Indoor Location Estimation

Bachelor End Project

TU Delft Coach
M. A. Zúñiga Zamalloa

TU Delft Bachelor Project Coordinators
Dr. ir. F.F.J. Hermans
Dr. ir. M.A. Larson
Ir. O.W. Visser

Company Coach
W. Van

Authors
S. Boor
M. den Toom

Date
July 12, 2016

Preface

This report has been written as part of the Bachelor study Computer Science at Delft University of Technology. It contains all relevant information to understand the project on indoor localization that we carried out in the second quarter of 2016. The goals of this project have been set based on conversations at bunq.

To develop an application that is able to do indoor localization without any experience in this field is a great challenge. Even more so, if the timespan is nine weeks. It was quite a ride, but we look back with satisfaction. We learned numerous things about localization and how software development at a company is done.

The project consisted of three parts: 1. doing research in indoor localization 2. implementing multiple indoor localization methods and test their accuracy and 3. implementing a system that detects which user is closest to an object. This object can for example be a gate at a festival. In this report, we want to inform the reader about all three parts. Furthermore, it contains a number of do's and don'ts with regards to indoor localization. With this we can, hopefully, prevent the reader from making the same mistakes as we made.

We would like to thank everybody at bunq for their friendliness. We especially liked the conversations during lunch. Our special thanks goes to Wessel, Marco and Ali who learned us numerous things about localization, coding, Android and how to run a bank.

Sven Boor
Matthijs den Toom
July 12, 2016

Summary

bunq is an innovative banking company that competes with other banks by providing the best services to their costumers. The functionality of a number of these services could be improved if the bunq applications would be able to locate the user within a building. An example of such functionality is NearPay. With NearPay a bunq user can see all other bunq users in the vicinity and exchange money with them. Indoor localization would make it easier to find the bunq users that are in the vicinity at, for example, a concert. Therefore, the goal of this project was to develop a general indoor localization application that can be used in the future by bunq to enhance their costumers' banking experience.

After reviewing the technologies that are currently used for indoor localization, Bluetooth Low Energy (BLE) beacons were chosen as the technology that is used in this project. The low costs and the ease to deploy the beacons are the main reasons for this choice.

The first approach to locate the user was by means of a method called trilateration. Trilateration uses the signal strength of each of the beacons in range of the user's phone to estimate the user's location. Experiments pointed out that, with an accuracy of 4.5 meters on average, this method was not going to be accurate enough for bunq's purposes.

Therefore, a second method, called a particle filter, was implemented. A particle filter estimates the users location based on, so called, particles that mimic the user's movements in a map of the building. Initially, the particles are everywhere. As the user moves, the particles follow the movements of the user. Particles that move through a wall are eliminated, reducing the number of locations with particles. The estimated location of the user is there where the density of particles is highest. The BLE beacons are used to make the particles converge to one location faster.

Besides an application that is able to localize the user indoors, with the methods mentioned above, a system that is able to detect the user that is closest to an object has been implemented. This system can be used to, for example, allow users to enter a concert, where the ticket check is automated. To do this, a BLE beacon has been added to each object. A web service determines by which object the user is standing based on the beacons that are in range of the user's phone. This speeds up the ticket check process, which currently is often done manually either by the user or by a security guard.

In short, this project consisted of two parts. The results of the first part are two indoor localization methods that can be used by bunq to improve the experience of bunq's users. The result of the second part of this project is a system that is able to identify the user that is closest to an object. With this system it is possible to, for example, allow only users with a valid ticket in a concert hall.

Contents

1. Introduction

As we, humans, are often indoors, being able to get a smartphone's location indoors introduces numerous new possibilities. It can be used to guide you in the supermarket to the products you want. In tours, it can give information via audio at the right moments. It can be used in augmented reality to give the user the illusion that he is walking in another room.

Outdoor localization can be done within a few meters accuracy using Global Positioning System (GPS) [50]. However, as GPS technology requires a direct line of sight with at least four satellites, it is not accurate indoors [54]. Therefore, other solutions are needed for indoor localization.

bunq, an IT company with a banking license [6], invited us to develop a smartphone application that is able to locate its users indoors. As a startup company, bunq strives to shake the banking world. With an innovative team, bunq tries to make banking easy and social. In the context of easy and social banking indoor localization comes in handy. For example, a bunq user would be able to find a friend's bunq account more easily in a crowded room using NearPay. Therefore, bunq requires a general indoor location estimation application that can be used in several situations. Furthermore, as bunq is a free bank for non-commercial costumers, a low cost solution is required.

In short, the project that is described in this report consisted of 4 parts:

1. Analyzing bunq's problem and creation of the product requirements.

2. Doing research into the technologies, such as Bluetooth Low Energy (BLE) beacons and WiFi access points, that are used for indoor localization. We completed this step with a founded decision to use BLE beacons for indoor localization. In addition to that, we found out during this step that trilateration and a particle filter are suitable methods for indoor localization using BLE beacons.

3. Implementing a prototype application that is able to perform indoor localization. In this application we have implemented two methods that are able to locate the user's phone indoors: trilateration and a particle filter. We use the BLE beacons as a source of information for localization.

4. Designing and implementing a system that is able to detect which person is closest to an object. This system can be used to, for example, allow users to enter a concert, where the ticket check is automated. We position a BLE beacon at each gate and open the gate only if the phone that has been detected closest to the gate based on the BLE measurements has a valid ticket.

In the following we take you on a journey to do indoor localization with just the user's smartphone and BLE beacons.

In Section 2 we analyze bunq's problem and define the product requirements.

In Section 3 we discuss a number of technologies that are currently being used for indoor localization and motivate why we used BLE beacons in this project.

In Section 4 we explain the most common methods that are used for indoor localization using BLE beacons and we motivate the methods we have chosen to implement.

In Section 5 we discuss a web service that we designed and implemented that is able to let the user create a map of a location. The created map can be used as input for the particle filter or to display the user's location.

In Section 6 we discuss how the first method that we used for indoor localization, trilateration, how it is implemented in the application. We also discuss the localization accuracy of trilateration based on our own experiments.

In Section 7 we discuss the second method that we used for indoor localization, a particle filter. We explain the concepts of a particle filter and how we have applied those concepts in the application. Furthermore, we discuss the extensions that we have implemented to make the particle filter more accurate and robust. Last, we show how the particle filter works in practice.

In Section 8 we introduce a system that is able to detect which user is closest to an object. We discuss its implementation and the obtained results.

In Section 9 we explain the software development process and how high code quality was ensured.

We finalize this report with future work and a conclusion, in Sections 10 and 11, respectively.

2. Problem Analysis

In this section we describe the problem bunq wants us to find a solution for. The original project description as specified by bunq can be found in Appendix A. For a good solution to bunq's problem, it is important to analyze the problem in detail and define goals based on this problem. In paragraph 2.1 we explain bunq's problem in detail. Then, in paragraph 2.2 we describe the general requirements of this project. Last, in Paragraph 2.3 we explain why indoor localization using BLE beacons, and therefore this project, is challenging.

2.1 Problem Definition

The usability of the bunq application can greatly improve if the application would be able to accurately perform indoor localization. It can be used to improve NearPay, one of the app's important functionalities. NearPay allows you to find people that are near you so that you can exchange money with them. With accurate indoor localization you will be able to obtain your friend's bunq account even when you are at a crowded indoor party. You would be able to pay the beer he got for you immediately.

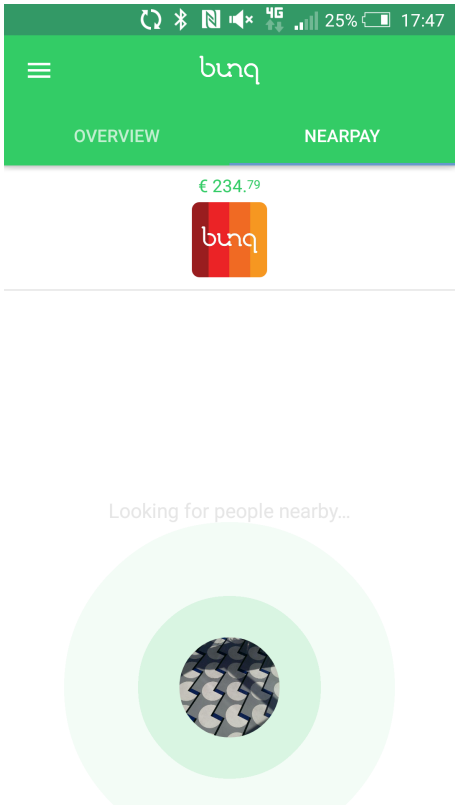


Figure 2.1: NearPay functionality in android app.

Currently, the location of the user is determined using the standard localization services of the phone's operating system. However, the standard localization services do not work well indoors as these use GPS and mobile networks to determine a phone's location. In Paragraph 2.3 we discuss why GPS and mobile networks are not able to estimate the user's location indoors.

Therefore, bunq needs a prototype app that estimates the location of its users indoors. The best indoor localization methods have to be reviewed and the most suitable method implemented into a prototype app. The problem can be formulated as follows:

The bunq app cannot localize the user with room-level accuracy indoors.

As the localization has to be performed on the user's phone, it is important to ensure that the used algorithms do not consume a lot of processor or battery power. As the method is going to be used by costumers of bunq, a hardware extension to the phone is also not allowed. Furthermore, the method has to be robust and generally applicable because it will be used at different locations and with different phones. bunq opted the use of BLE beacons for localization, because of their low costs and long battery life. Therefore, we are going to evaluate and compare several technologies, one of which is BLE beacons, for indoor localization.

2.2 General Requirements

At the beginning of the project the goal was to develop an application that is able to estimate the user's location with meter level accuracy. However, as the project progressed, we found out that meter level accuracy was not feasible with just BLE beacons. Therefore, we changed our goals to be able to estimate the user's location with room level accuracy.

Besides being able to locate the user's smartphone, bunq wanted us to design a system that is able to detect which bunq user is closest to an object. This can for example be used by bunq to speed up the ticket checks at festivals. It can be integrated in a system that only opens a gate at a festival if the user that is waiting for it has a valid ticket on his phone. In Section 8 we explain how this system works.

The methods used in both applications have to be applicable to recent iOS and Android smartphones so that localization functionality can easily be integrated in bunq's app. In order to ensure that the methods can be used in smartphones the complexity of the methods has to be low.

As the app is going to be used by bunq's developers in the future, the code should be easy to understand and maintain. The coding style has to be in line with bunq's coding style and of high quality according to SIG.

2.3 Indoor Localization Challenge

Indoor localization with BLE beacons is challenging because the often dense indoor environments influence the sensor measurements. Indoor obstacles, such as human bodies, walls and chairs, influence the sensor measurements in ways that are unpredictable. As localization with BLE beacons is RF-based, radio signals from the beacon are used to estimates the user's location. Two kinds of interference make it difficult to estimate a location based on the wave-propagation of RF-based signals. Shadow fading or slow fading means that obstacles reduce the wave-propagation, which results in a weaker signal [2]. Multi path fading or fast fading means that radio signals interfere with each other, resulting in changes in the signal strength [37].

These two kinds of interference are extremely hard to model. The constantly changing environment, for example if someone is walking through the room, makes the radio signal vary over time because of fading. Therefore, methods are required that deal with these kinds of interference, or at least work on the assumption that these kinds of interference will occur. In this report we describe two methods that are able to (to some extend) deal with these kind of interference: trilateration and a particle filter.

3. Indoor Localization Methods

In this section we give an overview of the most important technologies that are used for indoor localization and discuss their advantages and disadvantages when the technology would be used in this project. As bunq requires a smartphone app that can perform localization, we focus on techniques that can be used in combination with a smartphone. We evaluate Radio Frequency (RF)-based technologies as BLE and WiFi, Visual Light Communication (VLC), sound and Assisted-GPS (A-GPS). At the end of this section we motivate why BLE beacons were most suitable to be used in this project.

3.1 RF based localization Technologies

RF-based technologies are most used for indoor localization. These technologies use radio signals to determine the user's location. The signal strength is expressed in the Received Signal Strength Indicator (RSSI) value, which can be measured by the receiver of the signal. As discussed in Paragraph 2.3 the signals from all RF-based technologies are difficult to predict and model. Therefore, it is challenging to achieve a high accuracy with RF-based technologies. We discuss three technologies that are RF-based: BLE, WiFi and Near Field Communication (NFC).

3.1.1 Bluetooth Low Energy

BLE beacons have two features that make them attractive to be used for indoor localization: their low costs and their long lifespan.

Beacons can be obtained for under $50 [19]. These costs will drop as more manufactures are going to produce beacons. With a battery life of over 2 years, beacons do not require a lot of maintenance. Mobility is another advantage of BLE beacons. Even at sites without any facilities, battery powered beacons can be installed to enable indoor localization. A disadvantage of BLE beacons for indoor localization is that they send low power signals which increases the influence of the surroundings on the signals, resulting in a less accurate location estimate.

BLE is intended to provide the same communication range as Classic Bluetooth, while reducing the power consumption. BLE is supported by android devices with API-level 18 or higher, in other words from Android 4.3 'Jelly Bean' [3]. Examples of BLE beacons are Estimote beacons. Research has been done into using BLE for indoor localization [62] [60] [63].

3.1.2 WiFi

As lots of places have WiFi nowadays using it for localization can become a general method. If the WiFi access points that are installed in almost every building are used this would reduce the investment costs in additional hardware. However, the number of WiFi access points per area is usually small and therefore has to be expanded in order to do high accuracy location estimation. Furthermore, an infrastructure (of power cables) is required when using WiFi access points.

The technology, specified in the international standard IEEE 802.11 [9], is meant to be used for wireless communication between electronic devices. This makes WiFi access points usually

more expensive than BLE beacons. An advantage of using WiFi access points compared to BLE beacons is that they can send stronger signals, which can improve the accuracy of the location estimation. Several indoor localization systems, as Horus [68] and Radar [5], use WiFi access points for localization.

3.1.3 NFC

The biggest difference between NFC and the previous technologies is that NFC has a much smaller range [67]. This makes the technology suitable for ranging, in other words detecting when a device is close to a NFC tag. An advantage of NFC is that NFC tags do not need a power source..

