A 3D Ubiquitous Multi-Platform Localization and Tracking System for Smartphones

by

Seyyed Mahmood Jafari Sadeghi

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2017 by Seyyed Mahmood Jafari Sadeghi
Abstract

A 3D Ubiquitous Multi-Platform Localization and Tracking System for Smartphones

Seyyed Mahmood Jafari Sadeghi

Doctor of Philosophy

Graduate Department of Electrical and Computer Engineering

University of Toronto

2017

We have designed and implemented an indoor/outdoor localization system utilizing several sources of information to provide the accurate location of a smartphone/tablet both indoors and outdoors. In this system, we merge the traditional indoor localization techniques based on Wi-Fi fingerprinting with the recent methods, which are mostly based on Bluetooth Low Energy (BLE) beacons, to acquire a higher accuracy of positioning and also support a wider range of smartphones such as Android and iOS devices. A new format for the advertisement packets of BLE beacons was proposed which embeds all parameters of the beacon including its location. Also, a transparent scheme is proposed and implemented, which combines indoor localization techniques with Global Positioning System (GPS) to increase the accuracy of the indoor localization and also provides us with a soft switching between the GPS and indoor positioning. We have shown that using a Medium Access Control (MAC) filtering method, we can reduce the size of Wi-Fi radiomap for fingerprinting techniques and hence reduce their complexity and run time. The problem of tracking and floor detection is also investigated and promising results are achieved by using the sensors such as barometer and gyroscope. We have also addressed the problem of large scale indoor positioning in which we have the fingerprinting database for thousands of buildings worldwide. Efficient algorithms have been proposed to reduce the complexity and the management overhead of the Wi-Fi fingerprints. A cooperative method was proposed that allows iOS devices through BLE packets broadcast by Android phones/tablets to localize themselves using Wi-Fi fingerprints inside a building. Finally, a sensor transmission system for Android devices was built, which allows us to simulate our algorithms in real-time using real-world data on a PC running Matlab software.
Contents

List of Figures vi

List of Tables xi

List of Abbreviation xii

1 An introduction to indoor localization 1
   1.1 Background and related work ........................................... 1
   1.2 Block diagram of the proposed indoor positioning system ........ 6
   1.3 Contributions .................................................................. 9
   1.4 Organization of this thesis .............................................. 10

2 Android sensors transmission system 11
   2.1 An introduction to Android sensors ................................. 11
       2.1.1 Wi-Fi scanner .......................................................... 12
       2.1.2 Bluetooth scanner ..................................................... 12
       2.1.3 GPS .................................................................. 13
       2.1.4 Accelerometer .......................................................... 13
       2.1.5 Gyroscope ............................................................... 13
       2.1.6 Magnetometer .......................................................... 14
       2.1.7 Barometer ............................................................... 14
       2.1.8 Orientation ............................................................. 14
       2.1.9 Game rotation vector ................................................. 16
       2.1.10 Linear accelerometer ............................................... 16
   2.2 Server application (Android side) ...................................... 16
   2.3 Client application (MATLAB/Simulink side) ....................... 17
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Implementation of signal processing algorithms using Java Native Interface</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>Step counter for Pedestrian Dead Reckoning</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Floor detection using barometer</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>Challenges of using barometer for floor detection</td>
<td>27</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Drift over time</td>
<td>27</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Building diversity</td>
<td>28</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Device diversity</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Dynamic behavior of the barometer sensor</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Wi-Fi combined with the barometer sensor</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>GPS-compatible indoor localization</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Matching of the indoor maps to the outdoor maps</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>Floor to floor matching</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Using the GPS to assist indoor localization</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Indoor map representation as a graph</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Bluetooth Low Energy for indoor localization</td>
<td>44</td>
</tr>
<tr>
<td>5.1</td>
<td>iBeacon vs proposed advertisement packet format</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Bluetooth Low Energy experiment setup</td>
<td>46</td>
</tr>
<tr>
<td>5.3</td>
<td>Bluetooth Low Energy localization techniques</td>
<td>47</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Geometrical methods for BLE-based indoor positioning</td>
<td>47</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Bluetooth fingerprinting for indoor localization</td>
<td>51</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusion of indoor localization using BLE</td>
<td>51</td>
</tr>
<tr>
<td>6</td>
<td>Wi-Fi fingerprinting and sensor fusion using Kalman filter</td>
<td>58</td>
</tr>
<tr>
<td>6.1</td>
<td>The weighted kNN algorithm</td>
<td>58</td>
</tr>
<tr>
<td>6.2</td>
<td>MAC Filtering of Wi-Fi APs</td>
<td>61</td>
</tr>
<tr>
<td>6.3</td>
<td>3D location tracking using Kalman filter</td>
<td>62</td>
</tr>
<tr>
<td>6.4</td>
<td>The experimental results</td>
<td>68</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Comparison of BLE-based indoor localization methods</td>
<td>69</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Effect of Wi-Fi MAC filtering on localization accuracy</td>
<td>70</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Resetting the Kalman filter at turns</td>
<td>70</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Combining BLE-based indoor positioning and Wi-Fi fingerprinting</td>
<td>79</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Comparison of floor detection methods</td>
<td>81</td>
</tr>
</tbody>
</table>
6.4.6 Overall indoor positioning accuracy ................................. 84

7 Large-scale indoor positioning and cooperative localization 86

7.1 The proposed algorithm for large-scale indoor positioning ....................... 87
  7.1.1 Adding a reference point to the radiomap .................................. 90
  7.1.2 Deleting a reference point from radiomap .................................. 91
  7.1.3 Localizing the user by using online Wi-Fi scan ........................... 91

7.2 Evaluation of the performance of the server-based localizer ......................... 92

7.3 Cooperative indoor localization ........................................ 94

7.4 Conclusion and future work ........................................ 98

Bibliography ................................................................. 100
List of Figures

1.1 The block diagram of our indoor positioning system. ........................................ 6
1.2 Conventional localization server on the left vs. the proposed scalable localization server
on the right. ................................................................. 8

2.1 Device coordinates for Android phones/tablets [1]. ........................................ 13
2.2 Device orientation in terms of azimuth, roll and pitch. Speed of rotation over each of
these axes is measured by gyroscope. ........................................ 14
2.3 Real world coordinates for Android sensors [1]. ........................................... 15
2.4 Rotation matrix $R_{\text{phone}}$ of the phone translates the device coordinates shown on the left
side to the real-world coordinates shown on the right side for Android phones/tablets [1]. 15
2.5 Server application for Android sensor transmission system installed on a Nexus 5 phone. 17
2.6 Simulink model receiving real-time sensor data from Android phone/tablet. .......... 18
2.7 Configuration window of the ”Android phone” block in Simulink model. You need to set
the IP address and the sampling interval. ..................................... 18
2.8 A sample reading of 3D linear accelerometer in Simulink software. ................. 19
2.9 The block diagram of step counter. ......................................................... 20
2.10 Rotated 3D linear acceleration of the Android device with respect to real-world coordinates
of Figure 2.3. As seen, most of the acceleration variations are among the $z$ axis. .... 21
2.11 The cut function $C$, cuts the accelerations below $\frac{2\pi}{\tau}$ and passes through, the larger
accelerations. ................................................................. 22
2.12 On the top, the output of cut function, in the middle, the output of zero crossing detector
and on the bottom, the output of the step counter is shown. .......................... 23

3.1 Using of back and front window to monitor rapid changes in the altitude, to detect a floor
change is explained. This figure is borrowed from [2]. ................................. 26
3.2 Barometer readings for Bahen building on a sunny day. Floor number increases from top to the bottom. ................................................. 27

3.3 Histogram of the noise of the barometer sensor. ................................................. 28

3.4 The barometric pressure readings drift over time due to the variations in temperature, humidity, etc. ................................................. 29

3.5 Barometer readings for the Galbraith building on a sunny day. Floor number increases from top to the bottom. ................................................. 30

3.6 Barometer reading of two different Android phones at the same time. The absolute pressure reading is different but dynamic changes are the same. ................................................. 31

3.7 Abrupt changes in pressure as the user takes stairs from the fourth floor to the fifth floor of Bahen building. ................................................. 32

3.8 The barometric reading when the user takes stairs from fourth to the first floor. ........ 33

3.9 Changes in the pressure readings when the user takes the elevator from the first to the fourth floor. ................................................. 34

3.10 Changes in the pressure readings when the user takes the elevator from the fourth to the first floor. ................................................. 34

3.11 Differentiated barometric readings as the user takes stairs from fourth floor to the fifth floor 35

3.12 Differentiated barometric readings as the user takes stairs from fourth floor to the first floor. ................................................. 35

3.13 The vertical speed of the elevator calculated from readings of the pressure sensor in Figure 3.9. ................................................. 36

3.14 The vertical speed of the elevator calculated from readings of the pressure sensor in Figure 3.10. ................................................. 36

4.1 Raw maps of the fourth and the fifth floor of the Bahen building. As seen, the two maps have different scales and may not match on each other. ................................................. 40

4.2 The two floor plans are matched. Both have the same size and rotation and if they are put on the top of each other, all points match. ................................................. 41

4.3 The indoor map for the Bahen building by the Google Maps software [3]. Currently it only supports three floors of the Bahen building. ................................................. 42
4.4 On the left, you see a spot on the fourth floor of the Bahen building on which GPS signal is available. The true location is shown by a circle and the estimated locations using GPS is marked by stars. On the right, we have the main entrance of the building. As seen, the estimated locations are very close to the true location of the user in this case.

4.5 Graph representation of the fourth floor of the Bahen building.

5.1 TodHQ beacon on the left [4] and its heart, BLE112 module on the right [5].

5.2 The iBeacon packet format. Major and Minor IDs vary from one device to another.

5.3 The proposed BLE packet format. Coordinates and unique ID vary from one device to another.

5.4 Part of the experiment setup on the fourth floor of Bahen building, University of Toronto. The solid circles show the position of Wi-Fi reference points and the × marks show the location of BLE beacons and the solid triangles, which are labeled from A to F show the test points to measure the indoor localization error. The size of room BA-4148 is approximately 11m × 6m.

5.5 Positioning using the trilateration technique. The intersection of the three circles will be the estimated location of the user [6].

5.6 Part of the experiment setup on the fourth floor of Bahen building, University of Toronto. The solid circles show the position of BLE fingerprint reference points and the × marks show the location of BLE beacons and the solid triangles, which are labeled from A to E show the test points to measure the indoor localization error of BLE fingerprinting. The size of room BA-4148 is approximately 11m × 6m.

5.7 On the left the software for analyzing the proposed advertisement packets is shown and on the right the result for BLE localization inside our lab is displayed. The * is the true location and the circle is estimated location of the smartphone.

5.8 The 3D radiation pattern of BLE112 module from top, front and side views [5].

5.9 Propagation loss of free space channel vs. plane earth channel for BLE112 module [5].

6.1 The CDF of localization error for different BLE localization methods is depicted. Four methods of BLE-based indoor positioning, discussed in Section 5.3, are compared. It seems that localization using the trilateration technique works the best in our experiment.

6.2 Mean error of localization with and without MAC filtering for different training set sizes.

6.3 Standard deviation of localization error with and without MAC filtering for different training set sizes.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4</td>
<td>Probability of finding the correct nearest neighbor with and without MAC filtering for different training set sizes.</td>
</tr>
<tr>
<td>6.5</td>
<td>The CDF of localization error with or without MAC filtering for training size of 65.</td>
</tr>
<tr>
<td>6.6</td>
<td>The first path of the walk on the fourth floor of Bahen building in which the user makes two sharp 90° turns. Consider the rotation of the floor map compared to the geomagnetic north (θ ≈ −17° for Bahen building).</td>
</tr>
<tr>
<td>6.7</td>
<td>Tracking of the user using the simple Kalman filter for the walk of Figure 6.6.</td>
</tr>
<tr>
<td>6.8</td>
<td>Reading of game rotation vector for the walk of Figure 6.6.</td>
</tr>
<tr>
<td>6.9</td>
<td>Angular velocity over z axis measured by the gyro sensor, for the path of Figure 6.6. By using a proper threshold (solid red lines), we can detect the turns. The Kalman filter is reset at times t = 30s and t = 50s.</td>
</tr>
<tr>
<td>6.10</td>
<td>Tracking of the user using the Kalman filter integrated with game vector for the walk of Figure 6.6.</td>
</tr>
<tr>
<td>6.11</td>
<td>Tracking of the user using the Kalman filter integrated with game vector and map information for the walk of Figure 6.6. The red circles show the point on which we have reset the Kalman filter.</td>
</tr>
<tr>
<td>6.12</td>
<td>The second path of the walk on the fourth floor of Bahen building. The user makes only one turn.</td>
</tr>
<tr>
<td>6.13</td>
<td>Tracking of the user using the simple Kalman filter for the walk of Figure 6.12.</td>
</tr>
<tr>
<td>6.14</td>
<td>Tracking of the user using the Kalman filter integrated with game vector for the walk of Figure 6.12.</td>
</tr>
<tr>
<td>6.15</td>
<td>Tracking of the user using the Kalman filter and the map information for the walk of Figure 6.12.</td>
</tr>
<tr>
<td>6.16</td>
<td>The CDF of localization error for different BLE localization methods, combined with Wi-Fi fingerprinting.</td>
</tr>
<tr>
<td>6.17</td>
<td>The CDF of localization error for different hybrid localization methods.</td>
</tr>
<tr>
<td>6.18</td>
<td>Mean error of localization for different points in the vicinity of the testing area. The location of points A to F is shown in Figure 5.4.</td>
</tr>
<tr>
<td>6.19</td>
<td>The implementation of the system on Android OS. Location results are converted to geographic coordinates using (4.1) and displayed on Google Maps. The blue circle shows the margin of localization error. The star shows the true location and the circle shows the estimated location.</td>
</tr>
</tbody>
</table>
7.1 The block diagram of the server-based indoor positioning system. The Wi-Fi scan results are sent to the localization server via the Internet and the received location result is injected to the Kalman filter.

7.2 Connection diagram of the server-based indoor localization for Android devices. The Android smartphone collects the Wi-Fi scan and wraps them in JSON format and sends the data to localization server through HTTPS. The localization application finds the position of the user and responses to the client with its location, wrapped in JSON packet.

7.3 The response time of the server vs. the number of RPs in the radiomap is depicted for adding and removing RPs and also for localization using Wi-Fi scans. The number of APs for each RP is chosen uniformly and randomly between 1 to 39.

7.4 The response time of the server vs. the number of requests per second is displayed. The server has almost reached its capacity as the service rate is almost equal to the input rate of the queue.

7.5 Android phone localizes itself and broadcasts its location to iOS devices using BLE.

7.6 Android phone sends both Wi-Fi and BLE scans to the localization server.

7.7 The proposed localization schema for iOS devices.

7.8 The cooperation between Android and iOS devices is shown in this figure. We have put three Nexus 5X devices at the three corners of the lab BA-4148 and an iPhone 6s device in the middle. The Android devices localize themselves by Wi-Fi fingerprinting and then broadcast their locations through BLE. The blue circles are the reference point for the Wi-Fi fingerprinting based positioning.
## List of Tables

6.1 Three virtual APs are broadcast from the same physical AP. .................................................. 61
6.2 Number of Wi-Fi APs stored in the database before and after MAC filtering. .......................... 62
6.3 Parameters of indoor localizer of Bahen building. ................................................................. 68
6.4 Mean and variance of error of localization for different BLE methods, showing that the 
    trilateration technique, discussed in Subsection 5.3.1, has the least mean error. ..................... 69
6.5 Mean, variance and maximum error of localization without MAC filtering, with MAC 
    filtering using L2 norm and with MAC filtering using L1 norm. ............................................. 70
6.6 Mean and variance of error of localization for different Wi-Fi+BLE methods. ......................... 80
6.7 Mean and variance of error of localization for different hybrid methods. .............................. 81
6.8 Confusion matrix of floor detection only using Wi-Fi fingerprinting using kNN method. .......... 82
6.9 Confusion matrix of floor detection combining Wi-Fi fingerprinting and barometer sensor 
    (proposed method). .................................................................................................................... 82
6.10 Confusion matrix of floor detection using Wi-Fi fingerprinting using k-means-clustering [7]. 83
6.11 Confusion matrix of floor detection combining Wi-Fi fingerprinting and barometer sensor 
    [2]. ............................................................................................................................................. 83
6.12 Comparison of different floor detection methods. .................................................................. 83
6.13 Accuracy result of indoor localizer of Bahen building .......................................................... 85
7.1 AP-RP database describes the radiomap (a.k.a AP-RP-TABLE). .............................................. 87
7.2 Table of the reference points and their corresponding positions, the visible APs and their 
    RSS values for that AP (a.k.a RP-LOC-AP-TABLE). ............................................................... 88
7.3 Parameters of the PC running large-scale localization algorithms. ......................................... 92
7.4 Parameters of the radiomap used to verify the capacity of the server .................................. 93
7.5 Accuracy result of indoor localizer of Bahen building .......................................................... 98
List of Abbreviations

AP  Access Point
BLE  Bluetooth Low Energy
BST  Binary Search Tree
CH  Cluster Head
CS  Compressive Sensing
DOA  Direction Of Arrival
GPS  Global Positioning System
IMU  Inertial Measurement Unit
JNI  Java Native Interface
JSON  Java Script Object Notation
kNN  k-Nearest-Neighbors
LBS  Location-Based Service
MAC  Medium Access Control
NN  Neural Network
OS  Operating System
PDR  Pedestrian Dead Reckoning
RSS  Received Signal Strength
SVM  Support Vector Machine
TDOA  Time Difference of Arrival
TOA  Time Of Arrival
Chapter 1

An introduction to indoor localization

In the last decade, Location Based Services (LBS) have attracted considerable attention. These services include location based advertisement, user navigation, etc. The basis for all of these services is the acquiring of user’s location. As we know, there exists a complete solution to find user’s location for outdoor areas and that solution is the Global Positioning System (GPS). However, due to the lack of satellite signal, GPS does not usually work at all or works very inaccurately indoors. Researchers have been working to find a similar solution for indoor areas [8]. However, a comprehensive approach for the indoor positioning does not exist yet. There are many approaches to find user’s location inside buildings but currently, none of these approaches works on all phones and tablets.

In this thesis, our focus is first to develop a more comprehensive solution for indoor/outdoor localization and tracking of user’s phone/tablet, which can work on most of the commercial phones/tablets like Android/iOS/Windows and the second focus is to improve the performance and accuracy of indoor localization. Furthermore, we consider the problem of large scale indoor positioning in which we have the large database of Wi-Fi fingerprints from multiple buildings. This is discussed in more details in Chapter 7.

1.1 Background and related work

Due to the lack of signal from GPS satellites inside buildings, the problem of indoor positioning is different from outdoor localization. Even if we were able to receive the signal from GPS satellites,
the multi-path effect would have caused inaccurate results (since there is no line of sight to the GPS satellites). Researchers have investigated many diverse techniques to address the problem of indoor positioning. Some of these techniques are discussed here:

- **Wi-Fi fingerprinting**: The most widely used technique for indoor positioning is Wi-Fi fingerprinting [8][9][10][11]. There has been an intensive research on this topic. Wi-Fi fingerprinting has two steps: offline data collection phase and online localization phase. In the offline phase of Wi-Fi fingerprinting, a database of visible Access Points (APs) is created which is called a radiomap. This database contains the visible APs and their corresponding Received Signal Strength (RSS) for several locations called *Reference Points* (RPs). In the online phase of Wi-Fi fingerprinting, we can use a variety of machine learning algorithms, ranging from the traditional methods such as k-Nearest-Neighbor (kNN) [12], to the recent more complicated ones based on Compressive Sensing (CS) [13]. Using the machine learning techniques in the online phase, we estimate the users' locations by comparing the Wi-Fi online scans to the Wi-Fi scan of reference points. The main advantage of using this technique is that no extra hardware is required for indoor positioning. But on the other hand, the major problem is that iOS devices do not allow the third-party applications to scan for Wi-Fi APs.

