fingerprinting results in our minimization algorithm, we can replace *xyDistToRec* parameter with *xyDistToRecInFP* for this case . However, we only punish according to fingerprinting data if prefiltered RSSI value and *RSSIinFP* differ more than a threshold we determine.

- **CASE2: Receiver does not catch a signal for the current time step**

  If we have no signal value to make a calculation, the POI is most probably not close to the receiver. However, we know our lowest signal threshold (*minUsefulSignal*), and hence the maximum distance coverage (*maxCatchableSignalDistance*) of our receiver. If no signal is received at our receiver, the probability of the POI being more distant to the receiver than *maxCatchableSignalDistance* meters is more than the POI being in this coverage. Therefore, we punish the point $(x,\ y)$ if it is in the coverage area of the receiver. The punishment is disproportional to the *xyDistToRec* since the point $(x,\ y)$ should be as distant to the receiver as possible when in coverage area. Lastly, we take the square of the punishment term so that we punish large errors more.

  To use the fingerprinting results in our minimization algorithm, we can replace each *xyDistToRec* parameter with *xyDistToRecInFP* for this case.

We call the ONM algorithm *no-signal* since we make calculations considering all the

receiver positions even if some of the receivers do not get any signal. We can also make calculations even when none of the receivers get a signal if we know that the POI is inside the IE. We call the ONM algorithm *obstruction-aware* since we take obstruction information into account. In Algorithm 8, we can see the implementation of our multilateration algorithm.

Algorithm 8 Obstruction-Aware No-Signal Multilateration

function MULTILATERATION(prefilteredRSSIs)
  $mysum \leftarrow \infty$
  for $x \leftarrow 0 ; x < 1 + stepSize; x \leftarrow x + stepSize$ **do**
    for $y \leftarrow 0 ; y < 1 + stepSize; y \leftarrow y + stepSize$ **do**
      if $(x,\ y)$ on an obstacle **then**
        Skip to next iteration
      end if
      $tmpSum \leftarrow 0$
      for $i \leftarrow 0 ; i < numberOfReceivers; i \leftarrow i + 1$ **do**
        $strengtheningAmount \leftarrow$ GETSIGNALLOSSAMOUNT($receiverPositions[i]$,(x, y))
        $xyDistToRecInFP \leftarrow$ RSSITODIST($RSSIinFP[i, x, y] + strengtheningAmount$)
        $xyDistToRec \leftarrow$ EUCLIDEANDIST((x, y), $receiverPositions[i]$)
        if prefilteredRSSIs[i] is not None **then**
          $distToRecGivenRSSI \leftarrow$ RSSITODIST($prefilteredRSSIs[i] + strengtheningAmount$)
          $tmpSum \leftarrow tmpSum + (\text{ABS}(xyDistToRec - distToRecGivenRSSI)/distToRecGivenRSSI)^2$
          if ABS($prefilteredRSSIs[i] - RSSIinFP[i, x, y]$) $> maxSignalError$ **then**
            $tmpSum \leftarrow tmpSum + FPcoeff * (\text{ABS}(xyDistToRecInFP - distToReceiverGivenRSSI)/distToReceiverGivenRSSI)^2$
          end if
        else ▷ Use None info as well
          $maxCatchableSignalDist \leftarrow$ RSSITODIST($minUsefulSignal + strengtheningAmount$)
          if $xyDistToRec < maxCatchableSignalDist$ **then**
            $tmpSum \leftarrow tmpSum + FPcoeff * (\text{ABS}(xyDistToRec - maxCatchableSignalDist)/xyDistToRec)^2$
          end if
          if $xyDistToRecInFP < maxCatchableSignalDist$ **then**
            $tmpSum \leftarrow tmpSum + (\text{ABS}(xyDistToRecInFP - maxCatchableSignalDist)/xyDistToRecInFP)^2$
          end if
        end if

```
            end for
            if tmpSum < mySum then
                mySum ← tmpSum
                resultingPoint ← (x, y)
            end if
        end for
    end for
    return resultingPoint
end function
```

Explanations and Comments For Algorithm 8 :

- *xyDistToRecInFP* represents the distance between $(x,\ y)$ and receivers according to interpolated RSSI values in fingerprinting map.
- *xyDistToRec* represents the real distance between $(x,\ y)$ and the receiver.
- *distToRecGivenRSSI* represents the distance of the POI given the RSSI at the receiver.
- *maxCatchableSignalDist* represents the maximum distance of the POI that the receiver device can receive signal from. For this distance, we also consider the obstructions' effect on the signal.
- *RSSIinFP* represents the resulting interpolated and extrapolated fingerprinting map for all $(x,\ y)$ positions in the IE which is created given the RSSI values of the beacons we kept in the IE for a while.
- *strengtheningAmount* represents signal correction amount we would get if the POI were in $(x,\ y)$ position. If there were an obstruction between point $(x,\ y)$ and a receiver, then we would get a lower RSSI value at our receiver than in case there were no obstructions. We should increase the RSSI value for right distance calculations and therefore correct the signal by *strengtheningAmount*.

4.6 Particle Filter

For particle filtering, we firstly initialize all particles all around the map. Then, as time goes, the particles get close to the POI. By following the path of the mean of the particles for each time step, we can see what path that the POI follows. In the particle filtering algorithm, we have five main steps:

- **Initialization:**

  We initialize all particles randomly across the map of the IE. This step is only done once. Then, for each time step, we repeat the *prediction*, *weight update* and *resampling* steps of the particle filtering algorithm.

- **Prediction:**

  We predict the position of the object using human motion model. It means, we update all the particle positions considering how many meters a person could move in one time step. We use a parameter called *movingLimit* to represent the maximum amount of meters a person move in one time step. The particles make a movement in the amount between 0 and *movingLimit* in x and y directions where this amount is randomly determined. If we want to adjust to the movements of the POI faster, we can increase the value of *movingLimit* to make the particles move more freely with larger movements.

- **Weight Update:**

  We update the particle weights according to the difference between particle positions and measured position of the multilateration algorithm. We give higher weights to the particles which are closer to the measured position of the POI.

- **Resampling:**

  In the *prediction step* of the particle filtering, particles do not necessarily move in the right direction, that is, towards to the POI. Instead, they move randomly considering human motion relative to their current positions. However, we need the particles to move towards the POI. Hence, in *resampling step*; using N_{eff} value, we create particles all over again in the region where the most weighted particles reside. This step is the step where we form the probability distribution for our particles by making particles cumulate in the regions closer to the predicted POI position with higher probability and in other regions with lower probability. Without this step, the weight update would not have meaning and the particles would not be able to follow the POI.

- **Positioning:**

  At this step, some parameters of the probability distribution function of the particles in the IE map is found. These parameters are the *mean of the particles*,

max weighted particle and *covariance matrix* of the particles. Using the covariance matrix, we determine the uncertainty areas for our positioning system and we accept the *mean of the particles* as the predicted POI position.

We can see the implementation of the particle filtering in Algorithm 9. Since, particle initialization is done only once, we make this initialization before the main loop in Algorithm 1.

Algorithm 9 Particle Filter

```
1:  function PARTICLEFILTER
2:      IsPOIOutside ← True
3:      for each i ← 0; i < numberOfReceivers; i ← i + 1 do
4:          if prefilteredRSSIs[i] is not None then
5:              IsPOIOutside ← False
6:              break
7:          end if
8:      end for
9:      if IsPOIOutside = True then
10:         return                          ▷ No need further calculations
11:     end if
12:     PREDICT()
13:     UPDATEPARTICLEWEIGHTS()
14:     if thenNEFF() < NumberOfParticles/2
15:         RESAMPLE()
16:     end if
17:     meanOfTheParticleWeights ← FINDMEAN()
18:     maxWeightedParticle ← FINDMAX()
19:     covMatrix ← CALCCOVMATRIX()
20:     return meanOfTheParticleWeights
21: end function
```

Explanations and Comments For Algorithm 9:

- *IsPOIOutside* check whether the POI is still inside the IE or not. If not inside, there is no need for positioning inside.
- *Resample* function creates new particles in the area where the most weighted old particles reside after clearing all the old particles.

- *FindMean* function finds the mean of the particles which is the center of our prediction area and our prediction about the real position of the POI. It is the most probable result for the current real position of the POI according to the IP-PPFONM algorithm.
- *FindMax* function finds the max weighted particle.
- *CalcCovMatrix* calculates the covariance matrix of the particles to determine the uncertainty of our prediction about the real position of the POI.

4.6.1 Prediction Step

In this step, we update the particle positions considering human motion. Hence, we move particles by a certain amount which is limited by human movement which can be made in the time period between each calculation. This amount is represented by a parameter called *movingLimit*. Moreover, since humans generally move with similar velocities during their trajectories, we use a parameter called *pastCoeff* which determines how much we should rely on the previous velocity of the POI for determining the current position update amount. In Algorithm 10, we can see the implementation of the prediction step.

Algorithm 10 Prediction step of Particle Filter

1: **procedure** PREDICT
2: **for each** $i \leftarrow 0; i < numberOfParticles; i \leftarrow i + 1$ **do**
3: $xlow \leftarrow max(xmin, ParticlePositions[i].x - movingLimit) - ParticlePositions[i].x$
4: $xhigh \leftarrow min(xmax, ParticlePositions[i].x + movingLimit) - ParticlePositions[i].x$
5: $ylow \leftarrow max(ymin, ParticlePositions[i].y - movingLimit) - ParticlePositions[i].y$
6: $yhigh \leftarrow min(ymax, ParticlePositions[i].y + movingLimit) - ParticlePositions[i].y$
7: $walkingPredictionInXcoord \leftarrow$ UNIFORMLYDRAW(xlow,xhigh)
8: $walkingPredictionInYcoord \leftarrow$ UNIFORMLYDRAW(ylow,yhigh)
9: $walkingPredictionIn2D \leftarrow (walkingPredictionInXcoord, walkingPredictionInYcoord)$
10: **if** xPrev[i] and xPrevOfPrev[i] are set previously **then**
11: $pastVelocity \leftarrow xPrev - xPrevOfPrev$
12: $changeInPosIn2D \leftarrow (1 - pastCoeff) * walkingPredictionIn2D + pastCoeff * pastVelocity$
13: **else**
14: $changeInPos \leftarrow walkingAmountIn2D$
15: **end if**

16: **end for**

17: $ParticlePositions[i] \leftarrow ParticlePositions[i] + changeInPos$

18: **end procedure**

Explanations and Comments For Algorithm 10:

- In this step, we make a guess about the new particle positions considering human movement velocity. We make sure that particles do not go beyond the IE.
- *uniformlyDraw(a,b)* randomly draws some integer valued 1D coordinate between a and b.
- $xPrev[i]$ and $xPrevOfPrev[i]$ represent the positions of the particle at the previous iteration and at the position two iterations ago, respectively. *pastVelocity* gives us idea about how the POI will move in this iteration since humans would make a movement with a similar velocity according to the previous movement, most of the time. Hence, if *pastCoeff* value is close to 1, we guess that current movement will be very similar to the previous one regardless of current RSSI values. If it is close to 0, we do not look at previous movements when making the estimation for current POI motion.

4.6.2 Weight Update Step

We update the particle weights comparing with measured particle positions. Measured positions are calculated in our multilateration algorithm. The more a particle is close to the measured position, the more weight the particle gets. We can see the weight update implementation in Algorithm 11.

Algorithm 11 Weight Update step of Particle Filter

1: **procedure** UPDATEPARTICLEWEIGHTS

2: $sumOfDist \leftarrow 0$

3: **for each** $i \leftarrow 0; i < numberOfParticles; i \leftarrow i + 1$ **do**

4: $particlePOIdistance \leftarrow$ EUCLIDEANDIST(ParticlePositions[i], MeasuredPOIPos)

5: $sumOfDist \leftarrow sumOfDist + particlePOIdistance$

6: **end for**

7: **for each** $i \leftarrow 0; i < numberOfParticles; i \leftarrow i + 1$ **do**

```
8:        if ParticlePositions[i] is on a obstruction then
9:            ParticleWeights[i] ← 0
10:       else
11:           particlePOIdistance ← EUCLIDEANDIST(ParticlePositions[i],MeasuredPOIPos)
12:           ParticleWeights[i] ← ParticleWeights[i] * (sumOfDist/particlePOIdistance)
13:       end if
14:   end for
15:                                                          ▷ Normalize Weights
16:   sumOfWeights ← 0
17:   for each i ← 0; i < numberOfParticles; i ← i + 1 do
18:       sumOfWeights ← sumOfWeights + ParticleWeights[i]
19:   end for
20:   for each i ← 0; i < numberOfParticles; i ← i + 1 do
21:       ParticleWeights[i] ← ParticleWeights[i]/sumOfWeights
22:   end for
23: end procedure
```

Explanations and Comments For Algorithm 11:

- *particlePOIdistance* represents the distance between the $particle_i$ and the measured POI position according to our multilateration algorithm.
- If a particle happens to be on a obstruction as a result of prediction step explained in Section 4.6.1, then we should set that particle's weight to zero since the POI cannot be in the same position as an obstacle. After resampling step, the zero weighted particles disappear.

4.6.3 Resampling Step

Resampling is the process of sampling as many particles as we have initially. This step is also known as *sampling importance resampling* (SIR). In this step, we check whether the weights of our particles are good representatives for our object by checking N_{eff} value. If N_{eff} is low enough, we resample particles, Otherwise no resampling of particles takes place. The aim of the resampling is to avoid degeneracy problems where almost all particles diminish leaving only high-weighted particles.

Degeneracy prevents having a PDF that explains our possible object location with some uncertainty (variance).

We can use the mean or median of the particles to determine the POI position or the most weighted particle's position. We might also use only some particles according to their weights using the PDF we form. However, that is a implementational preference and we choose to use mean of the particles and the most weighted particles. We use different resampling algorithms like multinomial, residual, stratified or systematic resampling.

4.7 Common Functions

In Section 4.7.1, we explain how to calculate the value called *efficient number of samples* (N_{eff}) which gives and idea about whether we should resample the particles or not for the current time step. N_{eff} value is used in the *resampling step* of the particle filtering algorithm.

In Section 4.7.2, we explain how to correct signal strength values to get right distance information from RSSIs. We should have a higher RSSI value than what we receive if there is an obstruction between the signal source and the receiver. When we say *obstructions*, we mean *blocks and rooms* inside the IE in this thesis. We use obstruction material and thickness information for the signal correction amount.

In Section 4.7.3 section, we explain how to convert RSSI to distance and vice versa since we need to know the distance of the POI to receivers while applying multilateration algorithm and we need to extract the RSSI information from the distance in fingerprinting.

4.7.1 Efficient Number of Samples Calculation

This step is used for resampling. In this step, we calculate a value called N_{eff} which is useful to form a good probability distribution for the predicted POI position. If N_{eff} is low, it means that there are little number of particles whose weights are large (close to 1) and large number of particles whose weights are small (close to 0). Particles having

larger weights play a larger role to predict the POI position than the particles having smaller weights. We do not want to have only a little number of particles to determine the position of the POI; but we want to make use of most of our particles. Hence, we need to resample all of the particles in case N_{eff} is lower than a threshold we determine. Threshold for N_{eff} is generally chosen as half of the number of particles; but we may choose other values according to or application. We can see the N_{eff} formula in Equation 4.1 and the implementation of N_{eff} in Algorithm 12.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2} \tag{4.1}$$

Algorithm 12 Neff

```
1: function NEFF
2:     sumOfAllWeightsSquared = 0
3:     for each i ← 0; i < numberOfParticles; i ← i + 1 do
4:         sumOfAllWeightsSquared ← sumOfAllWeightsSquared + (ParticleWeights[i])^2
5:     end for
6:     return 1/sumOfAllWeightsSquared
7: end function
```

Explanations and Comments For Equation 4.1 and Algorithm 12 :

- In Equation 4.1, w represents particle weights, i represents each particle and N represents the total number of particles.
- In Algorithm 12, *sumOfAllWeightsSquared* represents the sum of all particle weights squared.

4.7.2 Signal Strength Correction

For distance estimations, obstructions between the receiver and the beacons cause signals to lose strength and hence, make our distance estimations higher than it should be. Therefore, we increase the RSSIs by how much signal loss caused by the obstructions for more accurate distance estimations. We can see the signal correction implementation in Algorithm 13.