NFC might be a good solution in combination with Dead Reckoning methods [40]. Dead Reckoning methods track the user from a begin point, based on the sensors in the phone. When the smartphone sees the NFC tag, its location accurately known. Sensors like the accelerometer, gyroscope and magnetometer can then take over to keep track of the user's position. It is, however, not convenient to calibrate your smartphone by coming close to a fixed point. Therefore, NFC is not an good solution for our project.

3.2 Visible Light Communication

A relatively new technology that can be used for indoor localization is VLC. An advantage of using VLC for indoor localization is the sub-meter accuracy with which the location can be estimated. Applications by Phillips [58] and by Bytelight [1] are already able to estimate its users' location. The basic principle of these methods is that all lights in a room send out a different signal. These signals are used to identify the lamps and the location of these lamps are used to estimate the position of the phone.

A promising technology that uses VLC is Luxapose [42]. This method uses the rolling shutter effect of the front facing camera of the user's phone to determine the location of the phone relative to the lights and is publicly available [57]. The rolling shutter effect means that the camera takes a picture by scanning across the camera sensor, instead of taking a shot of the whole sensor at once. Recent smartphones are capable of performing the algorithms used, as has been proven by a student at the TU Delft in his master thesis [65]. Other approaches need additional hardware, that is attached to the phone [47].

Limitation of VLC is that you must be able to set the shutter speed of the front camera of the smartphone. As this is not possible with all smartphones, VLC is not a suitable technology to be used during this project.

3.3 Ultrasound

As has been shown by the Active Bats system [11], ultrasound can also be used to locate a user. The time it takes for the sound to reach the user is used to estimate the user's location.

An advantage of using ultrasonic sound is the high accuracy that can be obtained when there is a direct line of sight with all speakers. A disadvantage of using ultrasonic sound to estimate the user's location is the method's low accuracy when the sound waves are interrupted. Furthermore, the system requires high investment costs, as all speakers have to be installed carefully [11].

Technology	Method	Accuracy (m)	Location Size	Name/Paper
WiFi	Bayesian Filter	<1	Building	Horus [68]
WiFi	Particle Filter	<7	Building	EZ [8]
WiFi	Fingerprinting	<5 (50%)	Building	Radar [5]
WiFi	Fingerprinting	1.5	Building	[49]
WiFi	Triangulation	<0.5	16 x 10	SpotFi [41]
Bluetooth	Trilateraton + Fingerprinting	<5	8 x 8	[63]
Bluetooth	Trilateration	<0.5	6 x 8	[60]
Bluetooth	Trilateration	<0.5	Building	[62]
Light	Triangulation	0.1	2 x 2	LuxaPose [42]
Sound	Trilateration	<0.5	4 x 6	Lok8 [21]
Phone Sensors	-	<3	Building	[46]

Table 3.1: A selection of indoor localization systems, their achieved accuracy and used method.

3.4 Assisted GPS

To achieve the same accuracy indoors as GPS has outdoors, A-GPS has been introduced. GPS does not work well indoors, because the signals are not strong enough to penetrate walls [69]. In addition to that, locking to the satellites may take minutes. To make the satellite locking faster, A-GPS has been introduced [14]. A-GPS combines the data from satellites and mobile networks to create a location estimation that is faster and more accurate.

As mobile networks signals are better in penetrating building walls, A-GPS might improve location determination indoors. However, the accuracy of A-GPS indoors is about 50 meters [14], which is not accurate enough for our purpose.

Indoor localization based on GPS can be improved when using, so called, pseudolites (pseudo-satellites) [23]. These devices generate a signal that is the same as the signal from a satellite. Pseudolites are, in contrast to satellites, positioned indoors, so that the signals are not blocked. An advantage of this technology, is that GPS receivers can use those devices with minimal modification. However, not many implementations of this technique are known and the available implementations cannot be implemented on a phone [16] [17].

3.5 Conclusion on Indoor Location Estimation Technologies

Table 3.1 shows the accuracies of a few localization technologies that have been discussed. Contrary to outdoor position estimation no general technology is used that works well in all situations. There is always a trade-off between accuracy and investment costs. The most accurate technique at this moment, if used well, is VLC. However, as the use of this technology in localization is still in research phase, we decided to use another technology for this project.

In Table 3.1 we see that there are methods using BLE, WiFi and sound that have an accuracy of within 1 meter. Of these three, BLE beacons also have low investment and maintenance costs. Therefore, we have chosen to use BLE beacons for indoor localization. During project we have kept an eye out for improvements and other technologies that we could use to improve the location estimate.

4. Indoor Location estimation using BLE beacons

To find the best methods that can be used in our project, the indoor localization methods that can be used in combination with BLE beacons have been reviewed. In this section we summarize our findings and motivate our choice for BLE beacons in combination with trilateration and a particle filter.

In literature, indoor localization methods can be divided into two groups: deterministic methods and probabilistic methods. We use this separation to highlight the known localization methods that can be used in combination with BLE beacons.

4.1 Deterministic methods

Deterministic techniques do not use probabilistics to estimate a location. Three deterministic indoor localization methods that can make use of BLE beacons are well known: fingerprinting [33] [48], triangulation [31] and trilateration [25] [51]. We have also seen hybrids that combine multiple methods. For example, fingerprinting and trilateration are combined in [63].

4.1.1 Fingerprinting

A well known method for indoor localization is fingerprinting. The fingerprinting method consists of two phases: a training (offline) phase and a testing (online) phase [33]. In the training phase, a fingerprint of the site is taken. This means that the location is divided into blocks and in each block the RSSI values for all the installed beacons are measured. In the testing phase, the user measures the RSSI values of the beacons in range. An algorithm matches the RSSI readings from the smartphone with the fingerprint of the site. Different algorithms can be used to match the user to a block. A well-known algorithm that is used for fingerprinting is the k-nearest neighbors algorithm [49].

As the complete site needs to be fingerprinted up front and the fingerprint determines the accuracy of the localization, this is not an optimal solution for this project. bunq does not have the manpower to scan each site they want to deploy indoor localization.

4.1.2 Triangulation

A method that is also used for outdoor localization is triangulation. For example, GPS uses triangulation to calculate the user's location [56].

Several method fall in the category of triangulation [31]. GPS calculates the location of a user based on the Angle of Arrival (AoA) of the signal. Other properties, as Time Difference of Arrival (TDOA) or Time of Flight (ToF), can also be used.

Because triangulation is widely used a lot of research has been conducted into triangulation. When using capable hardware, triangulation can accurately determine the user's location. A disadvantage of triangulation is that the current smartphones are not advanced enough to perform either of the three methods. The timing of a smartphone is not accurate enough to use the TDOA or ToF method and the sensors are not accurate enough to use the AoA method.

Another disadvantage of triangulation is that a direct line of sight with the signal source is required. If there is no direct line of sight, the received signal is weaker and inaccuracies are introduced. Therefore, this method can currently not be used to estimate the location of a smartphone without extra hardware.

4.1.3 Trilateration

In trilateration, the distance from the source to the receiver is used to estimate the location of the user. The trilateration method can provide an accuracy within 2 meters [25]. The RSSI values of each beacon in range are transformed into a distance. Each distance from a beacon to the smartphone can be seen as the radius of a circle. The position where the circles intersect is the estimated location of the smartphone. Of course, the circles do not always intersect. In that case, the intersection point of the circles needs to be estimated. Distance estimates can also be used in other ways. For example, Microsoft's RADAR uses the distance estimates and the K-nearest neighbor algorithm to estimate the room the user is in [5].

An advantage of trilateration is the setup cost. The only setup that is required is positioning the beacons and determining the best function to transform the RSSI values to distances. At the same time, the distance transformation function is a disadvantage of the system. A lot of factors are involved in determining the distance, based on an RSSI value. Furthermore, the RSSI value is not fixed at a distance. Inaccurate distance estimates make the whole method inaccurate.

Iterative trilateration is an extension to trilateration [43]. In this method an initial location estimate is iteratively updated. The location is updated so that the difference in measured distances and estimated distances is minimized. This makes the algorithm return the optimal solution based on the measurements.

4.2 Probabilistic Techniques

Probabilistic techniques use probabilities to estimate the location of the phone. We discuss the Bayesian filter and the particle filter, and highlight their advantages and disadvantages.

4.2.1 Bayesian Filter

Bayesian filtering is a technique that also uses training data to estimate a user's position [68] [29]. The idea of this method is that based on (initial) believes and on the measurements, the location of the user is estimated. The method starts with dividing the map into cells. Measurements of the RSSI values of all beacons are collected in each cell for 1-2 minutes. The measurements are used to create a mass distribution function, which gives the probability that you are in a specific room given an RSSI value for each of the beacons. If someone starts sensing with his smartphone the distribution function is used to estimate the cell the user is in. This goes in a process of sensing and moving, where in each iteration for each cell the probability that the user is in that cell is updated. The cell with the highest probability is the estimated location of the user. An advantage of this method is its accuracy after training and its simplicity to implement. Disadvantages are the training that is needed and that the accuracy is limited to the size of the cells. Therefore, we are not going to use this method to estimate the user's location.

4.2.2 Particle Filter

Particle filters do not require any training data [8]. The only thing needed is a map of the building and a set of sensors to obtain the user's movements. This training free method therefore has high potential for our project. The algorithm starts by randomly distributing a high amount of particles uniformly on the map. Each particle has a position (x, y). In addition to that, every particle has a weight w. Then a process of sensing, redeploying and moving is used to estimate the user's location. At the locations where the sensed measurements fit, w is increased. Then the particles are redeployed to the area's where particles have a high weight. If the user moves, the particles scatter again. What also happens during movement is that certain particles can be ruled out as the user's location. This is because the particles cannot travel trough walls and have to respect the map of the building. With this, the number possible locations of the user is reduced. This process is repeated until most particles are clustered at the user's location. A disadvantage of a particle filter is the relatively high computational power needed, as all the particles have to be moved each iteration. Because of the training-less localization, we used the particle filter as replacement for iterative trilateration when we found out that iterative trilateration did not perform as expected.

4.3 BLE Protocols

In order to retrieve information from the BLE beacon, a communication protocol has to be used. Useful information that can be obtained using a BLE communication protocol is, for example, the power with which the beacon transmits. Examples of BLE communication protocols are nearable [20] (Estimote), iBeacon [4] (Apple), Eddystone [26] (Google) and AltBeacon [53]. We decided to use Eddystone for this project because it is a widely adopted standard and because it is open source. This prevents a vendor lock-in and makes it possible to switch beacon supplier in the future.

4.4 Localization Method of Choice

All methods to do indoor localization with BLE beacons have their advantages and disadvantages. Triangulation has as disadvantage that the current smartphones do not have sophisticated enough hardware. Fingerprinting has as disadvantage that the investments costs are huge. Trilateration is a relatively simple method that requires the least installation effort and provides 2 meter accuracy. Therefore, trilateration is best option for location estimation in our project.

initially, we wanted to implement a method that combined trilateration with a particle filter. This method is described in [60] and [43]. That method gave an accuracy of within a meter in a 6 x 8 playing field for robots. We were interested how well it would work with a smartphone.

However, as will be discussed in Paragraph 6.4 the results of the trilateration algorithm were so inaccurate, that a change of method was needed. Therefore, halfway trough the project we changed from a partially deterministic method, trilateration in combination with a particle filter, to a probabilistic method, the particle filter. We used BLE beacons to make the particle filter more robust and more accurate.

5. Mapping the field

In this section we discuss an application that we designed to create maps which can be used for indoor localization. The map is needed to project the estimated location on. Furthermore, it is be used by a particle filter. Contrary to Estimote's map creating application, where the user is to walk besides each wall [18], we let the user enter a picture of the map in a web application. The picture of the map is displayed in the background and the user can indicate the walls that he wants to be included in the digital map on the picture.

To let the user draw the map we used a library called Leaflet [44]. With this library the user can draw walls that mark the border of the building, he can draw walls within the building and he can position beacons in the map. Furthermore, the user is able to enter the parameters that are required for the BLE beacons in the system. Figure 5.1 shows a screenshot of the application when a map is drawn.

An extra feature of the map creation tool is that the locations of the users that are located in the loaded map are displayed real-time on the map. This can be used by, for example, shop owners to review how their costumers move trough their shop.

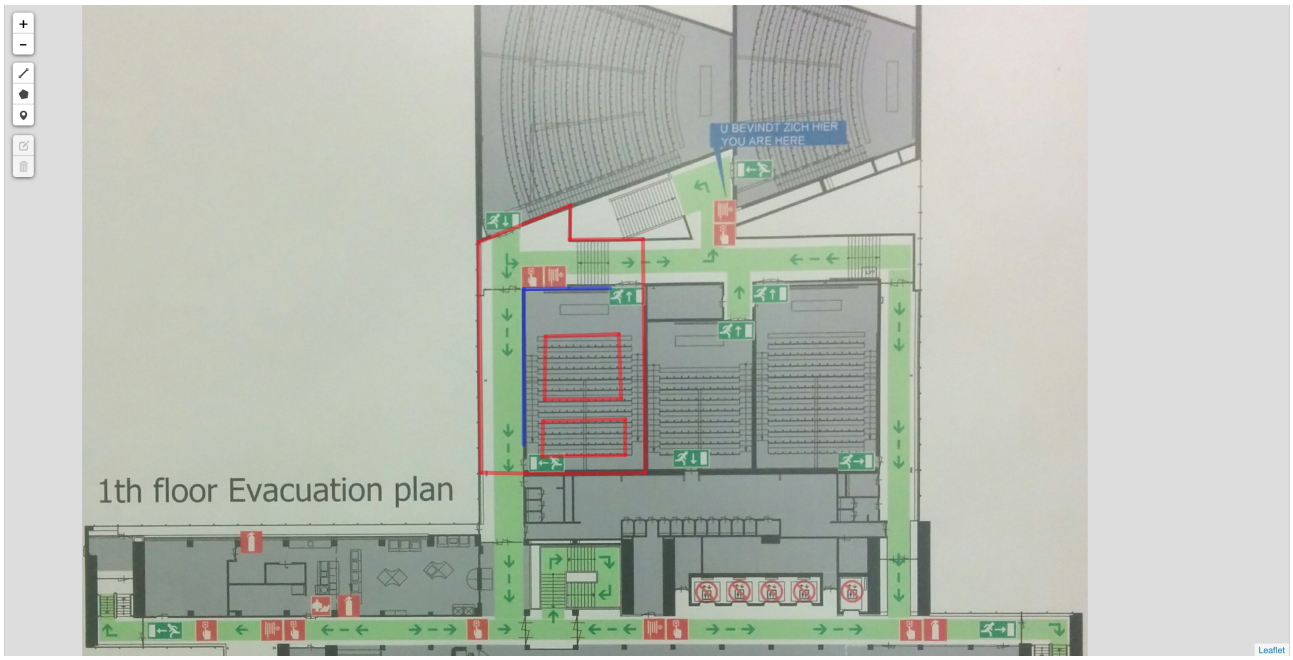


Figure 5.1: Screenshot of the map creation tool.