There also has been efforts to reduce the time required to build the radiomap. Automatic methods using crowd-sourcing mechanisms have been proposed in [14]. Usually a tracking system based on particle filters [15][16] or the Kalman filter is added to increase the accuracy of localization in such systems [10].

- **Bluetooth-based indoor localization**: The Bluetooth connectivity, and its complement BLE, has also been widely considered for indoor positioning [17][18][19][20][21][22][23][14][24][25][26]. Bluetooth-based indoor localization methods fall into two categories:

1. Bluetooth fingerprinting may also be used for indoor localization, exactly the same way as Wi-Fi fingerprinting [19][20][21][22]. Authors in [19] have studied three different machine learning algorithms kNN, Neural Networks (NNs) and Support Vector Machine (SVM) for indoor localization based on Bluetooth fingerprinting and have concluded that kNN regression works the best for Bluetooth fingerprinting.

   Unlike Wi-Fi scanning, Bluetooth scanning is supported on almost all platforms including iOS. Also, there has been research on the efficient placement of these Bluetooth devices, to achieve better accuracy of positioning [23]. Bluetooth fingerprinting is discussed in more
details in Subsection 5.3.2.

2. Geometrical methods: If the location of the Bluetooth beacons are known, we can use geometrical techniques to estimate the location of the receiver [17][18][24]. The trilateration method has been used in [18] to estimate the position of the user. Authors have proposed the proximity method (also known as the closest beacon) in [17], for BLE-based indoor localization. In [24], the weighted averaging of the location of Wi-Fi APs and Bluetooth beacons is recommended to find a more accurate estimation of the position.

The details of selected methods are discussed in Chapter 5 and the performance of these systems is compared to the proposed system in Subsection 6.4.1.

- **Time Of Arrival (TOA):** Another technique of indoor positioning is to place RF [27] or ultrasound beacons [28] at known locations inside a building and measure the distance of the smartphone to these beacons. The distance measurement is usually done via TOA or Time Difference of Arrival (TDOA) by estimating the travel time of the packets received from the access points. After the distance is measured, the methods such as Trilateration [6] might be used to localize the user. TOA and TDOA require the timing synchronization on the smartphone and the access point. The need for complicated hardware in the building and also on the smartphone makes this technique practically undesirable. Direction of Arrival (DOA) techniques can also be used for indoor localization [29]. Similar to TOA, the DOA technique requires extra infrastructure (multiple antennas) on the APs or on the smartphone, which is not suitable for the currently installed commercial infrastructure.

- **Proximity methods:** These methods rely on proximity sensors such as RFID tags or IR sensors [30][31], to obtain the location of the user as he/she passes by one of the installed tags at known positions. As these proximity tags usually have a short range, many of them should be installed in a building to cover the whole area.

- **Dead reckoning:** These systems only rely on the samples gathered from the inertial sensors of the device as well as the initial location of the user, which is usually obtained by GPS [32][33][34][35]. Using the sensors such as accelerometer and magnetometer, these systems find the displacement and the direction of the user to predict the future location. Since such systems have no independent observation to correct their estimated location, they accumulate positioning error and become inaccurate after a while. Authors in [25] recommend using of BLE beacons as independent observations to mitigate the accumulation of localization error. The barometer sensor has also
been investigated in some of these systems to get a 3D indoor localization based on dead reckoning [36][37][38][39][40].

- Vision-based indoor localization methods: The camera of smartphones has also been considered as a source of information for indoor localization [41]. An indoor localization error of below one meter is claimed to be obtained using Logo-based [42] or OCR-aided [43] methods. One major drawback of the camera-based indoor localization is that it requires the user to hold his/her smartphone in hand, pointing it to certain directions. Also the camera-based indoor localization techniques are usually suitable for fine localization in a limited test experiment setup and are expensive in terms of computation power requirement, so they are not good candidates to be implemented in our proposed system.

- Floor detection in 3D indoor positioning systems: Though most of the commercial indoor positioning systems lack the feature of floor detection, the floor detection has been addressed in literature frequently [7][44][2][40][36][38][37][39]. Some systems including [7] use only Wi-Fi fingerprints to detect the floor. A K-means-based method has been proposed, to cluster the reference points of different floors and also to detect the floor number. On the other hand, some systems merely rely on the barometer readings to detect the floor number [36][38][37][39]. The system proposed in [37] is designed to help the blind to find their desired floor number in a multi-floor building. This system relies on the barometric reading of a reference sensor, placed at a base station, to calculate the relative altitude from the base station (e.g. first floor). We show in Subsection 3.1.3 that this method does not work on smartphones. The system proposed in [38] uses dedicated hardware (IMU and barometer), mounted on the user. The proposed system uses the accelerometer sensor to verify the movement of the user and monitors the changes in the air pressure as the user moves, to detect the new height of the user.

Some papers have recommended the combination of Wi-Fi fingerprinting and barometric readings to detect the floor number more accurately [2]. To overcome the effects of environmental factors on floor detection, authors have proposed using a back window and front window in [2]. The initial floor is detected by using Wi-Fi fingerprinting in their proposed system. A crowd-sourcing approach is suggested in [44] to cluster of Wi-Fi fingerprints by using K-means clustering, aided by the barometer sensor.

The details of selected methods are discussed in Chapter 3 and the performance of these systems is compared to the proposed system in Subsection 6.4.5.
• Hybrid indoor localization systems: Some systems, including the system we have proposed in this thesis, combine multiple sources of information for indoor localization. The system discussed in [9] utilizes both the inertial sensors and the map information to get better accuracy of positioning. The system proposed in [25] fuses the localization of BLE along with the inertial sensors. A room-level indoor positioning has been obtained by combining Wi-Fi and BLE in [14]. Authors in [17] also propose using Wi-Fi in conjunction with BLE. A map of error of Wi-Fi and BLE is created in the offline phase, and in the online phase, either BLE or Wi-Fi fingerprinting might be used, based on their accuracy for that specific area. The indoor positioning technique discussed in [24] treats both Wi-Fi APs and BLE beacons in the same manner and estimates the position of the user by calculating a weighted average of the location of the Wi-Fi APs and BLE beacons. Chen et al. [24] assume that the location of Wi-Fi APs and iBeacons are both known. Then, finding the distance of the receiver to the APs and iBeacons, they estimate the position of the smartphone by weighted averaging of the location of APs and iBeacons. Chen et al. propose using an extended Kalman filter, to combine Pedestrian Dead Reckoning (PDR) with the location acquired by using Wi-Fi APs and iBeacons. As we assume that the locations of the Wi-Fi APs are not known in practice, we cannot directly compare their method with our proposed hybrid system. However, we compare their methods with other methods which combine BLE and Wi-Fi in Section 6.4.4.

As the indoor localization error highly depends on the test environment, and the amount of Wi-Fi scans collected in the offline phase, or the number of beacons installed for testing, it is difficult to compare these methods with each other, but generally, the hybrid methods have a mean accuracy of a few meters in real-world scenarios. In this thesis, we try to compare the proposed system with some other hybrid systems, using our own dataset (Wi-Fi radiomap, sensor samples and etc).

The goal of this thesis is to design and implement an indoor/outdoor localization system, which can be used on the currently available smartphones (including Android/iOS/Windows). As mentioned, some of the above techniques require expensive extra infrastructure and hence are not desirable to be used in our system. Our focus is on the techniques which can be implemented and used in a real-world scenario, with minimum extra hardware requirements.

Wi-Fi fingerprinting is probably the most important method for indoor localization. Wi-Fi fingerprinting needs a very dense radiomap to localize the smartphone with an accuracy of a few meters. The information from other sensors of a smartphone can be combined with Wi-Fi fingerprinting to achieve better accuracy of indoor positioning. In this thesis, we represent a 3D indoor/outdoor positioning system which uses Wi-Fi fingerprints, BLE beaconing, GPS, barometer and motion sensors of a smart-
phone, to accurately find the 3D location of the user (2D location and floor number). The system has been implemented on Android smartphones and promising accuracy of indoor positioning was achieved in real-world experiments (around one meter).

1.2 Block diagram of the proposed indoor positioning system

Figure 1.1 shows the block diagram of our proposed indoor localization system. Our system combines five sources of information to localize the user. These sources include:

- **Wi-Fi scanner**: This is the main source of information for Android devices. The Wi-Fi scanner finds the nearby APs and compares the scan results to the fingerprints of the reference points in database using the weighted kNN [12], which is discussed in Section 6.1.

- **Wi-Fi MAC filtering and merging**: This block reduces the redundancy of Wi-Fi fingerprinting by merging the virtual APs into a single physical AP and also by filtering the tethering devices. This is discussed in Section 6.2.

- **Wi-Fi localizer**: Wi-Fi fingerprinting is done in this block and the output of this block is 3D estimation of the location of the user. Wi-Fi fingerprinting may be implemented on the phone, as discussed in Section 6.1, or may be implemented on a localization server which is discussed in detail, in Chapter 7. The block diagram of our localization system, when we use a localization server is depicted in Figure 7.1.
• BLE scanner, BLE beacon filtering and BLE beacon localizer: In some areas of building, we have also installed BLE beacons, which are broadcasting their location. Using the techniques described in Section 5.3, these beacons can be used to localize the user. This method of localization is very important for iOS devices which do not allow the third-party applications to scan Wi-Fi APs in the vicinity of the phone.

• GPS receiver and GPS to pixel converter: Occasionally, we also receive signal from the GPS satellites, inside a building. We review the requirements of using GPS for indoor localization in Chapter 4.

• Barometer sensor and barometer differentiating and filtering: As discussed in Chapter 3, the barometer sensor can facilitate the problem of floor detection. In Section 6.3, we describe the fusion of this signal into the Kalman filter.

• Accelerometer and gyroscope (a.k.a game rotation vector) and reset generator: The game rotation vector shows the relative turns of the user and thus it helps us to reset the Kalman filter when the user makes a turn. Most of mobile platforms, including Android and iOS support this software sensor. The details of how this sensor works is discussed in Subsection 2.1.9 and the details on how we use this sensor for indoor localization is discussed in Section 6.3.

• The tracker module fuses the information from all sensors and obtains 3D position of the user. This module is a Kalman filter indeed. In Section 6.3, we show the procedure for the fusion of all sources of information, to get better accuracy of indoor positioning.

• Floor selector and map to GPS coordinates: The elevation $\hat{z}$ is quantized to select the appropriate floor plan and the 2D location $(\hat{x}, \hat{y})$ is converted to GPS coordinates and is displayed on Google maps.

Each subsystem of the block diagram of Figure 1.1 is discussed in more details in the next chapters.

In order to build a practical and expandable indoor localization system, which can be used globally, we have to investigate the problem of large-scale indoor positioning with thousands of buildings, millions of reference points and millions of Wi-Fi APs. It is obvious that such a large database can only be kept on a localization server, rather than on the smartphone. One way to do this, is to keep the radiomaps of different buildings separately on a server; the smartphone should request the resources (radiomap, floor plan etc.) for a specific building and server provides the phone with these resources.
An alternative method is to run the localization algorithm on the server, without sending the radiomap to the smartphone. In either case, we need to find very efficient algorithms to manage and query such a large localization database. We will be facing the problem of database management to implement such a server. In the literature, the clustering of the radiomap is investigated frequently [45][46][47][48]. The clustering of radiomap is time-consuming and any error in the clustering will lead to huge localization error.

![Figure 1.2: Conventional localization server on the left vs. the proposed scalable localization server on the right.](image)

We already have a localization server that works as shown in the left side of Figure 1.2. In our conventional server, the phone requests the resources (radiomap, floor map, etc.) for a specific building and server provides the phone with these resources. There are a few problems with this localization schema:

- The radiomap of all buildings can be acquired by unauthorized smartphones.
- The load of localization engine would be on the phone, which will draw a lot of battery.
- Smartphone still needs to determine the building in which it is located, to request the appropriate radiomap.
- Billing of the service is not possible on a per request basis.

To eliminate these problems, we propose the localization schema of the right side of Figure 1.2 in which the server does not send the radiomap to the phone but the user sends the Wi-Fi scans to the server and it will receive back its Wi-Fi fingerprinting based location. In this case, all the mentioned problems will be eliminated.
1.3 Contributions

The main contribution of this thesis is the implementation of a robust indoor localization system, which may be used nearly on all commercial phones/tablets and also works on a very large scale where we have many buildings in our database. The goal of our system is to combine several sources of information from different sensors to localize the user and also to gain better accuracy wherever and whenever possible.

To achieve this goal, we have designed and implemented the following subsystems:

1. Sensor fusion using the Kalman filter to gain better accuracy of indoor localization. We will show that there are many sources of information, which can contribute to the indoor localization of the user. Using a Kalman filter, we will explain their fusion to achieve a robust positioning. Combining Wi-Fi fingerprinting with indoor localization using Bluetooth Low Energy beacons, to achieve better accuracy of indoor positioning. Different methods for indoor localization based on BLE are investigated and the best method is chosen and a method for combining Wi-Fi fingerprinting with BLE beaconing, based on Kalman filter, is proposed. It is shown that our method is more accurate than other hybrid Wi-Fi BLE systems [49].

2. Proposed and implemented efficient server-based localization system capable of large scale indoor localization when we have many buildings and floor plans [50].

3. Robust 3D indoor localization is achieved by combining Wi-Fi RSS reading with barometric pressure sensor. The problem of floor detection is investigated and it is shown that methods which merely rely on Wi-Fi fingerprinting (either simple kNN or k-means-clustering) are not sufficient for accurate floor detection. A method which combines the Wi-Fi fingerprinting and the reading from barometer sensor is proposed and it is shown that our proposed method has better accuracy of floor detection than some other systems mentioned in literature.

4. Proposed and implemented a technique to eliminate the need of extra infrastructure for localization of iOS devices. In the proposed technique, Android device will work as BLE beacons to facilitate localization of iOS devices.

5. A new system to transfer real-time sensor data from Android phone or tablet to a PC running Matlab/Simulink software to simulate indoor localization on the PC in real-time and also a technique for fast implementation of signal processing algorithms using Simulink software and Java Native Interface (JNI). Using this technique we compile the Simulink models into C source code and then using JNI we run this C code in a Java/Android application.
In this thesis, we have put the focus on Android devices but the iOS devices (iPhone, iPad, etc.) have also a considerable share in the market. In Chapter 7, we will discuss our solution for indoor localization of such devices.

1.4 Organization of this thesis

The rest of this thesis is organized as follows:

Chapter 2 introduces the sensors embedded in most of the commercial phones/tablets and also our designed and implemented real-time sensor transmission system for Android devices, which will facilitate the simulation and implementation of any signal processing algorithm (including indoor localization algorithms) on these devices. At the end of Chapter 2, the design of a step counter is discussed, using our sensor transmission system. In Chapter 3, the problem of the 3D indoor localization using the barometer sensor is discussed. We will see that Wi-Fi fingerprinting is insufficient for 3D positioning and we shall use other sensors such as barometer to achieve accurate floor detection. Chapter 4 includes the explanation of the relation between indoor and outdoor positioning systems. We will show how these two maybe converted to each other and we will also see that GPS will occasionally help indoor positioning as well. Also, some techniques are proposed for transparent switching between indoor and outdoor areas. In Chapter 5, a new standard for Bluetooth connectivity, called BLE, is introduced and the ways it can help indoor localization are discussed. We will also introduce our novel packet format to embed the position of beacons inside their advertisement packets. In Chapter 6, first we will discuss our approach for MAC filtering and merging to reduce the size of the radiomap and also acquire a better accuracy of localization. Then, we will discuss the pedestrian tracking problem using a Kalman filter and we will show how all sensors may be fused into the Kalman filter. Later, the performance and accuracy results of our implemented indoor localization system are presented in Chapter 6 and finally our solution for large scale server-based indoor positioning and also our cooperative method for localization of iOS devices is discussed in Chapter 7.
Chapter 2

Android sensors transmission system

Nowadays smartphones and tablets come with a wide range of sensors including accelerometer, magnetometer, gyroscope, etc. The goal of this thesis is to combine the information from all of these sensors to provide a better accuracy of indoor/outdoor localization. In order to achieve this goal, at the very first stages of the project, we designed and implemented an Android sensor transmission system, which simplifies experimenting with the sensors of Android phones/tablets. This system simplifies the procedure of designing smartphone applications, which use sensors such as accelerometer, gyroscope, magnetometer, etc. To do so, a small server application is installed on an Android smartphone, which gathers the data from all sensors streams it to a host PC via Bluetooth, Wi-FI or USB cable. On the host PC, sensor data can be analyzed using software products like MATLAB/Simulink, NI Labview or any software with standard TCP/IP stack. We have developed MATLAB m-files and Simulink models required for data acquisition and processing. This system simplifies the data gathering from Android sensors in real-time and it allows the developers to test signal processing algorithms on the sensor data in real-time. The reader should also keep in mind that the information provided in this chapter is not limited to indoor localization application and the Android sensor transmission system can be used to simulate and/or implement any signal processing system on these sensors. At the end of this chapter at Section 2.5, a step counter is simulated and implemented using our proposed system. This step counter is later used for evaluation of [24].

2.1 An introduction to Android sensors

Smartphones come with a wide variety of sensors to have better interaction with the user [1]. In this section, we introduce some of these sensors that are likely to help us to localize the user.
Android sensors can be divided into two general categories:

- Hardware sensors: There exist a physical chip/module on the main board of the smartphone for these sensors. Hardware sensors include: Wi-Fi scanner, Bluetooth scanner, GPS, accelerometer, gyroscope, magnetometer, barometer, proximity, humidity, light, etc.

- Software sensors: These are not actual physical sensors but their samples are generated by processing of the samples of one or more of the hardware sensors. Examples of these sensors are: orientation, rotation matrix, linear accelerometer etc.

We should keep in mind that all of these sensors are not necessarily available on all devices. Presence of any sensor should theoretically increase the indoor positioning accuracy (or at least not decrease) but the absence of any sensor should not stop the system from working. In the following, some information on each of the mentioned sensors is provided. All reported power consumption are measured for a LG Nexus 5 phone.

### 2.1.1 Wi-Fi scanner

The Wi-Fi scanner searches for the nearby APs and returns a list of MAC addresses and the RSS values for those APs. Most of indoor positioning systems (including this system) rely on the fingerprinting of Wi-Fi APs installed in the environment. The sampling interval of this sensor is different among different devices and varies from 0.5 to 3 seconds. An Android device may support the 2.4-GHz and/or 5-GHz frequency bands.

One of the challenges of the indoor positioning systems is that Apple has restricted the access of third-party applications to Wi-Fi scanner. Therefore, when working on iOS devices, our system cannot rely on Wi-Fi fingerprinting like Android devices do. To overcome this problem, we have designed and implemented a cooperative localization system in which Android devices help iOS devices through Bluetooth Low Energy (BLE) beaconing. This is described later in Chapter 7. Also, the iOS devices may use the installed BLE beacons as described in Chapter 5.