Algorithm 13 Handle Signal Strength Due To Obstructions

```
1: function GETSIGNALLOSSAMOUNT(receiverPos, refPoint)
2:     line ← GETLINEBETWEEN(receiverPos, refPoint)
3:     strengtheningAmount ← 0
4:     for each room ∈ AllRooms do
5:         intersectionAmount ← GETINTERSECTIONAMOUNT(ROOM, LINE)(
6:         )materialOfRoom ← GETMATERIAL(room)
7:         signalDisturbanceOfMaterial ← GETSIGNALDISTURBANCE(materialOfRoom)
8:         strengtheningAmount ← strengtheningAmount + signalDisturbanceOfMaterial * intersectionAmount
9:     end for
10:    for each block ∈ AllBlocks do          ▷ Apply the same prodecure for the blocks
11:        intersectionAmount ← GETINTERSECTIONAMOUNT(block, line)
12:        materialOfRoom ← GETMATERIAL(block)
13:        signalDisturbanceOfMaterial ← GETSIGNALDISTURBANCE(materialOfBlock)
14:        strengtheningAmount ← strengtheningAmount + signalDisturbanceOfMaterial * intersectionAmount
15:    end for
16:    return strengtheningAmount
17: end function
```

Explanations and Comments For Algorithm 13:

- *refPoint* represents a 2D position where we want to find out how much of an obstacle is between this position and the receiver.
- $getintersectionAmount(room, line)$ returns the length of the intersection between the room and the imaginary line between the *receiverPos* and the *refPoint*.
- This function used to correct signals according to the effect of all obstructions or the portion of the obstructions which are between the receiver and the *refPoint*.
- $getSignalDisturbance(materialOfRoom)$ represents the value of disturbance value of *materialOfRoom* as indicated in *materialSignalDisturbanceCoefficients* parameter.

4.7.3 Conversions

We convert RSSI values to distances and vice versa according to the formulas used in the literature. For distances between $0m$ and $1m$ and their corresponding RSSI values, to the best of our knowledge, there is no generally accepted formula. Hence, we make our own algorithm by looking at the formula used for distances higher than or equal to $1m$. Our algorithm returns RSSI values in the range $[TX\_Power - 65,\ 0]$ for distances less than $1m$.

4.7.3.1 RSSI to Distance

We can convert RSSI to distance using Equation 4.2 for distances between $0m$ and $1m$.

$$Distance = 10^{((RSSI - rssiAtOneMeter)/-20)} \tag{4.2}$$

We can see the implementation of RSSI to distance conversions in Algorithm 14 and Algorithm 15. Algorithm 15 handles the RSSI values whose corresponding distances are between $0m$ and $1m$.

Algorithm 14 RSSI (dBm) to Distance (m) Conversion

```
1: function RSSIToDist(RSSI)
2:     dist ← 0
3:     if RSSI <= rssiAtOneMeter then
4:         dist ← 10^((RSSI − rssiAtOneMeter)/−20)
5:     else
6:         dist ← ZeroOneMeter_RSSIToDistance(RSSI )
7:     end if
8:     return dist
9: end function
```

Algorithm 15 RSSI to Distance between 0 and 1 meter

```
1: function ZeroOneMeter_RSSIToDistance
2:     return 10^(((RSSI − TX_Power) * log10(2))/(rssiAtOne − TX_Power)) − 1
3: end function
```

Explanations and Comments For Equation 4.2 Algorithm 14 and 15:

- In Algorithm 14, *RSSI* represents a value that we want to convert to a distance in meters. *rssiAtOneMeter* represents the RSSI value at the receiver when the beacon is $1m$ away. We use the Equation 4.2 to calculate the corresponding distance if the *RSSI* is lower than RSSI value that should be received at $1m$; however for RSSIs higher than this, we follow a different approach and use Algorithm 15 since Equation 4.2 is designed for RSSIs less than or equal to *rssiAtOneMeter*.

4.7.3.2 Distance to RSSI

We can see the implementation of RSSI to distance conversions in Algorithm 16 and Algorithm 17. Algorithm 17 handles the distance conversions for distances $0m$ and $1m$.

We can convert distance to RSSI using Equation 4.3 for distances between $0m$ and $1m$.

$$RSSI = -20\log_{10}(distance) + rssiAtOneMeter \tag{4.3}$$

Algorithm 16 Distance to RSSI Conversion

```
1: function DISTTORSSI(dist)
2:     RSSI ← 0
3:     if dist >= 1 then
4:         RSSI ← −20 ∗ log10(dist) + rssiAtOneMeter
5:     else
6:         RSSI ← ZEROONEMETER_DISTTORSSI(dist)
7:     end if
8:     return RSSI
9: end function
```

Algorithm 17 Distance to RSSI between 0 and 1 meter

```
1: function ZERO_ONE_METER_DISTANCE_TO_RSSI
2:     return TX_Power + (rssiAtOne − TX_Power) ∗ ((log10 dist + 1)/(log10 2))
3: end function
```

Explanations and Comments For Equation 4.3, Algorithm 16 and 17 :

- In Algorithm 16, *dist* represents a value that we want to convert to a RSSI in

dBm. *rssiAtOneMeter* represents the RSSI value at the receiver when the beacon is $1m$ away. We use the Equation 4.3 to calculate the corresponding RSSI if the *dist* is lower than $1m$; however for distances higher than $1m$, we follow a different approach and use Algorithm 17 since Equation 4.3 is designed for distances less than or equal to $1m$.

4.8 Use Cases of the IP-PPFONM

The IP-PPFONM algorithm is able to track a single person or multiple people in a small or large area. Furthermore, IP-PPFONM algorithm can track people without getting a single signal if we know that the POI is inside the IE. Hence, we can list the use cases of the IP-PPFONM algorithm as follows:

- **Single / Multiple POI Tracking:** Each POI has its own unique MACID and we use these MACIDs to distinguish each POI from the others. Calculations for each POI is the same and hence, we can apply IP-PPFONM algorithm for each POI on a different thread of our indoor positioning application simultaneously.
- **Small / Large Area Tracking:** IP-PPFONM works in both small and large areas. However, in large areas, IP-PPFONM takes more time since we make an exhaustive search to find the global minimum value minimizing our loss function in the multilateration part of the particle filtering step. The time that an exhaustive search takes also depends on the parameter *sensitivityOfResult*. For instance, for a *sensitivityOfResult* of $0.5m$ and *numberOfReceivers* of 5; the IP-PPFONM algorithm can process areas of $250m^2$, $540m^2$ and $800m^2$ in $1s$, $1.5s$ and $2s$ respectively.
- **Signal Usage:** Normally, BLE positioning algorithms need some signal to find out the position at each time step. Yet, in IP-PPFONM; even if we do not get any signal at some of the receivers in an IE, we still make calculations according to these receivers. We can even make calculations when none of the receiver devices catch a signal if we know that the POI is inside the IE. If we do not know whether the POI is inside the IE, getting no signal at none of our receivers would mean that the POI is probably outside the IE.

CHAPTER 5

SIMULATION TOOL AND PERFORMANCE ANALYSIS OF THE THEORETICAL EXPERIMENTS

The organization of this chapter can be summarized as follows:

- In Section 5.1, we explain a simulation tool that we design to simulate and visualize the IP-PPFONM algorithm.
- In Section 5.2, we explain some experiments performed by changing some parameters of IP-PPFONM algorithm to see how these parameters affect the positioning result in theory. For each experiment; we firstly explain how the setup environment is. Then, in the corresponding *result* section, we explain what the results are for the experiment. Lastly, in the corresponding *discussion* section, we interpret the results found.

5.1 Simulation Tool

In this section, we explain how the simulation tool simulates the POI motion and the RSSIs at receivers, how we place the initial position of the POI and obstructions and how the simulation tool visualizes the POI, obstructions and the IP-PPFONM results in an indoor environment.

5.1.1 Simulating the POI Motion and the RSSIs in the IE

The simulation tool moves the POI in the IE while making sure that the POI does not hit any obstructions or go out of the boundaries of the IE. The simulation tool uses the parameters called *movingLimit* and *pastCoeff* to determine how to move the POI.

For positioning calculations, the IP-PPFONM algorithm requires RSSI values at receivers. Hence, the simulation tool generates RSSIs at the receivers using the ground truth position of the POI, the distance between the ground truth and the receivers and the obstruction information in the IE. The simulation tool decreases the RSSI values if there are obstructions between the receiver and the POI considering the *maxSignalError* parameter value since obstructions affect the RSSIs in the real world.

5.1.2 Placing the POI and the Obstructions

We consider two types of obstructions which are blocks and rooms. The simulation tool firstly install blocks in the IE and make sure they do not intersect with each other, then install rooms and make sure rooms do not intersect with any other blocks or rooms and finally determine the initial position of the POI and make sure the POI does not intersect with any obstructions.

5.1.3 Visualizations:

In this section, we mention the visuals we use in our simulation to represent some objects and notions in IP-PPFONM algorithm:

- **The Indoor Environment:** It is drawn as the outermost rectangle which contains all other objects. It is the map of the *indoor environment* (IE) we represent.
- **The Obstructions (Rooms and Blocks):** Each block or room has a different coloring indication about the materials they are made of. Blocks are shown as gray, aqua or beige colored filled rectangles to represent concrete, glass and plastic materials respectively whereas rooms are either shown as gray or aqua colored unfilled rectangles to represent concrete and glass materials.
- **The Receivers:** They are the devices receiving the signals that beacons transmit. They are drawn as the dark blue square objects.
- **The POI:** It is the person we track and represented as the green circle.
- **The Mean of the particles:** It is the weighted average of all particles on the map and our the result of the IP-PPFONM algorithm for each iteration. It is represented as the purple circle.

- **The Most Weighted Particle:** It is the particle having the highest weight after the weight update step in particle filtering algorithm is completed. It is represented as the orange circle.
- **Three Confidence Ellipses:** We need to show our errors with some probability since we cannot know the exact position of the POI. We use three confidence ellipses formed around the mean of the particles to represent this probability. We give some penalty and enlarge confidence ellipses when we need to trust low power values.

  For drawing the ellipses we use a parameter called *strongSignalDistance*. This parameter represents the maximum distance threshold for accepting a signal as a strong signal. Signals with RSSIs having correspondent distance values higher than *strongSignalDistance* are accepted as weak signals whereas the others are accepted as strong signals.

  If there are no receivers receiving a signal, we do not show any ellipses. When there are less than three receivers, as we decrease the number of receivers, we enlarge our confidence ellipses. We enlarge confidence ellipses more in less number of receivers case than we do in the low power signal case to show we are more uncertain about the results when the number of receivers is less than three. If signals are received at three or more receivers and all signals are strong, then the inner most ellipse represents one standard deviation, middle ellipse represents two standard deviations and the outermost ellipse represents three standard deviation of error from the mean of the particles. If not, then the ellipses are larger than one, two and three standard deviations for innermost, middle and outermost ellipses respectively.

5.2 Experiments

This section shows how IP-PPFONM algorithm works in theory by performing some experiments using our simulation tool. For each experiment, we change only one parameter while holding the other parameters constant. We perform five performance analysis experiments to measure the accuracy of the IP-PPFONM algorithm in this section:

1. Multilateration Accuracy Test
2. Effect of Number of Receivers Test
3. Effect of Signal Noise Test
4. Effect of Obstruction Material Type and Thickness Test
5. Effect of Past Coefficient Test
6. Effect of Sensitivity Test

We also make two demonstrations to show what the IP-PPFONM algorithm can do:

1. Multiple POI Tracking Demonstration
2. Efficient Receiver Placement Demonstration

Efficient receiver placement demonstration shows how we place the receivers in the most efficient way possible whereas multiple POI tracking demonstration shows how we can track multiple POI using the IP-PPFONM algorithm.

To visualize the results, we use bar charts and our simulation tool mentioned in Section 5.1. For objects inside the environment created by our simulation tool, we use the visualizations mentioned in Section 5.1.3.

The implementation of the simulation tool was explained in the algorithms given in Chapter 4. In this chapter, we create two maps using this simulation tool where one is of small size ($5m \times 3m$) and the other one is of large size ($14m \times 11m$). In the experiments of this chapter, fingerprinting information is not used for the IP-PPFONM algorithm since fingerprinting is an information which is only meaningful for real world.

For material tests, we use two different materials which are *concrete* and *glass* where their signal losses are chosen as $6dBm$ and $16dBm$ for a thickness of $1m$ considering our observations and a few sources [38] [37]. These values may not represent the true signal loss values for every glass and every concrete form, but they are representative values which give an idea about how results differ for materials having these two signal loss values for materials having thicknesses of $1m$.

We perform the experiments in a small and large sized maps and predefine a trajectory for both small and large sized maps for the ground truth values. Some experiments

contain two concrete blocks and some do not contain any obstructions. The trajectories in small and large maps can be seen in the Figure 5.1 and Figure 5.2 respectively. We normally represent the position of the POI with green filled disc. However, in all the figures in Chapter 5, to distinguish between the starting and ending position of the POI from the other positions; we use yellow and red colors for starting and end positions respectively whereas all other positions are represented as green. The numbers inside the filled circles, if any, represent the time step of the POI in the IE.

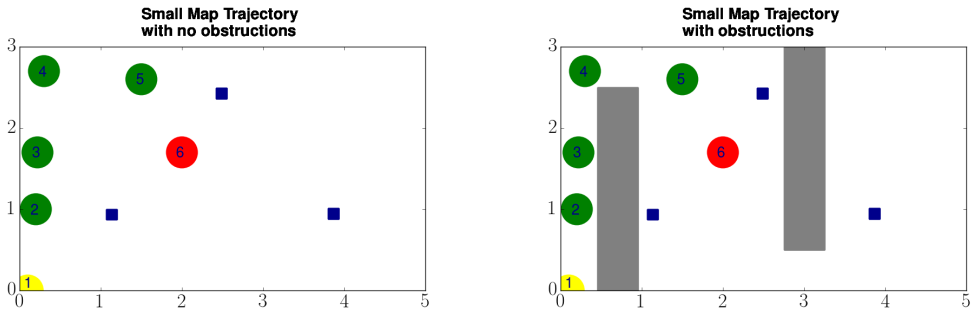


Figure 5.1. Small Map Ground Truth Trajectories

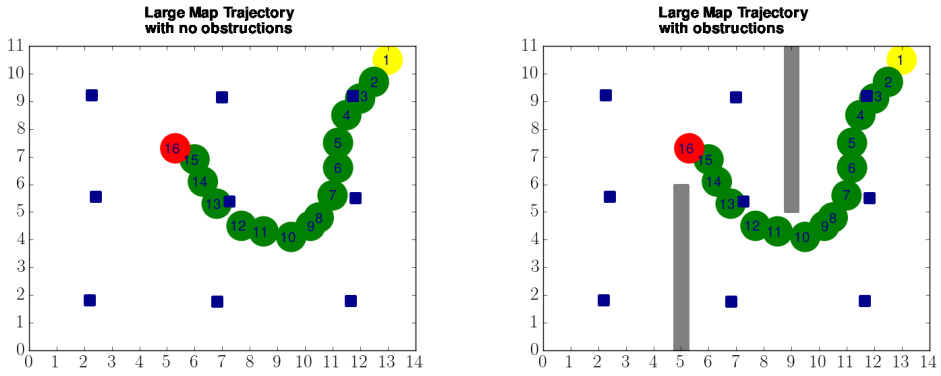


Figure 5.2. Large Map Ground Truth Trajectories

5.3 Performance Analysis Tests

For accuracy tests, we make comparisons for some parameters that we have used in the simulation. We hold all other parameters constant to compare different values of a parameter. For these experiments, we run our simulations for some predefined number of time steps and positions. For *number of receiver placement*, *signal noise*

and *obstruction material type and thickness* experiments, we use a map of size $14m \times 11m$ and for the other experiments we use a map of size $5m \times 3m$ as $width \times height$ for visualizations.

As for accuracy tests, we use two comparison values *Average Error* and *Number of No Signal Reception*. Average error is defined as:

$$AverageError = (\sum_{i=1}^{N} e_i)/N \tag{5.1}$$

where N is the number of time steps and where error at time $step_i$ is the Euclidean distance between the predicted position at time $step_i$ and the real position of the POI at time $step_i$:

$$e_i = EuclideanDist(MeanOfTheParticles_i, RealPos_i) \tag{5.2}$$

Number Of No Signal Reception indicates how many times we cannot catch any valid signal at any of the receivers and therefore we are not able to make a positioning calculation for that time step. The lower the average error and *number of no signal reception*, the better the results. We make the tests in two different sized maps since for some of the experiments, small sized map is enough whereas for others, a larger map is required.

Both *Average Error* and *Number of No Signal Reception* values are calculated for each run of IP-PPFONM algorithm once and each run may contain multiple time steps. As can be seen in Figure 5.1 and Figure 5.2, we limit the number of time steps spent in the indoor environment by six times and sixteen times for small and large map respectively. The reason of this limitation is just to be able to visualize definite number of time steps spent by the POIs.