6. Indoor localization using Trilateration

The first approach to indoor localization was by means of trilateration using Bluetooth Low Energy (BLE) beacons. This section describes the challenges of trilateration and how we have addressed those in the application. Trilateration works in two steps: 1. transforming the measured Received Signal Strength Indicator (RSSI) values into distance estimates for each of the beacons in range and 2. estimate the user's location using the estimated distance to each of the beacons.

Both steps are discussed in Paragraph 6.1 and 6.2, respectively. In Paragraph 6.3 we discuss how an extended version of trilateration, iterative trilateration, has been implemented in our localization app. In Paragraph 6.4 we discuss the accuracy of the trilateration algorithm that we implemented.

6.1 Mapping an RSSI value to a distance

The distance from the phone to a beacon can be estimated based on RSSI values that the phone measures originating from that beacon. It is a challenge to convert the measured RSSI values to a good distance estimate, because the measured RSSI values fluctuate heavily over time, as can be seen in Figure 6.1. Furthermore, the rate at which the strength of the signal reduces as the distance increases, the propagation loss, is different in every room.

To estimate a distance based on the measured RSSI values a well known *Radio Propagation Model* which is called Log-Normal Shadowing (LNS) is used [66] [63]. This model is expressed in Equation 6.1.

$$rssi = -(10n \log_{10}(d) + A) \tag{6.1}$$

In this model, $rssi$ is the RSSI value that is read, d is the distance from the source of the signal and A is the average RSSI value at 1 meter from the beacon. n is the environmental factor which describes the propagation loss of the environment the beacon is set up in [66].

In order to transform an RSSI value to a distance A and n have to be calculated. We build a setup activity which has to be performed at the location where the beacons are installed to calculate these parameters. This activity lets the user perform RSSI measurements of the beacon that is set up at predefined distances. Currently, measurements are taken at a meter interval, until the user decides that enough measurements are taken. Then the Ordinary Least Squares (OLS) algorithm [63] is used to estimate the values for A and n for which the variance of the errors between the observed RSSI values and the RSSI values that are calculated with the estimated parameters is minimal. We use the OLS algorithm because it is straightforward to implement and can be executed efficiently.

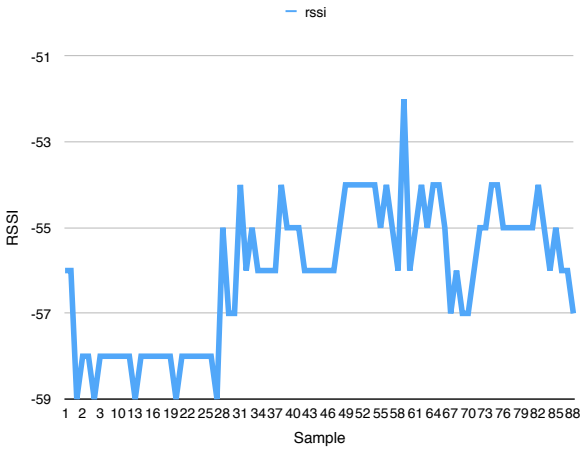


Figure 6.1: RSSI measurements from a beacon at 1 meter.

During our experiments we found out that it is difficult to calibrate LNS well. To remove the calibration step, we choose to use a static version of the LNS formula, which is expressed in Equation 6.2. A disadvantage of this static formula is that the distance estimates are not accurate above 10m.

$$d = 10^{\frac{(-A-RSSI)-41}{20}} \tag{6.2}$$

We obtain the signal strength at one meter, A, from the Eddystone data that the beacon sends.

To reduce the fluctuations in the distances, that are due to the fluctuations in the RSSI values, a Low Pass Filter (LPF) is used [38]. A LPF takes the previous RSSI values into account when the new measurement is coming in. This reduces the impact of outliers on the distance estimate. The filter is stated in Equation 6.3, where α is a constant between 0 and 1, T_n is the distance calculated at n and P_n is the reported distance after filtering at time n. We implemented a LPF in our application with $\alpha = 0.7$.

$$P_n = \alpha P_{n-1} + (1 - \alpha)T_n \tag{6.3}$$

6.2 Getting the user's location

When the distance to each of the beacons in range is known, the user's location can be estimated using trilateration. In this paragraph we first explain how basic trilateration works and discuss its downsides. Then, we explain an extended version of trilateration, iterative trilateration, and explain why this method is more useful to estimate the location of the user.

6.2.1 Basic Trilateration

Basic trilateration, also called *three-border positioning*, calculates a location based on the distance to at least three beacons [64] [36] [66]. For each of the BLE beacons in range B_i we can calculate the distance d_i, as described in the previous paragraph. For each beacon this means that the location of the smartphone is somewhere on the edge of a circle with radius d_i and as middle point the location of B_i. From now on we refer to this circle as a *distance circle*.

With two beacons the number of intersection points of their distance circles can be maximal 2, as can be seen from figure 6.2. This means that there are two possible locations of the smartphone, point A or point B. A third distance circle is needed to distinguish between point A and point B and get the final location, as can be seen in Figure 6.3.

Suppose we have three beacons that are located at $B_1 = (x_1, y_1)$, $B_2 = (x_2, y_2)$ and $B_3 = (x_3, y_3)$. The estimated distance from the phone to the beacons are d_1, d_2 and d_3, respectively. From the distance circles we know that:

$$\begin{aligned} d_1^2 &= (x - x_1)^2 + (y - y_1)^2 \\ d_2^2 &= (x - x_2)^2 + (y - y_2)^2 \\ d_3^2 &= (x - x_3)^2 + (y - y_3)^2 \end{aligned} \tag{6.4}$$

where x and y are the coordinates of the unknown location. Subtracting the first and the second equation, and the second and the third equation results in:

$$\begin{aligned} (-2 \cdot x_1 + 2 \cdot x_2)x + (-2 \cdot y_1 + 2 \cdot y_2)y &= d_1^2 - d_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 \\ (-2 \cdot x_2 + 2 \cdot x_3)x + (-2 \cdot y_2 + 2 \cdot y_3)y &= d_2^2 - d_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2 \end{aligned} \tag{6.5}$$

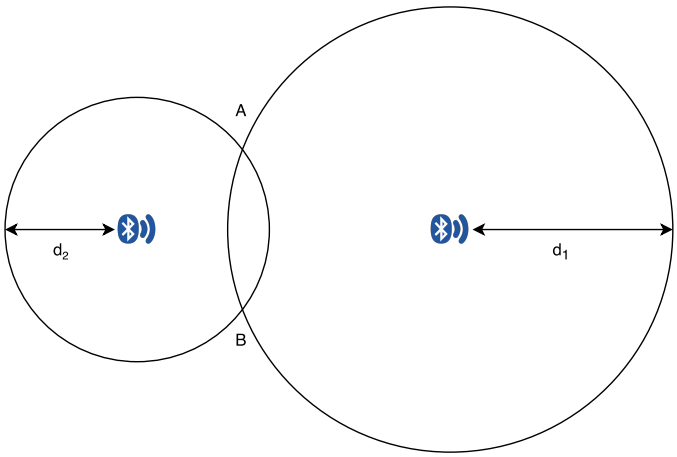


Figure 6.2: Two reference points give possible locations A and B

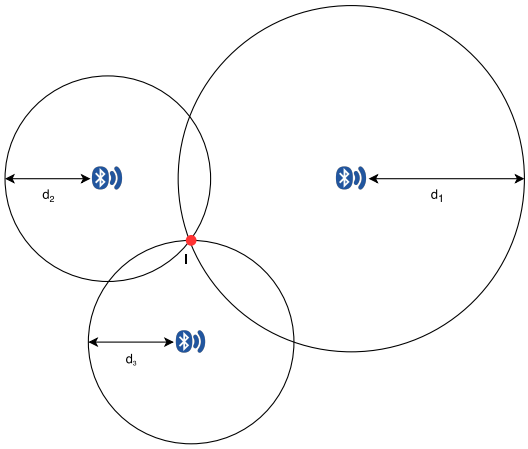
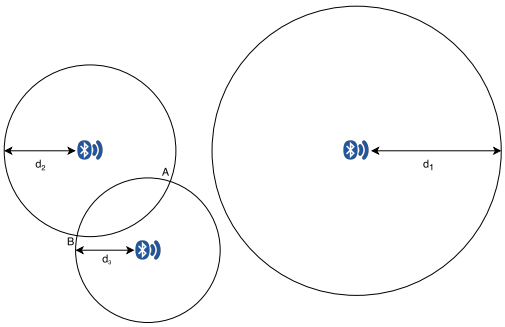
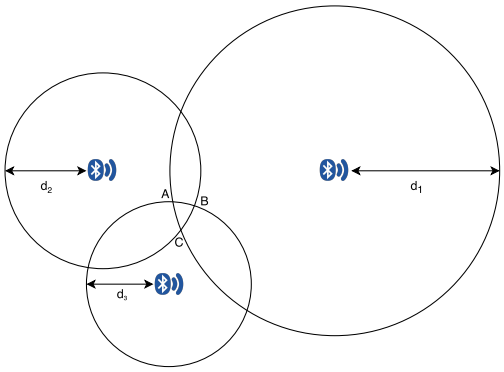


Figure 6.3: With three reference points only intersection point I remains.



(a) To less intersections.



(b) To many intersections.

Figure 6.4: Two situations where the outcome of the trilateration algorithm can be improved.

Which is a linear system in the form of

$$\begin{aligned} Ax + By &= C \\ Dx + Ey &= F \end{aligned} \tag{6.6}$$

that can be solved for x and y analytically as well as numerically to obtain a location.

However, there are situations, even with three reference points, where there is more than one intersection point. It can also happen that the distance circles do not intersect at all. Both situations are depicted in Figure 6.4. When using BLE beacons for localization such situations occur regularly as the distance estimates are not accurate. The distances are then not representative for the real world. Therefore, a more advanced trilateration method is needed.

6.2.2 Iterative Trilateration

To overcome the problem that trilateration does not return a good estimate in case not all distance circles intersect or the distance circles intersect at multiple points, iterative trilateration is used. Iterative trilateration estimates the location that matches the available data best [60]. This does not necessarily mean that the location the algorithm returns is accurate in the real world, because the RSSI values are not reliable enough to convert in an accurate distance.

Iterative trilateration uses the gradient descent minimization algorithm [52] to minimize the error between the measured distance for each of the reference points and the distance between the estimated location and each of the reference points. This error is known as the fractional error and is defined as

$$f_i = |\frac{d_i - \sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}}{d_i}| \tag{6.7}$$

where f_i is the fractional error for beacon i, d_i the measured distance to reference point i, (x_i, y_i) the location of beacon i and (x_e, y_e) the iteratively updated estimated location of the user. The absolute error is divided by d_i to give distances that are shorter more weight. The shorter distances are given more weight, because they are often more accurate.

The gradient descent algorithm calculates the slope of the fractional error at the estimated location and travels in the opposite direction of the slope. it stops when the difference in slope between two iterations is below a certain threshold indicating that a minimum has been found. Basic trilateration, which is discussed in 6.2.1, is used to calculate the initial estimated location.

Multiple methods can be used to implement the gradient descent algorithm. First, we followed the approach defined in [60]. However, as we did not manage to get locations that fit the data well, we changed the implementation to the method described in [24] which turned out to work well.

6.3 Trilateration Implementation

In this section we describe how the distance calculations and the iterative trilateration algorithm have been implemented in the localization app.

Figure 6.5 shows an Unified Modeling Language (UML) diagram of the most important classes involved in location estimation based on the RSSI measurement of the beacons. A pipeline, called `TrilaterationPipeline`, is used to perform the whole trilateration process. First, the beacons found during the last scan are retrieved in a `FoundBeaconList` object. The `BeaconMeasurement` objects in this list all represent a beacon in range. Each beacon that is in the `KnownBeaconList`, containing a list of beacons that belong to bunq, is given to a `DistanceCalculator` object, which calculates a distance based on the measured RSSI value. The `DistanceCalculator` object returns `LocationPair` objects, which contain a distance to and the location of a beacon. These objects are inputted in the `TrilaterationCalculator` to obtain an initial location estimate, a `Location` object. The initial location is inputted in a `IterativeTrilaterationCalculator` object which performs the iterative trilateration algorithm. Returned is a `Location` with the (local) minimum fractional error which, therefore, represents the measured data well.

6.4 Location Estimation Experiment for Trilateration

We evaluate the accuracy of the location estimate by the test setup depicted in Figure B.1 in Appendix B. The room is 14.5 by 14.6 meter, representing a small shop. To get a repeatable test 10 points divided over the room are chosen. At each test point we let the algorithm calculate a location three times. For this static method of testing is chosen, because it allows for a good view of the accuracy of the method. The accuracy of the location estimation method is the Euclidean distance between the actual location and the estimated location.