### 2.1.2 Bluetooth scanner

Beside the Wi-Fi fingerprinting of the environment, Bluetooth fingerprinting may also help the device to localize itself. The Bluetooth scanner works exactly the same way as the Wi-Fi scanner and returns a list of MAC addresses and the RSS values of the Bluetooth (or BLE) devices found in the environment (described in Chapter 5).
2.1.3 GPS

The GPS is mostly helpful when we are localizing the user outdoors but the GPS signal is sometimes available near large windows or under a glass roof. The GPS sensor returns the estimated position in terms of latitude, longitude and altitude. Further discussion on how GPS may be used in our system can be found in Chapter 4.

2.1.4 Accelerometer

Most of the Android devices have a 3D accelerometer sensor. This sensor has a lot of applications including: step counting, device orientation finding, gesture recognition, etc. Figure 2.1 shows the coordinate system of an Android device. This coordinate system is used for accelerometer, gyroscope, magnetometer and all other sensors. The accelerometer sensor measures the 3D acceleration of the device \((a_x, a_y, a_z)\) along \(x\), \(y\) and \(z\) axes in unit of \(\text{m/s}^2\). This sensor measures the acceleration based on the force exerted to the device, which means when the device is put at rest on a table, the gravity of the earth is measured by the accelerometer \((a_z \approx 9.8\ \text{m/s}^2)\). The current consumption of this sensor is approximately 0.4\(mA\) and its maximum sample rate is 200sps.

![Device coordinates for Android phones/tablets](image)

2.1.5 Gyroscope

The gyroscope sensor measures the angular velocity of the rotation of the device \((\omega_x, \omega_y, \omega_z)\) over \(x\), \(y\) and \(z\) axes in the units of \(\text{rad/s}\). Figure 2.2 explains these rotations.

The current consumption of this sensor is 3.2\(mA\) on a Nexus 5 device and the maximum sample rate is again 200sps.
Figure 2.2: Device orientation in terms of azimuth, roll and pitch. Speed of rotation over each of these axes is measured by gyroscope.

2.1.6 Magnetometer

The magnetometer sensor measures the intensity of the magnetic field \((H_x, H_y, H_z)\) along \(x\), \(y\) and \(z\) axes in the units of \(\mu T\). The magnetic sensor is used to derive the orientation of the device with respect to geomagnetic north of the earth. The current consumption of this sensor is 5\(mA\) on a Nexus 5 device and the maximum sample rate is 50\(sps\). Magnetometer readings are usually noisy and less reliable compared to other sensors. This is due to the magnetic disturbances of the environment (specially indoors). As a result, we expect that the estimation of the azimuth of the device to be less accurate compared to the estimation of its roll and pitch.

2.1.7 Barometer

The barometer sensor measures the air pressure of the environment surrounding the phone. As we show in Chapter 3, this sensor is be very useful to solve the problem of floor detection. The current consumption of this sensor is 0.004\(mA\), which is much lower compared to all other sensors, and the unit of measurement is \(mBars\).

2.1.8 Orientation

This software sensor is derived by the fusion of the samples of the accelerometer, gyroscope and magnetometer and it describes the 3D orientation of the device with respect to the real world coordinates shown in Figure 2.3. This orientation is expressed in terms of the rotations over \(x\), \(y\) and \(z\) axes and called pitch \((\beta)\), roll \((\gamma)\) and azimuth or yaw \((\alpha)\), respectively (Figure 2.2). As mentioned earlier, the magnetometer is used to obtain the azimuth and hence it is less reliable. The unit of the measurement of this sensor is degrees. As an example, if the phone is put on a table and the head is pointing toward the geomagnetic north pole, then the readings for pitch, roll and azimuth will be all be zeros.

Rotation matrix is another way of representing the 3D rotation of the phone. Using the pitch, roll
and azimuth obtained from the orientation sensor, an individual may calculate the rotation matrix as follows:

\[
R_{phone} = \begin{bmatrix}
\cos \alpha \cos \beta & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\
\sin \alpha \cos \beta & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\
- \sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma
\end{bmatrix}
\]  

(2.1)

Figure 2.4: Rotation matrix $R_{phone}$ of the phone translates the device coordinates shown on the left side to the real-world coordinates shown on the right side for Android phones/tablets [1].

In other words, the rotation matrix translates the coordinate system of the device depicted in Figure 2.1, to the coordinate system of real-world, depicted in Figure 2.3. This is shown in Figure 2.4. As an example, if the phone is put on a table and the head is pointing toward the geomagnetic north pole, then $R_{phone} = I_{3 \times 3}$. The orientation sensor is derived by combining the samples of accelerometer, gyroscope and magnetometer and hence its current consumption is the sum of these three, which is equal to $8.6mA$ and the maximum sample rate is $200sps$. 
2.1.9 Game rotation vector

Oftentimes, the absolute azimuth of the device is not required (and cannot be calculated due to magnetic disturbances of the environment). However, by using the accelerometer and gyroscope, the Android device may not know its absolute pitch and roll but only knows its relative azimuth. In this case, the origin of the azimuth is unknown but the relative changes of the azimuth of the device can be estimated by integrating the readings of gyroscope. For example, when the user makes a 90 degrees turn at a corner, this sensor can help us to reset the Kalman filter to avoid the divergence (Section 6.3).

The current consumption of the Game rotation vector is the sum of the consumption of the accelerometer and gyroscope, which is $3.6mA$ and the maximum sample rate is $200sps$.

2.1.10 Linear accelerometer

Linear accelerometer, which is a software sensor, is indeed the gravity compensated accelerometer. In many applications, including the pedometer, we have to remove the gravity from the accelerometer reading, which appears like a bias. However, removing this bias depends on the orientation of the device itself. For example, if the device is put on a table facing upward, then all the gravity is sensed along the $z$ axis. But if the device is put on the desk while standing tall, then all the gravity is sensed along the $y$ axis. Fortunately, Android sensor fusion system takes care of this and removes the gravity according to the sensed orientation of the device. The linear accelerometer uses the same unit as accelerometer and the readings along $x$, $y$ and $z$ are denoted by $l_x$, $l_y$ and $l_z$, respectively. The current consumption of this sensor is the consumption of accelerometer plus gyroscope, which is $3.6mA$. Figure 2.8 shows a sample trace of this sensor when the user walks while keeping the phone in his hand.

2.2 Server application (Android side)

To use the Android sensor transmission system, we have developed a small application, which should be installed on the Android phone/tablet. Figure 2.5 illustrates a screenshot of this application. Each hardware sensor can be turned on or off separately. Also, the IP addresses of all interfaces of the phone (Wi-Fi, Bluetooth, USB, data connection, etc.) are obtained and shown. This application opens three TCP ports (6000 for sensor data, 6001 for Wi-Fi scanner and 6002 for GPS) and waits for connection on these ports. Once a TCP client is connected to each of these ports, the application streams the live sensor data on that port.
2.3 Client application (MATLAB/Simulink side)

On the PC, MATLAB/Simulink is used to gather and process the samples of the sensors. Figure 2.6 shows the Simulink model, which is used to collect the sensor data from the Android phone. All we need to do on the client side is to set the IP address of the phone in the configuration window of the Simulink model. Figure 2.7 shows the configuration window for Android phone block. The server application (Figure 2.5) might show more than one IP address, which means there is more than one way to connect to the Android phone from PC (for example, we might have a Wi-Fi and a Bluetooth connection at the same time). Depending on the IP address you enter in the Simulink model, a different physical medium is chosen to stream the sensor data.

A challenge in the design of this system was the difference in sample rates of the sensors of different Android devices. Hopefully, we could find a common sample rate of 50\text{sps} between all sensors except the barometer. For the barometer sensor, a sample rate of 10\text{sps} is chosen and up-sampling with factor 5 is applied on the phone to match it to its sample rate to all other sensors.
2.4 Implementation of signal processing algorithms using Java Native Interface

At the first stage, we use the Android sensor transmission system to stream the real-time sensor data to a PC running Matlab/Simulink. All of indoor positioning algorithms are implemented in Simulink software and tested in real-time. When we are satisfied with the signal processing algorithm, we use the “Embedded Coder Toolbox” of Matlab [51] to convert the signal processing algorithm into the C code. This is done automatically by the software in less than one minute. The programming language for Android Operating System (OS) is Java but fortunately Java is capable of merging to a C code as a black box. The interface between Java and C code (also named as native code) is called Java Native Interface (JNI) [52]. So the whole localization system is be a black box, which is created by the Matlab software and works as core for the final indoor positioning application on the Android phone/tablet.
2.5 Step counter for Pedestrian Dead Reckoning

Figure 2.9 shows the block diagram of the proposed step counter. In order to count the steps of the user, we monitor the linear acceleration of smartphone, which is the gravity compensated acceleration [1], along the $z$ axis, so the steps is counted correctly, no matter how the smartphone is held in hand. As each step causes two zero crossings on the acceleration along the $z$ axis, we count the zero crossing of the acceleration along the $z$ axis, and divide it by two. The 3D linear acceleration is denoted by $l_k$. The Android smartphone measures the acceleration along its own coordinate system, which is depicted in Figure 2.1. However, the user may not hold his phone such that the device coordinates match the real-world coordinates, shown in Figure 2.3. To obtain the linear acceleration of the smartphone with respect to the real-world coordinates, we have to consider the rotation of the phone compared to the real-world coordinates. The rotation of device is given by the Android OS as a software sensor called rotation matrix, $R_{\text{phone}}$, explained in Figure 2.4. For example, if the phone is left with the head pointing toward geomagnetic north pole, then $R_{\text{phone}} = I_{3 \times 3}$. Rotation matrix is indeed the translation between the device coordinate system and the real-world coordinate system.

The rotated 3D linear acceleration of the device is defined as:

$$u_k = R_{\text{phone}} l_k$$

(2.2)
Then a moving average filter of length of 5 is used to filter the accelerometer data to obtain the signal $f_k$, shown in Figure 2.10, for a sample walk. Next, the rotated linear acceleration along the $z$ axis is selected ($f_{zk}$) and discard the reading along the $x$ and $y$ axes.

Then, the signal is fed to the non-linear cut function $C$, depicted in Figure 2.11, to remove the small variations of acceleration around zero and the signal $c_k$ is achieved. After that, by counting the zero crossing of the acceleration and dividing them by two, we can find the number of steps, taken by the user. This is shown in Figure 2.12.
Figure 2.10: Rotated 3D linear acceleration of the Android device with respect to real-world coordinates of Figure 2.3. As seen, most of the acceleration variations are among the \( z \) axis.
Chapter 2. Android sensors transmission system

Figure 2.11: The cut function $C$, cuts the accelerations below $\frac{2m}{s^2}$ and passes through, the larger accelerations.
Figure 2.12: On the top, the output of cut function, in the middle, the output of zero crossing detector and on the bottom, the output of the step counter is shown.
Chapter 3

Floor detection using barometer

Most of the indoor positioning systems, including Google Indoor Maps [3], ask the user to determine the floor he/she is located at and then localize the user on a 2D plane [8][11][13][32][33]. So why don’t most of these systems detect the floor automatically? Is it possible or not?

It is possible to detect the floor only using Wi-Fi fingerprinting but it might be erroneous, specially in widely open areas like shopping malls. Because when we are in the widely open areas of a building, the fingerprints of the reference points located at different floors may be very similar. This is due to the fact that the height of each floor is the only spacing between these reference points. So the system might not correctly find the floor if we only use Wi-Fi fingerprints. The confusion matrix of floor detection using Wi-Fi fingerprinting is depicted in Table 6.8, Chapter 6. As we see in Chapter 6, the error rate of floor detection without using barometer sensor is approximately 5%, which is not acceptable.

Generally, There are three approaches to detect the floor, in an indoor positioning system:

1. Using Wi-Fi fingerprinting and the k-Nearest-Neighbors method: This method is the most obvious way of floor detection using Wi-Fi fingerprinting. The drawbacks of this method are its complexity and also its low accuracy in widely open areas of the building. The online Wi-Fi scan should be compared to all reference points in the radiomap, to find the 3D location of the user.

2. Using Wi-Fi fingerprinting and the K-means-clustering technique: This technique is discussed in [7] and [44]. Razavi et al. [7] propose using the K-means-clustering technique to reduce the complexity of Wi-Fi fingerprinting using kNN, for floor detection. It is claimed that the performance of their proposed floor detection method is close to Wi-Fi fingerprinting using kNN. It should be noted that the target of that paper is not to increase the accuracy of indoor positioning, but to reduce
Chapter 3. Floor detection using barometer

its complexity. The objective of K-means-clustering method [53], is to cluster the Wi-Fi readings from all reference points into $N_c$ clusters and use these clusters for floor detection. To find the cluster heads (CHs) \( \{ \mu_k \}_{k=1}^{N_c} \), the following metric should be minimized:

\[
\sum_{k=1}^{N_c} \sum_{x_i} \| x_i - \mu_k \|_2^2
\]

(3.1)

where $x_i$ are the Wi-Fi scans. Using an iterative solution described in [7], we can find the CHs.

It should be mentioned that the number of clusters $N_c$ is not necessarily the same as the number of floors. Thus, each floor might contain more than one CH. There is a trade off between the complexity and accuracy of the floor detection by choosing the number of CHs. To investigate the performance of this system, the number of CHs is chosen the same as the number of floors.

Shen et al. [44] also propose combining Wi-Fi fingerprinting with barometric readings. Their focus is mainly on crowd-sourced and unsupervised using the K-means-clustering technique by using the barometer sensor. As a result, the accuracy of floor detection for this technique is less than or equal to the accuracy of the k-means-clustering method discussed in [7] and hence, it is not an interest of this thesis.

3. Barometer-assisted floor detection: It has been shown in [36] that there is enough difference between the air pressures of two consecutive floors of a building so that we can use the barometric pressure as a measure, to detect the floor number. There also has been attempts to use the barometer of the smartphones in [40] but most of these systems have considered the absolute value of the air pressure, which is affected by several environmental factors such as temperature, humidity, etc (usually a reference sensor is used at the first floor to estimate these variations). A customized hardware platform using a Micro-Electro-Mechanical Systems (MEMS) barometer sensor is suggested in [37], to help blind people detect the floor number. A pedestrian dead reckoning system using a special hardware is implemented in [38], which uses barometer sensor in conjunction with the accelerometer sensor, to detect the floor number.

Another system which combines Wi-Fi fingerprinting and barometric readings for floor detection is presented in [2]. This is probably the most relevant work to our floor detection system. Wang et al. [2] propose using two windows on the readings of the barometer sensor (or the equivalent estimated altitude), to monitor rapid changes in altitude, which is due to the change in floor number. The floor detection algorithm is straightforward. As shown in Figure 3.1, a front window of length $t_1$
and a back window of length $t_2$ with a time difference of $\Delta t$, is sliding on the altitude readings. Whenever the difference in the reading of front window $a_1$ and the back window $a_2$ is greater than a defined threshold a change in floor is detected. Otherwise, both windows slide on the data and track the slow variations of the air pressure, which is caused by the environmental factors. The drawback of this method is that it cannot detect a floor change, if the user changes the floor gradually (for example using the staircase instead of the elevator).

![Figure 3.1: Using of back and front window to monitor rapid changes in the altitude, to detect a floor change is explained. This figure is borrowed from [2].](image)

Wi-Fi fingerprinting is used to detect the initial floor. Whenever a change in the floor number is detected, Wi-Fi fingerprinting switches to the reference points of the destination floor. The performance of this system is investigated and compared to our proposed system, in Subsection 6.4.5.

In the rest of this chapter, we discuss our proposed method to utilize the barometer sensor, which is embedded in most of new phones nowadays [1], to solve the problem of accurate floor detection in a 3D indoor positioning system.

Figure 3.2 shows the barometric pressure reading on different floors of Bahen building. Obviously, the higher floors have a lower air pressure and the lower floors have a higher air pressure.
Figure 3.2 suggest that different floors of a typical building have enough air pressure difference (about 0.5 mBar for Bahen building) to be detected by the barometer sensor of a smartphone. The histogram of the noise of the barometer is depicted in Figure 3.3. This figure implies that barometer has a zero mean bell shaped (probably Gaussian) noise.

We also need to investigate the behavior of the readings of barometer sensor when the user is moving upward and downward. We also need to know if the readings are the same for different devices.

3.1 Challenges of using barometer for floor detection

Although using the barometer sensor might seem a very promising technique for floor detection, there are serious concerns, which are discussed next. Some of these challenges are discussed in [40], but we have conducted our own experiments to verify these issues.

3.1.1 Drift over time

We might expect that if we leave the smartphone on a table for one hour, the mean of barometer reading should not change. But the fact is that the barometric reading are changing gradually over time. Figure
Figure 3.3: Histogram of the noise of the barometer sensor.

3.4 shows a recording of the barometer for approximately one hour. This figure shows that changes in the reading of air pressure is significant compared to the difference in air pressure of two consecutive floors. So if we estimate the floor number using the absolute air pressure, we might find the user changing floors even if his/her phone is left on the table for some time. This is due to the fact that the air pressure is not only a function of the height but also a function of humidity, temperature, the weather conditions, etc. So when each of these parameters changes, we should also expect changes in the air pressure. The good news is that these parameters do not change rapidly and might be able to track them.

3.1.2 Building diversity

Figure 3.5 shows the barometer reading for the Galbraith building, University of Toronto. As shown in the figure, in certain buildings, we might experience more noise due to the air conditioning systems. This means if we only want to rely on barometric pressure, we might get huge errors in certain areas.

3.1.3 Device diversity

Another problem was observed when we recorded the barometer readings of two different Android phones (Nexus 4 and Galaxy Note) tied together. Our expectation was the reading should be the same or not
much different. But looking into the readings of the two devices depicted in Figure 3.6, we see a difference of 0.6 mBars, which is more than the difference of the air pressure of two consecutive floors.

We have even tried two identical devices of same brand and model (Nexus 4) and they also had significant differences in pressure readings. So the barometric reading of the each device can be modeled as follows:

$$P(t) = P_r(z(t)) + b_{\text{weather}}(t) + b_{\text{device}}(t)$$

(3.2)

where $P(t)$ is the reading of the barometer sensor, $z(t)$ is the elevation of the smartphone, $P_r(z(t))$ is the reference value of the pressure, which is assumed to be the function of elevation $z(t)$ only, $b_{\text{weather}}(t)$ is bias due to environmental conditions and $b_{\text{device}}(t)$ is the bias due to the differences of each device.

### 3.2 Dynamic behavior of the barometer sensor

So far we have seen that a barometer, which is embedded in a smartphone, might help us to detect the floor on which the user is located. But we have also seen that these readings are gradually drifting over time and they are also different from one device to another. Next, we want to investigate how
Chapter 3. Floor detection using barometer

Figure 3.5: Barometer readings for the Galbraith building on a sunny day. Floor number increases from top to the bottom.

the barometer readings change if the user changes floor using the staircases or the elevators. Figure 3.7 shows the barometer reading when the user walks from the fourth to the fifth floor ion Bahen Centre. As seen, the pressure readings drop by approximately 0.5 mBar. This change in pressure looks like a step function and it is easily detectable. Figure 3.9 and Figure 3.10 show the low pass filtered readings of barometer when the user is changing the floor using an elevator. Figure 3.8 shows the barometer readings when the user walks down from the fourth to the first floor of the Bahen building.

