Indoor Positioning Using IBeacon BLE Technology

by

Jake Davies

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

B.SC. COMPUTER SCIENCE HONOURS

in

Irving K. Barber School of Arts and Sciences

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

April 2016

Abstract

Indoor navigation is a widely discussed but mostly unsolved problem in the world today. There are many technologies that can be used, such as WIFI, Bluetooth or Sonar, but there is no de-facto standard in this space. Indoor navigation can be applied to many areas of society, from helping the blind navigate locations they depend on for everyday life, to helping people who are unfamiliar with a new location find the things they are looking for. We have explored the possibility of using iBeacon technology(A bluetooth device designed for bluetooth proximity detection on mobile devices) for indoor navigation when combined with the various sensors inside mobile devices. Two main methods have been explored, one being trilateration and the other being proximity based movement approximation. Thorough tests of both methods as well as the accuracy of the IBeacon technology will be displayed. Early analysis suggests that the trilateration method is not a viable method for positioning due to the vulnerabilities to inaccuracies that trilateration algorithms possess, but the proximity based movement detection is a more promising solution.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank Dr. Abdallah Mohammed for his guidance and insights throughout the year while working on this.

0.1 Terminology

- Magnometer

  A sensor that can read magnetic Field information. Often used for readings of Earths magnetic field.

- Accelerometer

  A sensor that can read acceleration in multiple axis.

- JSON

  Javascript Object Notation - A common file encoding for web APIs and configuration files.

Chapter 1

Introduction

iBeacon technology itself was developed for proximity based notification and alerting systems. We explored the possibility of using that same technology to get the absolute position of a device in 2d space.

1.1 Motivations

This project was suggested by a sight impaired colleague of Dr. Abdallah Mohammed. He made it clear that the challenges associated with sight impaired navigation indoors were largely unsolved, and that this area was a mostly unexplored research area. With this came the suggestion to use Apple's iBeacon technology to see if we could leverage the low cost bluetooth devices to create an easily deployed positioning solution for applications such as grocery stores and conferences.

1.2 Contributions

Our contributions consist of two different methods for positioning, a new flexible map format, a grid-based path-finding implementation, and a custom map renderer.

Chapter 2

Related Work

2.1 Positioning

Various work has been done both in the area of iBeacon based positioning as well as non iBeacon based positioning techniques. One other common technique is a wifi based method.

2.1.1 Other work using iBeacon's

The company that manufactures the iBeacon's we used in our tests (Estimote) has released a publicly available SDK for indoor navigation on iOS devices. Unfortunately they do not provide any android versions of this software. The reason for not having an android version of this software is because android devices can vary significantly in their capabilities and sensors, and it was not feasible to create software that worked on every device at the current time. Because we wanted to create software that was not restricted to certain devices, and could be applied to general purpose applications, this was not a solution for us.

Tsz Ming Ng talks about a method similar to our positioning method 2, which is what our implementation was based on. He called it "Destination directing with proximity and compassing". "The whole concept is to know all factors that will affect the resulting direction. By calibrating user's orientation with an objective indicator (compass points) and finding the direction between two locations, result can be obtained and shown to the user." [Ng15]

There is another method called the "Non-Linear Least Squares" method which is similar to our trilateration method, but it also factors in some correction techniques. Least Squares is a method to fit a parametrized function to a set of observations by minimizing the sum of the squared

residuals, where the residual is the difference between an observed value and the fitted value [rga]. This function is fitted to the beacons distance readings and then the same trilateration calculations are performed.

2.1.2 Wifi Based methods

Due to its wide deployment, WiFi is a great candidate for indoor positioning. One approach that Yang and Shao took was to use WiFi access points (APs) with multiple antennas as nearby anchors, while the positioning device could be any mobile platform with WiFi capability such as smartphone's or tablets. They proposed a method of trilateration by sending messages based on the time of arrival from the device to the AP, and then performing trilateration calculations once again. [YS15]

2.2 Mapping Formats

We developed our own map format because nothing existing quite fit our needs. One map format called GIS is very common for defining a traditional outdoor map with many layers, but this method was very large and bulky, and far more capable than we needed. It is also often limited to outdoor maps.
Another method that is commonly used is the CAD format, but again has the same issue as the GIS maps in that it is large and bulky.