As the room has several obstacles, such as a kitchen and several pieces of furniture, a normal environment is simulated. During the experiments people were allowed to use the room

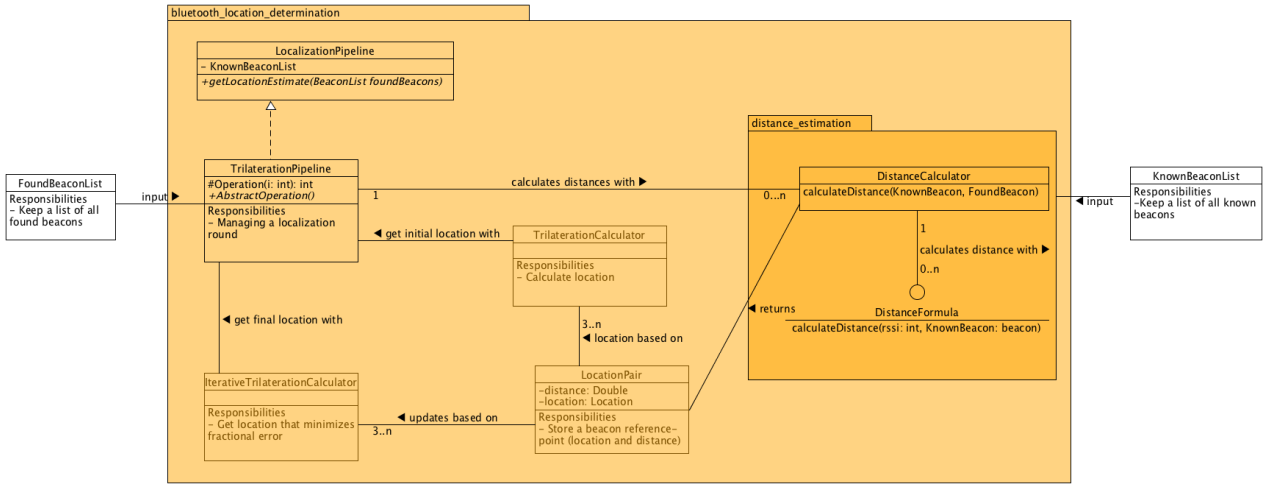


Figure 6.5: UML Diagram of the most important classes involved in the trilateration pipeline.

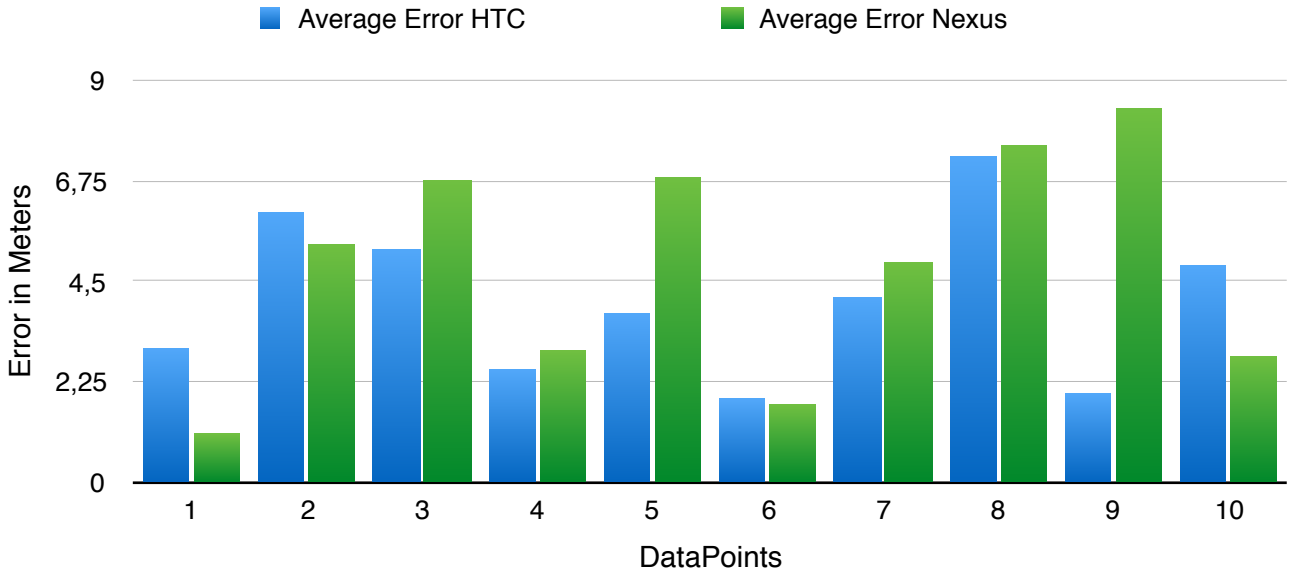


Figure 6.6: The average error per test location for iterative trilateration combined with the static distance formula.

as would be done in a real scenario. All tests were conducted with two phones: a Nexus 5 [45] and a HTC One M7 [34]. Using two phones allowed us to see the accuracy differences between phones.

The localization test that is discussed below has been conducted with the static LNS formula for calculating the distance and iterative trilateration for estimating the location. Other experiments, with basic trilateration and the calibrated LNS formula, resulted in an even lower accuracy.

The average error for both phones at each data point is shown in Figure 6.6. The estimated locations of testpoints 1 and 6, which are in the middle of the room, are within 2 meter of the actual location. However, testpoints that are closer to the wall, for example test point 3, are estimated to be more to the center of the room, resulting in a lower accuracy. This is due to inaccuracies at the distance estimation. If the user is more than 10 meter away the static distance formula returns distances that are to short. Therefore, the estimated location at the test points in the corner were 'pulled' to the center by the, too short, distance estimates.

However, as we modified the formula to return a better distance estimate for a long distance, the estimates at short distance got worse.

We tried to increase the number of beacons in the room to six, but that did not improve the average accuracy of the localization. What we did see, however, is that the location estimates are more accurate near the locations where an extra beacon was added. This suggest that adding even more beacons results in a higher accuracy. However, adding more beacons is a costly solution, and therefore we decided not to do this.

With an accuracy of roughly 4.5 meters on average we conclude that iterative trilateration is not going to be accurate enough for bunq's purposes. The inability to estimate the distance accurately and constant over time is the main reason that the accuracy is not accurate enough. Therefore, we have also looked at another method for indoor localization, namely a particle filter.

7. Indoor localization using Particle Filters

Another technique that we used to locate the user's phone is a particle filter. A particle filter is a probabilistic method that is able to estimate the user's location over time. We used this technique as it is a robust method, that does not necessarily rely on the unpredictable RSSI values. Instead it relies on the assumptions that the movements of the user can be obtained accurately and that a map of the building is available. In this section we describe the basic principles of a particle filter and how we implemented these principles in the prototype application.

A particle filter is an iterative process that consists of three steps: moving, sensing and redeploying. Because it is unknown where the user is initially, the initial belief is that the user can be anywhere in the map. This is represented by, so called, particles that are deployed uniformly at random over the map of the location. Each particle is given a weight, which represents the likelihood that the particle is at the correct location. The weights of all particles should sum to 1. At the beginning of the process all particles have equal weight.

The process of moving, sensing and redeploying changes how the particles are distributed in the map. In short, based on the movements of the user the particles move in the map. Particles that want to move trough a wall are eliminated. Therefore, particles are only at locations that could have been reached with the user's movements. The user's location can be estimated to be at the place where the density of the particles is highest. To find the place where the particle density is highest the map of the location is divided into 1 x 1 meter blocks. The number of particles in each block is counted. The block with the most particles is the estimated location of the user.

In the following paragraphs we describe how each of the three steps work. But first an overview of the most important components of particle filter system in our application.

7.1 Particle filter implementation

To get an overview of the particle filter system an Unified Modeling Language (UML) diagram of the most important classes of the particle filter implementation is shown in Figure 7.1.

The `ParticleFilterController` signals the particle filter each time the user moves or the user manually instructs the system to perform a particle filter round. The `ParticleFilter` then calls the methods `move()` on the implementation of `MotionModel` interface to move the particles, `sense()` on all the implementations of the `Senser` interface that are listed in the particle filter to reweight the particles of the particles and `redeploy()` on the implementation of the `RedeploymentMethod` interface to redeploy the particles. Those methods perform the moving, sensing and redeploying steps of the particle filter, respectively.

Particles are initially deployed with the `InitialParticleDeployer`, which deploys the particles uniformly at random over the site. The `ParticleRedeployer` redeploys the particles when the particle filter detects that the particles are stuck, as will be discussed in 7.4.1.

7.2 Moving

The moving step of the particle filter consists of two parts: obtaining the movements of the user in a motion model and translating those movements into movements of the particles. The

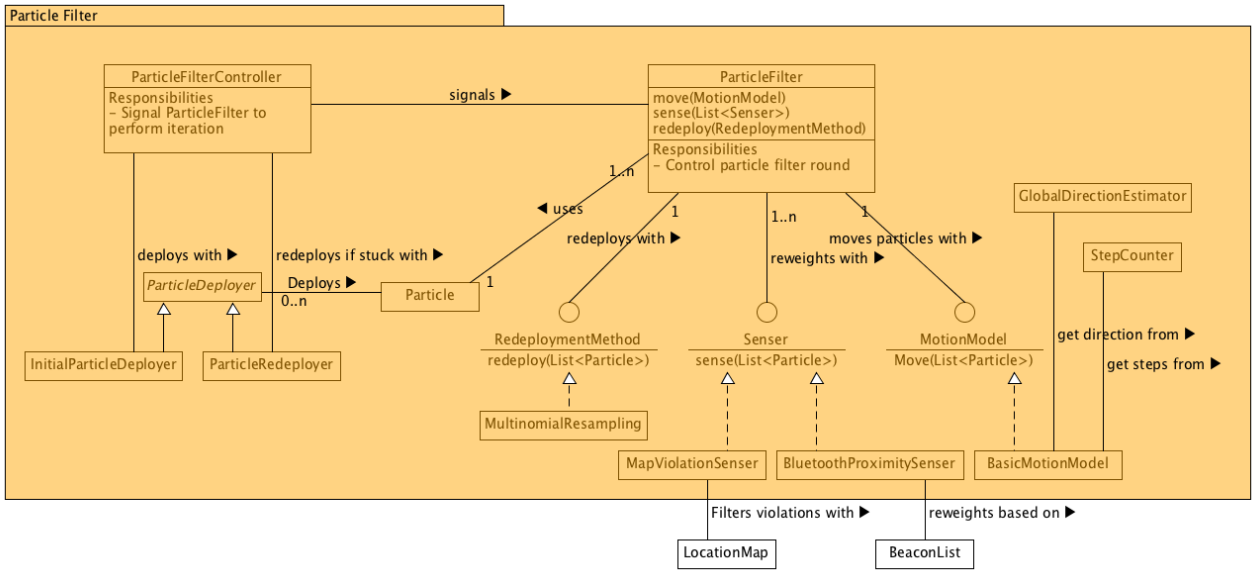


Figure 7.1: UML Diagram of the most important classes involved in the particle filtering.

motion model consist of a *heading detector* and a *step detector*. Both the detected heading and the detected number of steps are combined in a `Movement` object.

In the following we first explain how the step detector works. Then we explain the step counter. Last, we explain how we move the particles based on a list of `Movement` objects.

7.2.1 Heading Detector

As the detected heading determines the general heading of the particles it is important to accurately detect the heading of the user. Figure 7.2 shows what happens if heading detector is not accurate. Instead of going straight through the corridor the particles collide with a wall. Particles that collide with a wall are eliminated and therefore the corridor in the figure would lose the particles in it over time. This is not wanted as the user could have walked trough this corridor in the real world.

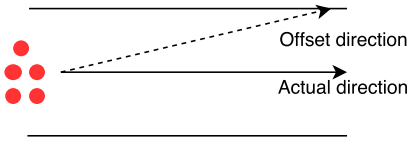


Figure 7.2: This figure shows the effect of an incorrect heading on the particles.

The heading of the movements is estimated using Android's build in method `SensorManager.getOrientation()`. This method uses the magnetometer and gyroscope to estimate the heading of the phone in its own orientation system. The heading is translated back to the world's orientation system by using the `Sensor.TYPE_ROTATION_VECTOR` sensor.

Two problems arise when using the magnetometer for getting the heading of the user. First, due to the sensor in the phone or other factors the magnetometer can have a static heading off-set. For example, our measurements show a deviation of around 20° in the EWI-building from the real north. This problem is solved by letting the particles determine the static heading offset of the magnetometer, as will be discussed in Paragraph 7.2.3. Second, the magnetometer is influenced by objects in the surroundings of the phone. We noticed, for example, that the magnetometer reading fluctuated highly when we held the phone close to a heater. We have solved this by adding a random offset between $[-25°, 25°]$ when moving each particle, as will be discussed in Paragraph 7.2.3. Furthermore, a circular Low Pass Filter (LPF) is used to subdue the high frequency fluctuations in heading measurements.

7.2.2 Step Detector

We count the number of steps the user has taken in two ways :

1. For phones with support for Step Detector API (introduced in API level 19, but requires a separate co-processor), the built-in step sensor is used to count the steps of the user.

2. Phones that not have a built-in step sensor use the accelerometer to count the number of steps taken.

The algorithm for detecting steps if the phone does not have a build-in step detector, consists of two parts: 1.) a part that detects if the user is walking and 2.) a part that is responsible for actually counting the steps. The algorithm is separated in two parts, because we do not want to count steps if the user is not walking. To detect whether the user is walking we use the fact that walking is a repetitive motion, as can be seen in 7.3. Autocorrelation is used to detect whether the user is walking or standing still. Autocorrelation detects how similar two samples are by multiplying the samples with different off-sets for one sample. Samples that have a lot in common indicate that the user is walking.

The number of steps taken are counted by Algorithm 1. If the user is walking this algorithm counts the peaks in the magnitude of the accelerometer.

Algorithm 1 This algorithm counts the number of steps within a sample.

```
1:  procedure COUNT STEPS
2:      magnitudes ← [Sample with accelerometer magnitude data]
3:      samplesBeforeCounting ← magnitudes.length/4
4:      intervalBetweenCounting ← magnitudes.length/2
5:      countedsteps ← 0
6:      samplesSinceStartPeak ← 0
7:      samplesSincePreviousCount ← 0
8:      for all magnitudes do
9:          if magnitude > 11.4 && samplesSincePreviousCount > intervalBetweenCounting then
10:             samplesSinceStartPeak++
11:         else
12:             samplesSinceStartPeak ← 0
13:         if samplesSinceStartPeak >= samplesBeforeCounting then
14:             counter++
15:             samplesSincePreviousCount ← 0
16:             samplesSinceStartPeak ← 0
17:         else
18:             samplesSincePreviousCount++
19:     return counter
```

Normally every peak represents a step. However, the algorithm takes in account that it's impossible to have peaks that only consist of one measurement or very short after each other. These events occur normally due to noise.