We repeat each of cases of the six performance analysis experiments explained in this section twenty times to control the variance of the results in terms of *Average Error* and *Number of No Signal Reception* values. After running the experiments twenty times, we calculate the mean and the standard deviation values for each of the cases covered in the experiments and show these results in bar charts. For each run, if exists, the place of the blocks and initial position of the POI are randomly determined

by our simulation. As for the simulation figures in our experiments, we only share the resulting images of a single run for each experiment.

5.3.1 Multilateration Accuracy Test

5.3.1.1 Results

At each time step, we calculate the position of the POI according to the signal powers we get using our multilateration algorithm. For this experiment, we use two different TX powers and two different sized maps. Then, we add two obstructions in the maps and check for multilateration accuracy again. We represent the starting point with yellow color and the end point with red whereas the other positions are represented by green. We indicate the time steps inside the circles.

We investigate eight cases for multilateration tests where each case shows multiple positions forming a trajectory of the POI where these positions are the results of our no-signal multilateration algorithm. These eight cases can be listed like this:

- **CASE1:** Trajectory of the POI on the small sized map with $SignalNoise = 0dBm$ and no obstruction.
- **CASE2:** Trajectory of the POI on the small sized map with $SignalNoise = 0dBm$ and 2 concrete blocks.
- **CASE3:** Trajectory of the POI on the small sized map with $SignalNoise = 10dBm$ and no obstruction.
- **CASE4:** Trajectory of the POI on the small sized map with $SignalNoise = 10dBm$ and 2 concrete blocks.
- **CASE5:** Trajectory of the POI on the large sized map with $SignalNoise = 0dBm$ and no obstruction.
- **CASE6:** Trajectory of the POI on the large sized map with $SignalNoise = 0dBm$ and 2 concrete blocks.
- **CASE7:** Trajectory of the POI on the large sized map with $SignalNoise = 10dBm$ and no obstruction.
- **CASE8:** Trajectory of the POI on the large sized map with $SignalNoise = 10dBm$ and 2 concrete blocks.

We represent the positions for each case as numbered circles where numbers indicate the time step in which the no-signal multilateration algorithm produce this position result. Therefore, by following the numbers from small to large, we can see the trajectory of the POI according to the no-signal multilateration algorithm. Since, in the *prefiltering* step we wait for at least three signals, there is no position information about the first and the second time steps for the multilateration calculations.

We can see figures showing multiltaration results (measured positions) for each small map case in Figure 5.3 and for large map case in Figure 5.4. In these figures, multilateration results are only shown for the time steps that they are not filtered out in the prefiltering step.

As a result of twenty repetitions for each of the eight cases, we get 0.06, 0.67, 1.06, 1.00, 0.04, 1.80, 2.26, 2.66 meters as the means and 0.01, 0.50, 0.46, 0.45, 0, 0.98, 0.43, 0.98 meters as the standard deviations of *average errors*. The errors between the real positions of the POI and the multilateration results are given in Figure 5.5.

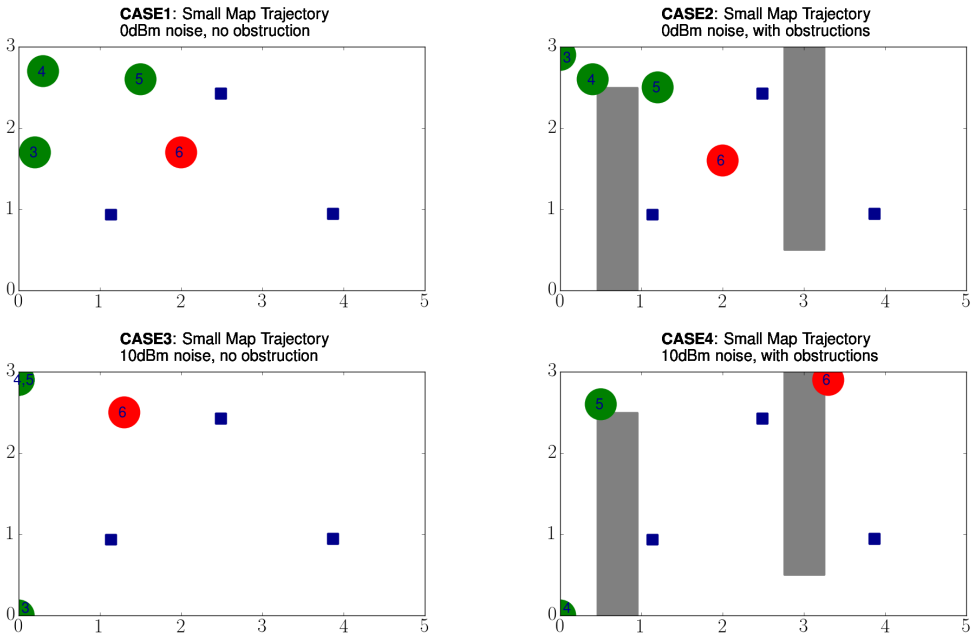


Figure 5.3. All Small Map Multilateration Cases

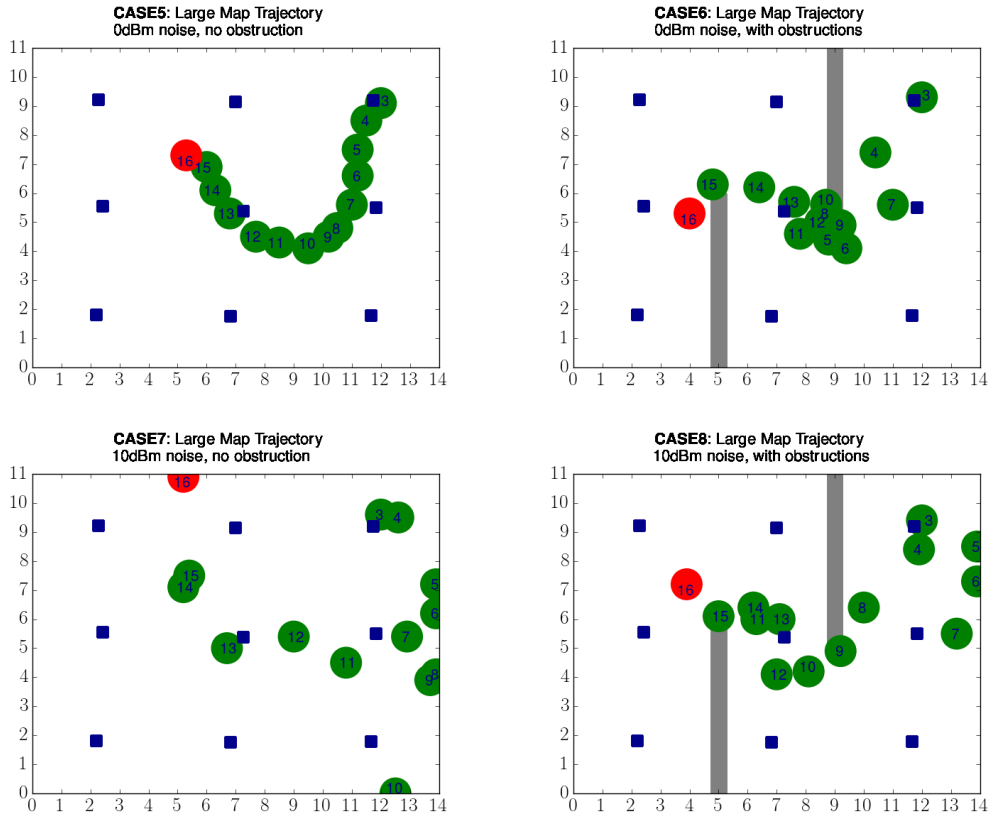


Figure 5.4. All Large Map Multilateration Cases

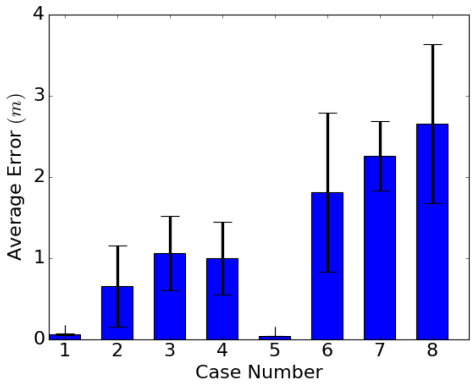


Figure 5.5. Multilateration Accuracy

We also run the IP-PPFONM algorithm for all of the eight cases to find out the final prediction results to compare these results with the multilateration results. So, we additionally applied particle filtering algorithm on the multilateration results to find out the predicted positions. For final prediction results, as a result of twenty repetitions

for each of the eight cases, we get 0.52, 0.83, 0.77, 0.71, 0.77, 1.91, 2.17, 2.33 meters as the means and 0.15, 0.32, 0.21, 0.27, 0.22, 0.86, 0.77, 0.89 meters as the standard deviations of *average errors*. The errors between the real positions of the POI and the predicted positions according to multilateration are given in Figure 5.5.

The errors between the real positions of the POI and the predicted positions of the IP-PPFONM algorithm are given in Figure 5.6

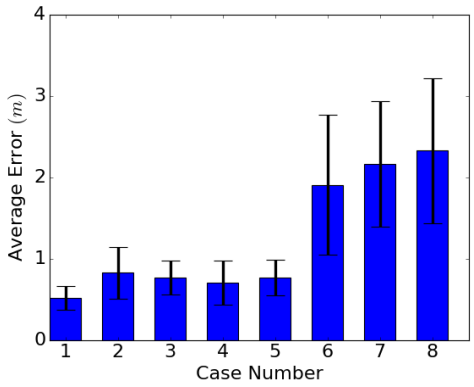


Figure 5.6. Final Accuracy

5.3.1.2 Discussion

For *Average Error* in Figure 5.5, we see that case1 is very close to the ground truth value positions, but not exactly accurate since the ground truth position the third ground truth position of the small map has 0.22 as x position which requires more sensitivity than what we use, 0.1 sensitivity. However, besides all positions are accurate. For case5, all positions are exactly accurate. So, case1 and case5 show that multilateration algorithm results in 100% accuracy with right parameters when there is no noise or obstruction in the indoor environment.

We also see that as the noise increases the multilateration accuracy decreases which is normal since noise gives misdirects the positioning information. Generally, getting in the coverage area of more receivers result in more accuracy as can be seen better in large map cases. Moreover, even though we handle the obstructions in the indoor environments, the obstructions may still decrease the accuracies since we cannot know exact signal loss we have due to the obstructions.

If we compare Figure 5.5 and Figure 5.6, we can see that our multilateration algorithm performs better before applying particle filtering when there are no noise or obstructions in the environments. However, when there are noise or obstructions, applying particle filtering on the multilateration results generally improve the results.

5.3.2 Effect of Number of Receivers Test

5.3.2.1 Results

We position the receivers in a way that we reach as much coverage area as possible. The more receiver we have, the more chance we receive a signal from the beacon of the POI. More receiver also means possible more distance information to the POI that we need to handle.

We analyze six cases where we experiment with six different number of receivers in the large map for this test to see how our algorithm behaves as we increase the number of receivers. The number of receivers we experimented are 1, 2, 3, 5, 7 and 9.

As a result of twenty repetitions for each of the cases, we get 3.18, 3.39, 3.03, 2.42, 2.69, 2.57 meters as the means and 2.58, 2.16, 2.09, 1.38, 1.15, 0.94 meters as the standard deviations of *average errors* whereas we get 2.2, 2, 2.05, 2.05, 2, 2 meters as the means and 0.4, 0, 0.22, 0.22, 0, 0 meters as the standard deviations for *number of no signal reception*.

We can see an example effect of number of receivers on positioning in Figure 5.7. and see the bar charts related to the average positioning error and the number of times we are not able to receive any signal at any of our receivers in Figure 5.8.

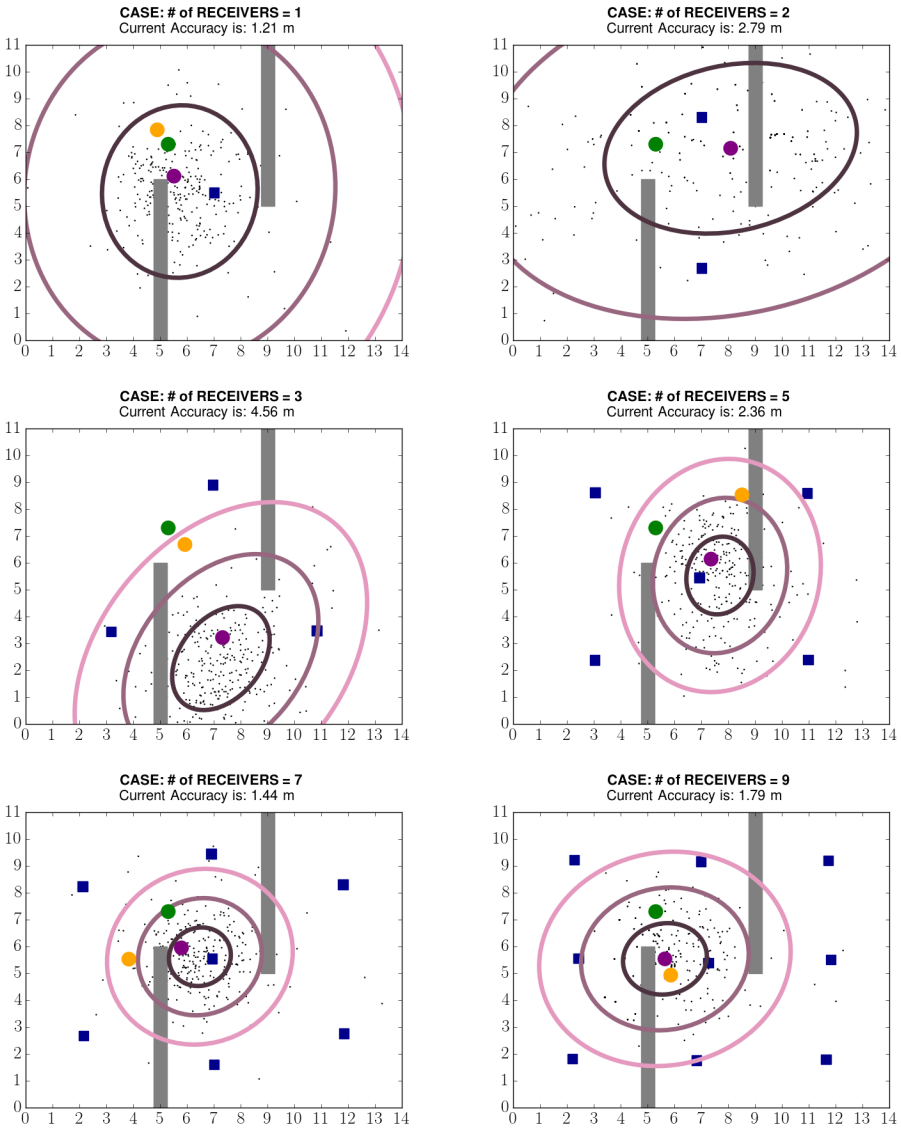


Figure 5.7. Number of Receiver Placement Effect

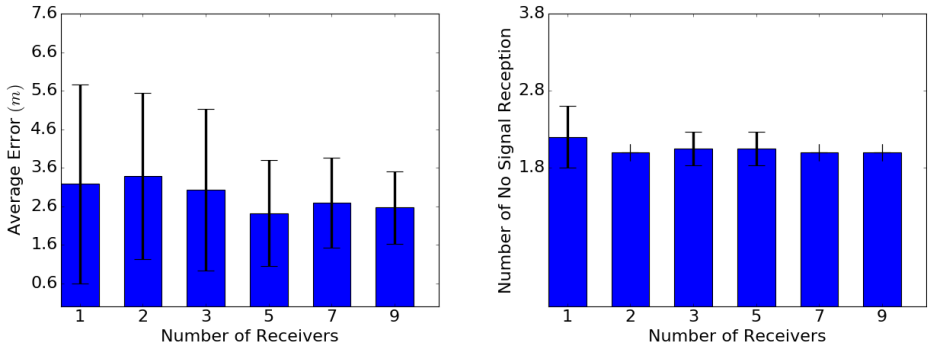


Figure 5.8. Number of Receiver Placement Error Info

5.3.2.2 Discussion

For *Average Error* in Figure 5.8 , we see that increasing number of receivers is generally better for increasing the accuracy of the IP-PPFONM. However, since obstructions and noise in the IL may misdirect the calculations at the receivers, increasing the number of receivers does not guarantee an increase in the accuracy as can be seen transitioning from 2 to 3 and 5 to 7 receivers.