Chapter 3

Methodology

We implemented two different algorithms to test the iBeacon's viability as an tool for indoor positioning. The first was a method called trilateration. The second is a method that utilizes the iBeacon's to improve the existing positioning methods that utilize the accelerometers and magnometers on the device.

3.1 Algorithm 1 - Trilateration

Trilateration is the process of determining absolute or relative locations of points by measuring distances using the geometry of circles. This is the same method that is utilized used to find the epicenter of earthquakes using the readings from various seismic stations. An example of how this works is seen in figure 1.

3.2 Algorithm 2 - Proximity Based

The proximity based algorithm uses the existing device sensors to perform a large portion of the calculations for positioning. The beacons are then Leveraged to make sure that the results of the calculations from the device are within the beacons range. This works by using a calculation based on the heading of the device, and the time delta, to calculate the movement vector. This movement vector is then added to the previous position to achieve the new position. The beacons serve the purpose of leashing the device, in that they are responsible for making sure the user is within the closest beacons range. The beacons have a 3 meter radius, and they must be positioned about 6m from any other beacons. If packed tightly; as is seen in figure 2, we can achieve good coverage of an area.

3.3 The System

The system consists of 2 layers, one is the phone application and the other is the external API server that implements the algorithm for calculating positions, and storing previous calculations as well as logging. This gives us the ability to improve readings and calculations without pushing an update to the device, and offload any intensive calculations to a more powerful server.

The android application implements a custom renderer that is responsible for displaying the custom map component, that pivots and moves with the device.

3.4 Accuracy Tests

A series of tests were performed on the iBeacon's to determine their accuracy, 3 different beacons were used in combination with a Galaxy S6 phone for the tests.

Chapter 4

Overview of Technology

4.1 The iBeacon

Apple released the iBeacon protocol in 2013. iBeacon compatible hardware are typically known as beacons. They are a BLE (Bluetooth Low Energy) device made for proximity sensing. Many vendors sell iBeacon compatible beacons.

The beacons we used for the experiment were made by Estimote, a third part beacon provider. The Beacons are about $30 each, and for our experiments we had 6 beacons available.

Estimote provides an SDK for Android that wraps the sensors inputs in an easy to use API.

4.2 Android

The main mobile platform we focused on was the Android platform. Android currently has 1.4 Billion devices worldwide and is the most accessible smart-phone platform. Therefor we choose to use the android for the project to maximize potential reach.

Android applications are written in Java and run in a JVM environment on the device. The app has access to all sensors available on the device.

Chapter 5

iBeacon Accuracy

We performed a series of tests on the iBeacon's to determine the accuracy of the devices. The tests were run to determine how accurate the iBeacon devices can be, and in what scenarios the devices are most accurate, and to make sure our specific devices act uniformly.

For each test, 10 readings were taken over a period of 10 seconds, and then results were calculated based on these 10 readings.

As is made clear by the results of the accuracy tests, seen in table 5.1, the Beacons are very inaccurate and readings can fluctuate significant amounts. At most readings fluctuated almost 1.5 meters.

Distance From Device, Beacon Position, Orientation	Mean reading(m)	Standard Deviation of readings(m)
0.3m - Right, Flat	0.039	0.01
1m - Right, Flat	2.436	0.50
2m - Right, Flat	1.418	0.11
0.5m - Right, Flat	0.460	0.12
2m - Right, Flat	1.925	0.31
3m - Right, Flat	2.872	0.79
3m - Right, Flat	3.812	1.14

Table 5.1: Results of testing accuracy of Estimote's iBeacon's

Chapter 6

Android Application

The system consists of 2 main layers, there is an android app, and a REST web server. The app is responsible for all user interaction and getting inputs, and delivering feedback. The REST server does the heavy lifting in calculating positioning and coordinating beacon data. A diagram of the relationship between the android applications and the HTTP server can be seen in Figure 7.1.

6.1 Estimote SDK

The Estimote SDK for Android is a library that allows interaction with Estimote beacons and stickers. The SDK system works on Android 4.3 or above and requires device with Bluetooth Low Energy (SDK's min Android SDK version is 9).

It allows for:

- beacon ranging (scans beacons and optionally filters them by their properties)
- beacon monitoring (monitors regions for those devices that have entered/exited a region)

6.2 Application Structure

Our android application consists of a few key pieces as are describe below.