In order to increase the accuracy of the motion model the direction of each step is stored in a `Movement` object. When the direction of each step is known the motion model can better mimics the user's movements and is therefore better at moving the particles accurately.

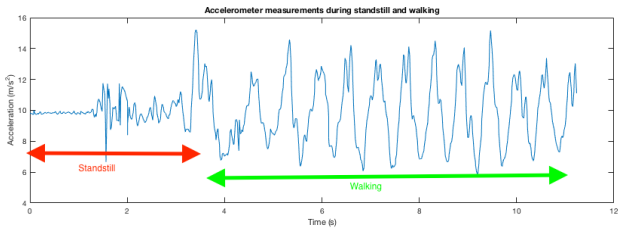


Figure 7.3: Plot of accelerometer measurements during walk and standstill.

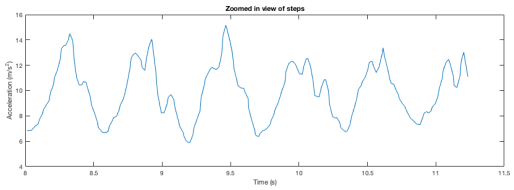


Figure 7.4: Plot of accelerometer measurements during walking.

7.2.3 Moving the Particles

The particles are moved based on the `Movement` objects that are created by the motion model. In these objects `heading` and `nrOfSteps` are stored.

Each `Particle` object contains a `location`, a `particleOrientationOffset` and a `strideLength`. The `particleHeadingOffset` stores the static heading offset that is given to to the particle at random from $[-45°, 45°]$. The purpose of this offset is to estimate the static offset in heading, that is for example caused by the phone of the user. Particles with an incorrect `particleHeadingOffset` will be eliminated as they will collide with a wall at some point. The stride length of the user is estimated in a similar fashion. Each particle is given a random stride length in $[0.5, 1.0]$ meter, as the stride length of most humans falls in this range. particles with a value `strideLength` that does not match the actual stride length of the user will be eliminated when they collide with a wall. Particles with a stride length that matches the stride length of the user will remain.

The information from the `Particle` and the `Movement` is combined to calculate the new location of a particle by:

$$newX = location.x + steps * (strideLength + strideRandom) * \cos(particleHeadingOffset + randomHeading + northOffset) \tag{7.1}$$

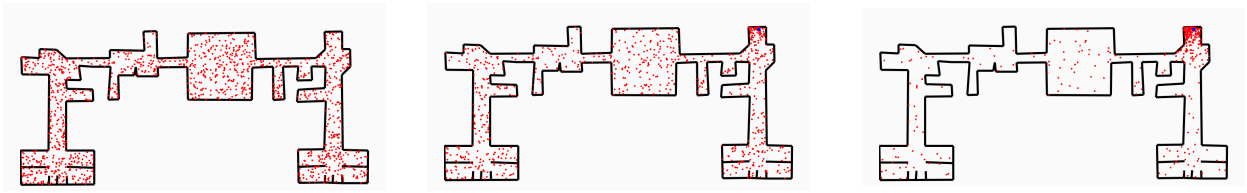
$$newY = location.y + steps * (strideLength + strideRandom) * \sin(particleHeadingOffset + randomHeading + northOffset) \tag{7.2}$$

where `strideRandom` is a random number in $[-0.15, 0.15]$ meter to account for variations in stride length, `randomHeading` is a random number in $[-25°, 25°]$ to account for variations in heading and `northOffset` that expresses the rotation of the map with respect to the north. The idea of using a variable stride length and heading offset has been adopted from [61]. It allows for the particle filter to be used by different persons with different phones.

7.3 Sensing

After all the particles have been moved based on the movements of the user, the particles are reweighted. All particles that have crossed a wall will be given a weight of 0. Those particles will be eliminated during the redeploy step.

One of the challenges of the particle filter is to make the particles converge to the user's location as fast as possible. In order to make the particles converge to the user's location faster a BLE beacon proximity senser has been added to the particle filter. This senser reweights the particles based on the measured distance to each of the BLE beacons that are in range. When a beacon is in range, the particles are reweighted by `oldWeight * (1 + 2 *`



(a) Initial belief (b) Round 1 (c) Round 2

Figure 7.5: Redeployment of particles based on beacon measurement. The beacon is positioned in the top right room.

`gaussian(distanceToBeacon))` where `gaussian(distanceToBeacon)` returns the probability of `distanceToBeacon` for a Gaussian distribution with as mean the estimated distance to the beacon. `distanceToBeacon` stores the distance from the particle to the beacon. During our experiments we found out that this feature allows the user's location to be determined within 2 particle filter rounds in areas were beacons are deployed as can be seen in Figure 7.5.

7.4 Reploying Particles

In the redeployment step, the particles are redistributed based on the weight given to each particle in the sensing step. This means that particles with an high weight will be duplicated and particles with a low weight will be removed. After redeployment all particles are given equal weight. This ensures that particles will not be 'ruled out' because their weight gets really low. Particles are redeployed with the multinomial re-sampling algorithm [32]. The weight of a particle determines the chance that a new particle is redeployed at the particle's position. So none of the particles will be redeployed at the location of a particle that crossed a wall.

After redeployment, the locations with a lot of particles are those where the map constraints have not been violated and the beacon measurements match best. Those locations have a higher probability of being the user's location.

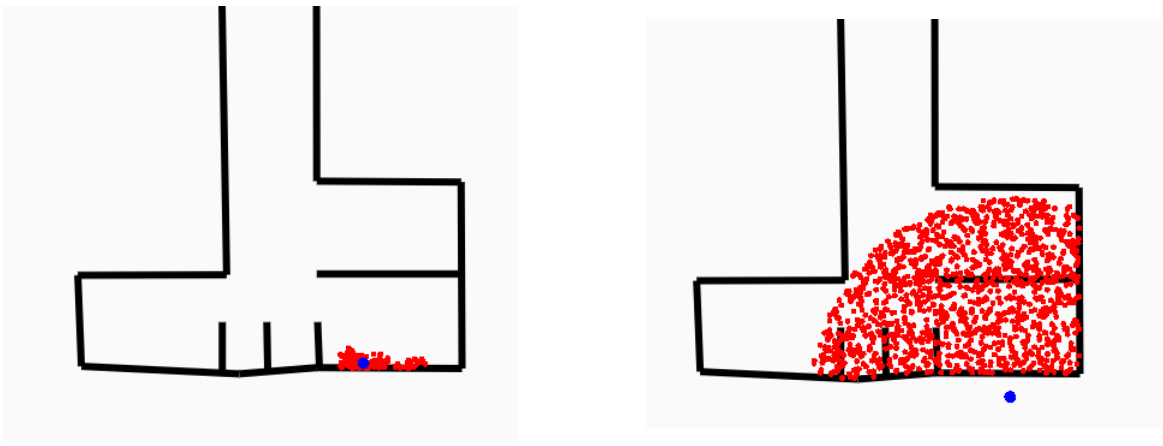
Besides redeployment during an iteration, particles are redeployed in two other situations. First, when the particles are stuck, all particles are redeployed around the last known location of the user. Second, to let the particle filter recover from particles that move incorrectly, a small part of the particles is redeployed when Bluetooth Low Energy (BLE) beacons are in range.

7.4.1 Redeploy Stuck Particles

Besides redeploying the particles in each iteration, a redeploy is also needed when all particles are stuck. This kind of redeploy is initiated at the moment that all particles want to cross a wall, indicating that all particles are stuck. All the particles are then redeployed within a circle with a diameter of 7 meter from the last known location of the user. After redeploying, the particles follow the normal procedure of moving, sensing, and redeploying to re-estimate the user's location. Figure 7.6 shows the particles before and after redeploying.

7.4.2 Redeploy Particles based on BLE Beacons

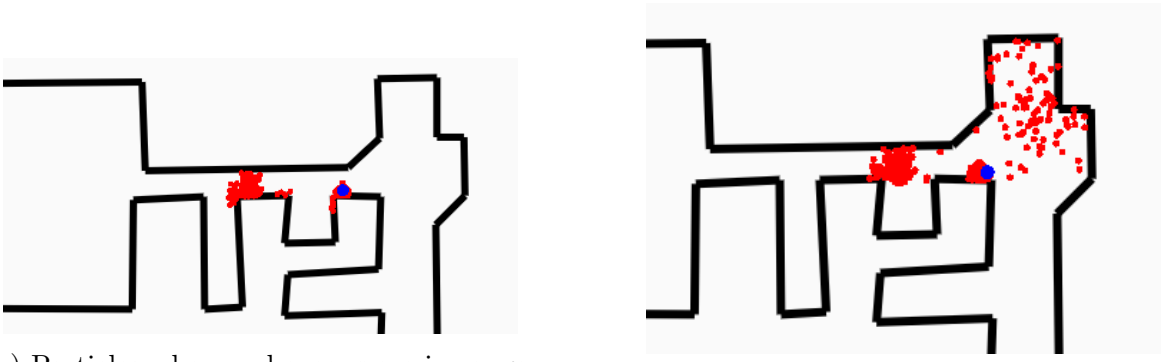
To reduce the impact of a fault in the motion model, 6% of the particles are redeployed based on the BLE beacons in the vicinity of the phone. A fault in the motion model, for example the



(a) Before redeploy

(b) After redeploy

Figure 7.6: Redeployment after the particles get stuck.



(a) Particles when no beacons are in range.

(b) Particles when beacon in top right room is in range.

Figure 7.7: Redeployment of particles when a beacon is in range.

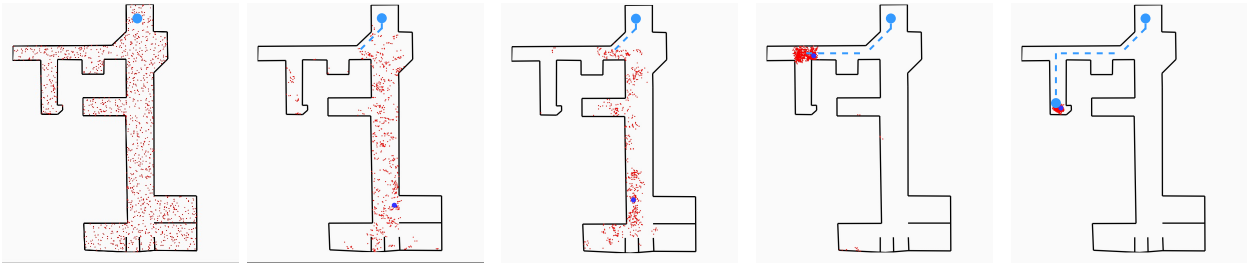
heading offset has been incorrectly estimated, causes the particles to move incorrectly. In an open space this will not immediately lead to the elimination of the incorrectly moving particles because the particles will not hit a wall. However, the particles would eventually hit a wall which would result in a redeploy as described in Paragraph 7.4.1.

To prevent such a redeploy, 6% of the particles are selected at random and are redeployed based on the BLE beacons in range, as can be seen in Figure 7.7. This 6% of the particles is divided equally over all beacons that are measured within 2.5 meters.

The 6% of the particles that are redeployed based on the beacons in range continue the localization process in case the other particles collide with a wall. At the same time, redeploying only 6% of the particles allows the other particles to remain the biggest group that determines the location as long as no collisions have occurred. In short, the particles that are redeployed based on the BLE beacons in range allow the particle filter system to recover from an incorrect movement estimate without a general redeploy.

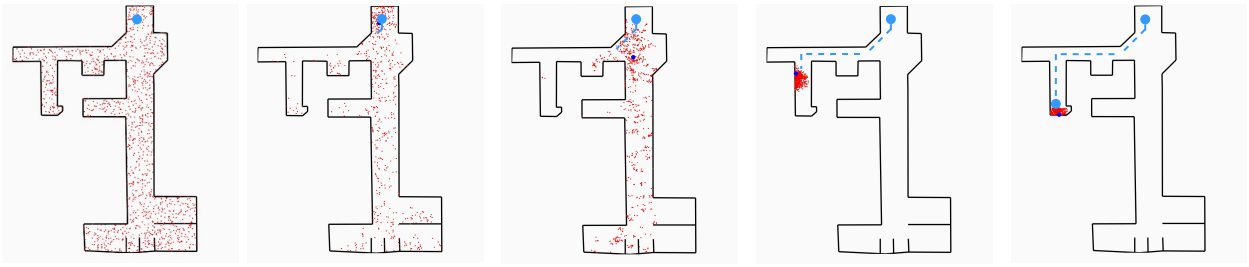
7.5 Particle Filter Results

In this section we show the working of the particle filters based on a number of example walks. Based on these examples we show how the particle filter works in practice and how the added



(a) (b) (c) (d) (e)

Figure 7.8: User walk with particle filter tracking without BLE based redeploy.