For *Number of No Signal Reception* graph in Figure 5.8, we can see that the *number of no signal reception* generally decreases as the number of receivers increases as we aim. However, the receivers in three and five receiver cases are not able to cover the areas that the receivers in two receiver case cover. According to this graph, this situation seems to cause an advantage in the favor of two receiver case compared to the three and five receiver cases in terms of *number of no signal reception*.

5.3.3 Effect of Signal Noise Test

5.3.3.1 Results

Signal noise is an undesired phenomenon which misdirects positioning calculations. We have six cases where we experiment with six signal noises for this test: $0dBm$ (no noise at all), $5dBm$, $10dBm$, $15dBm$, $20dBm$, $30dBm$.

We can see an example effect of the signal noise in the IE on positioning in Figure 5.9. As a result of twenty repetitions for each of the cases, we get 0.77, 1.29, 2.23, 3.42, 4.24, 3.06 meters as the means and 0.19, 0.31, 0.87, 1.13, 1.28, 1.53 meters as the standard deviations of *average errors* whereas we get 2, 2, 2, 2, 2, 3.75 meters as the means and 0, 0, 0, 0, 0, 1.41 meters as the standard deviations for *number of no signal reception*. We can see the bar charts related to average positioning error and the number of times we are not able to receive any signal at any of our receivers in Figure 5.10.

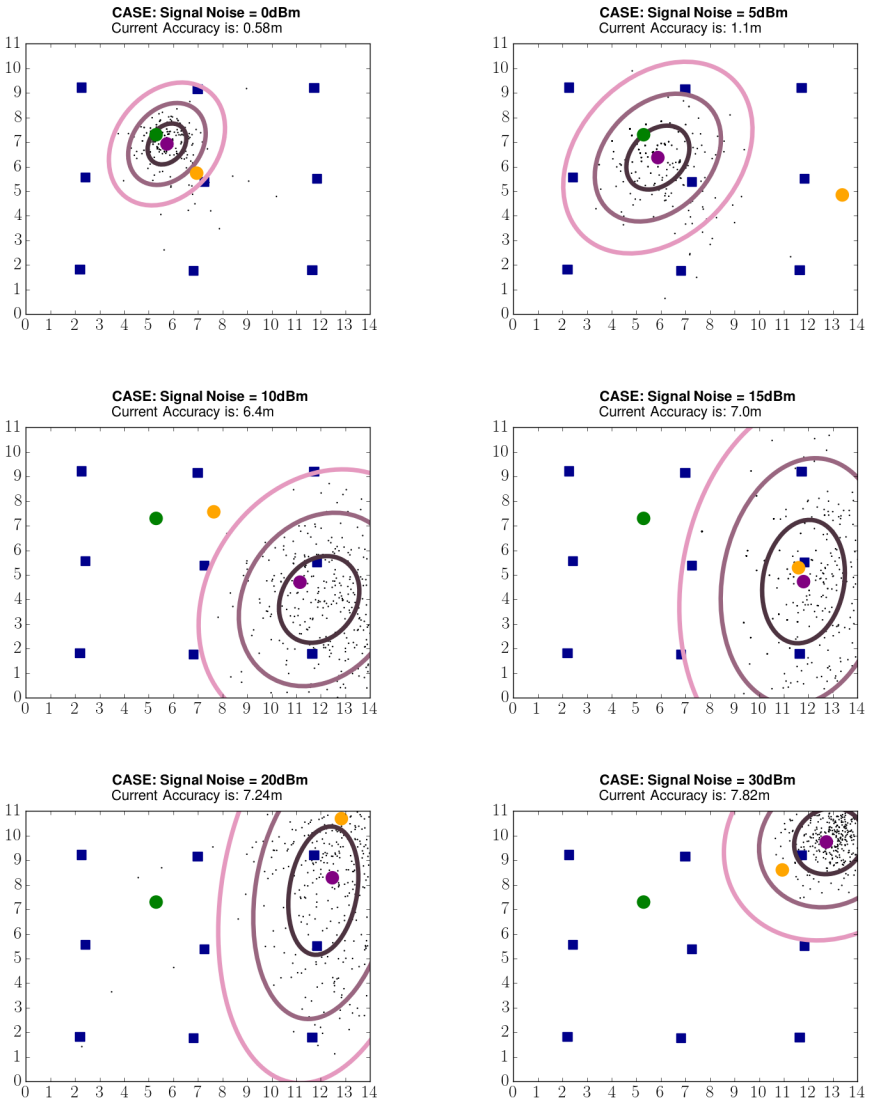


Figure 5.9. Signal Noise Effect

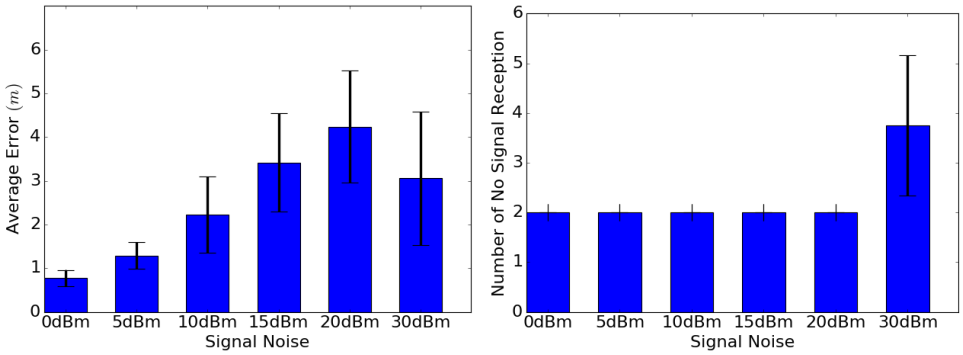


Figure 5.10. Signal Noise Error Info

5.3.3.2 Discussion

For *Average Error* in Figure 5.10, we see that as noise increases the algorithm deviates from the actual location more. Even though $-30dBm$ noise seems to cause a better accuracy than $-20dBm$ does, there are more cases we could not make a calculation for $-30dBm$ compared to $-20dBm$ noise as can be seen in *Number of No Signal Reception* graph. The reason why we have more cases that we cannot make any calculations for $-30dBm$ case is compared to other noise vales is, $-30dBm$ is a very high noise which decreases the BLE signals at the receivers to have very low values. If none of our receivers have high RSSI values according to the calculation in the prefiltering part of the IP-PPONM algorithm, we do not make any calculations for these steps.

For the time step shown in Figure 5.9, we can see that, as the noise increases, we declare a lower confidence as can be seen from the ellipse sizes. However, for the highest noise case, we seem very confident on a wrong result since the high noise makes the IP-PPFONM find that the POI is far away from most of the receivers and therefore finds the right upper corner as the right position with a high confidence (small ellipse size).

5.3.4 Effect of Obstruction Material Type and Thickness Test

5.3.4.1 Results

We analyze six cases in the small map where we experiment with three different thickness which are $30cm$, $50cm$, $70cm$ and two different materials which are concrete and glass in total:

1. $30cm$-thick concrete blocks
2. $30cm$-thick glass blocks
3. $50cm$-thick concrete blocks
4. $50cm$-thick glass blocks
5. $70cm$-thick concrete blocks
6. $70cm$-thick glass blocks

We know that as the thickness of the objects increase, the signal loss they cause increases. As for the material type, concrete causes more signal loss than glass does. We expect to see similar accuracies for different thickness values and materials to see our algorithm handles obstructions correctly.

We can see an example effect of material type and thickness on positioning in Figure 5.11. As a result of twenty repetitions for each of the cases, we get 0.77, 0.76, 0.85, 0.71, 0.71, 0.74 meters as the means and 0.32, 0.34, 0.39, 0.26, 0.29, 0.32 meters as the standard deviations of *average errors*. We can see the bar chart related to average positioning error in Figure 5.12. In Figure 5.11 and Figure 5.12, gray color represents the concrete blocks and the aqua blue color represents the glass blocks.

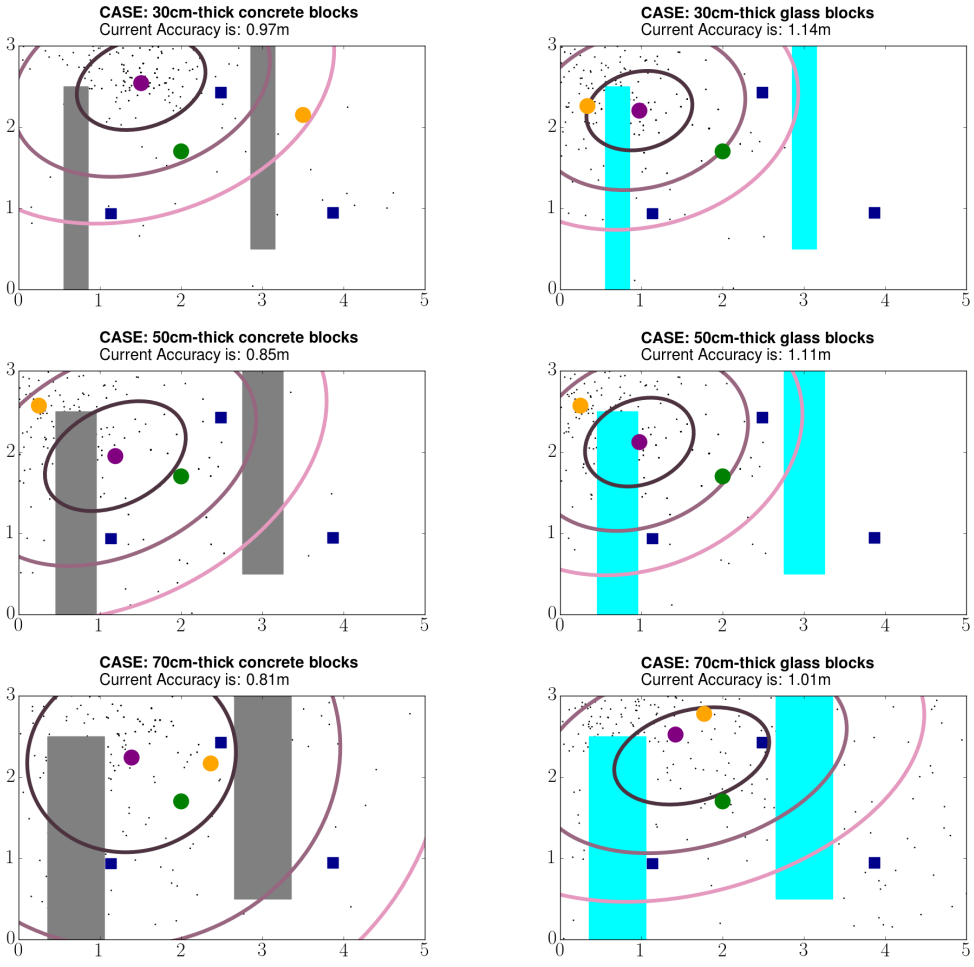


Figure 5.11. Obstruction Material Type and Thickness Effect

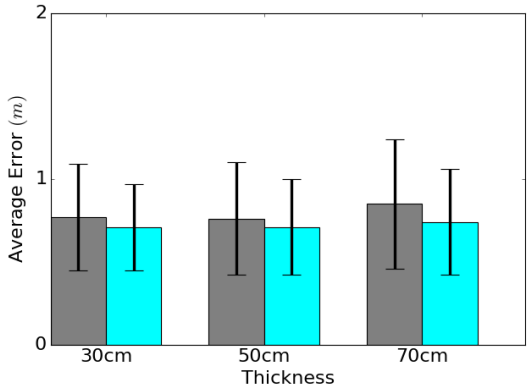


Figure 5.12. Obstruction Material Type and Thickness Error Info

5.3.4.2 Discussion

We see that IP-PPFONM algorithm can compensate for the noises that stem from obstructions regardless of their material or thickness by judging from the fact that the results for all cases have similar accuracies which are all between $0.70m$ and $0.85m$ as the mean errors and between $0.25m$ and $0.40m$ for the standard deviations.

5.3.5 Effect of Past Coefficient Test

5.3.5.1 Results

As explained in Section 4.1, $pastCoeff$ parameter represents how much of the previous prediction result should be used for the prediction calculation in the particle filtering. We experiment with six cases in the small map where we inspect $pastCoeff$ values 0, 0.2, 0.3, 0.5, 0.8 and 1.

We can see an example effect of *pastCoeff* on positioning in Figure 5.13. As a result of twenty repetitions for each of the cases, we get 0.8, 0.75, 0.76, 0.75, 0.77, 0.79 meters as the means and 0.34, 0.34, 0.34, 0.35, 0.35, 0.36 meters as the standard deviations of *average errors*. We can see the bar chart related to average positioning error in Figure 5.14.

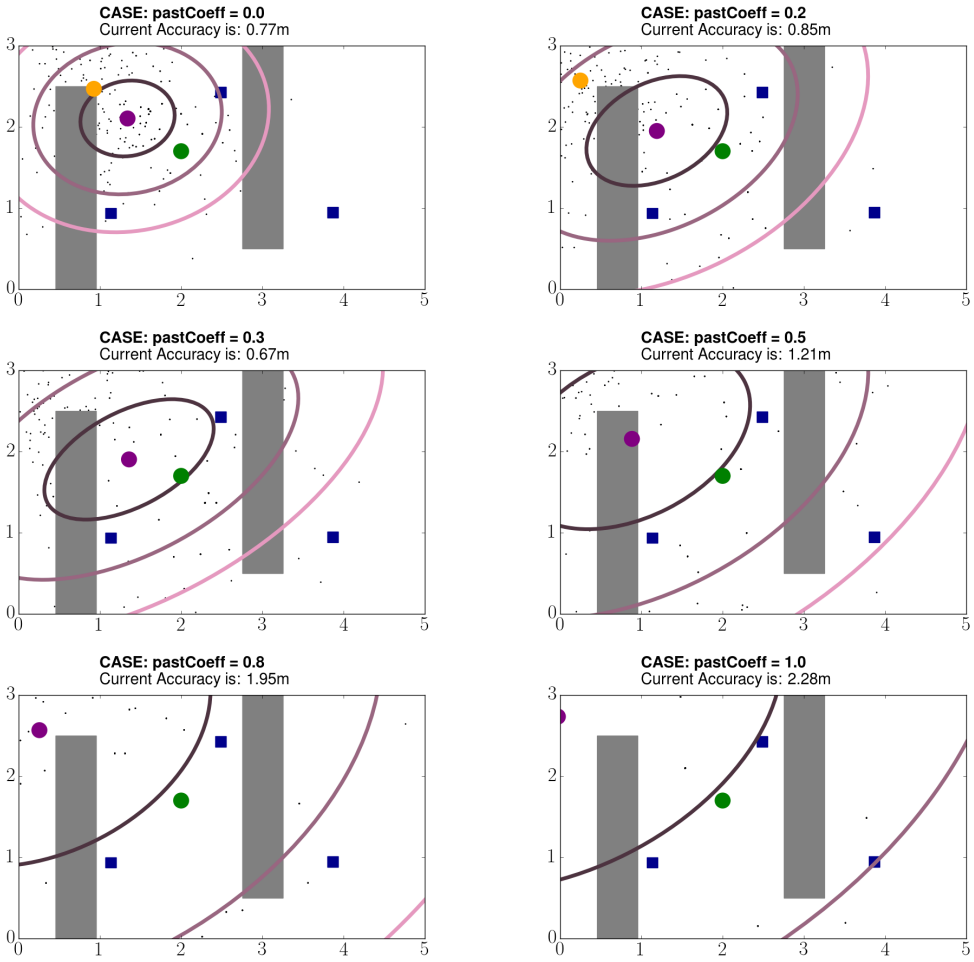


Figure 5.13. Past Coefficient Effect

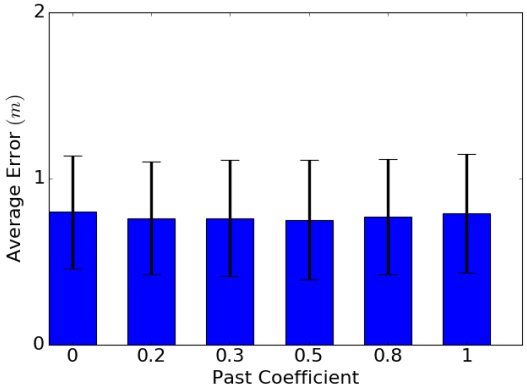


Figure 5.14. Past Coefficient Error Info

5.3.5.2 Discussion

We see that as the *pastCoeff* increases, the error increases. The results do not have to be like that all the time. This just shows that this POI, do not move along the same direction and velocity during the walking period in the experiments.