6.2.1 Heading Sensor

Android itself does not provide a sensor that can provide information about the mobile clients heading at any given time. To implement

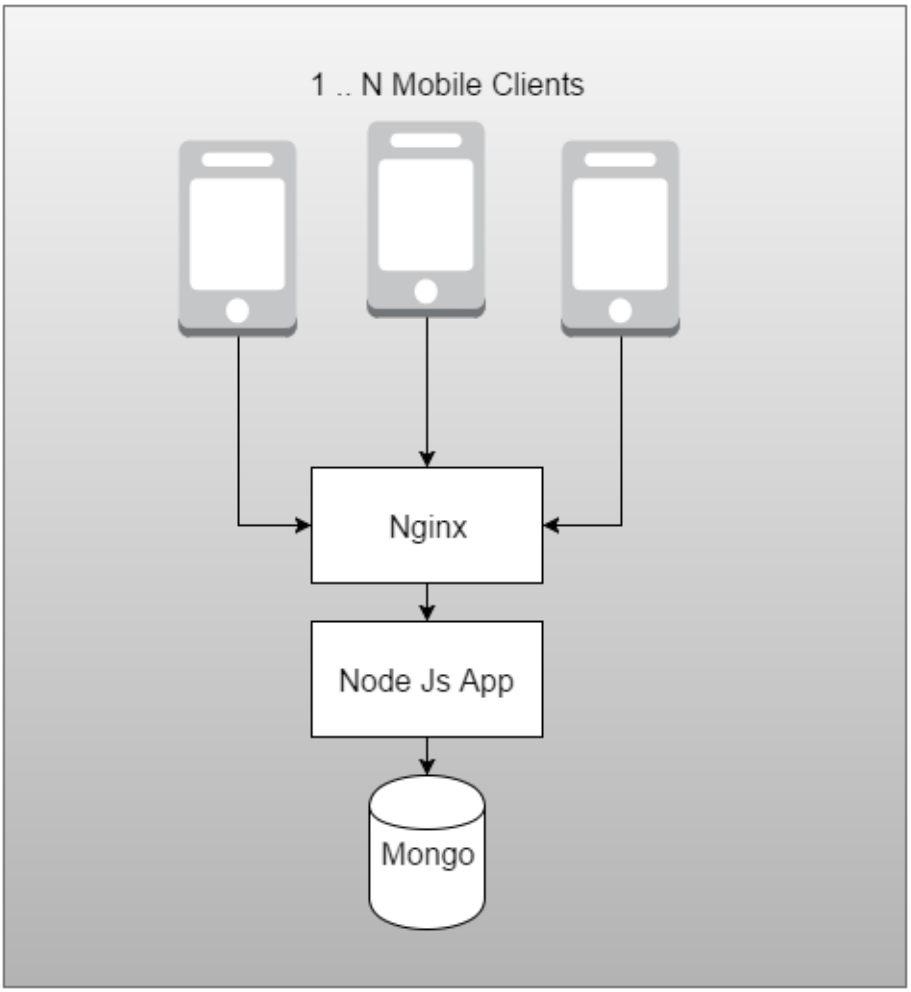

Figure 6.1: Architecture of system

this sensor we implemented a fusion sensor the magnometer sensors and accelerometers and to provide a custom android sensor class to the android application.

This custom sensor works by taking the orientation of the device in 3d space using the accelerometers to apply transformations on the readings from the magnometer to achieve an absolute heading reading [And16].

6.2.2 Map View

The view that displays the mapping information to the user implements a custom drawing system. Android has no built in way to provide custom mapping views. To overcome this a custom drawing and mapping view was implemented that takes the grid that the map is stored as, and draws it in blocks. The map then has the ability to pivot using matrix transformations on the existing map, with the bottom of the phone being the pivot point in the map.

6.2.3 Map

The Map class is where a lot of the work relating to creation and modification of the maps is done. The class handles the reading and interpreting of the Map Json. This class is responsible for taking the the JSON and turning it into a Map Object, that contains all the features like beacons, obstacles, and goals that the map also contains.

6.2.4 HTTP Layer

The HTTP Layer is the part responsible for communicating with the external REST API, this is where the maps originate as well as positioning information is gathered from.