The conclusion is that despite the facts that barometer reading is a function of a lot of parameters such as temperature, humidity, the type of device used, etc, we might consider all of these factor constant for a short period (a few minutes) and only monitor the rapid changes in the pressure as the changes in elevation of the user.

3.3 Wi-Fi combined with the barometer sensor

So far, we have shown that the barometer reading is a good source of information, which facilitates the process of floor detection in a 3D indoor positioning system. However, we have seen that the absolute pressure reading is not reliable and may vary from one device to another and from time to time. But
Chapter 3. Floor detection using barometer

Figure 3.6: Barometer reading of two different Android phones at the same time. The absolute pressure reading is different but dynamic changes are the same.

Fortunately, the relative changes in the air pressure reading reflect the changes in the altitude of the user. In order to get rid of the biases and the long term drift of the barometer readings, we suggest differentiating of the barometer readings in time domain.

The pressure is an exponential function of altitude [54]:

\[ P_r(z) \approx P_0 \exp\left(\frac{-gMz}{R_0T_0}\right) \]  

(3.3)

where \( P_r(z) \) is the reference pressure reading, \( z \) is the altitude from the sea level (for Bahen building, \( z \approx 100m \)), \( P_0 = 101325Pa = 1013.25mBars \) is the sea level standard atmospheric pressure, \( g = 9.80665\frac{m}{s^2} \) is the earth-surface gravitational acceleration, \( M = 0.0289644\frac{kg}{mol} \) is the molar mass of dry air, \( R_0 = 8.31447\frac{J}{mol.K} \) is the universal gas constant and \( T_0 = 288.15 \) is the sea level standard temperature. Differentiating with respect to the altitude we get:

\[ \frac{dP_r}{dz} = -P_0 \exp\left(\frac{-gMz}{R_0T_0}\right) \frac{gM}{R_0T_0} \approx -0.118 \frac{mBar}{m} \]  

(3.4)

We might also calculate this proportional constant by dividing the total difference in the air pressure of the top and the bottom of the building by the height of the building. Then, we should get a close
value to (3.4). The height of the Bahen building is 37m and the height of each floor is approximately 4.625m. So for the Bahen building:

\[
\frac{dP_r}{dz} = \frac{-0.5 \text{ mBar}}{4.625 \text{ m}} \approx -0.108 \frac{\text{mBar}}{m}
\]

or

\[
\frac{dz}{dP_r} \approx -9.25 \frac{m}{\text{mBar}}
\]

As mentioned, the environmental and device factors, which affect the barometer readings may be considered constant over a short time, so by assuming \( \frac{db\text{eater}}{dt} = 0 \) and \( \frac{db\text{device}}{dt} = 0 \) and differentiating (3.2), we have:

\[
\frac{dP}{dt} = \frac{dP_r}{dt}
\]

We linearize this exponential function when the building height is negligible compared to the altitude from the sea level. Now using the chain rule, we can find the velocity of the user \( \dot{z}(t) \) along the \( z \) axis:
Chapter 3. Floor detection using barometer

Figure 3.8: The barometric reading when the user takes stairs from fourth to the first floor.

\[ \dot{z}(t) = \frac{dz}{dt} = \frac{dz}{dP} \frac{dP}{dt} \approx -9.25 \frac{dP}{dt} \]  \hspace{1cm} (3.7)

So if we differentiate the barometer readings, we can simply calculate the velocity along the \( z \) axis. Figures 3.11 and 3.12 show the low pass filtered and differentiated readings of barometer when the user takes the elevator from the first to the fourth floor and vice versa. The estimated vertical speed of the user when he/she takes the elevator from the first to the fourth floor and vice versa are shown in Figures 3.13 and 3.14, respectively. In Section 6.3, we discuss how this estimated vertical speed can be fused with other sources of information such as Wi-Fi or BLE localization to increase the accuracy of floor detection. The accuracy of floor detection using our proposed method is described in Subsection 6.4.5.
Figure 3.9: Changes in the pressure readings when the user takes the elevator from the first to the fourth floor.

Figure 3.10: Changes in the pressure readings when the user takes the elevator from the fourth to the first floor.
Figure 3.11: Differentiated barometric readings as the user takes stairs from fourth floor to the fifth floor.

Figure 3.12: Differentiated barometric readings as the user takes stairs from fourth floor to the first floor.
Figure 3.13: The vertical speed of the elevator calculated from readings of the pressure sensor in Figure 3.9.

Figure 3.14: The vertical speed of the elevator calculated from readings of the pressure sensor in Figure 3.10.
Chapter 4

GPS-compatible indoor localization

The outdoor coordinates (GPS) are usually described in terms of latitude, longitude and altitude. These coordinates represent a unique point on or above the earth. However, the story is different for indoor maps. Most of the indoor localization systems lack the standard GPS coordinates and use their local reference positions (e.g. (x,y) pixel number in a JPEG image file). In order to build a transparent indoor/outdoor localization system, we have to standardize the coordinates between the floor plan and global GPS coordinates. To achieve this goal, we might either convert the GPS coordinates of outdoor maps to the pixel numbers of the indoor map or we might convert the pixels of the floor plan of the indoor map, into the GPS coordinates of the real world. If we use the first technique, to convert the GPS coordinates to pixel numbers of the indoor map, each building (or floor plan) would have its own local reference. Hence, we choose the second technique and convert the position of each pixel of the floor plan to the corresponding latitude and longitude. This conversion is explained in Section 4.1.

There is still one difference between the indoor coordinates and outdoor coordinates. In GPS systems, the displacement along the z-axis is described in terms of the distance from sea level in units of meters, feet, etc, which is called the altitude. This representation is not necessarily the best choice for the indoor maps. The most user-friendly way of representation of the elevation for indoor positioning systems is the floor number. So instead of the altitude, we use the floor numbers to describe the user’s location. Obviously the floor plan and global GPS coordinates are easily convertible knowing the altitude of the first floor of the building, the number of the floors and the height of a building.

The reader should note that this chapter is not claiming any innovation, but it is just a practical requirement of the implementation of an standard positioning system.
4.1 Matching of the indoor maps to the outdoor maps

In this section, we will describe how we can convert a pixel of the indoor map to its corresponding latitude and longitude on the earth. Assuming that the dimensions of the building are negligible compared to the radius of the earth \((W, L << 6371 \text{km})\), there will be a linear relation between the geo-coordinate and the pixel number of each point described as below:

\[
g = A \cdot p + b \quad (4.1)
\]

or in the expanded form:

\[
\begin{bmatrix}
g_{\text{lat}} \\
g_{\text{long}}
\end{bmatrix}
= 
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
p_x \\
p_y
\end{bmatrix}
+ 
\begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\]

where \(g\) is the geo-coordinate of each point, \(p\) is the pixel number of that point on the JPEG image file, \(A\) and \(b\) are conversion parameters, which should be estimated for each building. In (4.1), we have six unknowns to be estimated and hence, we need at least six equations to uniquely find these unknowns.

If we know the geo-coordinates of three pixels of the indoor map \(p_1, p_2, p_3\) and denote them as \(g_1, g_2, g_3\) respectively:

\[
p_1 = \begin{bmatrix} p_{x1} \\ p_{y1} \end{bmatrix}, \quad p_2 = \begin{bmatrix} p_{x2} \\ p_{y2} \end{bmatrix}, \quad p_3 = \begin{bmatrix} p_{x3} \\ p_{y3} \end{bmatrix}
\]

\[
g_1 = \begin{bmatrix} g_{\text{lat1}} \\ g_{\text{long1}} \end{bmatrix}, \quad g_2 = \begin{bmatrix} g_{\text{lat2}} \\ g_{\text{long2}} \end{bmatrix}, \quad g_3 = \begin{bmatrix} g_{\text{lat3}} \\ g_{\text{long3}} \end{bmatrix}
\]

The unknown parameters \(a_{11}, a_{12}, a_{21}, a_{22}\) can be estimated by solving two sets of linear equations (4.3) and \(b\) should be calculated using (4.4):

\[
\begin{bmatrix}
p_{x1} - p_{x2} & p_{y1} - p_{y2} \\
p_{x1} - p_{x3} & p_{y1} - p_{y3}
\end{bmatrix}
\begin{bmatrix}
a_{11} \\
a_{12}
\end{bmatrix}
= 
\begin{bmatrix}
g_{\text{lat1}} - g_{\text{lat2}} \\
g_{\text{lat1}} - g_{\text{lat3}}
\end{bmatrix}
\]

\[
\begin{bmatrix}
p_{x1} - p_{x2} & p_{y1} - p_{y2} \\
p_{x1} - p_{x3} & p_{y1} - p_{y3}
\end{bmatrix}
\begin{bmatrix}
a_{21} \\
a_{22}
\end{bmatrix}
= 
\begin{bmatrix}
g_{\text{long1}} - g_{\text{long2}} \\
g_{\text{long1}} - g_{\text{long3}}
\end{bmatrix}
\]

\[
b = g_1 - A \cdot p_1
\]
It is straightforward to match the three corners of the building from the image file to the outdoor maps using commercial software products such as Google Earth or Google Maps. Once we find the matrix $A$ and the vector $b$, every pixel of the floor plan can be mapped to a geo-coordinate using (4.1). Another useful parameter which will be used later, is the rotation of the building (the JPEG image of the floor plan) respective to geographical north (This is also explained in Figure 6.6). This is denoted by $\theta$ and can be calculated from $A$:

$$\theta = \tan^{-1}\left(\frac{a_{11}}{a_{12}}\right)$$  \hspace{1cm} (4.5)$$

if $a_{21} < 0$ then $\theta \leftarrow \theta + 180$

Then, the 3D rotation of the building compared to the geomagnetic north is expressed as:

$$R_{\text{building}} = \begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 \\
-\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}$$ \hspace{1cm} (4.6)$$

As we see in Section 6.3, the matrix $R_{\text{building}}$ is used to convert the 3D direction of the device, to 3D real world direction. For the Bahen building we have found the following values:

$$A = \begin{bmatrix}
0.19940 \times 10^{-6} & -0.61772 \times 10^{-6} \\
0.84940 \times 10^{-6} & 0.30126 \times 10^{-6}
\end{bmatrix}$$

$$b = \begin{bmatrix}
43.66003 \\
-79.39812
\end{bmatrix}$$

$$\theta \approx -17^\circ$$

4.2 Floor to floor matching

Often times, the maps provided for each floor of the building have different scales, or at least these maps are not aligned. By saying “not aligned”, we mean that the same pixel on each of these two maps, does not necessarily correspond to the same latitude and longitude. A simple way to match all floor plans to the GPS coordinates is to first match all floor maps to one of them and then match that floor to the GPS real world coordinates using the technique described in Section 4.1. One may replace the geo-coordinates $g$ with the pixel numbers of the image for another floor plan and find the transition
equation from one floor plan to another. An example of such floor plans before matching is shown in Figure 4.1.

Figure 4.1: Raw maps of the fourth and the fifth floor of the Bahen building. As seen, the two maps have different scales and may not match on each other.

After translating the floor map of the fifth floor to the fourth floor, the two images are on the same scale and rotation as shown in Figure 4.2. It seems that software products such as Google Map are utilizing the same technique to match indoor and outdoor maps [3]. Figure 4.3 shows the indoor map from the Google Map software for the Bahen building. We have uploaded three raw floor plan images on their website and the resulting indoor map is as shown.

4.3 Using the GPS to assist indoor localization

Generally the GPS signal is obstructed by the walls and the roof of the building and we get no strong GPS signal indoors, which is the reason for using the sophisticated techniques for indoor localization. However, sometimes when we have large windows or a glass roof, we get a rough location estimation by using the GPS. Also, when the user walks from outdoor area into the building, we can get a very good estimate of the initial location of the user inside the building. An example of the situations where GPS signal is available indoors is shown in Figure 4.4.

Research has also shown the use of GSM acquired locations for indoor positioning [55]. As we see in Section 6.3, this coarsely estimated location can also be injected into the Kalman filter to get
overall better accuracy of indoor localization and also transparent switching between indoor and outdoor positioning. In order to do that, we invert (4.1) to get:

\[ p = A^{-1}(g - b) \]  

so, knowing the building parameters \( A \) and \( b \), we can map the GPS readings to the pixels of the floor plan.

### 4.4 Indoor map representation as a graph

Maps are not only useful to represent the user’s location but they also ease the problem of tracking and navigation. The problem is that the floor plan image contains too much information about the walls, the rooms, the elevators, etc. In this system, we reduce the floor plan image to a graph consisting of some vertices and some edges. We assume that the user only moves, and should be localized, on this graph (on the vertices or on the edges). Any estimated location data will also be projected onto this graph. An example of the graph representation of a floor plan is depicted in Figure 4.5.
Figure 4.3: The indoor map for the Bahen building by the Google Maps software [3]. Currently it only supports three floors of the Bahen building.

Figure 4.4: On the left, you see a spot on the fourth floor of the Bahen building on which GPS signal is available. The true location is shown by a circle and the estimated locations using GPS is marked by stars. On the right, we have the main entrance of the building. As seen, the estimated locations are very close to the true location of the user in this case.
Figure 4.5: Graph representation of the fourth floor of the Bahen building.
Bluetooth Low Energy (BLE) is a new wireless connectivity standard designed and marketed by Bluetooth Special Interest Group [56]. Its main purpose is to serve wireless connectivity to very low power, low data rate, battery powered devices such as health monitoring, fitness, wireless sensor, beaconing, etc. Bluetooth Low Energy is not a newer version of the classic Bluetooth technology but it is a complement to the classic Bluetooth for low power applications. Fortunately, BLE is supported in most of the operating systems including Android, iOS, Windows, Linux, etc. BLE peripherals are very low power such that they can run even for a few years on a single cell coin battery [56]. But the transmitting range of a BLE device is limited to a few tens of meter. BLE shares the same ISM band with Wi-Fi and classic Bluetooth (2.4-GHz). The main idea behind BLE is that the peripheral device is in sleep mode in general and wakes up every one or two seconds and broadcasts an advertisement packet and goes back to sleep again. We want to use BLE as a low power battery based beaconing system to facilitate the problem of indoor localization. BLE beacons are very cheap (just a few dollars each). We can install hundreds of these beacons inside a building and they will not need any battery change for years. As mentioned earlier, iOS devices do not allow the scanning for Wi-Fi APs but they allow a third-party software to receive BLE advertisement packets from the beacons installed in the environment. So the most important motivation behind the using of BLE in localization is to support the indoor localization for iOS devices. However, these beacons can also increase the accuracy of indoor localization for Android devices.

Currently there are many BLE capable devices on the market. It should be mentioned that we want
to use these BLE devices as beacons only and we do not intend to connect to them. These beacons send advertisement packets on a specific rate. We have chosen the TodHQ BLE beacon for our experiments [4]. This beacon is shown in Figure 5.1. The beacon consists of a BLE112 module manufactured by Bluegiga [5]. This beacon is powered by a small 3V coin battery and the battery can last about year (depending on the beacon rate of the device and its transmitting power). In this thesis, we do not get into the details of BLE technology and we look at BLE as cheap battery powered beacons. We can install a lot of these beacons inside a building at known locations. These beacons help the smartphones to localize themselves. We have two approaches to utilizing these beacons for indoor positioning:

- Installing these beacons at known locations and measuring the distance of the user to each of these beacons and using the techniques described in Subsection 5.3.1, to localize the users.

- Installing these beacons at random locations in the building and doing the fingerprinting for these beacons, exactly the same way as we do with Wi-Fi APs [19][20][21][22]. This is discussed in Subsection 5.3.1.

The second approach might seem simpler, as we do not need to know the position of each individual beacon, but in fact, building of the radiomap consumes a lot of time and human resources. Another problem with the second approach is that considering the fact that BLE beacons are short ranged compared to Wi-Fi APs, so at each reference point, only a few of these beacons will be visible. The main reason that causes the Wi-Fi fingerprinting approaches to work much better than the geometrical approaches, such as Trilateration [6], is that Wi-Fi APs are transmitters with higher range.

As a result, we have chosen the first approach to utilize these beacons. Section 5.1 discusses the details of implementation of this approach. The performance of the Bluetooth fingerprinting methods is investigated in Subsection 6.4.1.

![Figure 5.1: TodHQ beacon on the left [4] and its heart, BLE112 module on the right [5].](image-url)
5.1 iBeacon vs proposed advertisement packet format

There are several defined standards for BLE advertisement packets. One of the most famous standards is created by Apple, which is called *iBeacon*. The packet format of iBeacon is depicted in Figure 5.2.

Inside an iBeacon advertisement packet, we have two numbers called major and minor IDs. These two IDs should be set uniquely for each beacon in the building. We also need to build and maintain a table, which shows the correspondence of each ID to geo-coordinates of the installed point. On the contrary, we have designed our proposed packet format to embed the geo-coordinates of the beacon into the advertisement packet. The proposed BLE advertisement packet is described in Figure 5.3. The description of each of the fields is as follows:

- **Identification Prefix**: Two bytes of data set to "00". This helps the receiver to distinguish between iBeacon and proposed format.
- **Coordinates**: We embed three IEEE 32-bits floating-point numbers in this field. These coordinates are set to latitude, longitude and the floor number of the installation point of the beacon.
- **TX power at 1 meter**: The transmitting power of the beacon at a distance of 1 meter.
- **Unique Identifier**: This optional field allows us to assign a unique identification number to this beacon.
- **Beacon name**: This field is also optional and can be useful in demos.

We have also created a simple wizard for programming of iBeacon or proposed packet format onto the beacons using the Matlab software. This tool allows us program tens of proposed beacons each hour. This tool is not described here in details.

It should be mentioned that changing the BLE advertisement packet format has absolutely no effect on indoor localization error. The reason for using the proposed BLE format is merely to embed the beacon information inside the packet, so we do not need to store the location of the beacons separately.

5.2 Bluetooth Low Energy experiment setup

Figure 5.4 shows the setup of indoor positioning using BLE beacons in our lab (Bahen building, room BA-4148). As seen in this figure, we have installed four TodHQ beacons on the four corners of the lab. We have also installed four beacons outside the lab. The advertisement packets of these beacons can be received by the smartphone almost everywhere inside the room. Each of these beacons is transmitting
its location using our proposed packet format discussed in Section 5.1. So when an Android/iOS phone
receives a BLE advertisement packet, it knows the location of the transmitter as it is embedded in the
packet. Let us define the notations of Figure 5.4:

- \((x_i, y_i)\) is the physical location of the transmitting beacon \(i\).
- \(TX_i\) is the transmitting power of beacon \(i\) at a distance of one meter.
- \(r_i\) is the physical distance between beacon \(i\) and smartphone, which should be estimated.
- \(RSS_i\) is the received signal strength from beacon \(i\) at the receiver (smartphone).

When we were inside the lab, the smartphone was receiving the signal from seven of the installed
beacons but in general any number of beacons might be heard at the receiver. So the receiver should
be able to estimate its location based on any number of beacons. Now we can move forward to the
localization algorithms.

5.3 Bluetooth Low Energy localization techniques

5.3.1 Geometrical methods for BLE-based indoor positioning

Unfortunately due to hardware restrictions, we cannot have any timing synchronization between the
beacons and/or smartphone so the only method that can be used to estimate the distance \(r_i\) is by using
Figure 5.3: The proposed BLE packet format. Coordinates and unique ID vary from one device to another.