(a) (b) (c) (d) (e)

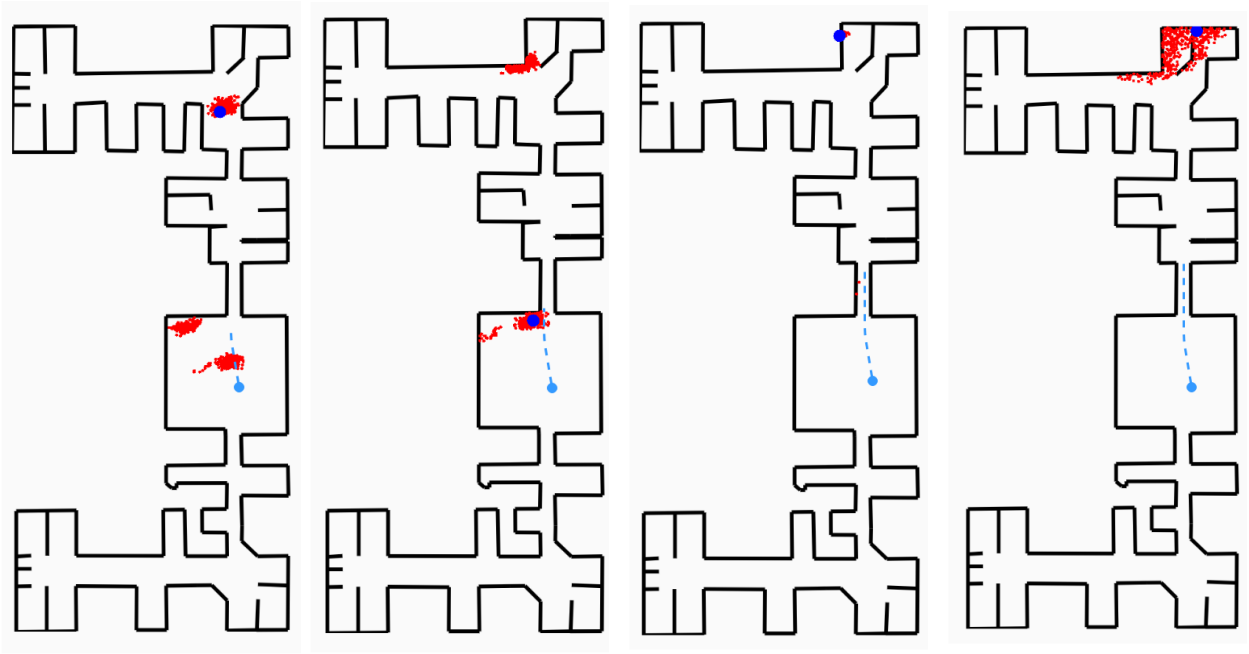
Figure 7.9: User walk with particle filter tracking with BLE based redeploy.

BLE features improve the location estimate.

Figure 7.8 shows how the particles behave during the user's walk without the added BLE based features. From this figure it is clearly visible that the particles, the red dots, follow the movements of the user, the dashed line. As expected, the particles converge to a single location because the particles that collide with a wall are eliminated. However, we can see that it takes until Figure 7.8d before the location of the user, the dark blue dot, is estimated near the actual position of the user.

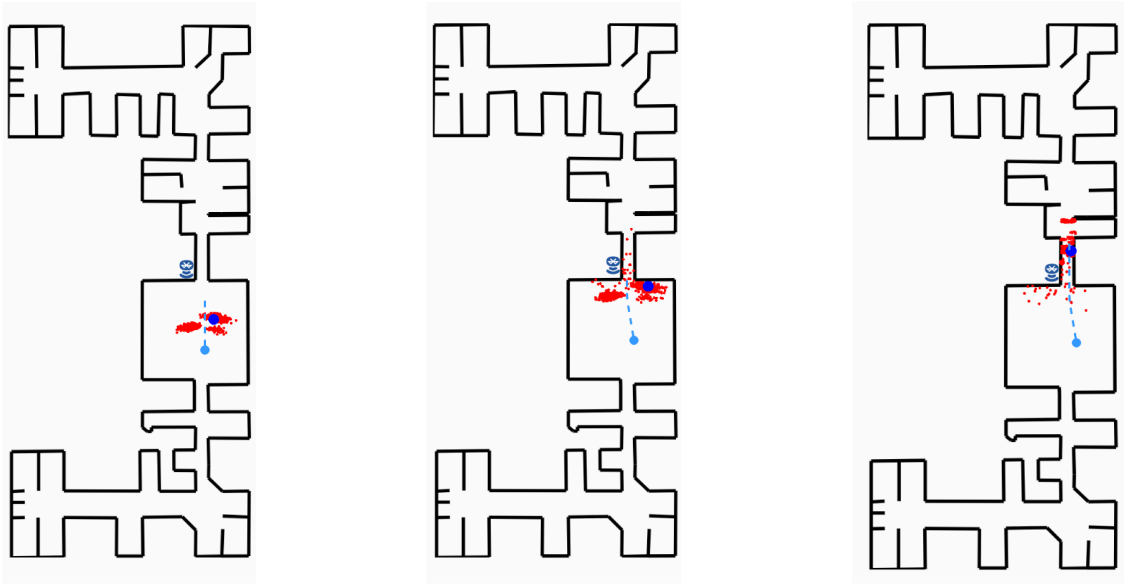
The same walk with BLE-based reweight and redeploy enabled is shown in Figure 7.9. During this walk, the estimated location of the user is already close to the real location of the user after a few steps as can be seen in Figure 7.9b. This shows that the particles converge faster to the user's location when the BLE-based reweight and redeploy are intended.

The added value of the BLE-based redeploy is more clearly shown in a another example of an user's walk. In Figure 7.10 the user walks straight in a larger room. However, due to an incorrect heading offset all particles are redeployed at a location that is not even close to the user's location (Figure 7.10d). Redeployment based on the BLE beacons in the phone's range fixes this problem, as can be seen in 7.11. As part of the particles is redeployed around the beacon (Figure 7.11b), the redeployed particles are able to continue moving even after the incorrectly moving particles are eliminated (Figure 7.11c), thereby preventing a redeploy. Because particles have more room to move incorrectly in larger rooms, the BLE beacons can best be deployed at the exits of larger rooms. The beacons will then be detected by the phone when the user passes and particles are deployed around the beacon. When the other (possibly) incorrectly moving particles then collide with a wall, as happens in Figure 7.10c, all particles are redeployed the the locations of the particles that were redeployed based on the BLE beacons. The beacons at the exits then pick up the phone when the user passes and the particles with an incorrect heading offset are eliminated as they collide with a wall.



(a) User moves straight. (b) Particles move. (c) Particles eliminated. (d) Unwanted Redeploy.

Figure 7.10: An user's walk where an undesired redeploy happens because of an error in the heading offset.



(a) Particles move. (b) Beacon detected. (c) Particles continue moving.

Figure 7.11: An user's walk where the BLE beacons allow the particles to recover from an error in the heading offset.

8. Identifing the Closest User

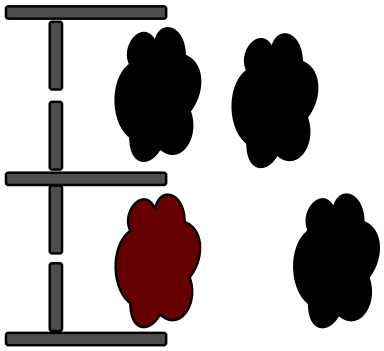


Figure 8.1: Example scenario of detecting the user closest to a gate. Only the person with brown hair should be identified as the person at the gate.

To apply the knowledge obtained from the localization experiments to a real world scenario we implemented a system that uses Bluetooth Low Energy (BLE) beacons to detect which user is closest to an object.

Imagine yourself waiting with a ticket on your phone for a concert at Utrecht Jaarbeursplein. Currently, your ticket is checked at the the gates after which you are checked again by security guards, which is a time consuming process. The speed at which the people are allowed to access the concert hall can be sped up if the users' tickets could be checked automatically, while the security guards are checking your stuff. A system in the gate detects the phone of the user that is currently walking though the gate. After the phone is detected, the ticket on the phone is downloaded by the system and checked. If the ticket is valid, the gate will open and allow the waiting person. This would reduce amount of personal needed at the concert hall's entrance.

Therefore, a robust method is needed that is able to identify the user that is standing closest to an object, for example, a gate. Figure 8.1 depicts an example scenario of the concert hall's gate setup. The percentage of correct identifications should be near 100% as this method is going to be used to let a lot of users enter a concert hall and we do not want to open the gate where person with an invalid ticket is waiting or disallow someone with a valid ticket from the concert hall.

8.1 Implementing Closest User Detection

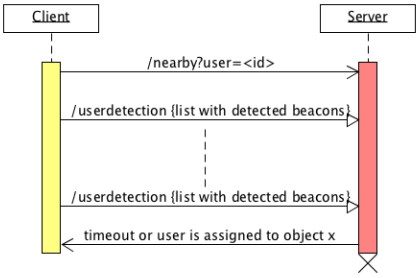


Figure 8.2: Sequence diagram of network interactions to detect the user closest to an object.

We decided to position a BLE beacon at each object and use the user's phone to collect the Received Signal Strength Indicator (RSSI) values of each of the beacons in range. Figure 8.2 shows the communication between the phone and the server. After each measurement the client pushes the beacons in range and the corresponding RSSI values to the server. Furthermore, as it starts scanning it sets up a long-polling connection to wait for a response [30]. This response is a message from the object or a time-out which means that the phone has not been detected closest to an object.

As with trilateration the variance of the RSSI measurements and the difficulty to transform them in a distance makes it hard to assign a tab to the closest person. Figure 8.3 shows one user that is waiting for an object within 0.5 meter but also has an object at 1.5 meter. This figure shows that the waiting user sometimes measures the beacon at 1.5 meter, for which he is not waiting, as strong as or even stronger as the beacon he is waiting for. It is the responsibility of the web service, described in Paragraph 8.1.1, to filter out the beacons that are in range, but not closest.

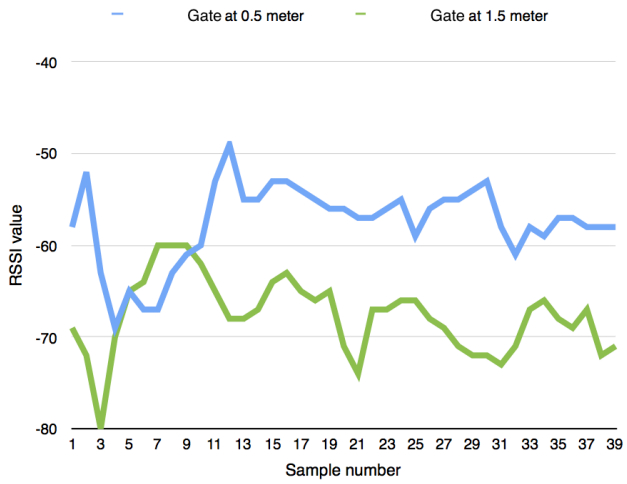


Figure 8.3: A phone that has a gate at 0.5 and at 1.5 meter.

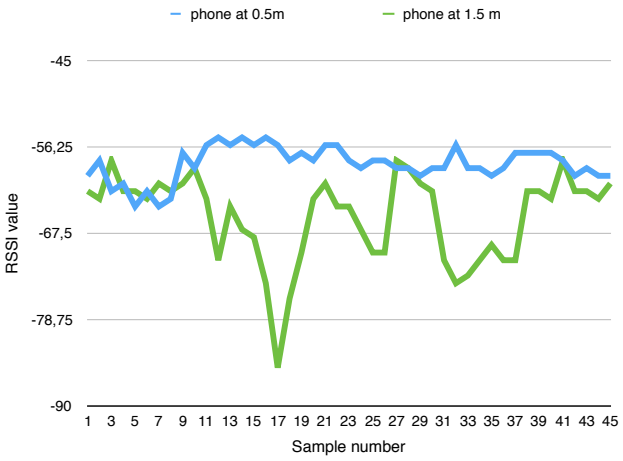


Figure 8.4: Two phones at 0.5 and 1.5 meter in range of the same gate.

Figure 8.4 shows two users, that are 0.5 meter and 1.5 meter from the same object, that are sending RSSI data to the server. This figure shows that the RSSI values measured by the phone at 1.5 meter are sometimes as high or higher than the RSSI values measured by the phone at 0.5 meter. An algorithm is needed that detects the closest person even if the measured RSSI values of two phones are at the same level. We discuss such algorithm in Paragraph 8.1.2.

8.1.1 Web service

A web service determines which object the user is closest too. For each time that it is necessary for the object to get the closest phone, the server creates a thread that decides which user is closest to the object.

The web service receives the beacons in range of the user's phone and uses this data to find the object that the user is close enough to. First, the server extracts the beacon with the highest RSSI value that belongs to a gate. With this, it filters out all measured beacons that are not closest to the phone in this measurement 'round'. This ensures that each measurement round only influences the decision of one of the objects. The user can only be standing for one object, so it is not logical to let the measurement influence multiple objects at once. Note that the selected beacon can be of a object that is not actually closest to the phone, as can be seen in Figure 8.3. However, the algorithm described in the next paragraph ensures that object to which the beacon belongs that is closest over multiple measurements receives the information of the phone's user.

After that, the RSSI value of the best measurement is transformed into a distance by the formula described in Paragraph 6.1. If the distance to the closest beacon is more than 1 meter, the measurement is ignored, as the phone is not close to any gate. Otherwise, the distance is passed to the thread of the gate that the beacon belongs to.

8.1.2 Identifying the Waiting Costumer

The thread that is assigned to an object uses the distances, that are passed to it from the web service, to send the information of the closest user to the object. This thread uses a simple principle to assign detect the closest user: The first user that has 3 times complied with the following two conditions is labeled closest to the object. The conditions are: 1.) the user's

phone has measured the beacon of the object closest and 2.) the calculated distance from the user's phone to the beacon is smaller than 1.2 times the values for previously smallest distance. As a BLE measurement takes about 300 ms on a modern phone, a the closest phone is detected in about 1 second. Other values for the constants have been tried, but these values give the best trade-off between speed and accuracy.

If the user has been detected to be waiting for an object, it cannot be standing at another object. The open connection that has been created when the user started sending data is used to send the data of the object to the user. The complete algorithm for detecting the closest object is given in Algorithm 2. This algorithm ensures short waiting times of less than one second, but also makes sure that variations in signal strength do not cause the wrong person to be detected closest to an object.

Algorithm 2 This algorithm determines which of the users in the area is standing closest to an object.

```
1:  procedure Closest User Determination Procedure
2:      WHEN Object does request c to find closes user.
3:      c.smallestDistance ← Float.MAX_VALUE
4:      c.counts[i] ← Number of times phone i has close measurement
5:
6:      WHEN beacon closest to phone i belongs to c
7:      beacon ← beacon closest to user
8:      if beacon.distance ≤ 1 && beacon.distance ≤ 1.2 * c.smallest  then
9:          if beacon.distance ≤ 0.5 * c.smallest then
10:             c.smallest ← 0.5 * c.smallest
11:         else
12:             c.smallest ← beacon.distance
13:         c.count[i] + +
14:         if c.count[i] == 3 then
15:             Push object information to person i
16:             Exclude i from other objects for time period
```

8.2 Testing Closest User Detection System

In order to evaluate the robustness of the Closest User Detection system a thorough experiment has been designed. Because the maximum distance between a phone and an object that is evaluated is 1 meter and we assume that there is more than 1 meter between each object, testing for more than 2 objects is not necessary. With 2 objects 4 situations can occur:

1. One object wants to know which user is closest and one person has the application open.
2. One object wants to know which user is closest and two persons have the application open.
3. Two objects want to know which user is closest and one person has the application open.
4. Two objects want to know which user is closest and two persons have the application open.