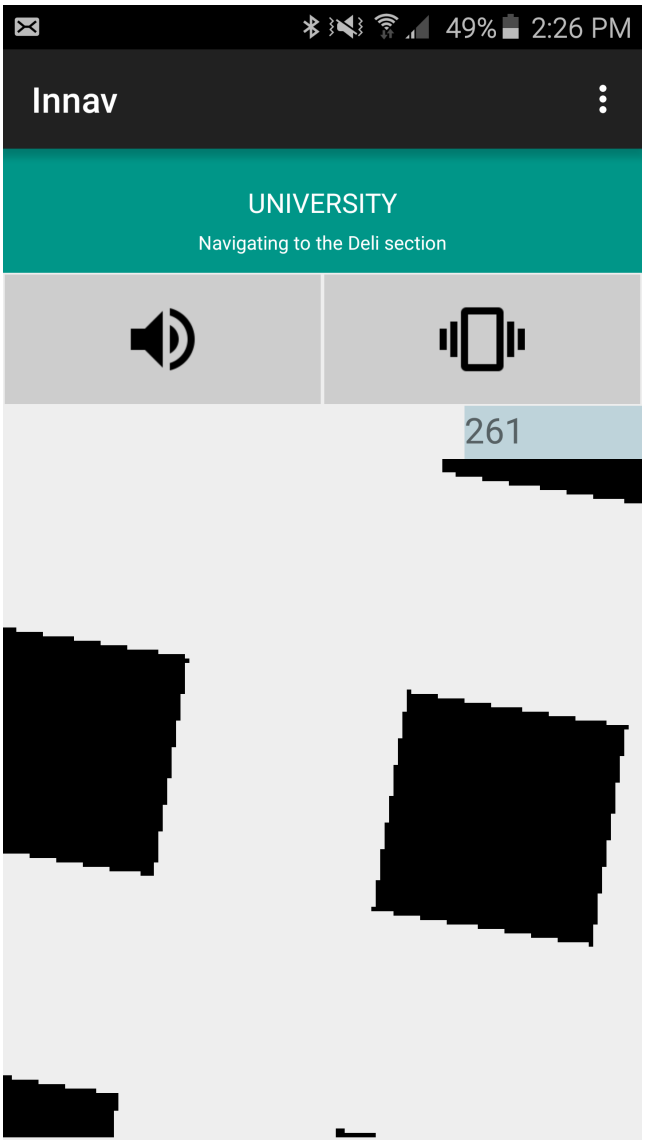

Figure 6.2: The custom map view drawing, rotated 22 degrees

Chapter 7

HTTP Server

7.1 Technologies and Librarys

NodeJS has one of the richest and most expansive package ecosystems, providing libraries for a range of tasks. Here are some of the key technologies and libraries that were used to build the backend HTTP server.

7.1.1 Node JS

Node js is an open-source, cross-platform runtime environment for developing server-side Web applications. The runtime environment interprets JavaScript using Google's V8 JavaScript engine. Node.js has an event-driven architecture capable of asynchronous I/O.[nod]

7.1.2 Express JS

Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications. It is the de facto standard server framework for node.js.

7.1.3 MongoDB

MongoDB is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. MongoDB is developed by MongoDB Inc. and is published as free and open-source software under a combination of the GNU Affero General Public License and the Apache License. As of July 2015, MongoDB is the fourth most

popular type of database management system, and the most popular for document stores [mon]

7.2 Request flow

The web server consists of a simple MVC architecture that is responsible for positioning calculations. The main API endpoint called calculate, consumes HTTP requests with JSON payloads. The mobile device makes request to the server that look like the following.

```
{
    "uuid": 1234,
    "beacons": [
        {
            "id": 1,
            "major": 1
        },
        {
            "id": 2,
            "major": 2
        }
    ],
    "heading": 0
}
```

The server consumes this payload and then in turn looks up any related data in the Mongo database. Using the information from previous requests from this device as well as the information from the current request, the positioning calculations are done. This implements the Algorithm 9.1 to perform the calculation based on the available data.