The received signal strength and the propagation equation. The propagation equation for free space \[57\] is explained in (5.1):

\[
L_P(dB) = 92.45 + 20\log(F) + 20\log(d) \tag{5.1}
\]

where \(L_P\) is the path loss in \(dB\), \(F\) is the frequency in GHz, which is 2.4-GHz and \(d\) is the distance. We have measured the approximate transmitting power of our BLE beacons at a distance of one meter (-61 \(dBm\)). Instead of placing the transmitting power of beacon \(P_i\) in the Friis transmission equation, we have measured the transmitting power of the beacon \(TX_i\), at the distance of one meter \(d = 1\) from the beacon. So the distance \(r_i\) can be calculated by using the difference between the received signal strength and the transmitting power of the beacon at the distance of one meter:

\[
TX_i = P_i - L_P|_{d=1} \tag{5.2}
\]

\[
= P_i - 92.45 - 20\log(F) + 20\log(1)
\]

\[
= P_i - 92.45 - 20\log(F)
\]

and

\[
RSS_i = P_i - L_P|_{d=r_i} \tag{5.3}
\]
\[ P_i = 92.45 - 20 \log(F) - 20 \log(r_i) \]

\[ = TX_i - 20 \log(r_i) \]

so the distance \( r_i \) is:

\[ r_i = 10^{\frac{TX_i - RSS_i}{20}} \]  \hspace{1cm} (5.4)

where \( TX_i = -61\,dBm \) in our experiments and the denominator in the exponent is called the attenuation factor \( A \), thus \( A = 20dB \).

As mentioned before, this model is best for free space. For indoor areas, attenuation factors of 30\( dB \) to 50\( dB \) are recommended in literature. A technique to estimate this attenuation factor has been proposed in [25]. The attenuation factor is estimated by averaging the attenuation factor for different locations, while knowing the distance of the receiver to the beacon:

\[ \hat{A} = E\left(\frac{TX_i - RSS_i}{\log(r_i)}\right) = \frac{1}{NK} \sum_{i=1}^{K} \sum_{j=1}^{N} \frac{TX_i - RSS_{ij}}{\log(r_{ij})} \]  \hspace{1cm} (5.5)

where \( N \) is the number of test points and \( r_{ij} \) is the known distance of the test point \( j \) to the beacon \( i \), and \( RSS_{ij} \) is the RSS of beacon \( i \) at the test point \( j \).

The estimated value of attenuation factor for our environment was \( \hat{A} \approx 30dB \).

Once the distance of each beacon from the smartphone is estimated, we can estimate the position of the smartphone knowing the locations of all beacons. We have tried three different techniques:

- The closest beacon [17]
- Weighted average of beacons [24]
- Trilateration [18]

Assume that \( K \) of the installed beacons are heard at the smartphone, then each of these techniques is as follows.

**The closest beacon**

If we have a dense installation of BLE beacons inside a building, it is usually enough to find the nearest beacon to the smartphone and report its location as the location of the smartphone. The closest beacon method is described as follows:
\[(\hat{x}, \hat{y}) = (x_n, y_n) \mid n = \arg \max_i RSS_i \equiv \arg \min_i r_i \quad (5.6)\]

Basically, we assume that the highest power (RSS) is received from the nearest beacon. This might not always be true due to the factors described in Section 5.4. The closest beacon technique is very simple in terms of implementation.

**Weighted average of beacons**

If we receive advertisement packets from more than one beacon, it is reasonable to combine their locations to get better positioning accuracy. In this averaging, we give higher weights to the closer beacons and the lower weights to those that are far. The estimated location of the smartphone \((\hat{x}, \hat{y})\) is:

\[
(\hat{x}, \hat{y}) = \sum_{i=1}^{K} \left(\frac{x_i, y_i}{\sum_j \frac{1}{r_j}}\right) \quad (5.7)
\]

Implementation of this technique is again simple. If we only receive advertisement packet from one beacon only, then choosing the closest beacon is equivalent to the averaging of beacons.

**Trilateration**

Another geometric technique to localize the user is the trilateration technique [6]. The trilateration for the case of three BLE beacons is described in Figure 5.5. If we can measure the distance of the smartphone to at least three points with known coordinates, then we can find the position of the smartphone using the trilateration technique.

We define matrix \(A\) and vector \(b\) as follows:

\[
A = 2 \begin{pmatrix}
 x_2 - x_1 & y_2 - y_1 \\
 x_3 - x_1 & y_3 - y_1 \\
 \vdots & \vdots \\
 x_K - x_1 & y_K - y_1
\end{pmatrix} \quad (5.8)
\]

\[
b = \begin{pmatrix}
 r_1^2 - r_2^2 + x_2^2 + y_2^2 - x_1^2 - y_1^2 \\
 r_1^2 - r_3^2 + x_3^2 + y_3^2 - x_1^2 - y_1^2 \\
 \vdots \\
 r_1^2 - r_K^2 + x_K^2 + y_K^2 - x_1^2 - y_1^2
\end{pmatrix} \quad (5.9)
\]
Then, using the least-squares method we can find the location of the smartphone:

\[ Ax = b \xrightarrow{LS} \hat{x} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = (A^T A)^{-1} A^T b \]  \hspace{1cm} (5.10)

We need at least three beacons to be able to use trilateration on a 2D plane and at least four beacons to do it in a 3D space. However, in some areas the smartphone might hear only one or two beacons. In this case we are not be able to use the trilateration technique. Trilateration is more complicated in terms of implementation as it requires the calculation of an inverse of a matrix.

### 5.3.2 Bluetooth fingerprinting for indoor localization

On the contrary to geometrical methods for BLE-based indoor localization, we might use BLE-fingerprinting which is discussed in many papers [19][20][21][22]. In BLE fingerprinting, first a radiomap of RSS of the beacons at different locations is formed. In the localization phase, the online RSS is compared to the RSS of the reference points, using various machine learning algorithms and the location of the receiver is estimated. Since BLE fingerprinting is very similar to Wi-Fi fingerprinting, which is discussed in Section 6.1, we skip the mathematics here. Figure 5.6 shows the setup of the experiment of BLE fingerprinting.

The benefit of using BLE fingerprinting is that there is no need to use signal propagation model and hence the ranging error does not affect its performance. However, as we see in Subsection 6.4.1, the geometrical methods are more accurate in ideal cases.

### 5.4 Conclusion of indoor localization using BLE

So far, we have introduced four methods for BLE-based indoor localization. If we only receive signal from one BLE beacon then we have to use the closest beacon technique. If we receive the signal from two or more beacons, then we can use either the closest beacon or the weighted average technique. If we hear from three or more beacons, then we can use the closest beacon, the weighted average or the trilateration technique. The benefit of trilateration technique is that it can theoretically localize the user anywhere on a 2D plane but the weighted average technique will always localize the smartphone inside a polygon, which connects all of beacons (in this case, inside the room BA-4148). Another problem with trilateration is that it is more sensitive to the errors of range measurement than weighted averaging and the closest beacon techniques.

We have developed two applications on Android, which are shown in Figure 5.7. The first software is a BLE scanner and analyzer. This software scans for BLE advertisement packets and extracts the
location of the beacon, transmitting power, etc. This software analyzes both iBeacon and our proposed packet formats. The second software is the localization engine, which uses the techniques described in Section 5.3. As shown in Figure 5.7 the error of localization using only BLE beacons, is around 2 meters inside our lab (BA-4148). Later in Section 6.3, we explain how the results of BLE localization can be fused with Wi-Fi fingerprinting and other sensors to get a better accuracy.

Although the localization using BLE might seem very promising, there are also some cons to this method of indoor positioning. First of all, it needs an extra infrastructure despite Wi-Fi based indoor positioning systems. Secondly, the estimation of ranges $r_i$ has some error due to the following reasons:

- Signal fading and shadowing: The RSS of each beacon is a random process at receiver. The value of RSS changes randomly due to several factors including channel fading. Certain object like concrete columns may shadow the BLE signal from these beacons.

- Non-isotropic pattern of BLE beacons: The antennas of the BLE beacons are not ideally isotropic. This means that the antenna has different gains in different directions. The antenna pattern of BLE112 module is shown in Figure 5.8.

- Effect of plane earth and the environment: The earth has a great impact on the RSS of the signal at the receiver. This effect is described in Figure 5.9.

- Effect of the geometry of the building: Objects like walls, columns, etc, affect the level of the received signal and hence cause errors in the estimation of the distance.

In conclusion, the indoor localization using BLE technology is a promising method in the sense that it can be used for almost all smartphones including Android, iOS, etc. The error can be also decreased by increasing the number of beacons and limiting the range of each beacon. If we have sufficiently large number of beacons and they have short ranges, then the localization error is reduced to range of each of these beacons. Other researchers have also shown promising results for positioning using BLE over Wi-Fi [26]. The main disadvantage of using this technology is its need for installing and maintaining an infrastructure that can be expensive and time-consuming. In Section 7.3, we describe how we can utilize this technology without the need for a new infrastructure.
Figure 5.4: Part of the experiment setup on the fourth floor of Bahen building, University of Toronto. The solid circles show the position of Wi-Fi reference points and the $\times$ marks show the location of BLE beacons and the solid triangles, which are labeled from A to F show the test points to measure the indoor localization error. The size of room BA-4148 is approximately $11m \times 6m$. 
Figure 5.5: Positioning using the trilateration technique. The intersection of the three circles will be the estimated location of the user [6].
Figure 5.6: Part of the experiment setup on the fourth floor of Bahen building, University of Toronto. The solid circles show the position of BLE fingerprint reference points and the × marks show the location of BLE beacons and the solid triangles, which are labeled from A to E show the test points to measure the indoor localization error of BLE fingerprinting. The size of room BA-4148 is approximately $11m \times 6m$. 
Figure 5.7: On the left the software for analyzing the proposed advertisement packets is shown and on the right the result for BLE localization inside our lab is displayed. The * is the true location and the circle is estimated location of the smartphone.
Figure 5.8: The 3D radiation pattern of BLE112 module from top, front and side views [5].

Figure 5.9: Propagation loss of free space channel vs. plane earth channel for BLE112 module [5].
Chapter 6

Wi-Fi fingerprinting and sensor fusion using Kalman filter

In this chapter, first in Section 6.1, we see how the indoor localization using Wi-Fi (or BLE) fingerprinting is performed. Then in Section 6.2, we show that by filtering of the redundant Wi-Fi APs from the radiomap, we can significantly reduce the complexity of Wi-Fi fingerprinting. In Section 6.3, we discuss how the different sources of information, which were discussed in the previous chapters, can be combined to acquire better accuracy of indoor positioning. These sources of information include the sensor data, the Wi-Fi scans, the BLE scans, the GPS data and the map information. The next section discusses the pre-processing of Wi-Fi scans, which is essential to do Wi-Fi fingerprinting. At the end, in Section 6.4 we present the comparisons for BLE-based indoor localization, different floor detection methods and also the comparisons for the whole system with respect to other hybrid systems.

6.1 The weighted kNN algorithm

In this section, we describe how the Wi-Fi fingerprinting localization based on weighted kNN works and find its computational complexity and its memory requirements. The Wi-Fi radiomap is expressed as matrix $\Psi$:

\[
\Psi = \begin{bmatrix}
\psi_{11} & \cdots & \psi_{1n_r} \\
\vdots & \ddots & \vdots \\
\psi_{n_a1} & \cdots & \psi_{n_an_r}
\end{bmatrix}
\]  

(6.1)
where $\psi_{ij}$ is the RSS reading of AP number $i$ at RP number $j$, $n_a$ is the total number of APs in the building and $n_r$ is the total number of RPs in the database. The radiomap is created in the offline phase by gathering the Wi-Fi scans at the specific reference points in a building. Each column of the radiomap represents the Wi-Fi APs nearby at a specific reference point. To fill out the radiomap $\Psi$, if the AP number $p$ is not visible at RP number $q$, then we set $\psi_{pq} = -110\text{dBm}$.

In the online phase, the smartphone scans for APs nearby and gives the online vector $s$:

$$
\begin{bmatrix}
s_1 \\
s_2 \\
\vdots \\
s_{n_o}
\end{bmatrix}
$$

where $s_k$ is the RSS of AP number $k$, and $n_o$ is the total number of APs in the online Wi-Fi scan. To find the location of the user, the online reading vector $s$ should be compared to all columns of the radiomap $\Psi$. In general, the number of APs in the online scan is different from the number of APs in the radiomap, thus $n_a \neq n_o$. First, we need to match these numbers to be able to calculate the distance between the online readings and the columns of the radiomap. To do so, we need to do the following:

- If an AP is visible in the Wi-Fi scan, but is not present in the radiomap, it should be discarded from the scan, as we do not have any prior information for that AP.
- If an AP is present in the radiomap, but is not visible in the online scan, we put a fill value of $-110\text{dBm}$ in the online scan to fill its place.

After this, the resized online reading $s'$ have the same dimension as the columns of the radiomap $\Psi$. The distance of $s'$ to the columns of the radiomap $\Psi$ in the RSS space is calculated using L1 or L2 norm. If we use the L2 norm, we have:

$$
d_m = \| s' - \Psi_{:,m} \|_2^2 = \sum_{l=1}^{n_a} (s_l - \psi_{lm})^2
$$

where $\Psi_{:,m}$ is the $m$th column of matrix $\Psi$ and $d_m$ is the distance of the resized online reading $s'$ to the RP number $m$ in the RSS space. We define the ascending sorted vector of distances $d$ as:

$$
d = [d_1 d_2 \ldots d_{n_a}]^T
$$
Then, the location can be estimated by calculating the weighted average of the locations of the RPs corresponding to the \( k \) first elements of \( \mathbf{d} \), using the following equation:

\[
\mathbf{z}_{\text{wifi}} = \begin{bmatrix}
\hat{x}_{\text{wifi}} \\
\hat{y}_{\text{wifi}} \\
\hat{z}_{\text{wifi}}
\end{bmatrix} = \frac{1}{\sum_{p=1}^{k} \frac{1}{d_p}} \sum_{p=1}^{k} \left( x_p, y_p, z_p \right) \times \frac{1}{d_p}
\tag{6.5}
\]

where \((x_p, y_p, z_p)\) is the location of the reference point \( p \) and \( d_p \) is the distance of online Wi-Fi scan to the fingerprint of reference point \( p \) in RSS space. We choose the variance of localization of Wi-Fi fingerprinting to be proportional to the distance to the closest RP in the RSS domain.

\[
\sigma_{\text{wifi}}^2 \propto \arg \min_m d_m, \forall m
\tag{6.6}
\]

The selection of the value of proportional constant, which we call it \textit{Wi-Fi accuracy constant} \((c_{\text{wifi}})\), depends on the number of APs in the radiomap and hence depends on the average distance in RSS space. The variance \( \sigma_{\text{wifi}}^2 \) will later be used as the observation error of Kalman filter in Section 6.3.

To find the complexity of the weighted kNN algorithm, consider (6.3). We need to calculate \( n_r \) distances where each distance calculation needs \( 2n_a \) multiplications for squaring, \( n_a - 1 \) additions to compute the summation and \( n_a \) subtractions. To find the location of (6.5), we do not need to sort all the distances but we only need to find the \( k \) smallest distances that needs \( k \times n_r \) operations. So the total number of operations considering the summation of (6.5) is:

\[
n_r(2n_a + n_a - 1 + n_a) + k \times n_r + 2k + k - 1 = 4n_r n_a + (k - 1)n_r + 3k - 1
\]

As \( k \) is usually a small constant (in our system \( k = 2 \)), the total number of required calculations is \( O(n_r n_a) \). The memory required to store the radiomap \( \Psi \) is obviously proportional to the number of elements of \( \Psi \), which is again \( O(n_r n_a) \). In general, we can assume that the number of access points in the environment \( n_a \) is proportional to the number of reference points \( n_r \) (to have a fair Wi-Fi coverage and reasonable indoor positioning error), which is denoted by \( n \). So the complexity and the memory requirement of the weighted kNN algorithm is \( O(n^2) \). Hence, it is impossible to use the conventional kNN method when we have millions of APs and millions of RPs in our database. The problem of large scale indoor positioning is addressed in Chapter 7 and our solution to reduce the complexity of the
weighted kNN to $O(\log(n))$ is discussed.

Next, we will show how the size of the radiomap may be reduced by proper filtering and merging of the Wi-Fi APs in the database.

### 6.2 MAC Filtering of Wi-Fi APs

Looking into the Wi-Fi scans in the Bahen building, we observed that many of the discovered APs look very similar in terms of MAC addresses and the RSS values but they have different SSIDs. These are often called virtual APs, which means all of these virtual APs are broadcast from the same physical AP. As the physical layer for all of them are the same, theoretically they should have the same RSS. Table 6.1 shows an example.

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>SSID</th>
<th>RSS(dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:0c:e6:48:0c:00</td>
<td>eduroam</td>
<td>-50</td>
</tr>
<tr>
<td>00:0c:e6:48:0c:01</td>
<td>UofT</td>
<td>-49</td>
</tr>
<tr>
<td>00:0c:e6:48:0c:02</td>
<td>compsci</td>
<td>-50</td>
</tr>
</tbody>
</table>

Table 6.1: Three virtual APs are broadcast from the same physical AP.

These virtual APs may be combined to one physical AP in order to reduce the redundancy of our database and also to increase the accuracy of indoor localization by averaging the RSS of different virtual APs. Another issue in the Wi-Fi scanning is the presence of the tethering devices such as smartphones, laptops, etc. These devices should be opted out of the database because of their mobility. If these devices are included in the radiomap, they eventually increase the indoor positioning error. Here is a three step procedure for MAC filtering and merging:

1. Mask the first 24-bits of the MAC address to find the vendor (in this example 00:0c:e6). If the vendor (in this example Meru Networks) is a provider for tethering devices, then discard this AP.
2. Mask the last hex digit of all the remaining MAC addresses (in this example it will be 00:0c:e6:48:0c:00 for all three virtual APs).
3. Merge all similar MAC addresses and average their RSS values (MAC: 00:0c:e6:48:0c:00 and RSS: -49.66 dBm).

Note that the structure of MAC addressing of virtual APs may be different in other buildings, so the mac filtering and merging algorithm should be modified accordingly for each building, to merge virtual...
APs to one single physical AP.

The MAC filtering impacts the size of the database significantly and this hugely impacts the complexity of the computation of location using kNN or other algorithms such as [13]. Table 6.2 summarizes the number of APs before and after MAC filtering in the Bahen building for a specific radiomap.

<table>
<thead>
<tr>
<th>Floor Number</th>
<th>No MAC filtering</th>
<th>MAC filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>496</td>
<td>138</td>
</tr>
<tr>
<td>2</td>
<td>368</td>
<td>111</td>
</tr>
<tr>
<td>3</td>
<td>364</td>
<td>113</td>
</tr>
<tr>
<td>4</td>
<td>340</td>
<td>105</td>
</tr>
<tr>
<td>5</td>
<td>311</td>
<td>99</td>
</tr>
<tr>
<td>6</td>
<td>261</td>
<td>82</td>
</tr>
<tr>
<td>7</td>
<td>267</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>179</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 6.2: Number of Wi-Fi APs stored in the database before and after MAC filtering.

As seen in Section 6.1, the complexity and the memory requirement of the weighted kNN algorithm is $O(n_an_r)$ where $n_a$ is the total number of APs and $n_r$ is the total number of RPs in the radiomap. So reducing the number of APs approximately by a factor of 3, as seen in Table 6.2, reduces the complexity and the memory requirement of the weighted kNN algorithm by a factor of 3. The effect of Wi-Fi MAC filtering and merging on the accuracy of localization is discussed in Subsection 6.4.2.