5.3.6 Effect of Sensitivity Test

5.3.6.1 Results

This value is the distance between each of the search points in the multilateration algorithm as mentioned in Chapter 4. The less this value, the more sensitive we are and the more area we search for our multilateration algorithm. We experiment with six cases with two different sensitivity and three different obstruction information where each obstruction has $50cm$ width for this test:

1. $0.1m$ sensitivity with concrete blocks
2. $0.5m$ sensitivity with concrete blocks
3. $0.1m$ sensitivity with glass blocks
4. $0.5m$ sensitivity with glass blocks
5. $0.1m$ sensitivity with no obstructions
6. $0.5m$ sensitivity with no obstructions

We can see effect of sensitivity on positioning in Figure 5.15. As a result of twenty repetitions for each of the cases, we get 0.76, 0.84, 0.67, 0.67, 0.59, 0.6 meters as the means and 0.34, 0.41, 0.23, 0.32, 0.19, 0.18 meters as the standard deviations of *average errors*. We can see the bar chart related to average positioning error in Figure 5.16. In Figure 5.15 and Figure 5.16, gray color represents the concrete blocks and the aqua blue color represents the glass blocks. For Figure 5.16, the black color represents no obstruction case.

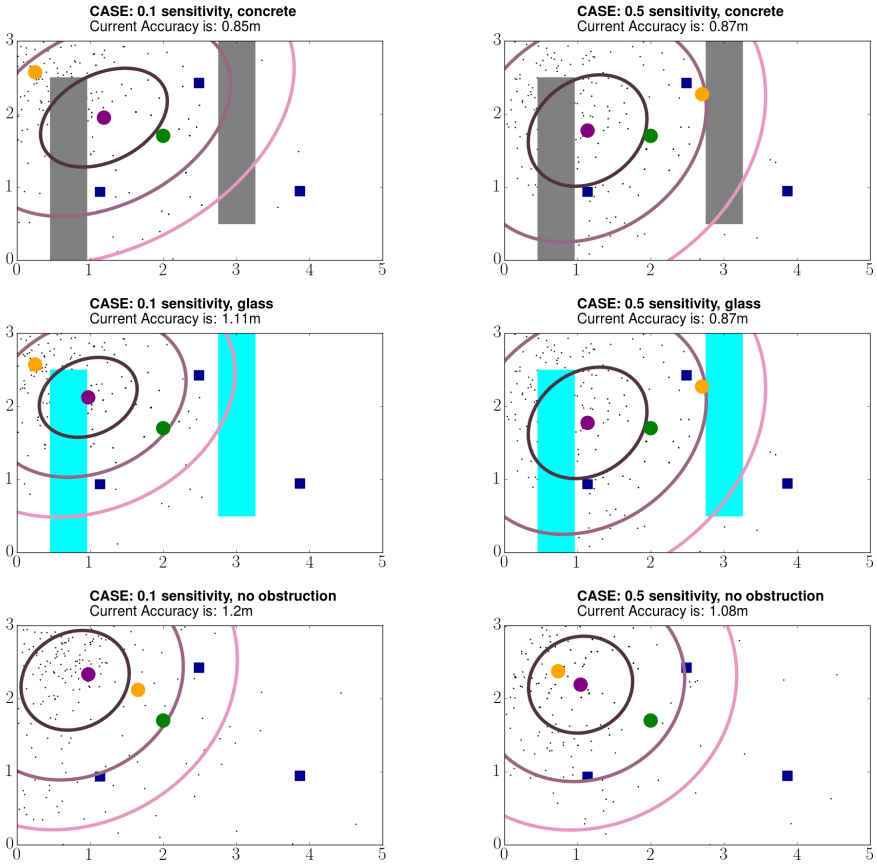


Figure 5.15. Sensitivity Effect

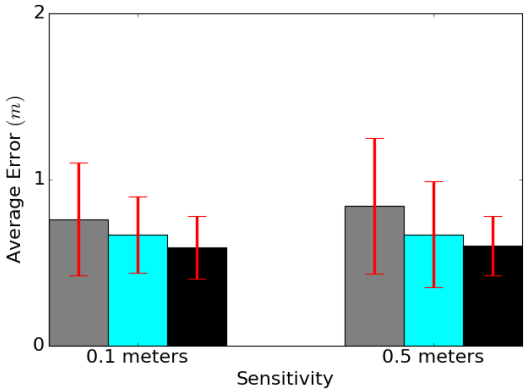


Figure 5.16. Sensitivity Error Info

5.3.6.2 Discussion

As we increase the sensitivity, the accuracy generally increases. The reason why the accuracy does not always increase shows that the multilateration algorithm does

not always find the closest possible position about the real position of the POI and therefore searching for more points in the IE does not always increase the accuracy. However, as we expect, the accuracy generally increases which shows that our multilateration algorithm generally finds a close position to the real position of the POI.

5.4 Demonstrations

5.4.1 Multiple POI Tracking

5.4.1.1 Results

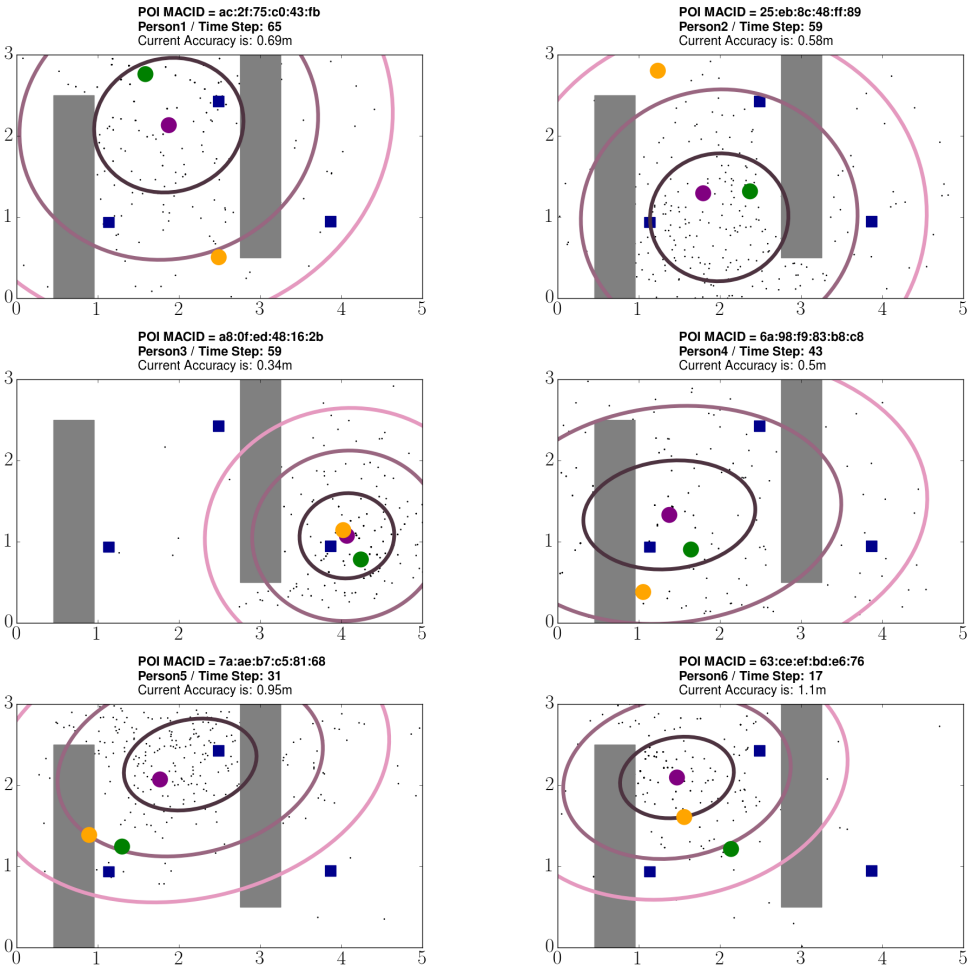


Figure 5.17. Simultaneous BLE Tracking for Multiple POI

In this section, we show that we can track multiple people simultaneously. For this,

we simulate six people entering into an indoor environment represented by the small map with at most 20 seconds between the entrance time of these people. In Figure 5.17, we show the tracking results of six people. Each person is identified by a MACID which is shown as *POI MACID* in Figure 5.17.

5.4.1.2 Discussion

In Figure 5.17, we can see that we track six different MACIDs where all POIs have their own ellipses, particles and accuracy values for the current time step. Moreover, except for the Person2 and Person3, all POIs have different dwell times as can be seen by *TimeStep* indicated in the title of the simulations. Even though Person2 and Person3 enter the IE at the same time, our algorithm does not mix up the trajectories or particles for these people. Hence, we see that our algorithm works equally well and correctly when tracking multiple people in real time.

5.4.2 Efficient Receiver Placement

5.4.2.1 Results

For *efficient receiver placement*, we position the receivers in a way that we reach as much coverage area as possible. We choose nine placements each having different number of receivers installed in an example area of $14m \times 11m$. The number of receivers tested are 1, 2, 3, 5, 7, 9, 15, 20 and 30. The placements can be seen in Figure 5.18.

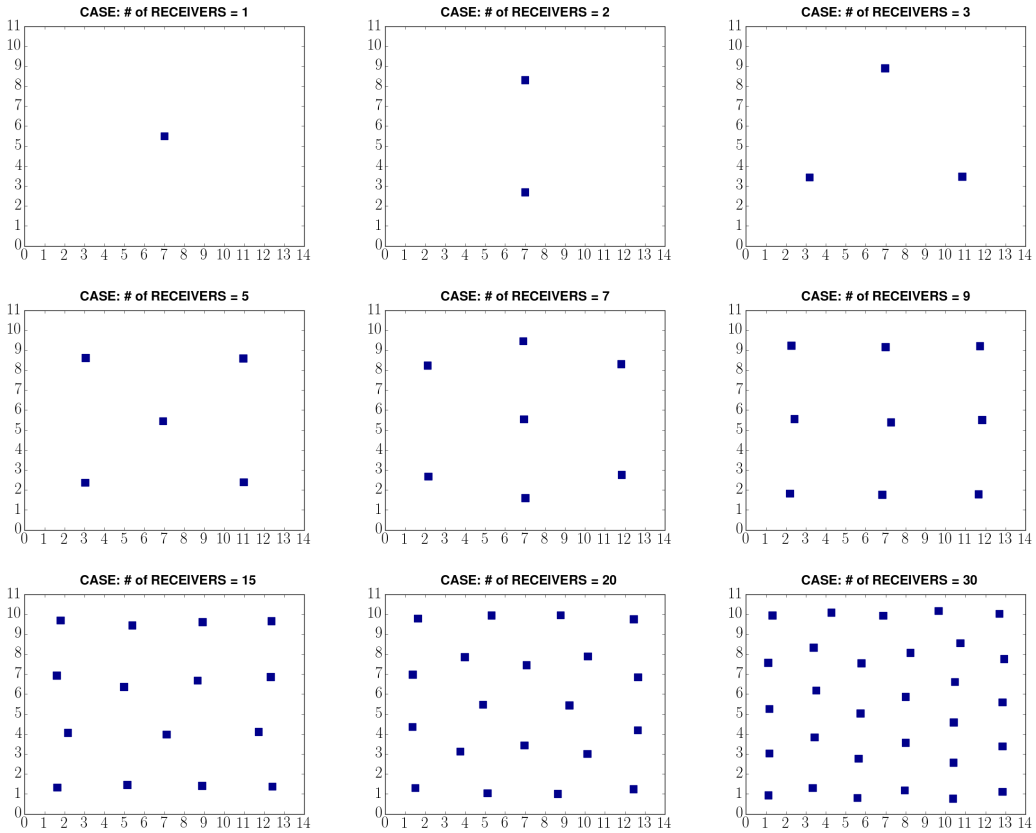


Figure 5.18. Efficient Receiver Placement

5.4.2.2 Discussion

In Figure 5.18, we see that all receivers are almost evenly distributed in the IE which. Moreover, we see that receiver devices are positioned away from the walls so that signal coverage area of the receivers are not limited by the walls. These show our k-means approach for efficient receiver placement which is mentioned in Chapter 4 helps maximizing signal coverage area of receivers in an area.

CHAPTER 6

PERFORMANCE EVALUATION OF THE REAL WORLD EXPERIMENTS

In this chapter we conduct two types of experiments in real world environments. The first type of experiments are performed to show how *indoor positioning via pre-filtering and particle filtering with obstruction-aware no-signal multilateration* (IP-PPFONM) algorithm is used in a real world environment and how accurate the results are. We call the first type of experiments *the positioning accuracy experiments*.

The second type of experiments are performed to understand how BLE signals behave in general. We call the second type of experiments *the behavioral experiments*. To come up with a solution using BLE signals, we need to know how BLE signals behave and how these signals should be used in different situations. For this purpose, we conduct the following experiments:

- **Signal Blocking Material Tests:** We put some signal blocking materials around the receiver device to check if this will prevent the signal from being received.
- **Antenna Role:** We remove the external antenna to see the effect of the antenna for RSSI.
- **Beacon Carrying Position Test:**. We check whether carrying the beacons in hand or in pocket makes a difference with different TX powers. We try different TX powers to see if high powered signals still get affected by the beacon carrying position.
- **BLE Chipset / Receiver Device Role:** We use two receiver devices having different BLE chipset to see the effect of the chipset on RSSI.
- **Obstruction Effect:** We check how obstruction affects signal penetration.

- **Transmitter / Beacon Factor:** We test different beacons from different vendors where all the parameter settings of the beacons are the same to see if vendors are important to receive a quality signal (high RSSI).

- **Transmission Power and Period Factor:** We test the same beacons with different parameter settings to see the effect of these parameters on signal stability (constantly receiving some signal for tracking) and RSSI.

For each experiment; we firstly explain how the setup environment is. Then, in the corresponding *result* section, we explain what the results are for the experiment. Lastly, in the corresponding *discussion* section, we interpret the results found. To visualize the results we use bar charts, line charts and the visualizations mentioned in Section 5.1.3.

6.1 Positioning Accuracy Experiments

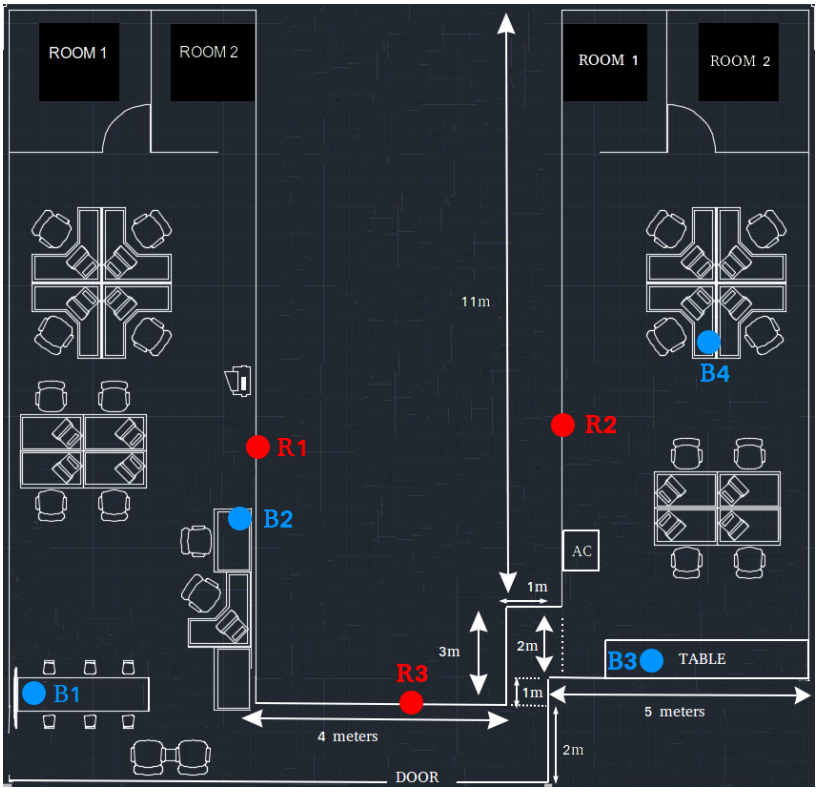


Figure 6.1. Indoor Environment for the Real World Positioning Test

We perform the experiments in an office environment of size $15m \times 16m$ where the IE layout is shown in Figure 6.1. In Figure 6.1, blue dots (B) represent fingerprinting beacons and red dots (R) represent receiver devices on the wall.

We also need sizes of blocks and rooms for our algorithm and these information which can also be seen in Figure 6.1. All receivers are $2m$ above the ground and all the fingerprinting beacons approximately $1m$ above from the ground. The position of the receivers and the fingerprinting beacons in the IE are shown in Table 6.1 and the 2D distances of the beacons to the receivers in $x,\ y$ coordinates, are shown in the Table 6.2.