Chapter 8

Map Definition Format

For the project we required a simple format to store layout and object information about our maps. No existing solution (Some examples are in the prior work section) proved to be simple enough while still giving the flexibility to create custom features. The challenge was that our map needed to both encode layout information, but also information about the beacons and their locations and meta data.
The format that was settled on was a custom JSON schema. The schema contains all the layout information needed to display the map as well as the meta-data needed by the applications to perform calculations and give the user correct feedback. Utilizing this schema also allows for a lot of flexibility when it comes to adding additional meta-data to the maps, since the schema is not enforced and is free flowing.

```
{
  "features": [
    {
      "height": 30,
      "type": "isle",
      "width": 8,
      "x": 14,
      "y": 10
    },
    {
      "height": 30,
      "type": "isle",
      "width": 8,
      "x": 30,
      "y": 10
    },
  ]
  "beacons": [
```

```
    {
      "x": 1,
      "y": 1,
      "id": "1"
    },
    {
      "x": 19,
      "y": 38,
      "id": "2"
    },
    {
      "x": 38,
      "y": 1,
      "id": "3"
    }
  ],
  "intersections": [
    {
      "canBeGoal": true,
      "x": 19,
      "y": 19,
      "name": "Middle"
    }
  ],
  "name": "Box with beacons and sections",
  "height": 40,
  "width": 40
}
```

As can be seen the JSON above, some general map metadata like height, width and name are at the highest level, and then there are three main pieces, the beacons, the features and the intersections.
The beacons array contains data about where the beacons are located and their ID's
The features array is data about things within the space, there are 4 supported feature types at current. The supported feature types are as follows, wall, isle, obstacle, void. Each feature type is drawn differently and has different characteristics when it comes to the mapping compo-

nent and positioning.
The intersections array is used to define where users are allowed to turn(For navigation and route calculation). Because of the need to create the straightest paths possible, we have the user define where intersections are. This makes sense when thought of in a grocery store type scenario, where the intersections and pathways are clearly defined.

Chapter 9

Algorithms Implemented

There were two different approaches taken when trying to come up with a method for positioning. The first method was a well known algorithm for positioning in 2d space called trilateration. The second method was a method that I will be referring to as the proximity based positioning method that used vectors calculated from the device heading, and elapsed time, to calculate position deltas while using the beacons as a leashing mechanism.

9.1 Trilateration

In geometry, trilateration is the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles, spheres or triangles. An illustration of this can be seen in figure 9.1.

In addition to its interest as a geometric problem, trilateration does have practical applications in surveying and navigation, including global positioning systems (GPS). In contrast to triangulation, it does not involve the measurement of angles.

[tri16]

9.2 Proximity based positioning

The proximity based algorithm uses the existing device sensors to perform a large portion of the calculations for positioning. The beacons are then Leveraged to make sure that the results of the calculations from the device are within the beacons range. This works by using a calculation based on the heading of the device, and the time delta, to calculate the movement vector. This movement vector is then added to the previous position to achieve the new position. The beacons serve