In the next section, we will explain how we used a Kalman filter to combine all sources of information to acquire a better accuracy of indoor localization.

### 6.3 3D location tracking using Kalman filter

We use a linear Kalman filter [58] to fuse the sources of information to get better accuracy of indoor localization. In this section, we show how Kalman filter may be applied to achieve this goal. It is assumed that the reader is familiar with the Kalman filter and hence only the problem statement in terms of Kalman filter is discussed here. A generic Kalman filter is described in (6.7) and (6.8):

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{6.7}$$

$$z_k = H_k x_k + v_k \tag{6.8}$$

where $x_k$ is the state vector, $F_k$ is the transition matrix, $B_k$ is the control-input, $u_k$ is the control vector, $w_k$ is the process noise, $z_k$ is the observation vector, $H_k$ is the observation model and $v_k$ is
the observation noise. The formulations of the prediction and update stages of the Kalman filter is not
provided in this thesis and can be found in [58]. Next, we model the indoor pedestrian tracking problem
in accordance to the Kalman filter framework.

In this system, we define the states as the 3D location and the 3D speed of the user:

\[
x_k = \begin{bmatrix} x_k \\ y_k \\ z_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix}
\] (6.9)

Assuming a constant acceleration of the user at each time step \((dt = 0.02s)\), we can use the following
equation to find the next predicted location and the predicted speed of the user:

\[
x(t) = x(t_0) + v(t_0).dt + \frac{1}{2}a(t_0))dt^2
\] (6.10)

\[
v(t) = v(t_0) + a(t_0).dt
\]

where \(x(t)\) is the location at time \(t\), \(x(t_0)\) is the location at time \(t_0\), \(v(t)\) is the speed at time \(t\), \(v(t_0)\)
is the speed at time \(t_0\) and \(a(t_0)\) is the acceleration at time \(t_0\). The controlled input of the Kalman
filter is the acceleration of the user, which is measured by the linear accelerometer sensor \(l_k\) as discussed
in Sub-Section 2.1.10. However, this raw controlled input is measured with the reference to the device
coordinates and is useless, unless we consider the rotation of the device as mentioned in Sub-Section
2.1.8 and the rotation of the building compared to the real-world coordinates mentioned in Section 4.1.
So the controlled input \(u_k\) with the reference to the floor map coordinates is:

\[
u_k = R_{\text{building}}R_{\text{phone}}l_k
\] (6.11)

where
\[ l_k = \begin{bmatrix} l_{x_k} \\ l_{y_k} \\ l_{z_k} \end{bmatrix} \]

The noise of accelerometer is assumed \( N(0, A) \) where the covariance matrix of noise is \( A = \sigma_a^2 I_{3 \times 3} \).

Using (6.10), we can find matrices \( F_k \) and \( B_k \):

\[
F_k = \begin{bmatrix}
1 & 0 & 0 & dt & 0 & 0 \\
0 & 1 & 0 & 0 & dt & 0 \\
0 & 0 & 1 & 0 & 0 & dt \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (6.12)

\[
B_k = \begin{bmatrix}
\frac{1}{2} dt^2 & 0 & 0 \\
0 & \frac{1}{2} dt^2 & 0 \\
0 & 0 & \frac{1}{2} dt^2 \\
dt & 0 & 0 \\
0 & dt & 0 \\
0 & 0 & dt
\end{bmatrix}
\]

The process noise is assumed to be Gaussian in Kalman filter framework, so \( w_k \sim N(0, Q_k) \) where:

\[
Q_k = \sigma_a^2 B_k R_{\text{building}} R_{\text{phone}} (B_k R_{\text{building}} R_{\text{phone}})^T
\]

Next, we can define the matrices of the observation equation of the Kalman filter (6.8). We have different observations in our system and they may not arrive at the same time, so we define different observation vectors, noises and models. The observations in this system include the following:

1. Wi-Fi based location: Once a Wi-Fi scan is available, it is compared to the radiomap using the weighted kNN algorithm as discussed in Section 6.1 and a 3D location is provided to the Kalman tracker. The observation parameters in this case are:
\[ z_{k}^{\text{wifi}} = \begin{bmatrix} \hat{x}_{k}^{\text{wifi}} \\ \hat{y}_{k}^{\text{wifi}} \\ \hat{z}_{k}^{\text{wifi}} \end{bmatrix} \] \tag{6.13}

\[ H_{k}^{\text{wifi}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] \tag{6.14}

\[ v_{k}^{\text{wifi}} \sim N(0, R_{k}), R_{k} = \sigma_{\text{wifi}}^{2}I_{3 \times 3} \] \tag{6.15}

The value of observation variance is dynamically chosen as discussed in Section 6.1. If the online readings are similar to any of the Wi-Fi fingerprints in the radiomap then the observation error will be a small value. But if the online readings are not that similar to any of the fingerprints inside the radiomap then we use a larger value \( \sigma_{\text{wifi}}^{2} \) to reduce the effect of this observation in the Kalman filter.

2. BLE based location: The observation model for BLE is very similar to the observation from Wi-Fi fingerprinting. The only difference is that the observation error (localization error) of BLE is usually less than the observation error of Wi-Fi due to limited range of BLE beacons, thus \( \sigma_{\text{ble}}^{2} < \sigma_{\text{wifi}}^{2} \). So the Kalman gain for observation from BLE beacons is larger compared to Wi-Fi locations. Indeed, we rely more on BLE beacons whenever they are available.

\[ z_{k}^{\text{ble}} = \begin{bmatrix} \hat{x}_{k}^{\text{ble}} \\ \hat{y}_{k}^{\text{ble}} \\ \hat{z}_{k}^{\text{ble}} \end{bmatrix} \] \tag{6.16}

\[ H_{k}^{\text{ble}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \] \tag{6.17}

\[ v_{k}^{\text{ble}} \sim N(0, R_{k}), R_{k} = \sigma_{\text{ble}}^{2}I_{3 \times 3} \] \tag{6.18}

Again in this system we have used a dynamic value for \( \sigma_{\text{ble}}^{2} \). This variance is chosen to be the distance to the closest BLE beacon seen in the area:
\[ \sigma_{\text{ble}}^2 = \arg \min_i r_i^2, \forall i \] (6.19)

where \( r_i \) is the distance to the BLE beacon \( i \), calculated using (5.4). This means that if the user is close to a BLE beacon, then the localization using BLE beacons is more accurate.

3. GPS based location: The observation from GPS signal is first converted to the indoor coordinates as discussed in Section 4.3. The GPS provides us with a 2D location since the error in altitude is much higher than the height of each floor. Also, the localization error is much higher compared to BLE or Wi-Fi positioning (\( \sigma_{\text{ble}}^2 < \sigma_{\text{wifi}}^2 \ll \sigma_{\text{gps}}^2 \)). So the observation model for GPS is:

\[
\begin{bmatrix}
\hat{x}_{\text{gps}}
\hat{y}_{\text{gps}}
\end{bmatrix}
\]

\[
H_{\text{gps}}^k = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
v_{\text{gps}}^k \sim N(0, R_k), R_k = \sigma_{\text{gps}}^2 I_{2 \times 2}
\]

The value of \( \sigma_{\text{gps}}^2 \) is obtained by squaring the error of the GPS reported by the smartphone. An interesting result of fusing the GPS into the Kalman filter is that when the user walks to the outside of a building, where there is no Wi-Fi radiomap or BLE beacon, the system transparently uses only the GPS to estimate the location and when the user walks back into the building, the system switches back to indoor positioning automatically. In the indoor areas where GPS signal is available, the system uses its location and its reported error.

4. Barometer sensor: As shown in Section 3.3, the barometer can be used to estimate the vertical speed of the user along the \( z \) axis. Next, we fuse this to the Kalman filter as an observation:

\[
\begin{bmatrix}
\dot{z}_{\text{baro}} \\
\dot{\hat{z}}_k
\end{bmatrix}
\]

\[
H_{\text{baro}}^k = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[ z_{\text{baro}}^k = \begin{bmatrix}
\dot{z}_{\text{baro}} \\
\dot{\hat{z}}_k
\end{bmatrix}
\]
Chapter 6. Wi-Fi fingerprinting and sensor fusion using Kalman filter

\[ v_{k}^{\text{baro}} \sim N(0, \sigma_{\text{baro}}^2) \]  \hspace{1cm} (6.25)

Our measurement shows the value of \( \sigma_{\text{baro}}^2 = 0.008 \) for a Nexus 4 device.

5. Game rotation vector sensor: We use this sensor to detect rapid turns. The game rotation vector describes the rotation of devices over its three axes. The difference between the rotation described by the rotation matrix \( R_{\text{phone}} \) and this vector is that, in this sensor, the rotation over the \( z \) axis (heading) is relative (It only uses gyroscope and not magnetometer).

This orientation is expressed in terms of rotations over \( x \), \( y \) and \( z \) axes, which are called \( \text{pitch} \) (\( \beta \)), \( \text{roll} \) (\( \gamma \)) and \( \text{azimuth} \) or \( \text{yaw} \) (\( \alpha \)), respectively (Figure 2.2).

The Kalman filter tracks the user very well once he/she is walking on a straight line. But if the user makes sharp turns, the Kalman filter takes a long time to converge to the new direction of movement. This results in an increase in the localization error. To avoid this problem, when a turn is detected using the gyroscope sensor, we reduce the 2D speed of the user to a small value of \( \epsilon \), so that the Kalman filter finds enough time to adapt itself to the new direction of movement.

It seems reasonable to assume that the user cannot have a high speed while he/she is making a sharp turn. The parameters \( \epsilon = 0.1 \) and \( \sigma_{\text{gyro}}^2 = 0.001 \) are selected to get the best performance.

The observation of gyroscope sensor is inserted to the Kalman filter:

\[ \mathbf{z}_k^{\text{gyro}} = \begin{bmatrix} \epsilon \\ \epsilon \end{bmatrix} \]  \hspace{1cm} (6.26)

\[ \mathbf{H}_k^{\text{gyro}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{1cm} (6.27)

\[ \mathbf{v}_k^{\text{gyro}} \sim N(0, \sigma_{\text{gyro}}^2) \]  \hspace{1cm} (6.28)

The importance of resetting the Kalman filter is elaborated in Subsection 6.4.3.
6.4 The experimental results

We have tested our indoor positioning system in the Bahen Centre for Information Technology, at University of Toronto. Table 6.3 summarizes the parameters of this experiment. The BLE beacons have been installed in BA-4148 only as mentioned in Section 5.2. So we expect to get better accuracy of positioning inside room BA-4148 compared to the other parts of building.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of visible Wi-Fi APs in building</td>
<td>355</td>
</tr>
<tr>
<td>Number of reference points</td>
<td>320</td>
</tr>
<tr>
<td>Number of reference points per floor</td>
<td>40</td>
</tr>
<tr>
<td>Total number of BLE beacons</td>
<td>8</td>
</tr>
<tr>
<td>Transmitting power of BLE beacons ( TX_i ) @ 1 meter</td>
<td>-61dBm</td>
</tr>
<tr>
<td>Kalman filter time step ( dt )</td>
<td>0.02s</td>
</tr>
<tr>
<td>Wi-Fi accuracy constant ( c_{wifi} )</td>
<td>0.15</td>
</tr>
<tr>
<td>Variance of noise of accelerometer ( \sigma^2_a )</td>
<td>0.001</td>
</tr>
<tr>
<td>Variance of noise of barometer ( \sigma^2_{baro} )</td>
<td>0.008</td>
</tr>
<tr>
<td>Variance of noise of gyroscope ( \sigma^2_{gyro} )</td>
<td>0.001</td>
</tr>
<tr>
<td>Reset value of Kalman filter ( \epsilon )</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6.3: Parameters of indoor localizer of Bahen building.
6.4.1 Comparison of BLE-based indoor localization methods

Figure 6.1 compares the different BLE-based indoor localization techniques. The Cumulative Distribution Functions (CDFs) of error of localization for different methods are presented. As seen in this figure, the trilateration technique mentioned in [18] seem to have the best localization accuracy. Hence, we chose this technique and combined it with Wi-Fi fingerprinting in our proposed hybrid system.

Figure 6.1: The CDF of localization error for different BLE localization methods is depicted. Four methods of BLE-based indoor positioning, discussed in Section 5.3, are compared. It seems that localization using the trilateration technique works the best in our experiment.

Table 6.4 summarizes the performance of different BLE-based localization methods.

<table>
<thead>
<tr>
<th>BLE Localization method</th>
<th>Mean of Error (m)</th>
<th>Variance of Error (m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The closest beacon (mentioned in [17])</td>
<td>2.6</td>
<td>1</td>
</tr>
<tr>
<td>Weighted averaging (mentioned in [24])</td>
<td>2.3</td>
<td>0.9</td>
</tr>
<tr>
<td>Fingerprinting (mentioned in [19][20][21][22])</td>
<td>2.3</td>
<td>1</td>
</tr>
<tr>
<td>Trilateration (mentioned in [18])</td>
<td>2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 6.4: Mean and variance of error of localization for different BLE methods, showing that the trilateration technique, discussed in Subsection 5.3.1, has the least mean error.
6.4.2 Effect of Wi-Fi MAC filtering on localization accuracy

Figures 6.2 and 6.3 summarize the effect of MAC filtering on the mean and standard deviation of localization error for different sizes of training database.

Figure 6.4 compares the probability of finding the correct nearest neighbor (1-NN) for the systems with and without using the MAC filtering and Figure 6.5 compares the CDF of error for these two.

Although MAC filtering can show a small improvement in localization error, its more important goal is to reduce the radiomap size and the complexity of localization. Table 6.5 summarizes the performance of localization using 1-NN technique for both systems. In the 1-NN technique, we also compared the results of using the L1 and L2 norms.

<table>
<thead>
<tr>
<th></th>
<th>Mean of Error (m)</th>
<th>Variance of Error (m^2)</th>
<th>Max. Error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No MAC filtering</td>
<td>5.02</td>
<td>1.08</td>
<td>9.78</td>
</tr>
<tr>
<td>MAC filtering-L2 Norm</td>
<td>4.71</td>
<td>0.68</td>
<td>8.31</td>
</tr>
<tr>
<td>MAC filtering-L1 Norm</td>
<td>4.60</td>
<td>0.80</td>
<td>9.53</td>
</tr>
</tbody>
</table>

Table 6.5: Mean, variance and maximum error of localization without MAC filtering, with MAC filtering using L2 norm and with MAC filtering using L1 norm.

Next, the behavior of the tracking module, when the user makes sharp turns, is investigated.

6.4.3 Resetting the Kalman filter at turns

To clarify the importance of resetting the Kalman filter when the user turns at the intersections of corridors, we have done two test walks. Figure 6.6 shows the first path of walk. As seen, the user makes two sharp 90° turns in this walk. Figure 6.7 is the tracking result using a simple Kalman filter. It is obvious that the Kalman filter fails to follow the user at turns due to its linear nature. The readings of the game rotation vector sensor is shown in Figure 6.8 and its derivation, which is the reading of the gyroscope (The velocity over z axis) is shown in Figure 6.9. Now, if we reset the Kalman filter whenever a turn is detected by the gyroscope (by using a proper threshold as shown in Figure 6.9), the tracking result is more accurate, which is depicted in Figure 6.10. However, we may improve the performance even further if we also inject the map information so that we know the exact location of the turn. The result is displayed in Figure 6.11.
Figure 6.2: Mean error of localization with and without MAC filtering for different training set sizes.

Figure 6.3: Standard deviation of localization error with and without MAC filtering for different training set sizes
Figure 6.4: Probability of finding the correct nearest neighbor with and without MAC filtering for different training set sizes.

Figure 6.5: The CDF of localization error with or without MAC filtering for training size of 65.
Figure 6.6: The first path of the walk on the fourth floor of Bahen building in which the user make two sharp 90° turns. Consider the rotation of the floor map compared to the geomagnetic north ($\theta \approx -17^\circ$ for Bahen building).

Figure 6.7: Tracking of the user using the simple Kalman filter for the walk of Figure 6.6.
Figure 6.8: Reading of game rotation vector for the walk of Figure 6.6.

Figure 6.9: Angular velocity over z axis measured by the gyro sensor, for the path of Figure 6.6. By using a proper threshold (solid red lines), we can detect the turns. The Kalman filter is reset at times $t = 30s$ and $t = 50s$. 
Figure 6.10: Tracking of the user using the Kalman filter integrated with game vector for the walk of Figure 6.6.

Figure 6.11: Tracking of the user using the Kalman filter integrated with game vector and map information for the walk of Figure 6.6. The red circles show the point on which we have reset the Kalman filter.
A reasonable question is what if we reset the Kalman filter at each turn without using the gyroscope to see if the user has actually made a turn. To answer this question, we have experimented the walk of path shown in Figure 6.12. In this path, the user makes only one turn. Again Figure 6.13 shows the failure of the Kalman filter when the user makes the turn. Figure 6.14 shows the tracking using a Kalman filter combined with game rotation vector sensor. Figure 6.15 only uses the map information to reset at each intersection and hence the Kalman filter fails to follow the user to the end of the corridor. So the suggested method is combining both the map information and the game rotation vector to decide whether an actual turn is made and a reset to the Kalman filter is required.

Next, the evaluation of performance of the floor detection mechanism is presented.
Figure 6.12: The second path of the walk on the fourth floor of Bahen building. The user makes only one turn.

Figure 6.13: Tracking of the user using the simple Kalman filter for the walk of Figure 6.12.
Figure 6.14: Tracking of the user using the Kalman filter integrated with game vector for the walk of Figure 6.12.

Figure 6.15: Tracking of the user using the Kalman filter and the map information for the walk of Figure 6.12.
6.4.4 Combining BLE-based indoor positioning and Wi-Fi fingerprinting

We found two related works for combining Wi-Fi fingerprinting and BLE-based indoor localization in literature.

1. Authors in [17] propose using Wi-Fi fingerprinting in conjunction with BLE-beaconing. A map of error of Wi-Fi and BLE is created in the offline phase and each of these two might be used, based on their accuracy for that specific area, in the online phase. The proximity method (a.k.a the closest beacon) is selected for BLE-based indoor localization. Wi-Fi fingerprinting is used for localization using Wi-Fi APs. In order to combine the Wi-Fi fingerprinting and BLE-based localization, a spatial distribution of localization error $P(E|\hat{x})$ is estimated in the offline phase (where $\hat{x}$ is the Wi-Fi reading $\hat{x}_{\text{wifi}}$, or the BLE reading, $\hat{x}_{\text{ble}}$). Then, in the online phase the location $\hat{x}_H$ is estimated by switching between Wi-Fi and BLE:

$$
\hat{x}_H = \begin{cases} 
\hat{x}_{\text{ble}} & \text{if } \hat{x}_{\text{ble}} < \hat{x}_{\text{wifi}} \\
\hat{x}_{\text{wifi}} & \text{otherwise}
\end{cases}
$$

(6.29)

This methods is called Hybrid Wi-Fi BLE in the comparing figures.