Situation	Correct assignments	Incorrect assignments
1	100	0
2	98	2
3	91	9
4	98	2
total	387	13

Table 8.1: A table containing the number of correct and incorrect assignments per tested situation.

As it takes only one second for the system to determine the user that is stating closest to an object, situation 1 is most likely to happen.

To simulate different environments each of the setups is tested 20 times on 5 different beacon locations. We have put the beacons i.) in a normal room, ii.) in a room where the beacons are hung on the wall, iii.) in a bigger room where people are walking in and out, iv.) in a paper box and v.) behind a glass wall to simulate a situation where the beacons are covered.

We paid equal attention to the corner case situations as to the likely situation to make sure that the system is robust. Table 8.1 shows the results of the measurements per situation. An incorrect assignment means that a user standing in front of the object but is not detected at the object, a user is standing closest the object but is not the user that is reported closest by the system or a user that is not standing in front of the object but the system reports that the user is closest to the object.

From the results we can conclude that our system is robust. It most often happens that a user that is not closest to the object, is detected closest to the object. This can be prevented by reducing the maximum distance for which the users are taken into account in the closest detection process. However, if this distance is reduced to much, the number of objects that do not get a closest user might increase. In the most likely scenario, scenario 1, the algorithm never failed during the experiments. This means that the closest user detection system system could work really well in practice. During experiments with the covered beacons we did not see a significant reduction in the number of passing tests. This means that the method works even in difficult circumstances.

9. Software Quality and Development

As the goal of the prototype app is to integrate it in bunq's production application we need to make sure that all functionalities work as expected. For example, the user should be able to see a map without any glitches. The localization algorithms have to return the best estimates based on the measurements. Therefore, the application has to be thoroughly tested to minimize the risk of errors or unexpected results. In this section we discuss how the application has been tested before the experiments were conducted and how we ensured a high code quality.

9.1 Testing

To avoid suprises during the experiments, all code was tested with unit tests. As our application has been written in Java, we choose JUnit [39] as the used testing framework. Each class, except a few GUI-only classes, has its own test class. Because part of our classes depend on the Android API for getting sensor readings, it was not possible to create valid instances of each class that the tested classes depend on. In order to test functionality that depends on other classes, Mockito was used. Mockito is a mocking framework with which a stub of the class can be inserted in an object instead of a real instance of that class [10]. An even more powerful tool, called PowerMock [35] is used to mock static classes such as Android's `Log` class. With this framework we could test classes that depend on the Android API or that have other dependencies.

The Espresso framework [12] and the UI Automator framework [13] are used to test the interactions of the user with the GUI. As it is important for the user interactions to work properly, all basic interactions are tested. In addition to the basic interactions, we also tested for exceptional interactions, such as pressing a button twice in fast succession.

We used the Jacoco code coverage plugin [15] which combines both the coverage of the unit tests and the application tests in a single coverage report. This enabled us to easily view the line and branch coverage of the code and find the classes that needed further testing. With this coverage tool we made sure that the branch coverage of our code was always above 85%.

9.2 Code Quality Assurance

Besides a testing framework and a code coverage tool, other tools were used to assure a high code quality. CheckStyle [7] was used to make sure that our code layout was consistent. With FindBugs [22] and PMD [59] we tracked down bugs that could be found with static analysis. To maintain the SOLID principles [55] we used the static analysis tool InCode [28]. With this tool we could detect design flaws that are not in accordance with the design principles.

Our code was also evaluated by SIG. They look at how easy it is to maintain and expand the code. Their first feedback can be found in Appendix C. We addressed their feedback by breaking up the complex methods. Furthermore, we have used libraries for the low level functionality such as writing a file. Last, we have reviewed the long methods in our code base and split those into smaller methods if that were possible.

To keep each other sharp all code was peer reviewed before a merge request was accepted. With peer reviewing we ensure consistent layout of the code. Furthermore, we were able to

track down a number of bugs that were overlooked and refactor certain classes to a better implementation. Most important, it allowed us to give each other tips on coding style and suggest improvements to each other. This allowed both of us to become better programmers.

9.3 Software development process

Besides tools to review the code quality, the quality of the code also increases if a clear software development process is used. During the project, Git is used as version control system. As repository manager we use GitLab as this is internally used by bunq. We used separate repositories for the Android and server software.

To keep the repository clear, a new branch has been created for every feature. After completing a feature, the project member who wrote the code has the responsibility that the code is tested and complies with code style guidelines. With a pull request the code is merged into the master. This ensures that the other project member checks the code and if needed made a remarks the writer on improvements. With this we ensured that the master branch always has a working version of the product.

10. Future work

There are a number of challenges that need to be overcome by bunq's developers before the localization methods and the closest user identification system can be used in practice. Currently, localization methods, like ours, drain the battery so fast that they cannot run as a background service. Smart solutions are needed to reduce the battery drain of localization methods. Both localization methods and the closest user identification system are currently working from an Android `Activity`. When the battery drain of the methods is reduced, the methods have to be extracted from the `Activity` and put in a `Service` in order to run in the background.

The biggest drawback of our application is that it is not able to give more than room level accuracy. The accuracy is mainly limitted because the accuracy of the used sensors is limited. Research is needed into accurate sensors that can be used for localization and that are integrated in the current smartphones. As the sensor technologies get more advanced the localization accuracy will increase.

The iterative trilateration algorithm can be improved by designing a system that can better estimate the distance to each of the beacons. An example of a method that might improve the distance estimate is Bayesian filtering.

The accuracy of the particle filter can be improved by implementing a motion model that mimics the user's movements better. Research is needed into step detection to reduce the delay, of currently 1 second, before the first step is detected. This ensures that the motion model responds faster to the user's movements and therefore reproducing the movements more accurate. The heading detector can be improved by algorithms or sensors that reduce the influence of the phone's surroundings on the magnetometer measurements. A drawback of the heading detector is that it assumes that the user is holding the phone in his hand in portrait mode. The motion model should be extended so that it can also accurately detects the user's movements when the phone is in the user's bag or pocket.

Before the Closest User Detection system can be implemented in, for example, concert hall gates on large scale, a number of things need to be done. First, a pilot is needed, where the performance of the system is evaluated in a real life context. Because there are a lot of external dependencies that such system relies on, this could not be realized during the project.

Second, the security of the used BLE beacons has to be evaluated. Measures should be taken to prevent spoofing of the beacons, as this would trick people into receiving messages from objects that are not in range. A possible solution for this called Ephemeral ID (EID) has recently been introduced in the Eddystone protocol [27]. With EID the identifier of a beacon changes at over time and only the owner of the beacons can verify the identifier. This ensures that the beacons can only be used by their rightful owner.

11. Conclusion

bunq wanted us to create an application that is able to locate the user's phone indoors. To find the method that suits bunq's needs best, research was done into the different indoor localization methods. During this research we found out that BLE beacons suit bunq's needs well, as they are cheap and easy to deploy. Therefore, we used BLE beacons to estimate the location of the user's phone.

The first approach to indoor localization using BLE beacons was with a method called iterative trilateration. This method calculates the location of the phone based on the estimated distance to each beacon in range. Experiments showed that this method was not going to be accurate enough, with an average accuracy of 4.5 meters.

Therefore, we changed our methodology from trilateration to a particle filter. The particle filter estimates the location of the user based on a set of participles that mimic the movements of the user in a map of the building that the user is in. The estimated location of the phone is there where the density of particles is highest. The robustness of the particle filter was enhance by using BLE beacons to speed-up the localization process.

Experiments point out that all features of the particle filter work as expected, making the particle filter a robust room-level accurate indoor localization method.

In the second part of the project, we have used the knowledge gained when developing the localization application to implement a system that is able detect which user is standing closest to an object. In this system each object has been equipped with a BLE beacon. We use an algorithm that prevents that a user that is not standing close to an object is detected close to an object due to inaccuracies in the distance estimate of the phone. Our experiments, which were conducted in different environments, showed that the system is robust and works well even in scenario's with multiple phones and objects.

In short, we implemented two methods that are able to perform indoor localization: iterative trilateration and a particle filter. Of these methods, the particle filter is the most robust method. Furthermore, we have implemented a system that is able to detect which user is standing close to an object, hereby increasing the speed at which users can enter a concert hall. Experiments showed that this method is accurate and robust.

bunq can use the localization methods and the closest user detection system to expand their production app's location-aware functionalities to make the banking experience even better.

Bibliography

[1] Acuitibrands. *ByteLight Services: Indoor Positioning*. URL: http://www.acuitybrands.com/solutions/services/bytelight-services-indoor-positioning (visited on 04/21/2016).

[2] Piyush Agrawal and Neal Patwari. "Correlated link shadow fading in multi-hop wireless networks". In: *Wireless Communications, IEEE Transactions on* 8.8 (2009), pp. 4024–4036.

[3] Android. *Bluetooth Low Energy*. n.d. URL: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html (visited on 04/21/2016).

[4] Apple. *iBeacon*. 2016. URL: https://developer.apple.com/ibeacon/ (visited on 04/25/2016).

[5] Paramvir Bahl and Venkata N Padmanabhan. "RADAR: An in-building RF-based user location and tracking system". In: *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 2. Ieee. 2000, pp. 775–784.

[6] Bunq. *Bunq*. 2016. URL: www.bunq.com (visited on 04/20/2016).

[7] *Checkstyle*. 2016. URL: http://checkstyle.sourceforge.net (visited on 05/19/2016).

[8] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N Padmanabhan. "Indoor localization without the pain". In: *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM. 2010, pp. 173–184.

[9] Brian P Crow et al. "IEEE 802.11 wireless local area networks". In: *Communications Magazine, IEEE* 35.9 (1997), pp. 116–126.

[10] *D820 Sprint Black*. URL: http://mockito.org (visited on 05/19/2016).

[11] Gabriel Deak, Kevin Curran, and Joan Condell. "A survey of active and passive indoor localisation systems". In: *Computer Communications* 35.16 (2012), pp. 1939–1954.

[12] Android Developers. *Testing UI for a Single App*. n.d. URL: https://developer.android.com/training/testing/ui-testing/espresso-testing.html (visited on 05/28/2016).

[13] Android Developers. *Testing UI for Multiple Apps*. n.d. URL: https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html (visited on 05/28/2016).

[14] Goran M Djuknic and Robert E Richton. "Geolocation and assisted GPS". In: *Computer* 34.2 (2001), pp. 123–125.

[15] EclEmma. *JaCoCo Java Code Coverage Library*. 2016. URL: http://eclemma.org/jacoco/ (visited on 05/28/2016).

[16] Leading Edge and G Jobs. "Centimeter-accuracy indoor navigation using GPS-like pseudolites". In: (2001).

[17] Rikard Eriksson and Vlad Badea. "Indoor navigation with pseudolites (fake GPS sat.)" In: (2005).

[18] Estimote. *Estimote Indoor Location*. n.d. URL: http://estimote.com/indoor/ (visited on 04/21/2016).

[19] Estimote. *Estimote Real-world context for your apps*. n.d. URL: http://estimote.com (visited on 04/21/2016).

[20] Estimote. *Nearables protcol*. 2016. URL: http://developer.estimote.com/nearables/ (visited on 04/26/2016).

[21] Viacheslav Filonenko, Charlie Cullen, and James D Carswell. "Asynchronous ultrasonic trilateration for indoor positioning of mobile phones". In: *Web and Wireless Geographical Information Systems*. Springer, 2012, pp. 33–46.

[22] *FindBugs*. 2015. URL: http://findbugs.sourceforge.net (visited on 05/19/2016).

[23] Kostas Fouskas et al. "On the potential use of mobile positioning technologies in indoor environments". In: *BLED 2002 Proceedings* (2002), p. 33.

[24] Ravi Garg, AvinashL Varna, and Min Wu. "An efficient gradient descent approach to secure localization in resource constrained wireless sensor networks". In: *Information Forensics and Security, IEEE Transactions on* 7.2 (2012), pp. 717–730.

[25] Tengqingqing Ge. "Indoor Positioning System based on Bluetooth Low Energy for Blind or Visually Impaired Users: Running on a smartphone". In: (2015).

[26] Google. *Eddystone*. 2016. URL: https://github.com/google/eddystone (visited on 04/25/2016).

[27] Google. *Eddystone eid*. 2016. URL: https://developers.google.com/beacons/eddystone-eid (visited on 04/29/2016).

[28] Christian Grobmeier. *INTOOITUS AND THEIR INCODE*. 2013. URL: https://www.grobmeier.de/intooitus-incode-20092013.html (visited on 05/28/2016).

[29] Andreas Haeberlen et al. "Practical robust localization over large-scale 802.11 wireless networks". In: *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM. 2004, pp. 70–84.

[30] Joe Hanson. *What is HTTP Long Polling?* 2014. URL: https://www.pubnub.com/blog/2014-12-01-http-long-polling/ (visited on 06/02/2016).

[31] Jeffrey Hightower and Gaetano Borriello. "Location sensing techniques". In: *IEEE Computer* 34.8 (2001), pp. 57–66.

[32] Jeroen D Hol, Thomas B Schon, and Fredrik Gustafsson. "On resampling algorithms for particle filters". In: *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*. IEEE. 2006, pp. 79–82.

[33] AKMM Hossain, Hien Nguyen Van, and Wee-Seng Soh. "Fingerprint-Based location estimation with virtual access points". In: *Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on*. IEEE. 2008, pp. 1–6.

[34] HTC. *HTC One*. 2016. URL: http://www.htc.com/nl/smartphones/htc-one-m7/ (visited on 05/19/2016).

[35] *jayway/powermock*. n.d. URL: https://github.com/jayway/powermock (visited on 06/09/2016).

[36] John. *Find X location using 3 known (X,Y) location using trilateration*. 2014. URL: http://math.stackexchange.com/questions/884807/find-x-location-using-3-known-x-y-location-using-trilateration (visited on 04/29/2016).