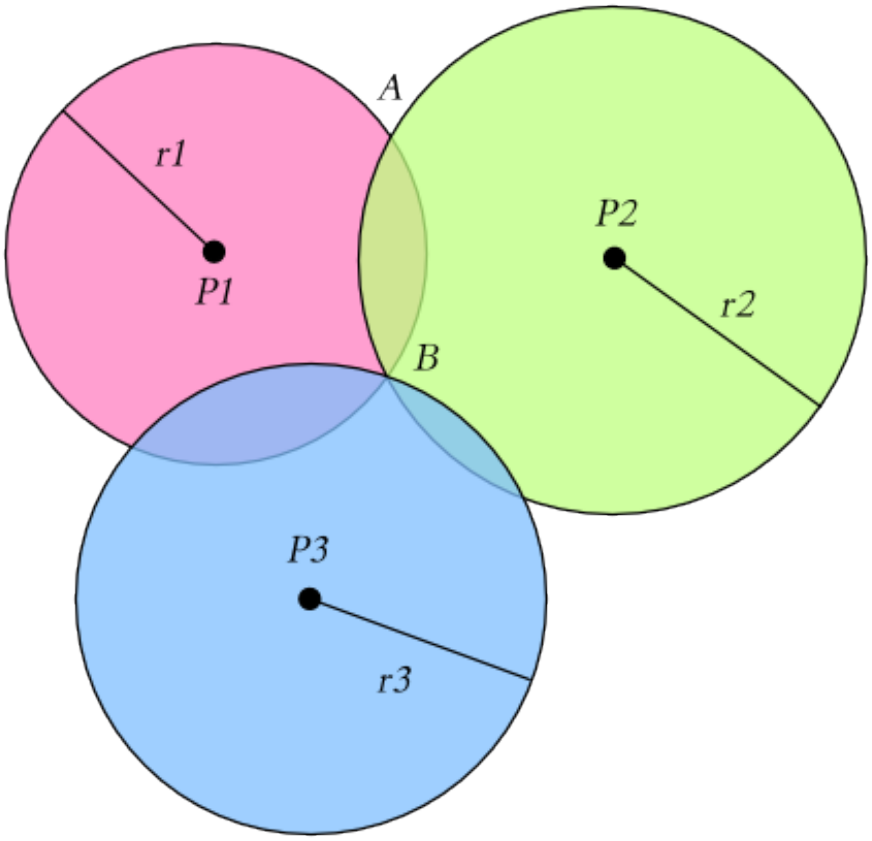

Figure 9.1: Illustration of trilataration

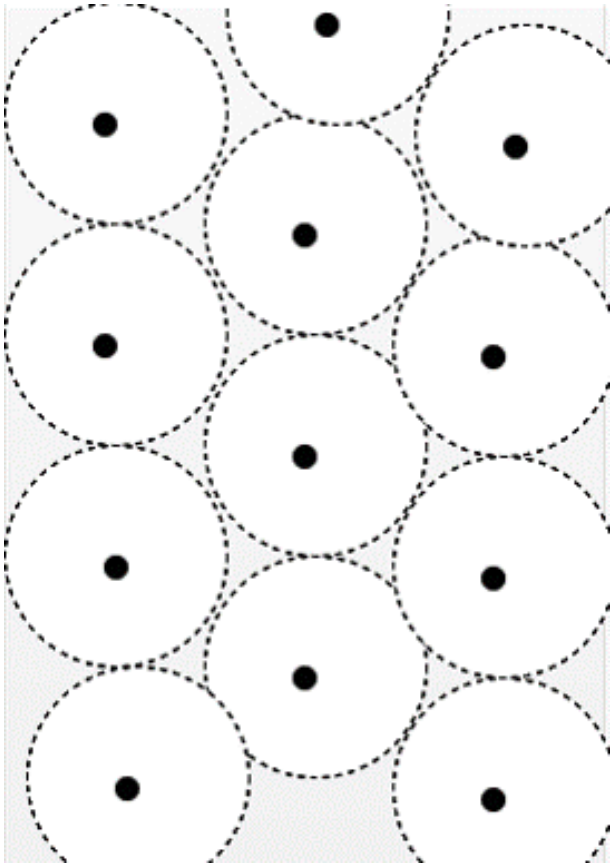

Figure 9.2: Layout of iBeacons for proximity based positioning

the purpose of "Leashing" the device, in that they are responsible for making sure the user is within the closest beacons range. The beacons have a 3 meter radius, and they must be positioned about 6m from any other beacons. If packed tightly; as is seen in figure 9.2, we can achieve good coverage of an area using this method.

9.3 Results

We determined that algorithm 1; the implementation of trilateration proved to be very inaccurate. Readings frequently fluctuated 5+ meters away from the true location of the device. The reason for this is

that the trilateration algorithm is extremely vulnerable to input inaccuracies, which is very much the situation with the iBeacon's.
Algorithm 2 was a much more promising solution. We were able to test the implementation of this algorithm in the real application and the results were very close to being good enough to navigate in a situation such as a grocery store or conference

Chapter 10

Limitations

Our system and algorithms were designed with a certain scope in mind. Various parts of the application are constrained to specific environments.

10.1 Navigation

The navigational algorthims are designed to work inside of a grocery store, and to navigate in straight lines. This is done by defining valid intersections inside the mapping format. While this works well for the scope we had defined, it introduces the limitation of not being able to function as well in situations that do not lend themselves well to grid based movements, such as an open gym or similar open area.

10.2 Mapping format

The mapping format is implementated as a simple json structure as was seen above, this allows us to add and remove features as is needed. While we can add features to the mapping format, the system only supports rendering of the small set of items we currently use. This includes the "Features", "Intersections" and "iBeacon" object types.

10.3 System Requires Wifi

While this limitation is an advantage in some aspects in that it broadens the scope of computations and data we can perform location calculations on, the downside is that it introduces the limitation that the user be connected to wifi that has a low latency to the server that does the calculations.