2. The indoor positioning technique discussed in [24] treats both Wi-Fi APs and BLE beacons in the same manner and estimates the position of the user by calculating a weighted average of the location of the Wi-Fi APs and BLE beacons. Chen et al. [24] assume that the locations of Wi-Fi APs and iBeacons are both known. Then, finding the distance of the receiver to the APs and iBeacons, they estimate the position of the smartphone by weighted averaging of the location of APs and iBeacons. Chen et al. propose using an extended Kalman filter, to combine Pedestrian Dead Reckoning (PDR) with the location acquired by using Wi-Fi APs and iBeacons. The step counter discussed in Section 2.5 is used in evaluation of this system. As we assume that the locations of the Wi-Fi APs are not known to us in practice, we cannot directly compare their method with our proposed hybrid system. The following explains the combining method of [24]:

$$
(\hat{x}, \hat{y}) = \sum_{\text{BLE beacons}} \frac{(x_i, y_i) \times \frac{1}{r_i}}{\sum_j \frac{1}{r_j}} + \sum_{\text{Wi-Fi APs}} \frac{(x_i, y_i) \times \frac{1}{r_i}}{\sum_j \frac{1}{r_j}}
$$

(6.30)

where $(x_i, y_i)$ is the position of either Wi-Fi AP or BLE beacon and $r_i$ is the estimated distance of the receiver to each AP or beacon.
Figure 6.16 compares different methods for combining Wi-Fi fingerprinting with BLE-based indoor localization.

![Cumulative Distribution Function of Error](image)

Figure 6.16: The CDF of localization error for different BLE localization methods, combined with Wi-Fi fingerprinting.

As seen in Figure 6.16, our proposed method, which first finds the BLE-based localization using trilateration technique and then combines it with Wi-Fi fingerprinting using a Kalman filter, has better accuracy than the other methods mentioned in literature. Table 6.6 summarizes the performance of our proposed Wi-Fi fingerprinting combined with BLE trilateration, with other hybrid systems.

<table>
<thead>
<tr>
<th>Localization method</th>
<th>Mean of Error (m)</th>
<th>Variance of Error (m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE Trilateration [18]</td>
<td>2</td>
<td>0.8</td>
</tr>
<tr>
<td>Hybrid Wi-Fi + BLE [17]</td>
<td>1.5</td>
<td>0.8</td>
</tr>
<tr>
<td>BLE Chen et al. [24]</td>
<td>1.8</td>
<td>0.8</td>
</tr>
<tr>
<td>BLE Trilateration (Kalman filtered) (Proposed for iOS devices)</td>
<td>1.7</td>
<td>0.9</td>
</tr>
<tr>
<td>Proposed Wi-Fi + BLE + Sensor fusion</td>
<td>1.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 6.6: Mean and variance of error of localization for different Wi-Fi+BLE methods.

As mentioned earlier, since iOS devices do not allow Wi-Fi scanning, we only rely on BLE-based localization (with tracking) for such devices. Figure 6.17 compares the performance of Wi-Fi fingerprinting, BLE-based positioning and the proposed hybrid system.
Figure 6.17: The CDF of localization error for different hybrid localization methods.

Table 6.7 summarizes the performance of Wi-Fi fingerprinting, BLE-based positioning and the hybrid systems, in terms of mean and variance of error of indoor localization.

<table>
<thead>
<tr>
<th>Localization method</th>
<th>Mean of Error (m)</th>
<th>Variance of Error (m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE Trilateration [18]</td>
<td>2</td>
<td>0.8</td>
</tr>
<tr>
<td>BLE Trilateration (Kalman filtered) (Proposed for iOS devices)</td>
<td>1.7</td>
<td>0.9</td>
</tr>
<tr>
<td>Wi-Fi fingerprinting [8]</td>
<td>4.6</td>
<td>0.8</td>
</tr>
<tr>
<td>Wi-Fi fingerprinting (Kalman filtered)</td>
<td>4</td>
<td>0.7</td>
</tr>
<tr>
<td>Proposed Wi-Fi + BLE + Sensor fusion</td>
<td>1.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 6.7: Mean and variance of error of localization for different hybrid methods.

6.4.5 Comparison of floor detection methods

To get a better view of the difference in the performance our proposed floor detection technique, compared to other systems, consider the confusion matrix depicted in Table 6.8 for floor detection using Wi-Fi fingerprinting using kNN only vs. the confusion matrix of Table 6.9, which is related to our proposed technique for Wi-Fi combined with barometer. The element \((i, j)\) in confusion matrix is the probability of detection of floor \(i\) as floor \(j\), so the closer it is to the identity matrix, the better. The error rate of
floor detection without using barometer is approximately 3% for Bahen building but it is expected to be much larger for shopping malls. As shown in Table 6.8, the error rate between the first and second floor where there are more widely open areas, is about 13%. Using the barometer sensor, we were able to reduce the total error rate to 0.025%. Tables 6.10 and 6.11 show the performance of floor detection system of [7] and [2] respectively.

<table>
<thead>
<tr>
<th></th>
<th>Floor 1</th>
<th>Floor 2</th>
<th>Floor 3</th>
<th>Floor 4</th>
<th>Floor 5</th>
<th>Floor 6</th>
<th>Floor 7</th>
<th>Floor 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor 1</td>
<td>0.875</td>
<td>0.115</td>
<td>0.008</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 2</td>
<td>0.010</td>
<td>0.990</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 3</td>
<td>0.01</td>
<td>0.097</td>
<td>0.892</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.990</td>
<td>0.010</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.004</td>
<td>0.996</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.002</td>
<td>0.998</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.999</td>
<td>0</td>
</tr>
<tr>
<td>Floor 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>0.990</td>
</tr>
</tbody>
</table>

Table 6.8: Confusion matrix of floor detection only using Wi-Fi fingerprinting using kNN method.

<table>
<thead>
<tr>
<th></th>
<th>Floor 1</th>
<th>Floor 2</th>
<th>Floor 3</th>
<th>Floor 4</th>
<th>Floor 5</th>
<th>Floor 6</th>
<th>Floor 7</th>
<th>Floor 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor 1</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 3</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.999</td>
<td>0.001</td>
</tr>
<tr>
<td>Floor 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 6.9: Confusion matrix of floor detection combining Wi-Fi fingerprinting and barometer sensor (proposed method).
Table 6.10: Confusion matrix of floor detection using Wi-Fi fingerprinting using k-means-clustering [7].

<table>
<thead>
<tr>
<th></th>
<th>Floor 1</th>
<th>Floor 2</th>
<th>Floor 3</th>
<th>Floor 4</th>
<th>Floor 5</th>
<th>Floor 6</th>
<th>Floor 7</th>
<th>Floor 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor 1</td>
<td>0.960</td>
<td>0.110</td>
<td>0.010</td>
<td>0.020</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 2</td>
<td>0.01</td>
<td>0.98</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 3</td>
<td>0.002</td>
<td>0.097</td>
<td>0.882</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.990</td>
<td>0.010</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.990</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.010</td>
<td>0.990</td>
<td>0</td>
</tr>
<tr>
<td>Floor 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.010</td>
<td>0.990</td>
</tr>
<tr>
<td>Floor 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.025</td>
</tr>
</tbody>
</table>

Table 6.11: Confusion matrix of floor detection combining Wi-Fi fingerprinting and barometer sensor [2].

<table>
<thead>
<tr>
<th></th>
<th>Floor 1</th>
<th>Floor 2</th>
<th>Floor 3</th>
<th>Floor 4</th>
<th>Floor 5</th>
<th>Floor 6</th>
<th>Floor 7</th>
<th>Floor 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor 1</td>
<td>0.970</td>
<td>0.020</td>
<td>0.010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 2</td>
<td>0</td>
<td>0.980</td>
<td>0.010</td>
<td>0.010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 3</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.990</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Floor 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.010</td>
<td>0.980</td>
<td>0.010</td>
</tr>
<tr>
<td>Floor 7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.990</td>
<td>0.010</td>
</tr>
<tr>
<td>Floor 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.020</td>
</tr>
</tbody>
</table>

Table 6.12 compares the error rate of different floor detection methods. The total error rate for all floors of Bahen building and also error the rate for confusing the first to the second floor, which is the most erroneous due to having open areas, are listed.

<table>
<thead>
<tr>
<th>Floor detection method</th>
<th>Total error rate (%)</th>
<th>Open area error rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi-Fi kNN</td>
<td>3.4</td>
<td>12.5</td>
</tr>
<tr>
<td>Razavi et al. Wi-Fi k-means-clustering [7]</td>
<td>4.3</td>
<td>4</td>
</tr>
<tr>
<td>Wang et al. Barometer + Wi-Fi [2]</td>
<td>1.3</td>
<td>3</td>
</tr>
<tr>
<td>Proposed Wi-Fi + Barometer + Kalman filter</td>
<td>0.025</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6.12: Comparison of different floor detection methods.
6.4.6 Overall indoor positioning accuracy

Figure 6.18 compares the mean error of localization for points A to F, using different methods. As seen in this figure, the mean error of indoor localization is decreased by combining the BLE beaconing, Wi-Fi fingerprinting and sensor fusion. At point A to D, where we have good coverage of BLE beaconing, the mean localization error is approximately one meter. As we move to points E and F, the mean error of localization using BLE beaconing begins to increase. Hence, the total mean error is increased but it is still less than the error of localization using Wi-Fi fingerprinting only.

The accuracy of our indoor positioning system is summarized in Table 6.13. As shown in Table 6.13, in order to predict the localization error for an iOS device (iPhone, iPad, etc), we have disabled the Wi-Fi scanner so the device relies only on BLE beacons wherever present. In the areas that no BLE beacons were installed, the indoor localization error accumulates after a short time, as there are no independent observations to correct the predictions of the Kalman filter. Wi-Fi fingerprinting is not usable on iOS devices but other sensors such as accelerometer, gyroscope, magnetometer and barometer are accessible on iOS devices as well. As a result, these devices will have less accuracy of indoor positioning. In Section 7.3, we show how Android devices may work as soft BLE beacons to aid the indoor positioning of iOS devices.

Figure 6.19 shows the developed application on Android OS. As shown in this figure, the location results, which are in the coordinates of the floor map, are converted back to GPS coordinates using (4.1) and then displayed on Google Maps.

Figure 6.18: Mean error of localization for different points in the vicinity of the testing area. The location of points A to F is shown in Figure 5.4.
Table 6.13: Accuracy result of indoor localizer of Bahen building

<table>
<thead>
<tr>
<th>Description</th>
<th>Error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean error of Wi-Fi fingerprinting (no tracking)</td>
<td>≈4.6m</td>
</tr>
<tr>
<td>Mean error of Wi-Fi fingerprinting (basic Kalman filter)</td>
<td>≈4m</td>
</tr>
<tr>
<td>Mean error of Wi-Fi fingerprinting and BLE beaconing</td>
<td>≈1.1m</td>
</tr>
<tr>
<td>Mean error of BLE beaconing for iOS emulation (tracking)</td>
<td>≈1.7m</td>
</tr>
<tr>
<td>Floor detection error rate (Wi-Fi and barometer)</td>
<td>0.025%</td>
</tr>
<tr>
<td>Floor detection error rate (Wi-Fi only)</td>
<td>3%</td>
</tr>
<tr>
<td>Floor detection error rate of 1st and 2nd floors (Wi-Fi only)</td>
<td>13%</td>
</tr>
</tbody>
</table>

Figure 6.19: The implementation of the system on Android OS. Location results are converted to geo-coordinates using (4.1) and displayed on Google Maps. The blue circle shows the margin of localization error. The star shows the true location and the circle shows the estimated location.
Chapter 7

Large-scale indoor positioning and cooperative localization

So far, we have focused on building our localization system for the Bahen building at University of Toronto. Our final goal is to base an indoor positioning system that can be used all around the world for each building. In order to do that, we have to look into the problem of large-scale indoor positioning in which we have thousands of building, millions of reference points and millions of Wi-Fi or Bluetooth APs.

It is obvious that such a large database cannot be maintained on a smartphone and should be kept on a localization server. However, we need to find very efficient algorithms to manage and query such a large localization database. As it is usually difficult to stream the real-time sensor data to the localization server, the tracking module and the sensor processing systems still run on the smartphone, as shown in Figure 7.1. Figure 7.2 shows the connection diagram of the server-based indoor positioning on Android devices. We have are running our final localization server on Google Cloud to simplify the maintenance of the server and the database. As shown in Figure 7.2, the Android smartphone collects the Wi-Fi scan and wraps them in Java Script Object Notation (JSON) format and sends the data to localization server through Hyper Text Transfer Protocol-Secured (HTTPs). The localization application finds the position of the user and responses to the client with its location, wrapped in JSON packet.

We already discussed the complexity and the memory requirement of the weighted kNN algorithm in Section 6.1. Next, in Section 7.1 we explain our solution to large-scale indoor positioning and we find its complexity and memory requirements.
Chapter 7. Large-scale indoor positioning and cooperative localization

Figure 7.1: The block diagram of the server-based indoor positioning system. The Wi-Fi scan results are sent to the localization server via the Internet and the received location result is injected to the Kalman filter.

7.1 The proposed algorithm for large-scale indoor positioning

To reduce the complexity of the weighted kNN algorithm, which was discussed in Section 6.1, we consider the following facts:

1. Due to the limited range of Wi-Fi APs, each AP is visible at a limited number of reference points. The number of RPs in which an AP is visible depends on how close the RPs are selected but it is usually less than 20. In other worlds, the radiomap matrix $\Psi$ is sparse.

2. The MAC addresses of the APs are unique. Hence, the AP might be used as a primary key in the database.

Now let us deform the radiomap $\Psi$ into two separate tables, Table 7.1 and Table 7.2.

<table>
<thead>
<tr>
<th>AP$_1$</th>
<th>RP$_1$</th>
<th>RSS$_s11$</th>
<th>RP$_2$</th>
<th>RSS$_s12$</th>
<th>...</th>
<th>RP$_{m1}$</th>
<th>RSS$_{s1m1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP$_2$</td>
<td>RP$_1$</td>
<td>RSS$_s21$</td>
<td>RP$_2$</td>
<td>RSS$_s22$</td>
<td>...</td>
<td>RP$_{m2}$</td>
<td>RSS$_{s2m2}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>AP$_{n}$</td>
<td>RP$_1$</td>
<td>RSS$_{sn1}$</td>
<td>RP$_2$</td>
<td>RSS$_{sn2}$</td>
<td>...</td>
<td>$P_{map}$</td>
<td>RSS$<em>{snm</em>{map}}$</td>
</tr>
</tbody>
</table>

Table 7.1: AP-RP database describes the radiomap (a.k.a AP-RP-TABLE).

In Table 7.1, each row represents an access point and the reference points in which this AP is visible. AP$_i$ is the MAC address of the AP, which is unique as mentioned, RP$_j$ is an ID assigned to reference point $j$ and RSS$_{ij}$ is the received signal strength of AP$_i$ at RP$_j$. The number of RPs for which an AP...
Figure 7.2: Connection diagram of the server-based indoor localization for Android devices. The Android smartphone collects the Wi-Fi scan and wraps them in JSON format and sends the data to localization server through HTTPs. The localization application finds the position of the user and responds to the client with its location, wrapped in JSON packet.

is visible is \( m_i \), which is a small number as mentioned (in our experiments \( m_i < 20, \forall i \)). As we see in Subsection 7.1.1, we build Table 7.1 sorted upon its primary key, which is the MAC address of APs. So each row of Table 7.1 may be expressed as:

\[
\text{AP}_1 \rightarrow \{(RP_i, RSS_{ij}) \mid \text{AP}_j \text{ is visible at } RP_i\}
\] (7.1)

Indeed, each row of Table 7.1 is a linked list of pairs \((RP_i, RSS_{ij})\) connected to the key of \(\text{AP}_1\).

| RP \(_1\) | Lat\(_1\) | Long\(_1\) | Floor\(_1\) | AP\(_1\) | RSS\(_{11}\) | AP\(_1\) | RSS\(_{12}\) | \(\ldots\) | AP\(_{n_1}\) | RSS\(_{n_1}\) |
|-----|------|--------|---------|-------|--------|-------|--------| \(\ldots\) | ------- | ------ |
| RP \(_2\) | Lat\(_2\) | Long\(_2\) | Floor\(_2\) | AP\(_1\) | RSS\(_{12}\) | AP\(_2\) | RSS\(_{22}\) | \(\ldots\) | AP\(_{n_2}\) | RSS\(_{n_2}\) |
| \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) | \(\vdots\) |
| RP \(_{n_r}\) | Lat\(_{n_r}\) | Long\(_{n_r}\) | Floor\(_{n_r}\) | AP\(_1\) | RSS\(_{1n_r}\) | AP\(_2\) | RSS\(_{2n_r}\) | \(\ldots\) | AP\(_{n_{n_r}}\) | RSS\(_{n_{n_r}}\) |

Table 7.2: Table of the reference points and their corresponding positions, the visible APs and their RSS values for that AP (a.k.a RP-LOC-AP-TABLE).

In Table 7.2, we store the RPs, their geo-coordinates and the APs, which are visible at that RP. We also build Table 7.2 such that it is sorted based on its primary key, which is the enumeration number of the RPs. Table 7.2 is in fact another representation of Table 7.1 plus the geo-coordinates of the RPs. Each row of this table shows the APs, which are visible at a certain RP and their corresponding RSS values. The number of APs at each RP is assumed to be limited \((n_i < 20, \forall i)\). The memory requirement of Table 7.1 is \(O(n_a)\) and the memory requirement of Table 7.2 is \(O(n_r)\). So the total required memory is \(O(n_a + n_r) \equiv O(n)\), compared to the conventional kNN, which was \(O(n^2)\). Each row of Table 7.2 may
be expressed as:

\[ \mathbf{RP}_i \rightarrow \{ \mathbf{g}, (\mathbf{AP}_j, RSS_{ij}) \mid \mathbf{AP}_j \text{ is visible at } \mathbf{RP}_i \} \quad (7.2) \]

where \( \mathbf{g}_i = (\text{Lat}_i, \text{Long}_i, \text{Floor}_i) \) is the geo-coordinates of \( \mathbf{RP}_i \). Indeed, each row of Table 7.2 consists of the geo-coordinates of the RP \( \mathbf{g}_i \) and a linked list of pairs \( (\mathbf{AP}_j, RSS_{ij}) \) connected to the key of \( \mathbf{RP}_i \).

Assume that a Wi-Fi scan is expressed as the following set of pairs:

\[ S = \{ (MAC_1, rss_1), (MAC_2, rss_2), \ldots, (MAC_n, rss_n) \} \]

where \( MAC_i \) is the MAC address of the AP and \( rss_i \) is its received signal strength. Now define the mentioned operations. Now we define the following operations and we find their complexity:

- Adding a reference point to the radiomap described in Subsection 7.1.1.
- Deleting a reference point from radiomap described in Subsection 7.1.2.
- Localizing the user by using online Wi-Fi scan described in 7.1.3.

Different data structure and search algorithms may be used to implement the mentioned operations on a database. We have used Binary Search Trees (BST) \[59\]. Here are the definition of BST operations and their complexity:

- \( BST - Query(\text{tab}, \text{key}) \): Searches the primary key \( \text{key} \) in the table \( \text{tab} \) and returns its corresponding entry. This function returns \text{null} if \( \text{key} \) is not found in the table.
- \( BST - Insert(\text{tab}, \text{key}, \text{list}) \): Searches the primary key \( \text{key} \) in the table \( \text{tab} \) and adds the linked list \( \text{list} \) to the key. This function returns \text{null} if \( \text{key} \) is not found in the table.
- \( BST - Delete(\text{tab}, \text{key}) \): Searches the primary key \( \text{key} \) in the table \( \text{tab} \) and deletes the key and all its data from the table \( \text{tab} \). Does nothing if \( \text{key} \) is not found.