Table 6.1. Fingerprinting Beacon and Receiver Positions

Object	2D Position	
	x	y
Beacon1	0.25	2.25
Beacon2	5	6
Beacon3	11.5	3.5
Beacon4	12.5	9
Receiver1	5	7.5
Receiver2	10.3	8
Receiver3	7.4	2

Table 6.2. Receiver-Fingerprinting Beacon Distance (m)

	Rec1	Rec2	Rec3
Beacon1	7.08	11.58	7.15
Beacon2	1.5	5.66	4.66
Beacon3	7.64	4.66	4.37
Beacon4	7.65	2.42	8.66

Figure 6.2. Card Beacon

Figure 6.3. Receiver Device on the Wall

We use card shaped BLE beacons that use iBeacon technology, to emit BLE signals which is shown in Figure 6.2. The width, length and thickness of the BLE beacons we use are approximately $86mm$, $55mm$ and $4mm$ respectively. We use an embed-

ded device with a single-board computer that has a BLE antenna which is shown in Figure 6.3.

We know the places of the fingerprinting beacons. Therefore, RSSI values of these beacons for different receivers give us some idea about the signal power value and the positions on the map. To construct a fingerprinting map to be used in IP-PPFONM algorithm, we place four beacons at certain positions in an office environment and collect signal power results for approximately three days. We call these beacons as *fingerprinting beacons*. We use MACIDs of beacons to identify each POI. The TX powers and advertising intervals of the beacons are $0dBm$ and $1000ms$ for these measurements.

Real world experiments are conducted in an office environment. We have three cases where we perform the real world experiments in different conditions. We performed the carrying in pocket case twice, but did not perform carrying in the hand case twice since in real world people generally carry beacons in their pockets or bags, but not in their hands. The cases can be numerated as:

1. **Case1:** In-pocket, crowded case (carrying the beacon in our pocket in a crowded environment)
2. **Case2:** In-hand, crowded case (carrying the beacon in our hand in a crowded environment)
3. **Case3:** In-pocket, non-crowded case (carrying the beacon in our pocket, in a non-crowded hour)

For the ground truth position values when moving, we use a fish-eye lensed camera results where we use the position values only in sight of the camera. We save each of the three experiments' RSSI results into csv files to read the real world results into the IP-PPFONM algorithm. In our experiments, *crowded* means that the population density changes between $0.07\ people/m^2$ and $0.10\ people/m^2$ whereas *non-crowded* means that the density is $0.3\ people/m^2$ where these densities are calculated over the areas where the camera sees. We walk at average speeds of $0.41m/s$, $0.61m/s$ and $0.80m/s$ with standard deviations of $0.60m/s$, $0.32m/s$ and $0.30m/s$ for the case1, case2 and case3 respectively.

While obtaining the ground truth positions, an important issue is to match the camera time and the time RSSI values received at all of our receivers. To that end, we use a server to synchronize times for each of our receiver devices and the camera. We make sure that they are perfectly synchronized in terms of seconds. Another issue is we need to collect some data to understand the signal behavior in the IE. Hence, in this experiment, we collect the signal power values for three days at 10 minute time intervals and then, average them.

We apply our algorithms and methods on our receivers which are embedded devices that has Bluetooth 4.0 chipset supporting BLE. Data is acquired via BLE antennas and BLE chipsets that are built into our receiver devices. These BLE chipset are built-in on a single-board computer chip, but the antennas are replaceable.

To find how realistic our simulation tool is, we perform two main experiments for the *positioning accuracy experiments* for the same real world environment:

1. **Real World RSSI Data Experiments:** These are the experiments performed in an office environment with real data. These experiments are the ones which determine the accuracy of the IP-PPFONM algorithm in real life.
2. **Generated RSSI Data Experiments:** These are the experiments performed in an office environment with real data except for the RSSI data acquired by wandering around the office. These are the experiments we perform to show how realistic is our simulation tool in terms of generating RSSI data for a real life case.

In *Real World RSSI Data Experiments*, we firstly note down the obstruction information in the office environment. Then, we wander around the IE for 50 seconds and then collect the RSSI values at each receiver during the time of wandering. After that, watching the recordings of a fish-eye lensed camera, we note down the ground truth positions for each second. Lastly, before running the IP-PPFONM algorithm; we enter the obstruction position information in the IE, ground truth positions and the collected RSSI information to the algorithm. Using the entered information, our algorithm predicts a position in the IE with some confidence and calculates how the predicted position differs from the ground truth positions at each time step.

In *Generated RSSI Data Experiments*; we apply the same procedure as in *Real World Data Experiment* except for the RSSI collecting part. In these experiments, we use RSSI data that our simulation tool generates by looking at the ground truth positions resulted from wandering around the office and obstruction information in the real world environment.

In both *Real World RSSI Data Experiments* and *Generated RSSI Data Experiments*, we use BLE fingerprinting data that we collect for three days in the real world environment.

6.1.1 Results

As we run the IP-PPFONM algorithm for *Real World Data Experiments*, we save two images showing results for case1 results at two different time steps as can be seen in Figure 6.4 and Figure 6.5. We also show the resulting images for case2 and case3 in Figure 6.6 and Figure 6.7 respectively. These three images are chosen in a way that we can show the positioning accuracy where the POI is in different positions.

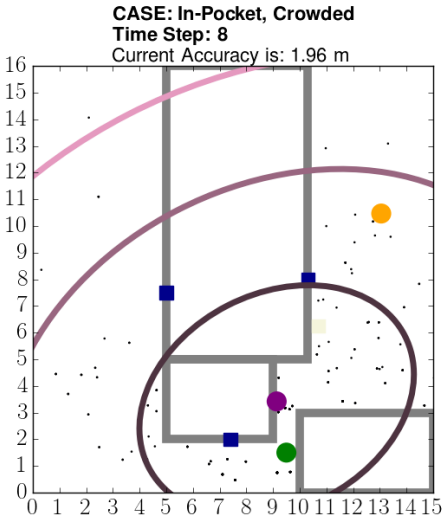


Figure 6.4. Case1: 8th second

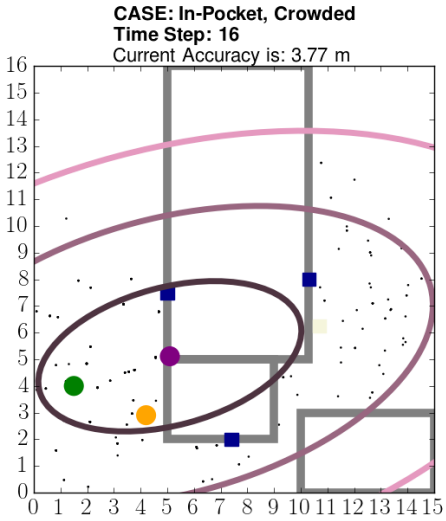


Figure 6.5. Case1: 16th second

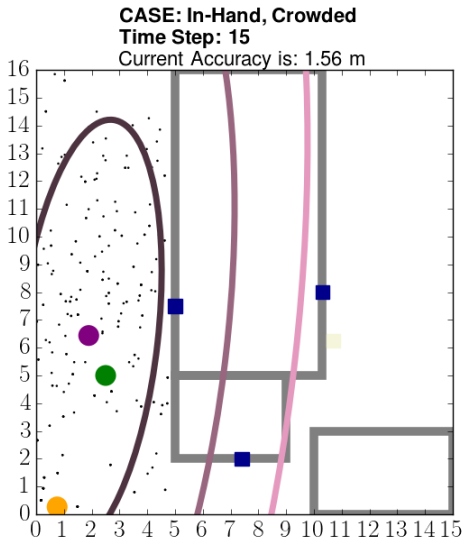


Figure 6.6. Case2

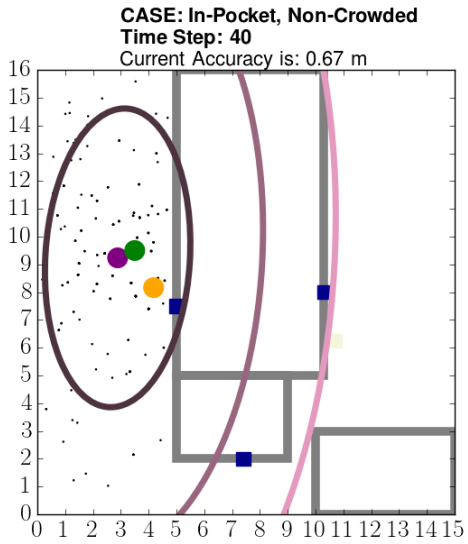
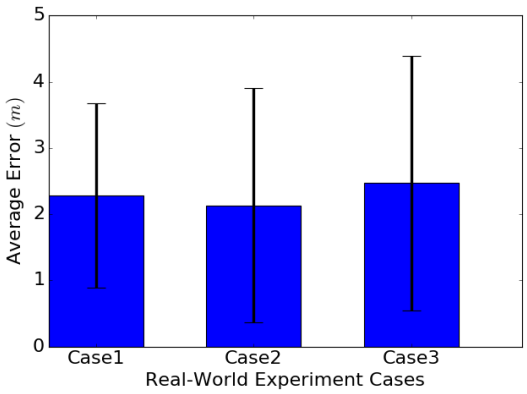
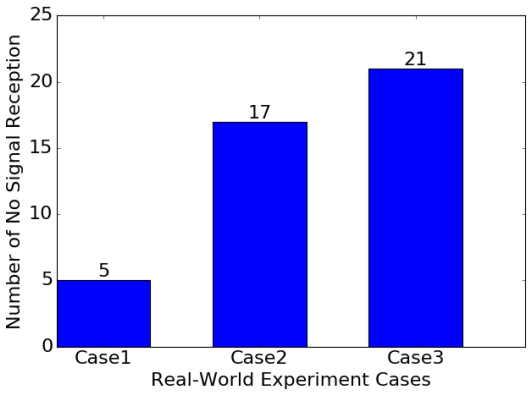


Figure 6.7. Case3

For *Real World RSSI Data Experiments*, the IP-PPFONM algorithm achieves an average accuracy of $2.28m$, $2.13m$ and $2.47m$ with standard deviations of $1.39m$, $1.77m$ and $1.92m$ for case1, case2 and case3 respectively as shown in Figure 6.8a. Moreover, for all cases combined, our algorithm achieves an average accuracy of $2.29m$ with a standard deviation of $1.67m$. We can also see the number of times we could not make a positioning calculation as *number of no signal reception* in Figure 6.8b.



(a) Average Accuracy



(b) Number of No Signal Reception

Figure 6.8. Real World RSSI Data Experiment Results

As a result of running IP-PPFONM algorithm for *Generated Data Experiments*, the

corresponding visual positioning results for the Figure 6.4, Figure 6.5, Figure 6.6 and Figure 6.7 are found to be as in Figure 6.9, Figure 6.10, Figure 6.11 and Figure 6.12 respectively.

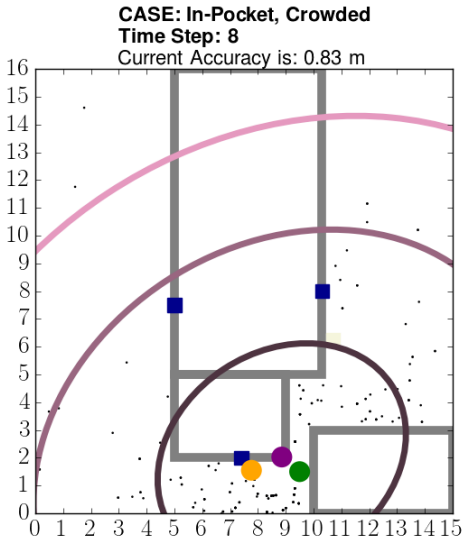


Figure 6.9. Case1 with Generated RSSIs: 8th second

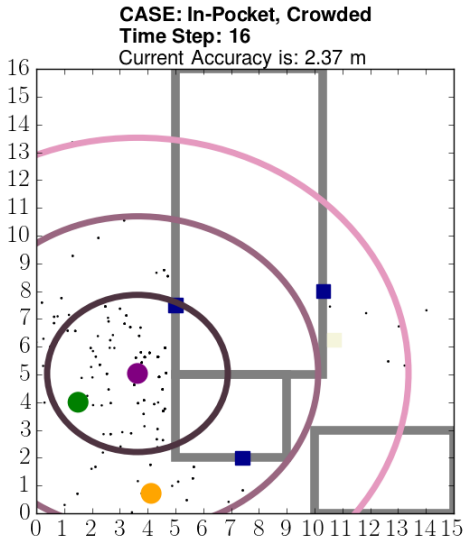


Figure 6.10. Case1 with Generated RSSIs: 16th second

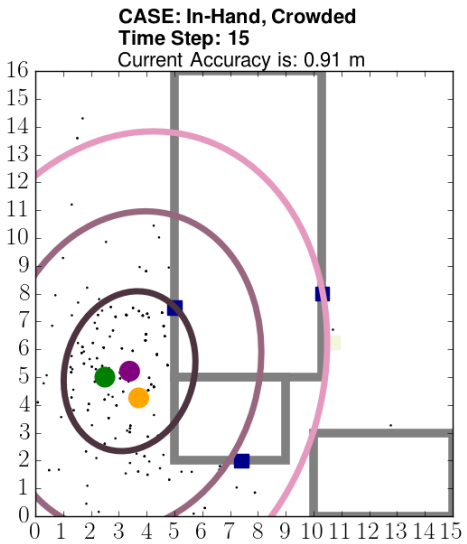


Figure 6.11. Case2 with Generated RSSIs

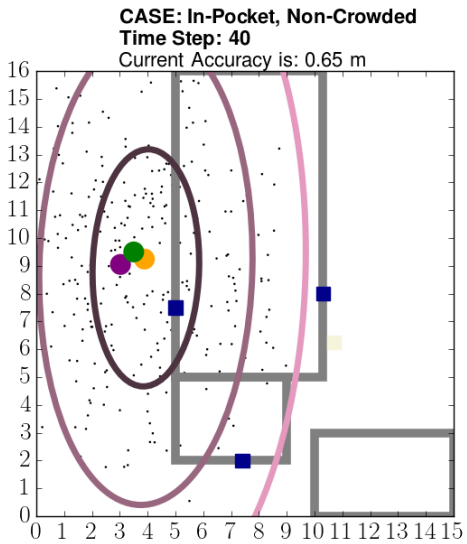
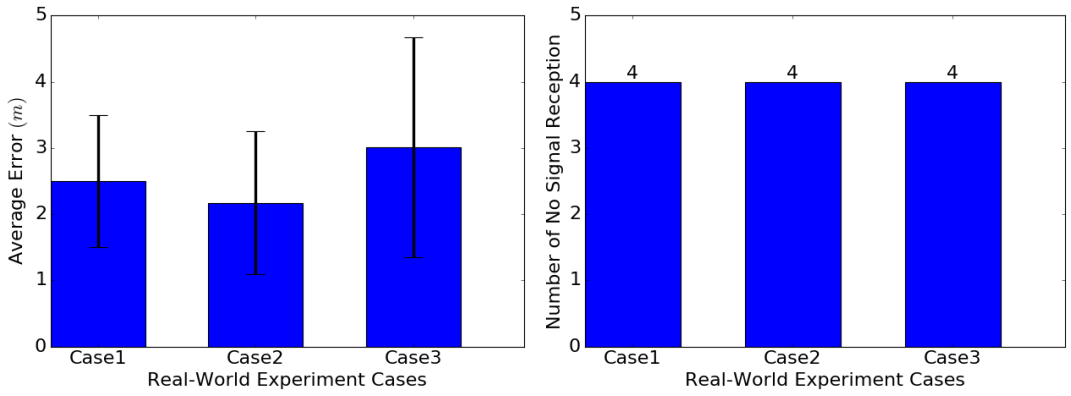


Figure 6.12. Case3 with Generated RSSIs

For *Generated RSSI Data Experiments*, the IP-PPFONM algorithm achieves an average accuracy of $2.50m$, $2.17m$ and $3.01m$ with standard deviations of $1.00m$, $1.08m$

and $1.66m$ for case1, case2 and case3 respectively as shown in Figure 6.13a. Moreover, for all cases combined, our algorithm achieves an average accuracy of $2.58m$ with a standard deviation of $1.33m$. We can also see the number of times we could not make a positioning calculation as *number of no signal reception* for *Generated Data Experiments* in Figure 6.13b.



(a) Average Accuracy

(b) Number of No Signal Reception

Figure 6.13. Generated RSSI Data Experiment Results

The resulting fingerprinting RSSIs in Table 6.3 are used in the multilateration part of the IP-PPFONM algorithm in both of the *positioning accuracy experiments*. These are the RSSI values of four fingerprinting beacons at three receivers averaged over three days in the office environment where we make the positioning test. We use fingerprinting information for the results of the simulation tool as well to be able to show how realistically our simulation generates RSSI data compared to the real world data.