Chapter 11

Lessons Learned

11.1 iBeacons are inaccurate

As was shown in the accuracy experimentation's for the accuracy of the iBeacon devices, it was determined that the devices themselves are not very accurate. The intended use of the devices is proximity based interaction, which they work very well for. But when used in an application that requires a much higher degree of accuracy, they have proven to not be the best tool for the job. While our application was able to achieve mild success, iBeacon's were not meant for this application and struggle to achieve the required accuracy for indoor navigation. The same results have been found in other independent studies [RF] .

11.2 Trilateration algorithms are not fit for inaccuracy prone applications

When we set out to use trilateration as a method of positioning in 2d space, it seemed like the obvious method. It is commonly used to position earthquakes based on readings at seismic activity stations, but these applications are very accurate. When applied to inaccurate readings often times the algorithm was unable to produce a result, or the result given was so far from the true position it was unusable.

11.3 Non Technical Challenges

We had some additional challenges that we can learn from. The first being that being dependent on hardware is okay as long as the hardware is readily accessibly or time is not a constraint. In our case, we had an issue getting the iBeacon hardware needed for experiments, and when we finally received the hardware there was not sufficient time to

properly explore all methods of using the iBeacon's.

The other challenge was that because of the delays in getting our beacon hardware, we focuses on other parts of the system. When we received the hardware we needed the focus was shifted to the positioning part of the application. While a lot of code was developed for the extraneous parts of the application, much of it is no longer relevant. Going forward I would recommend to focus on one thing and do that well, and then spend extra time on other parts when the primary focus is at a point where it can be considered "done".

Chapter 12

Future Work

12.1 Improved UI

When work began on the UI of the application we set out to create a UI that would be accessible for sight impaired people. This lead to a UI that does not lend itself as well to usage by sighted people. And then part way through the project, the focus of the application shifted to researching the viability of iBeacons as a navigational tool. This lead to a UI that is usable but not pleasant to use.

12.2 Sight Impaired Navigation

The project was originally to provide a way for sight impaired people to navigate indoors, but as we progressed it became clear that this technology could be applied to much broader applications. This lead to the scope of the project being broadened and not restrict the applications of the software to sight impaired people. With some small adaptations the same technology could be used and reapplied for users with a sight impaired, in an app with that as the focus.

12.3 Better Map Rendering

The map renderer was implemented on top of androids existing canvas drawing engine. This is because it was the easiest way to get graphics onto the screen and to respond to user input. The downside of this method is that the performance is badly optimized and many calculations are needed every time the screen is refreshed. If this was done at a lower level using a custom low level map renderer implemented on top of the low level graphics API's available to android devices, the app could achieve much better performance characteristics and use less

resources.

Chapter 13

Conclusions

As was shown by the accuracy testing of the beacons, the beacons themselves are very inaccurate and readings fluctuate wildly. Because of this, iBeacon's do not lend themselves well to trilateration, which is highly vulnerable to inaccuracies. iBeacon technology is best used as a tool that can be used in combination with other sensor data, to achieve rough positioning information. This can be seen in the second algorithm that was developed that uses heading data as well as beacon data to achieve more accurate readings. Further combining more inputs could increase the accuracy even further than what we have achieved. iBeaconâĂŹs are not a good technology for absolute indoor positioning, but when combined with other sensor data they can be a powerful tool in increasing accuracy. Neither of our implementations for positioning is cut out for real world deployments in its current state, but the capabilities of iBeacon's still have lots of room for exploration.

Bibliography

[And16] Android. Position sensors [online]. 2016 [cited April 26, 2016]. → pages 12

[mon] Mongodb [online]. → pages 15

[Ng15] Tsz Ming Ng. From âĂIJwhere i amâĂİ to âĂIJhere i amâĂİ: Accuracy study on location-based services with ibeacon technology. 22–1:23–31, 2015. → pages 3

[nod] Node js [online]. → pages 14

[RF] UK R. Faragher, University of Cambridge. An analysis of the accuracy of bluetooth low energy for indoor positioning applications [online]. → pages 24

[rga] Determining indoor position using ibeacon [online]. → pages 4

[tri16] Trilateration [online]. 2016. → pages 19

[YS15] Chouchang Yang and Huai-Rong Shao. Wifi-based indoor positioning. pages 150–157, 2015. → pages 4