The complexity of all the three mentioned operations is \( O(\log(n_{rows})) \) where \( n_{rows} \) is number of rows in the table.

We also need the following operations on linked lists:

- \( AddToLinkedList(\text{list}, \text{obj}) \): Adds the object \( \text{obj} \) to the linked list \( \text{list} \). This operation can be performed in \( O_{n_{list}} \), where \( n_{list} \) is the number of objects in the linked list.
• *DeleteFromLinkedList(list, obj)*: Removes the object *obj* from the linked list *list*. This operation can also be performed in $O_{n_{list}}$.

### 7.1.1 Adding a reference point to the radiomap

To add a reference point to the radiomap, we need to enter its Wi-Fi scans in the Table 7.1 and Table 7.2. To do so, we look into the MAC address of APs found and search for them in Table 7.1. If the AP already exist in the table, we add the pair of RP number and its RSS value $(r_{pj}, r_{ss})$ to its row. If the AP does not exist in Table 7.1, first we need to create a row for it in the Table 7.1 and then add the mentioned pair. After that, we add a row in Table 7.2 for the RP. This row consists of the RP number, its geo-coordinates and the visible APs for that RP and their corresponding RSS values. Algorithm 1 is the pseudo-code for adding an RP to the database.

**Algorithm 1:** Adding a reference point to the radiomap

```plaintext
input: Wi-Fi scan $S = \{(MAC_1, r_{ss_1}), (MAC_2, r_{ss_2}), \ldots, (MAC_{n_o}, r_{ss_{n_o}})\}$ and the reference point $r_{pj}$ and its geo-coordinates $g$

output: Augmented radiomap

1 for $MAC_i \in S$ do
2 \hspace{1em} rplist $=$ BST-Query(AP-RP-TABLE, $MAC_i$) /* Is $MAC_i$ is already in the table? */
3 \hspace{1em} if rplist $=$ null then
4 \hspace{2em} Initialize rplist /* Initialize an empty list of RPs for this AP */
5 \hspace{2em} BST-Insert(AP-RP-TABLE, $MAC_i$, rplist) /* Creates a row for $MAC_i$ in the AP-RP-TABLE */
6 \hspace{2em} AddtoLinkedList(rplist, $(r_{pj}, r_{ss})$) /* Adds $r_{pj}$ and its RSS value to the $MAC_i$ row of AP-RP-TABLE */
7 end
8 BST-Insert(RP-LOC-AP-TABLE, $r_{pj}$, ($g, \{(MAC_1, r_{ss_1}), (MAC_2, r_{ss_2}), \ldots, (MAC_{n_o}, r_{ss_{n_o}})\}$))
```

To calculate the complexity of Algorithm 1, we remind the reader that the number of APs in the Wi-Fi scan $n_o$ and the number of RPs on which an AP is visible $m_i$ are both limited constant numbers, which are assumed to be less than 20. As seen in Algorithm 1, the worst case scenario happens when none of the APs in the online Wi-Fi scan exists in the table then $n_o$ queries and insertions need to be done on Table 7.1. Also, $n_o$ adding operations to a linked list of size $m_i$ should be done. Finally, an insertion on Table 7.2 should be performed. So the number of required operations are:

\[
n_o \times (O(\log(n_o)) + O(\log(n_o)) + O(\log(n_r)) + O(n_r)) \equiv O(\log(n_o)) + O(\log(n_r)) \equiv O(\log(n)) \tag{7.3}
\]
7.1.2 Deleting a reference point from radiomap

To delete a RP from the radiomap, first we need to acquire the list of visible APs from Table 7.2. Then, we have to search Table 7.1 to find these APs and remove the desired RP from their linked lists. Algorithm 2 explains the removing of an RP from the database.

Algorithm 2: Deleting a reference point from the radiomap

<table>
<thead>
<tr>
<th>input : The reference point $r_p_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output: Reduced radiomap</td>
</tr>
</tbody>
</table>

1. $aplist =$BST-Query(RP-LOC-AP-TABLE,$r_p_j$) /* Get the list of APs for $r_p_j$ */
2. for $MAC_i \in aplist$ do
3. $rplist =$BST-Query(AP-RP-TABLE,$MAC_i$) /* Find $MAC_i$ in the AP-RP-TABLE */
4. DeletefromLinkedList(rplist,$r_p_j$) /* Deletes $r_p_j$ from the $MAC_i$ row of AP-RP-TABLE */
5. end
6. BST-Delete(RP-LOC-AP-TABLE, $r_p_j$)

Similar to the computation required for adding a RP, we may calculate the required operations for removing a RP from the database:

$$O(n_r) + n_o \times (O(log(n_a)) + O(m_i)) \equiv O(log(n_a)) + O(log(n_r)) \equiv O(log(n)) \quad (7.4)$$

7.1.3 Localizing the user by using online Wi-Fi scan

The most frequently used operation is the localization of the user by using the online Wi-Fi scans. To find the location of the user, first we build a much smaller subset of the radiomap denoted as $\Psi'$. To do so, we look into the APs in the online Wi-Fi scan and search for them in Table 7.1. Then, we gather all RPs and their RSS values for which this AP was visible. Finally, we put them in the reduced-size radiomap matrix $\Psi'$. This is explained in detail in Algorithm 3.

Algorithm 3: Choosing the appropriate subset of radiomap $\Psi'$

<table>
<thead>
<tr>
<th>input : Wi-Fi scan $S = {(MAC_1, rss_1), (MAC_2, rss_2), \ldots, (MAC_{n_o}, rss_{n_o})}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output: Subset of radiomap to use for localization $\Psi'$</td>
</tr>
</tbody>
</table>

1. Initialize matrix $\Psi'$
2. for $MAC_i \in S$ do
3. $rplist =$BST-Query(AP-RP-TABLE,$MAC_i$) /* Find $MAC_i$ in the AP-RP-TABLE */
4. for $RP_j \in rplist$ do
5. $\Psi'_{ij} = RSS_{ij}$ /* Find all RPs which have $MAC_i$ and put their RSS values in $\Psi'$ */
6. end
7. end

First, let us find the complexity of Algorithm 3. Similar to what we have seen in the operations of adding and removing RPs from the database, we may calculate the complexity of Algorithm 3 as follows:
\[ n_o \times (O(\log(n_a)) + O(m_i)) \equiv O(\log(n_a)) \quad (7.5) \]

It is essential to mention that the size of the reduced radiomap matrix \( \Psi' \) does not depend on the number of APs \( n_a \) nor the number of RPs \( n_r \) in the database. The maximum size of \( \Psi' \) will be \( n_o \times m_i \) and as both numbers are limited to 20, the maximum dimension of \( \Psi' \) will be \( 20 \times 20 \) no matter how large the actual database is.

Now to localize the user, we can run any machine learning algorithm such as the weighted kNN algorithm on the reduced-size radiomap \( \Psi' \). As the maximum size of the reduced radiomap is now constant, no matter how complicated our localization algorithm is, it does not affect the growth of the computational complexity of the localization, as the radiomap expands. To acquire the geo-coordinates of the desired RPs we also have the complexity of \( O(\log(n_r)) \). So the total number of computations is:

\[ O(\log(n_a)) + O(\log(n_r)) \equiv O(\log(n)) \quad (7.6) \]

We conclude that all required operations including adding and removing a RP and also localizing the user may be performed in \( O(\log(n)) \) time, which is a significant improvement, compared to the conventional method, which was \( O(n^2) \).

### 7.2 Evaluation of the performance of the server-based localizer

To evaluate the performance of our proposed large-scale indoor positioning algorithms, which was discussed in Section 7.1, we have implemented the mentioned algorithms and we have benchmarked them in real world. To evaluate the performance of our proposed large-scale indoor positioning algorithms, which were discussed in Section 7.1, we have implemented and benchmarked these algorithms on a normal PC. Table 7.3 describes the specification of the PC used to run the proposed algorithms.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU type</td>
<td>Intel(R) Core(TM) i7-3615QM</td>
</tr>
<tr>
<td>CPU clock frequency</td>
<td>2.30 GHz</td>
</tr>
<tr>
<td>CPU cores number</td>
<td>4</td>
</tr>
<tr>
<td>Installed memory (RAM)</td>
<td>8.00 GB</td>
</tr>
<tr>
<td>OS type</td>
<td>Windows Server 2008 R2 Standard 64-bit</td>
</tr>
<tr>
<td>Database application</td>
<td>MySQL Sever 5.6.30 64-bit</td>
</tr>
<tr>
<td>Database query engine</td>
<td>InnoDB</td>
</tr>
</tbody>
</table>

Table 7.3: Parameters of the PC running large-scale localization algorithms.
The complexity of adding and removing reference points and also the complexity of the localization operation, versus the number of the RPs in the database, is depicted in Figure 7.3. As shown in this figure, the localization operation, which is used much more often than adding and removing RPs, is the least complex operation among the three operations. Adding RP operation is more time-consuming as it needs the writing to memory (and disk,) and the removing RP operation is the most time-consuming, which is used rarely.

In another experiment, we have verified the capacity of our localization server in terms of the number of localization requests per second, which it can handle. Table 7.4 describes the parameters of the radiomap for this experiment and the response time of server for different load works is depicted in Figure 7.4. The number of APs for each RP is chosen randomly between 1 and 39, so the average number of APs for each RP will be 20.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of Wi-Fi APs in database</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Total number of reference points in database</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Number of APs per reference point</td>
<td>Uniform from 1 to 39</td>
</tr>
<tr>
<td>Number of APs in online Wi-Fi scan</td>
<td>Uniform from 1 to 39</td>
</tr>
</tbody>
</table>

Table 7.4: Parameters of the radiomap used to verify the capacity of the server.
Figure 7.4: The response time of the server vs. the number of requests per second is displayed. The server has almost reached its capacity as the service rate is almost equal to the input rate of the queue.

As shown in Figure 7.4, the server reaches its capacity at approximately 850 localization requests per second. If each Wi-Fi scan takes two seconds on a typical Android device, then our server would be able to localize 1700 Android devices in real-time.

7.3 Cooperative indoor localization

In this thesis, we mostly discussed the indoor positioning for Android devices. However, the indoor positioning still remains a challenge for iOS devices due to several restrictions of this operating system. In Chapter 5, we showed that it is possible to implement an indoor positioning system for both iOS and Android devices based on BLE technology. However, this technology needs installation and maintenance of an extra infrastructure, which might be expensive or even not allowed in many buildings such as hospitals. To overcome this problem, we propose a cooperation between Android and iOS devices using BLE connectivity. There are two approaches to achieve this goal.

The first schema for the cooperative indoor localization of iOS devices is shown in Figure 7.5. In this schema, the Android phone localizes itself by sending the Wi-Fi scans to the localization server.
After it receives its location, it broadcasts its location as a BLE beacon to assist iOS devices localizing themselves.

![Figure 7.5: Android phone localizes itself and broadcasts its location to iOS devices using BLE.](image)

The iOS device uses the Android devices as BLE beacons. It measures the RSS of each of them to estimate its distance to each of these smartphones, which we call them soft beacons. As the location of the Android devices are known to the iOS device (with some error), it can estimate its location using the methods described in Section 5.3. The positioning of the iOS devices will be less accurate using the soft BLE beacons, due to the error in the positioning of each of these beacons. However, as this cooperative method mitigates the requirement of the BLE infrastructure, it can be very helpful specially when there are many Android smartphones around to use. Despite what is shown in Figure 7.5, the Android devices may also localize themselves without using the localization server if they have a copy of the radiomap.

The problem with the first schema is that only a few models of Android phones (Nexus 5x, Nexus 6, Nexus 9, etc) are capable of broadcasting BLE packets. Fortunately, most of the Android phones are capable of receiving BLE advertisements while most of iOS devices are capable of transmitting these packets. So we reverse the direction of the transmission of the BLE advertisement packets and propose the second schema as shown in Figure 7.6.

In this schema, again the Android phone localizes itself by sending the Wi-Fi scans to the localization server. It also scans for the BLE advertisements sent by iOS devices and reports them to the server. The iOS devices send out BLE packets with specific ID and at the same time they send that ID to the
localization server using the Internet connectivity. If we have at least one Android device close to the iOS device, we can easily localize the iOS device using the techniques discussed in Section 5.3. Figure 7.2 and Figure 7.7 describe the required software components for the implementation of the cooperative localization on iOS devices.

In order to test our cooperative localization approach, we have done an experiment in which three Android phones localize themselves using Wi-Fi fingerprinting (in conjunction with tracking and sensor
fusion methods discussed in Chapter 6) and then broadcast their locations through BLE, as discussed in the first schema. Figure 7.8 shows the setup of this experiment. No hardware BLE was used in this experiment and in fact the Android devices were used as soft BLE beacons.

Figure 7.8: The cooperation between Android and iOS devices is shown in this figure. We have put three Nexus 5X devices at the three corners of the lab BA-4148 and an iPhone 6s device in the middle. The Android devices localize themselves by Wi-Fi fingerprinting and then broadcast their locations through BLE. The blue circles are the reference point for the Wi-Fi fingerprinting based positioning.

The accuracy of localization results are shown in Table 7.5.

Having no extra BLE beacons in the environment, we can compare the achieved mean error of indoor localization for Android devices (1.5m) to the mean error of indoor localization of iOS devices (2m). It is
expected to have less accuracy in localization of iOS devices, as their location is acquired by aggregation of three or more erroneous positions reported by Android devices.

We have not tested the second schema but we expect to get the same results, because of the reciprocity of the BLE transmitter and the receiver.

### 7.4 Conclusion and future work

In this thesis, we represented a 3D indoor/outdoor localization system, which uses not only Wi-Fi fingerprinting, but also other sources of information such as barometer sensor, accelerometer, gyroscope, BLE scanner, GPS, etc, to find the position of a smartphone inside/outside a building.

We showed that promising results may be achieved by using the barometer sensor for floor detection. Our indoor localization system not only uses Wi-Fi scanner, but also uses the barometer sensor to facilitate the problem of floor detection. We showed that the barometer sensor provides a good source of information for floor detection, but the absolute value of pressure may not be used to detect the floor number. Differentiating of the barometer readings were examined and we have shown that the readings from the barometer sensor may be combined with the location acquired through Wi-Fi fingerprinting using a Kalman filter. Real-world experiments have shown the high accuracy of the system in detection of the floors.

We also discussed how GPS may contribute to the problem of indoor localization whenever it is available. We showed that by combining the GPS signal with readings from Wi-Fi and BLE scanner, we can transparently and softly switch between indoor and outdoor positioning. We discussed how we may convert the GPS location to pixel numbers on a floor plan and vice versa. Using the discussed technique for mapping the pixel numbers to GPS coordinates, the results of indoor/outdoor positioning was shown on Google Maps.

Using of BLE technology for indoor localization was considered and instead of using iBeacon packets, we have developed a new packet format for BLE beacons, which embeds the location, the transmitting power and other characteristics of the beacon. We have proposed and implemented a technique to combine two well-known indoor positioning methods, Wi-Fi fingerprinting and BLE beaconing, using the Kalman filter. The system achieves better positioning accuracy by combining the two sources of
information using a Kalman filter. The experimental results show that indoor localization by combining Wi-Fi fingerprinting and BLE beaconing is more accurate than using each of them separately.

It was shown that virtual APs cause huge redundancy in the radiomap of indoor localization using Wi-Fi fingerprinting. The filtering and merging of the redundant fingerprints of Wi-Fi APs was also investigated and it was shown that by merging the virtual APs into a single physical AP and also filtering of tethering devices, we can gain accuracy in localization and can reduce the size of radiomap and the complexity of localization significantly. Hence, the computational complexity of Wi-Fi fingerprinting was reduced.

We showed how different sensor information may be fused using a linear Kalman filter and we investigated the problem of tracking of the user at sharp turns. We saw that using the gyroscope sensor, we can reset the Kalman filter when necessary to avoid issues at turns. We also discussed how the different sources of information contribute to indoor positioning and showed that by fusing the mentioned information into a Kalman filter, the accuracy of indoor localization is increased significantly. Accuracy results were provided, which indicated the robustness of this system. The devices which do not support Wi-Fi scanning are expected to be less accurate in indoor positioning due to the restriction on Wi-Fi scanning for third-party applications. However, our system can still be used on those devices using BLE beacons and the inertial and environmental sensors.

To simulate and implement our proposed positioning system, we have also built an Android sensor transmission system that facilitates this process. This system simplified the procedure of designing smartphone applications, which use sensors such as accelerometer, gyroscope, magnetometer, etc. All of indoor positioning algorithms are implemented in Simulink software and tested in real-time. So the whole localization system will be a black box, which is created by the Matlab software and works as core for the final indoor positioning application on the Android phone/tablet.

The problem of large-scale indoor positioning in which the radiomap contains the Wi-Fi fingerprints of thousands of buildings, was also addressed and efficient algorithms for indoor localization were proposed and implemented. We proposed algorithms to make Wi-Fi fingerprinting based indoor localization more efficient, when the radiomap covers many buildings, with a large number of RPs and APs. Efficient algorithms for adding RPs to and removing RPs from the database were explained. It was shown that using an efficient search, the size of the radiomap for online phase of indoor positioning can be reduced to only relevant RPs and APs. Using the significantly reduced-size radiomap, sophisticated fingerprinting algorithms can be carried out, independent of the original radiomap size. The computational complexity of the proposed algorithm was investigated and the complexity of $O(\log(n))$ for the weighted kNN was achieved. The proposed algorithms were implemented using SQL database on a normal PC and the
The complexity of the system was numerically evaluated. Our server-based Wi-Fi fingerprinting localizer was migrated to Google Cloud, after benchmarking.

In conclusion, the indoor localization accuracy of around one meter is achievable on commercial smartphones, without requiring expensive hardware infrastructure. Also, the performance of Wi-Fi fingerprinting can be significantly improved by using other sensors of the smartphone and tracking the user with a Kalman filter.

A cooperative approach was suggested, to facilitate the positioning of iOS devices without an extra BLE infrastructure. In the cooperative schema, Android phones localized themselves using Wi-Fi fingerprinting and then helped iOS devices find their locations. Promising accuracy results were obtained, which indicated that this method may replace using of physical BLE beacons.

The proposed indoor/outdoor localization system was implemented on Android smartphones and tested in real-time. An average indoor localization accuracy of around one meter was obtained by combining all the mentioned techniques. Although the system was not implemented on iOS, the predicted localization error was emulated by disabling the Wi-Fi scanner of the Android smartphones.

The following are suggested as possible future works for indoor localization:

- Using magnetic fingerprinting for indoor localization.
- Combining vision-based indoor localization methods with Wi-Fi fingerprinting and BLE beaming to obtain sub-meter accuracy.
- Considering the difference in the Wi-Fi RSS of different smartphones (due to different antenna gains, etc.), to get better accuracy of Wi-Fi fingerprinting.
- Implementation of the proposed ubiquitous localization system on iOS devices using Objective-C/Swift programming languages.
- Implementation of the second cooperative schema in which, the iOS device broadcasts BLE packets and Android devices estimate their distances to the iOS device by measuring the RSS of the received packets and reporting the distances to a localization server. The server then estimates the location of the iOS device.
- Cooperation between Android/iOS devices using BLE to find their distances to each other and improve the localization accuracy of each other.
Bibliography


[38] Y. Kim, Y. Hwang, S. Choi, and J. Lee, “Height estimation scheme of low-cost pedestrian dead-reckoning system using Kalman Filter and walk condition estimation algorithm,” in *Advanced