Table 6.3. Receiver-Fingerprinting Beacon Signal Power (dBm)

	Rec1	**Rec2**	**Rec3**
Beacon1	-76.0	-84.0	-82.0
Beacon2	-72.0	-81.0	-77.0
Beacon3	-86.0	-66.0	-85.0
Beacon4	-82.0	-73.0	-90.0

6.1.2 Discussion

We see that we are able to reflect the effect of large objects and rooms residing in the environment in our results. However, there is also the human factor and small objects in the environment which can be dynamically identified and determined only. Our algorithm takes static objects residing in the environment into account and try to mitigate the effect of these objects for positioning calculations.

Our real world environment has a difficult plan for tracking with small number of receivers. But, it is a good way to show how we try to handle obstructions in the way. For positioning, normally, at least three receivers are needed for distance calculations based on RSSI. However, our algorithm always tries to find a position even when we have one RSSI value at only one receiver. We prevent making calculations when none of our receivers have RSSI values in the real world experiment since it could mean the person went out of the indoor environment. However, if we know that the POI is always in the IE, then the IP-PPFONM algorithm can make positioning calculations even without receiving a single RSSI value since not receiving any signal may mean being a certain distance away from the receivers.

In positioning experiments, since our beacons do not transmit BLE signals frequently, all of the receivers may not catch signals at each time step which may lower the accuracy of our multilateration algorithm. Therefore, we determine a way to use signals for the time steps we do not receive any. To that end, we firstly search for the previous time step's RSSI value and if not found, then wait for the next time step's RSSI value. We use a parameter to determine the maximum value for how many time steps into the past or into the future we should look at until we find a RSSI value. For our real world test, we search for at maximum two past and two future values. If we do not find any, then we mark the signal value *None*. In this approach, our real time algorithm may make the current time step's calculations after up to two time steps (2 seconds in our case) later to find a signal. Since this approach may use the same signal multiple times, our sliding window size requirement for the prefiltering step of IP-PPFONM should be increased to be more selective and hence, we set the minimum size requirement of each sliding window to 5 instead of 3 where 3 is the value we use in our simulation results in chapter5.

We realize that when carrying beacons in hand or in pocket, moving affects the signal powers. Therefore we add $4dBm$ to RSSIs before making the calculations for the cases carrying in hand and add $6dBm$ for the cases when we carry the beacons in our pockets. We expected carrying in hand would be much more accurate than carrying in pocket case, but after adding these offset power values, results did not change much in our case. We also observe that moving beacons decreases the RSSI values at receivers compared to keeping them still.

According to our observations which will be mentioned in Section 6.2.7.1, BLE signals having $<= 300ms$ advertising interval results in a stable signal environment. By stable signal environment, we mean an environment where receivers can get enough signals to update the positioning result more often to catch up with the POI motion. However, since we use $1000ms$ as the BLE signal advertising interval, we often lose the signal at our receivers which prevents making positioning calculations more often. Also, we use low number of receivers for a large office environment where there is no clear line of sight between any of our receivers and there are lots of wireless devices like cell phones and Wi-Fi access points interfering with the Bluetooth signal. Yet, since we can handle cases where we receive no signal at a receiver, our algorithm can still achieve accuracies similar to the studies mentioned in Chapter 3.

We also use fingerprinting information in our multilateration algorithm to find the best matching position given RSSIs. Even though, our fingerprinting results does not seem consistent with formulas for conversions between RSSI and distance, they naturally take environmental factors into account which helps us make more accurate positioning predictions. In our case, fingerprinting does not always increase the accuracy for some positions and sometimes decreases the accuracy. Yet, if we have more beacons positioned for fingerprinting, we might get better results for these positions as well.

We play with different parameters mentioned in Chapter 4. Then, comparing different parameter settings, we try to find the best combination. Therefore, parameter tuning is an important step which is better not to skip for different indoor environments having different characteristics. For example, we previously set $movingLimit$ parameter to $1m$ for simulations. However, for real world examples; we observed

that to easily adapt to the new positions fast, we have to increase this parameter to $5m$ so that the particles make larger movements than before. Moreover, we use $0.5m$ for $sensitivityOfResult$ instead of $0.1m$ as we use in simulation results in Chapter 5.

As a result of 50 seconds of wandering around the office for each case separately, IP-PPFONM achieves an average accuracy of $2.28m$, $2.13m$ and $2.47m$ with standard deviations of $1.39m$, $1.77m$ and $1.92m$ for case1, case2 and case3 respectively. If we generate the RSSIs via our simulation tool, IP-PPFONM achieves an average accuracy of $2.50m$, $2.17m$ and $3.01m$ with standard deviations of $1.00m$, $1.08m$ and $1.66m$ for case1, case2 and case3 respectively. We see that for case1 and case2, the *Generated RSSI Data Experiments* perform better whereas for the case3, *Real World RSSI Data Experiments* perform better by comparing the sum of the corresponding means and the standard deviations. However, we have only 4 time steps that we do not make any calculations for case3 in *Generated RSSI Data Experiments* whereas we have 21 in *Real World RSSI Data Experiments*. Therefore, *Generated RSSI Data Experiments* may have said to perform better for case3 as well. This shows us that simulation tool is more optimistic about the results than it should be in reality. Furthermore, simulation tool generally assumes to receive a signal if the beacons are in the range; yet, we see that, in real world; the receivers do not get signals all the time even if the receivers are in the range of the beacons according to the TX powers of the beacons.

Even though the accuracies in *Real World RSSI Data Experiments* are not enough for strict positioning applications which require sub-meter accuracy, they are accurate enough to show which part in a room the POI is close to. These accuracies meet our need to have a system with long battery life and low hardware costs.

6.2 Behavioral Experiments

6.2.1 Signal Blocking Material Tests:

We use two $2.4GHz$ Wi-Fi dongles in this experiment in which one of them is positioned at right and the other is positioned at left with $20cm$ space between them.

We put two aluminum barriers having thicknesses of $0.5cm$ between the two dongles where each barrier covered top, bottom and left position for the right dongle and top, bottom and right positions for the left dongle. We use Wi-Fi signals for this test and the results should be similar to the results below for BLE signals as well except for the fact that BLE signals would be lower in strength. We use aluminum since aluminum is known to block high frequency electromagnetic signals like Bluetooth well [40]. We perform this test while being approximately $50cm$ away from both of the dongles, starting at the right dongle which is positioned at $0°$. The degrees and positions are determined using the coordinate system in Figure 6.14.

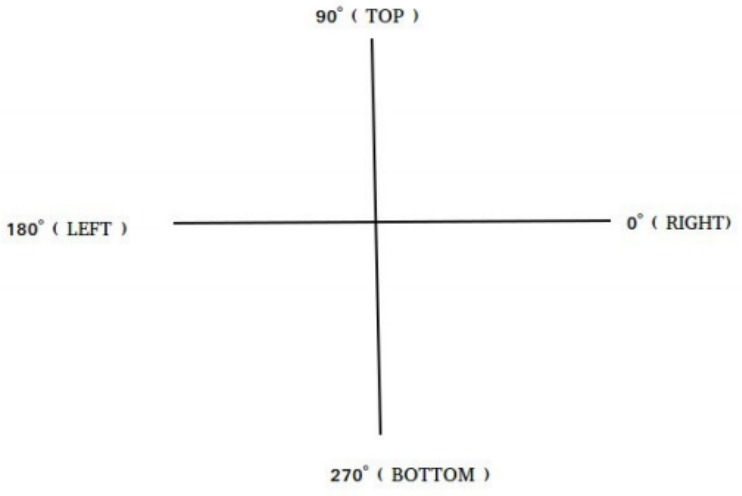
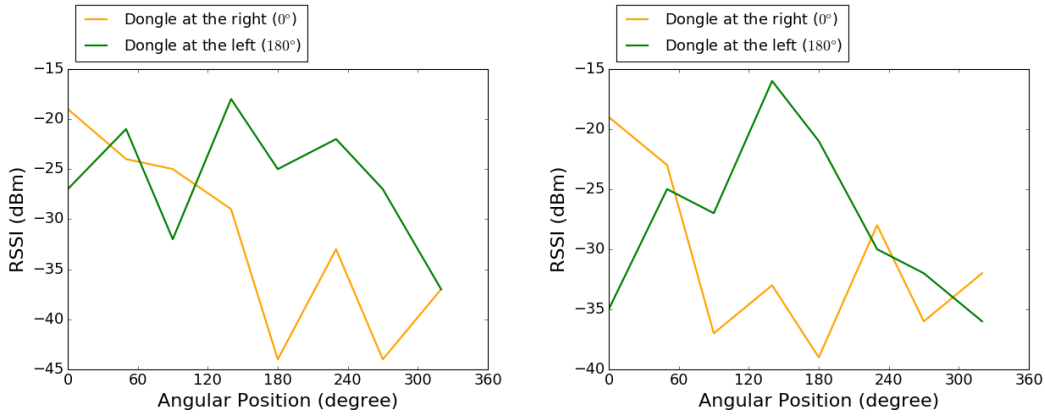


Figure 6.14. Reference Coordinate System Used

6.2.1.1 Results

In fig 6.15, we can see the RSSI results for a beacon that gets hold in different angles and positions to the dongles (receiver devices) to see the effect of material just near the receiver devices.



(a) RSSI with Block

(b) RSSI without Block

Figure 6.15. RSSI Comparison

6.2.1.2 Discussion

As can be seen in Figure 6.15, having a block in between, decreases the RSSI for the dongle at the left preventing the beacon and the left dongle being in line of sight with each other. But at $100°$, somehow, the right dongle has higher RSSI value in the case with blocks than in the case without block. Hence, we can say that the position of the beacon and the receiver devices and the angle between them is also a factor that needs to be taken into account besides the presence of an obstruction alone. To receive stronger signals, we should place the receivers in a position and angle that their BLE chipset is in sight with the surrounding BLE signals as much as possible.

6.2.2 Antenna Role:

For this test, we take RSSI measurements of a beacon at a receiver. Then, we remove the antenna of the receiver and make measurements again. We collect RSSI values for a while and then compare for each case. The beacon we use has $0dBm$ as TX power and therefore; has an expected RSSI of approximately $-70dBm$ at $3m$.

6.2.2.1 Results

As a results of our tests, we observe that the antenna plays a huge role for receiving the RSSI values. The receiver gets RSSI values of approximately $-70dBm$ when we are $3m$ away from the receiver and get signals after $15m$ easily. However, after removing the antenna the RSSI values drop to $-93dBm$ for $3m$ and we cannot get any signal beyond $7m$.

6.2.2.2 Discussion

As the results show, an antenna is a must-to-have for receiver devices. As a further thought, we think we can still use receiver devices when the antenna is removed to check if a beacon is near because if these types of receivers receive a signal, then it must be due to a nearby beacon. In this way, we eliminate the need for checking RSSI values for near proximities.

6.2.3 Beacon Carrying Position Test:

We experiment with different TX powered beacons to determine which one to use for our tests. In our results, we find out that lower TX power beacons are not well suited for indoor environments using signal power. However, they are still suited for indoor environments if we just check whether we receive a signal in our receivers or not. Low TX powered beacons could be used to find positions in indoor environments more accurately since we would only receive the signals of these beacons within a small proximity.

We make the tests with different beacons from different firms. All of these beacons use the iBeacon protocol to transmit signal powers, the signal power results are more or less the same for all of the beacons we test, therefore we do not categorize the firms as A-Firm , B-Firm beacon and so on. Not every firm has the same signal range and same *transmission* (TX) powers. In total, we investigate eight different TX powers which are -40, -30, -20, -16, -12, -8, -4, $0dBm$.

We test the beacons in two ways:

- Carrying them in our pockets.
- Holding them in our hands without blocking the BLE chip of the beacon.

Then, we note the maximum achievable range of the beacons. Note that, when beacons are carried in a pocket no other object is on top of the beacons since otherwise the range information accuracy gets lower significantly. It shows us that if there are objects between the beacons and the receivers and these objects are next to our beacons, it affects the signal propagation considerably. We test all TX powers to see what we can do with different TX powered beacons in the field.

6.2.3.1 Results

We expect that carrying the beacon in pocket blocks the BLE signal and therefore decreases the maximum range achievable by this signal whereas by carrying the beacon in hand, BLE signal can reach a larger range range due to being in line of sight with the receiver. The maximum ranges that can be reached by the emitted BLE signals and the TX powers of the beacons providing these ranges can be summarized as in the Table 6.4.

Table 6.4. TX Power vs Maximum Achievable Range

TX Power (dBm)	**Range (m)**	
	In Hand	**In Pocket**
0	50	25
-4	27	18
-8	22	13
-12	15	9
-16	10	5
-20	7	2
-30	2	0.5
-40	1	0

6.2.3.2 Discussion

As we expect, results show that carrying in pocket decreases the signal power a lot, hence lowering the maximum distance coverage. Therefore, a beacon should not be carried in a bag or next to other objects inside a pocket since carrying in this way would prevent receiving a strong signal from the beacon. We can also see that low powered beacons can be used for proximity solutions since their coverage is very low.

6.2.4 BLE Chipset / Receiver Device Role:

We test different receivers with two different well-known and quality chipsets. We wander around an office environment and check the RSSI values at the receiver devices having different chipsets to see whether the chipset affects the RSSIs.

6.2.4.1 Results

According to our observations, chipsets also affect the signal reception since one of our receiver devices catches signals that the other one cannot catches most of the time.

6.2.4.2 Discussion

According to our results, BLE chipset or receiver device quality is an important factor for accurate positioning and one should buy a quality receiver device.

6.2.5 Obstruction Effect:

For these tests, we hold the beacons when there is a $30cm$ wall completely blocking the way between the receiver and the beacons. Then, we test when there is no wall between the receiver and the beacon to see the effect of the wall for BLE signal penetration.

6.2.5.1 Results

One of the receivers is able to receive signals from a beacon beyond a wall while the other one is not. It shows signal gets severely degraded due to wall and with a receiver having a weak chipset, it is very hard to obtain any signals when there is a thick obstruction between the receiver and the beacons.

6.2.5.2 Discussion

According to the results, we can say that the receivers should be placed in a way that the obstructions between the receivers and the beacons are minimal.

6.2.6 Transmitter / Beacon Factor:

We tested several beacons from different firms, but after some tests for decision making we chose three of them each belonging to a different firm. One of the beacons has Bluetooth 5.0 and the other two has Bluetooth 4.0.

6.2.6.1 Results

After testing signal catching frequency and number of signal signals we caught we saw that Bluetooth 4.0 beacons were enough for us since we did not need more distance coverage and faster signals which come with Bluetooth 5.0.

6.2.6.2 Discussion

The higher the version of Bluetooth, the better the beacon is if the receivers are compatible with the versions of the Bluetooth. As can be seen from the results, if receivers do not have as high Bluetooth version as the beacons, then having a higher versioned beacon may not be necessary.

6.2.7 Transmission Power and Period Factor:

In total, we investigate eight different TX powers which are -40, -30, -20, -16, -12, -8, -4, $0dBm$ and four transmission periods which are $100ms$, $300ms$, $800ms$ and $1000ms$.

6.2.7.1 Results

According to our experiments, we observe that increasing TX power increases the signal stability and provides more healthy decisions for positioning. However as the transmission period decreases, the battery consumption increases which is a trade-off that one should consider. According to our observations in general, BLE signals having $<= 300ms$ advertising interval (transmission period) results in a stable signal environment for signal positioning calculations. We choose $0dBm$ TX power and $1000ms$ advertising interval for real world experiments. Since the advertising interval we choose is lower than $300ms$, our signal is not very stable. Yet, it is necessary to make battery last longer. We do not decrease the TX power to have a higher battery life since low TX powers fade away easily in noisy or large environments which is against our aim to be able to track the POI in large and noisy environments.

6.2.7.2 Discussion

As expected, as the TX power increases or advertisement period decreases, the stability of the signal increases improving the indoor positioning accuracy.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Indoor positioning is getting increased attention with the improvements in wireless signal positioning methods. Some of the use cases of indoor positioning include redirecting people in indoor environments to find their ways just like GPS does in outdoor environments or providing store owners the ability to give location based services to their customers and make customer dwell time analysis.

The wireless signal positioning methods are generally cheaper compared to camera-based positioning methods due to cheaper hardware (no camera) and software requirements (no image processing required). However, signals like *Bluetooth low energy* (BLE) are not very reliable to make positioning estimations without processing the signals first.

In this thesis, we explain different wireless indoor positioning technologies. Among these technologies, we use BLE due to its low cost, low power consumption (high battery life) and ubiquity. The BLE technology improves as the new versions of Bluetooth provides increased speed and signal range capabilities for BLE. Hence, now, this technology is more suitable for indoor positioning compared to previous years.

Moreover, we mention studies performed in the field of indoor positioning with wireless technologies. We explain the environmental setup and accuracy information of these studies. There are different methods and approaches using the BLE technology which are proposed in the literature as mentioned in Chapter 3. We see that some of these methods achieve accuracies higher than our method does. However, proposed approaches generally rely on frequent ($< 1000ms$) and high-powered signals trans-

mitted from beacons or frequent receiver placement in the indoor environment. Also, to the best of our knowledge, the obstructional information is not taken into account directly in the proposed methods in the literature. In the related studies mentioned, we see that obstructional information is considered using indirect methods like fingerprinting without integrating this information into the core of the positioning algorithm itself. Hence, in this thesis, we propose a cost-effective algorithm which does not rely on frequently installed receivers or frequently transmitting beacons where obstructional information in the environment is integrated into the core of the algorithm.

We propose a wireless indoor positioning algorithm IP-PPFONM that can predict the position of a person by receiving only one signal by using the receiver position information for the receivers not getting any signal. If we know that the POI is inside the IE, then the IP-PPFONM algorithm can predict the position without receving any signal, hence the *no-signal* in the algorithm's name. IP-PPFONM is an *obstruction-aware* algorithm since it takes obstructional information in the IE environment into consideration and uses this information to get true distance information given the RSSI. In IP-PPFONM algorithm, to overcome the non-reliable nature of BLE signals, we process the signals by applying prefiltering and postfiltering methods. IP-PPFONM algorithm applies the following methods in order, for positioning the POI:

- efficient receiver placement to increase the possibility of getting BLE signals regardless of the position of the POI in the indoor environment,
- BLE fingerprinting to mitigate multipath effect,
- prefiltering BLE signals to determine which signals should be let to pass for positioning calculations using *running average filtering* (RAF) algorithm,
- obstruction-aware no-signal multilateration algorithm to find the measured position of the POI,
- particle filtering to locate the POI with some uncertainty.

We design a simulation tool to visualize the output of IP-PPFONM algorithm where we can easily point the POI, receiver and obstruction positions in the IE. This tool also visualize how much we are sure about the positioning result using confidence ellipses

formed around the POI position. We show different simulations for different scenarios to show the theoretical accuracies of the IP-PPFONM algorithm for different cases.

We perform the real-world experiments in an office environment which is of size $15m \times 16m$ to see the behavior of our system in a real world indoor environment. We use a BLE fingerprinting method, running average filtering, multilateration, particle filtering and k-means algorithms in the IP-PPFONM algorithm. As a result of $50seconds$ of wandering around in a real world environment, the IP-PPFONM achieves an average accuracy of $2.29m$ with a standard deviation of $1.67m$.

For real-world experiments, we try to use little number of receivers to reduce the hardware cost. We try to find the appropriate beacon parameter settings like advertisement interval and TX power settings to have high battery-life. Having high-battery life means, the less need to replace the beacons with new ones which is an important factor for easing the maintenance of an indoor positioning system.

In this thesis, we use the following materials:

- Receiver devices that support BLE technology.
- BLE beacons which transmit BLE signals.
- A fish-eye lensed camera to verify the positions resulted from our indoor positioning system.

This study can be further extended by trying different TX power and advertisement interval values. Moreover, in this thesis we assume that BLE signals propagate linearly. However, in real life, BLE signals do not propagate linearly and get affected from multipath effect. Hence, in further studies, in terms of collision detection of signals with obstructions, suitable propagation model for BLE signals can be taken into account as well.

REFERENCES

[1] M. Computing, "Data acquisition handbook." https://www.mccdaq.com/handbook/chapt_1.aspx. Accessed on 2019-04-01.

[2] N. G. NM. Sanjeev Arulampalam, Simon Maskell and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE TRANSACTIONS ON SIGNAL PROCESSING,*, 2002.

[3] A. Azeez Khudhair, S. Jabbar, D. Sulttan, D. Wang, and M. Toty, "Wireless indoor localization systems and techniques: Survey and comparative study," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 3, pp. 392–409, 08 2016.

[4] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, pp. 1067–1080, Nov 2007.

[5] M. Uradzinski, H. Guo, X. Liu, and M. Yu, "Advanced indoor positioning using zigbee wireless technology," *Wireless Personal Communications*, vol. 97, pp. 6509–6518, Dec 2017.

[6] "Ieee standard for telecommunications and information exchange between systems - lan/man - specific requirements - part 15: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans)," *IEEE Std 802.15.1-2002*, pp. 1–473, June 2002.

[7] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.1a: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpan)," *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pp. 1–700, June 2005.

[8] B. S. Proprietary, "Bluetooth core specification v5.1."

https://www.bluetooth.com/specifications/bluetooth-core-specification/, Jan 2019. Accessed on 2019-06-08.

[9] M. Woolley, "Exploring bluetooth 5 – going the distance." https://www.bluetooth.com/blog/exploring-bluetooth-5-going-the-distance/, Feb 2017. Accessed on 2019-06-08.

[10] N. Gupta, "Inside bluetooth low energy," 2013.

[11] "Getting started with ibeacon." https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf, 06 2014.

[12] "Eddystone format." https://developers.google.com/beacons/eddystone. Accessed on 2019-06-10.

[13] "Eddystone ephemeral identifier." https://developers.google.com/beacons/eddystone-eid. Accessed on 2019-06-10.

[14] "Who we are." https://www.wi-fi.org/who-we-are/history. Accessed on 2019-06-08.

[15] "Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-1997*, pp. 1–445, Nov 1997.

[16] "Standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pp. 1–512, Nov 1998.

[17] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, June 2007.

[18] "Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

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

[20] Y. Rahayu, T. Abd Rahman, R. Ngah, and P. Hall, "Ultra wideband technology and its applications," pp. 1 – 5, 06 2008.

[21] "Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems." `https://www.fcc.gov/document/revision-part-15-commissions-rules-regarding-ultra-wideband`, August 2010. Accessed on 2019-06-10.

[22] M. Z. Win, D. Dardari, A. F. Molisch, W. Wiesbeck, and J. Zhang, "History and applications of uwb [scanning the issue]," *Proceedings of the IEEE*, vol. 97, pp. 198–204, Feb 2009.

[23] M. Malajner, P. Planinšič, and D. Gleich, "Uwb ranging accuracy," in *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 61–64, Sep. 2015.

[24] G. R. Hiertz, Y. Zang, J. Habetha, and H. Sirin, "Ieee 802.15.3a wireless personal area networks - the mboa approach," in *11th European Wireless Conference 2005 - Next Generation wireless and Mobile Communications and Services*, pp. 1–7, April 2006.

[25] "Message from wimedia." `https://www.wimedia.org/en/index.asp`. Accessed on 2019-06-10.

[26] "Our scope and goals." `https://uwballiance.org/`. Accessed on 2019-06-10.

[27] H. Xu, Y. Ding, P. Li, R. Wang, and Y. Li, "An rfid indoor positioning algorithm based on bayesian probability and k-nearest neighbor," *Sensors*, vol. 17, no. 8, 2017.

[28] H. Desa, M. Rosbi Mohd Sofian, and Z. Shaiful, "Study of integration 2.4ghz and 5.8ghz in rfid tag," 10 2009.

[29] L. Chang, H. Wang, Z. Zhang, Y. Li, and Z. Feng, "Compact single-feed dual-mode antenna for active rfid tag application," *IEEE Transactions on Antennas and Propagation*, vol. 63, pp. 5190–5194, Nov 2015.

[30] L. Lessing, "Looking back [man of high fidelity: Edwin howard armstrong]," *IEEE Potentials*, vol. 3, pp. 41–41, Oct 1984.

[31] H. Roder, "Amplitude, phase, and frequency modulation," *Proceedings of the Institute of Radio Engineers*, vol. 19, pp. 2145–2176, Dec 1931.

[32] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, pp. 1–1, 11 2015.

[33] S. Subedi and J.-Y. Pyun, "Practical fingerprinting localization for indoor positioning system by using beacons," *Journal of Sensors*, vol. 2017, pp. 1–16, 12 2017.

[34] H. Ai, W. Huang, Y. Yang, and L. Liao, "The ble fingerprint map fast construction method for indoor localization," in *Algorithms and Architectures for Parallel Processing* (J. Vaidya and J. Li, eds.), (Cham), pp. 326–340, Springer International Publishing, 2018.

[35] Q. W. Yang Li, Jianhua Zhang, *Adaptive Sliding Mode Neural Network Control for Nonlinear Systems*, pp. 151–152. Elsevier Science, 2018.

[36] W. , X. Wang, O. Bischoff, R. Laur, and S. Paul, "Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications," *Procedia Chemistry*, vol. 1, 09 2009.

[37] O. S. Adeniran .A.O., Ajao S.O., "The basics of signal attenuation." `https://www.dataloggerinc.com/wp-content/uploads/`

2016/11/16_Basics_of_signal_attenuation.pdf. Accessed on 2019-06-06.

[38] R. Wilson, "Propagation losses through common building materials." https://www.am1.us/wp-content/uploads/Documents/E10589_Propagation_Losses_2_and_5GHz.pdf, August 2002. Accessed on 2019-06-06.

[39] D. Duran and H. Kadoglu, "Research on electromagnetic shielding with copper core yarns," *Tekstil ve Konfeksiyon*, vol. 22, pp. 354–359, 10 2012.

[40] R. L. William Benjamin, "Radio frequency patch antenna and system for permitting secure access to a restricted area." https://patents.google.com/patent/US20160300413, 10 2016.

[41] A. L. Felicie, *Multipath Propagation*. Salve Regina University, 2012.

[42] S. Faruque, *Multipath Propagation*, pp. 27–38. 01 2015.

[43] V. Cantón Paterna, A. Calveras, J. Paradells Aspas, and M. Alejandra Pérez Bullones, "A bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and kalman filtering," *Sensors*, vol. 17, p. 2927, 12 2017.

[44] R. Ma, Q. Guo, C. Hu, and J. Xue, "An improved wifi indoor positioning algorithm by weighted fusion," *Sensors (Basel, Switzerland)*, vol. 15, pp. 21824–43, 09 2015.

[45] C. Park and S. H. Rhee, "Indoor positioning using wi-fi fingerprint with signal clustering," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 820–822, Oct 2017.

[46] A. Sahar and D. Han, "An lstm-based indoor positioning method using wi-fi signals," pp. 1–5, 08 2018.

[47] K. Wang, X. Yu, Q. Xiong, Q. Zhu, W. Lu, Y. Huang, and L. Zhao, "Learning to improve wlan indoor positioning accuracy based on dbscan-krf algorithm from rss fingerprint data," *IEEE Access*, vol. PP, pp. 1–1, 05 2019.

[48] W. Chen, Q. Chang, H. tao Hou, and W. ping Wang, "A novel clustering and kwnn-based strategy for wi-fi fingerprint indoor localization," *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, pp. 49–52, 2015.

[49] M. Uradzinski, H. Guo, X. Liu, and M. Yu, "Advanced indoor positioning using zigbee wireless technology," *Wireless Personal Communications*, 08 2017.

[50] C. Ou, C. Chao, F. Chang, S. Wang, G. Liu, M. Wu, K. Cho, L. Hwang, and Y. Huan, "A zigbee position technique for indoor localization based on proximity learning," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 875–880, Aug 2017.

[51] V. Bianchi, P. Ciampolini, and I. De Munari, "Rssi-based indoor localization and identification for zigbee wireless sensor networks in smart homes," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, pp. 566–575, Feb 2019.

[52] H. Yin, W. Xia, Y. Zhang, and L. Shen, "Uwb-based indoor high precision localization system with robust unscented kalman filter," in *2016 IEEE International Conference on Communication Systems (ICCS)*, pp. 1–6, Dec 2016.

[53] P. Dabove, V. Di Pietra, M. Piras, A. A. Jabbar, and S. A. Kazim, "Indoor positioning using ultra-wide band (uwb) technologies: Positioning accuracies and sensors' performances," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 175–184, April 2018.

[54] R. W. C. Ling, A. Gupta, A. Vashistha, M. Sharma, and C. L. Law, "High precision uwb-ir indoor positioning system for iot applications," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pp. 135–139, Feb 2018.

[55] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," vol. 2, pp. 775 – 784 vol.2, 02 2000.

[56] L. Ni, Y. Liu, Y. Cho Lau, and A. Patil, "Landmarc: Indoor location sensing using active rfid," vol. 10, pp. 407– 415, 04 2003.

[57] S. Saab and Z. Nakad, "A standalone rfid indoor positioning system using passive tags," *Industrial Electronics, IEEE Transactions on*, vol. 58, pp. 1961 – 1970, 06 2011.

[58] H. Xu, Y. Ding, P. Li, R. Wang, and Y. Li, "An rfid indoor positioning algorithm based on bayesian probability and k-nearest neighbor," *Sensors*, vol. 17, p. 1806, 08 2017.

[59] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "Fm-based indoor localization," 06 2012.

[60] S. Yoon, K. Lee, Y. Yun, and I. Rhee, "Acmi: Fm-based indoor localization via autonomous fingerprinting," *IEEE Transactions on Mobile Computing*, vol. 15, pp. 1318–1332, June 2016.

[61] A. Popleteev, "Indoor localization using ambient fm radio rss fingerprinting: A 9-month study," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 128–134, Aug 2017.

[62] F. Subhan, H. Hasbullah, A. Rozyyev, and S. T. Bakhsh, "Indoor positioning in bluetooth networks using fingerprinting and lateration approach," in *2011 International Conference on Information Science and Applications*, pp. 1–9, April 2011.

[63] N. Khan, "Positioning in bluetooth networks using lateration approach-a comparative study," *Sci.Int.(Lahore),26(5),2077-2082,2014*, 12 2014.

[64] A. A. Kalbandhe and S. C. Patil, "Indoor positioning system using bluetooth low energy," in *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, pp. 451–455, Dec 2016.

[65] P. Kriz, F. Maly, and K. Tomáš, "Improving indoor localization using bluetooth low energy beacons," *Mobile Information Systems*, vol. 2016, pp. 1–11, 04 2016.

[66] A. Ozer and E. John, "Improving the accuracy of bluetooth low energy indoor positioning system using kalman filtering," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 180–185, Dec 2016.

[67] S. Memon, M. M. Memon, F. K. Shaikh, and S. Laghari, "Smart indoor positioning using ble technology," in *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pp. 1–5, Nov 2017.

[68] M. Sie and C. Kuo, "Indoor location estimation using ble beacon with multiple transmission power levels," in *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, pp. 323–324, June 2017.

[69] Q. H. Nguyen, P. Johnson, T. T. Nguyen, and M. Randles, "Optimized indoor positioning for static mode smart devices using ble," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, Oct 2017.

[70] I. Radoi, G. Gutu, T. Rebedea, C. Neagu, and M. Popa, "Indoor positioning inside an office building using ble," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 159–164, May 2017.

[71] M. Terán, J. Aranda, H. Carrillo, D. Mendez, and C. Parra, "Iot-based system for indoor location using bluetooth low energy," in *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1–6, Aug 2017.

[72] V. Cantón Paterna, A. Calveras, J. Paradells Aspas, and M. Alejandra Pérez Bullones, "A bluetooth low energy indoor positioning system with channel diversity, weighted trilateration and kalman filtering," *Sensors*, vol. 17, p. 2927, 12 2017.

[73] S. Kajioka, T. Mori, T. Uchiya, I. Takumi, and H. Matsuo, "Experiment of indoor position presumption based on rssi of bluetooth le beacon," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pp. 337–339, Oct 2014.

[74] Z. Zuo, L. Liu, L. Zhang, and Y. Fang, "Indoor positioning based on bluetooth low-energy beacons adopting graph optimization," *Sensors*, vol. 18, p. 3736, 11 2018.

[75] H. Yanagaimoto, K. Hashimoto, and T. Matsuo, "Indoor positioning estimation using ble beacons," in *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, pp. 1–6, Nov 2018.

[76] M. Teran, H. Carrillo, and C. Parra, “Wlan-ble based indoor positioning system using machine learning cloud services,” pp. 1–6, 11 2018.

[77] H. Li and H. Ma, “A low complexity low power indoor positioning system based on wireless received signal strength,” in *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6, Sep. 2018.

[78] K. Huang, K. He, and X. Du, “A hybrid method to improve the ble-based indoor positioning in a dense bluetooth environment,” *Sensors*, vol. 19, p. 424, 01 2019.

[79] K. Mekki, E. Bajic, and F. Meyer, “Indoor positioning system for iot device based on ble technology and mqtt protocol,” 04 2019.