[37] Damien B Jourdan et al. “Monte Carlo localization in dense multipath environments using UWB ranging”. In: *Ultra-Wideband, 2005. ICU 2005. 2005 IEEE International Conference on*. IEEE. 2005, pp. 314–319.

[38] J Jung, D Kang, and C Bae. “Distance estimation of smart device using Bluetooth”. In: *Personal Computing Platform Research Team* (2013), pp. 13–18.

[39] JUnit. *JUnit*. URL: `http://junit.org/junit4/` (visited on 05/19/2016).

[40] Wonho Kang and Youngnam Han. “SmartPDR: smartphone-based pedestrian dead reckoning for indoor localization”. In: *Sensors Journal, IEEE* 15.5 (2015), pp. 2906–2916.

[41] Manikanta Kotaru et al. “Spotfi: Decimeter level localization using wifi”. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM. 2015, pp. 269–282.

[42] Ye-Sheng Kuo et al. “Luxapose: Indoor positioning with mobile phones and visible light”. In: *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM. 2014, pp. 447–458.

[43] Erin-Ee-Lin Lau and Wan-Young Chung. “Enhanced RSSI-based real-time user location tracking system for indoor and outdoor environments”. In: *Convergence Information Technology, 2007. International Conference on*. IEEE. 2007, pp. 1213–1218.

[44] *Leaflet.draw*. URL: `https://github.com/Leaflet/Leaflet.draw` (visited on 06/08/2016).

[45] LG. *D820 Sprint Black*. 2016. URL: `http://www.lg.com/us/cell-phones/lg-D820-Sprint-Black-nexus-5` (visited on 05/19/2016).

[46] Fan Li et al. “A reliable and accurate indoor localization method using phone inertial sensors”. In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM. 2012, pp. 421–430.

[47] Liqun Li et al. “Epsilon: A visible light based positioning system”. In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 2014, pp. 331–343.

[48] Peng Lin et al. “A real-time location-based services system using WiFi fingerprinting algorithm for safety risk assessment of workers in tunnels”. In: *Mathematical Problems in Engineering* 2014 (2014).

[49] Eladio Martin et al. “Precise indoor localization using smart phones”. In: *Proceedings of the international conference on Multimedia*. ACM. 2010, pp. 787–790.

[50] Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[51] Esmond Mok and Günther Retscher. “Location determination using WiFi fingerprinting versus WiFi trilateration”. In: *Journal of Location Based Services* 1.2 (2007), pp. 145–159.

[52] Matt Nedrich. *An Introduction to Gradient Descent and Linear Regression*. 2014. URL: `https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/` (visited on 06/09/2016).

[53] Radius Networks. *Altbeacon*. 2015. URL: `http://altbeacon.org` (visited on 04/25/2016).

[54] Shahriar Nirjon et al. “Coin-gps: indoor localization from direct gps receiving”. In: *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM. 2014, pp. 301–314.

[55] Samuel Oloruntoba. *S.O.L.I.D Object Orientated Design*. 2015. URL: https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (visited on 05/28/2016).

[56] Dhruv Pandya, Ravi Jain, and Emil Lupu. "Indoor location estimation using multiple wireless technologies". In: *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*. Vol. 3. IEEE. 2003, pp. 2208–2212.

[57] Pannuto. *Source code luxapose*. 2015. URL: https://github.com/lab11/vlc-localization (visited on 04/26/2016).

[58] Philips. *Philips LED indoor positioning technology at Carrefour*. May 21, 2015. URL: https://www.youtube.com/watch?v=uQw-o6bjrec (visited on 04/21/2016).

[59] *PMD - don't shoot the messenger*. URL: https://pmd.github.io (visited on 05/19/2016).

[60] Aswin N Raghavan et al. "Accurate mobile robot localization in indoor environments using bluetooth". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 4391–4396.

[61] Anshul Rai et al. "Zee: zero-effort crowdsourcing for indoor localization". In: *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM. 2012, pp. 293–304.

[62] Mohamed Er Rida et al. "Indoor location position based on Bluetooth Signal Strength". In: *Information Science and Control Engineering (ICISCE), 2015 2nd International Conference on*. IEEE. 2015, pp. 769–773.

[63] Daan Scheerens. "Practical indoor localization using Bluetooth". In: (2012).

[64] Kyle Schultz. *From Barns to Satellites: An Introduction to the Mathematics of Global Positioning Systems*. n.d. URL: http://jwilson.coe.uga.edu/EMAT6680Fa05/Schultz/6690/Barn_GPS/Barn_GPS.html (visited on 04/25/2016).

[65] Michail Vasilakis. *DynaLight: A Dynamic Visible Light Communication Link for Smartphones*. 2015. URL: http://www.es.ewi.tudelft.nl/msc-theses/2015-Vasilakis.pdf (visited on 04/26/2016).

[66] Yapeng Wang et al. "Bluetooth positioning using RSSI and triangulation methods". In: *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*. IEEE. 2013, pp. 837–842.

[67] Roy Want. "Near field communication". In: *IEEE Pervasive Computing* 3 (2011), pp. 4–7.

[68] Moustafa Youssef and Ashok Agrawala. "The Horus WLAN location determination system". In: *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM. 2005, pp. 205–218.

[69] Vasileios Zeimpekis, George M Giaglis, and George Lekakos. "A taxonomy of indoor and outdoor positioning techniques for mobile location services". In: *ACM SIGecom Exchanges* 3.4 (2002), pp. 19–27.

A. Original Project Description

Where is my phone? That's a question we would like to see answered with the highest level of accuracy. And to do so, you will only get to use BLE and a smartphone. That's it.

We are looking for ways to improve NearPay and make interacting with everyone around you even faster, easier and more battery-friendly. To do so, you design a flawless system to locate smartphones within a defined area, using BLE beacons. Then you test, optimise and repeat until you can pinpoint your phone's position down to a few centimeters. While testing, you don't forget to analyse the accuracy of different models of bluetooth transmitters in client devices. Yep, we know there are A LOT of those but who doesn't like a challenge?

Our goal is to estimate locations with sub-meter precision and to have a prototype up and running.

A.1 Possible research questions

- What is the best setup of BLE beacons to locate smartphones with a high level of accuracy?
- Would combining different technologies (think NFC, Wi-Fi) lead to higher levels of precision?
- What parameters influence localisation accuracy?

A.2 Company description

Youĺl join in on the fun of 50+ smart, dedicated people (10+ nationalities) that are on a common mission to revolutionise the financial sector. You'll be based in Amsterdam, right next to Sloterdijk station. We'll of course supply you with a laptop and a solid monthly pay. Being a tech startup, we have all the perks to match, from pingpong and free beer to epic bi-monthly team events.

A.3 Auxiliary information

Be passionate about your work and have a drive to learn. That's all we expect. We value creativity and smart solutions. In return for your hard work, we give you a personal supervisor that knows what he is talking about. Any questions? Don't hesitate to get in touch!

B. Maps of Locations used in Experiments

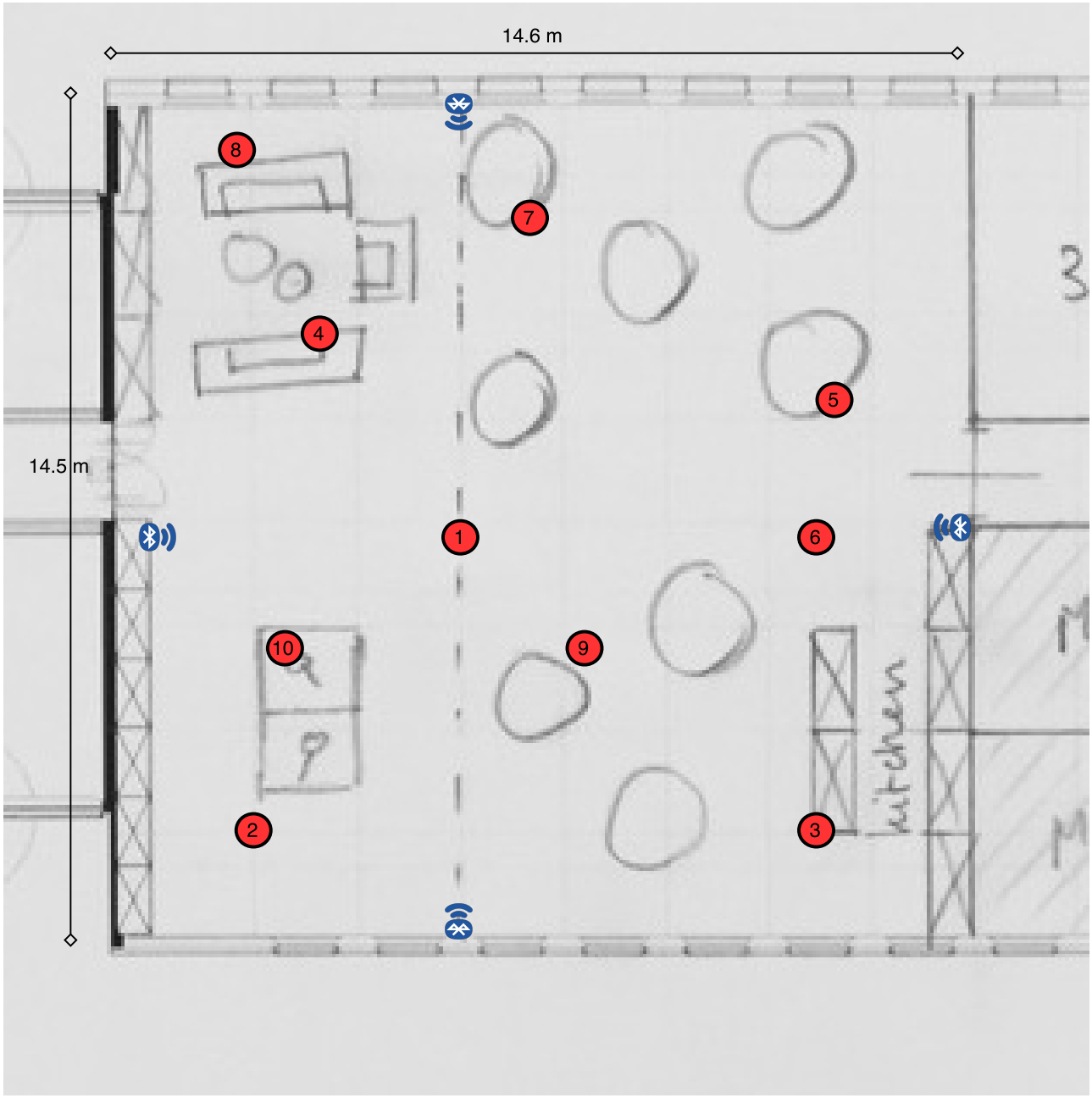


Figure B.1: Map of location 'The Club' that is used to test the trilateration algorithm

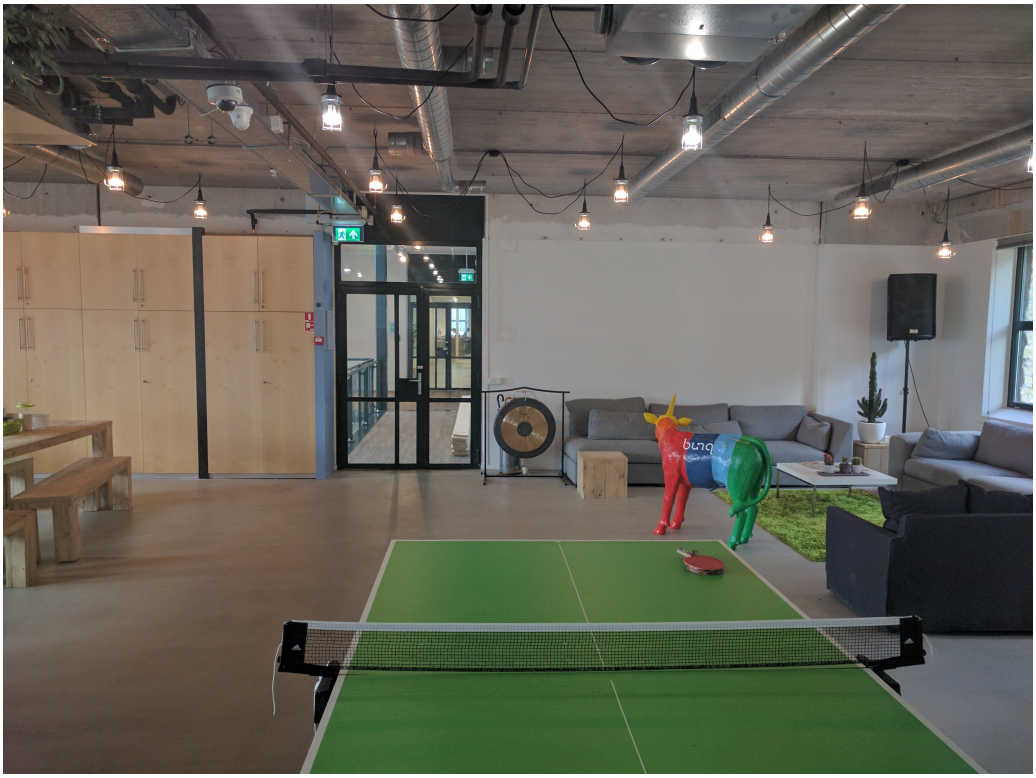


Figure B.2: Image of location ‘The Club’ that is used to test the trilateration algorithm

C. First feedback SIG

Als we alleen naar de echte code kijken scoren jullie 4,5 ster in ons model, dus dat ziet er een stuk beter uit (jullie zijn met deze score één van de best scorende groepen). Over de server hebben we dan ook weinig te melden, aangezien het hier om weinig code bestaat die over het algemeen uit kleine, duidelijke methode bestaat.

Bij de Android app zien we wél een aantal kleine punten waar nog verbetering mogelijk is. In Wall.onSegment en Wall.isCrossed staan relatief complexe boolean-condities. Dit soort condities zijn vrij moeilijk leesbaar, zeker omdat orientation1, orientation2, etc. niet de beste namen ooit zijn. Jullie hebben dat algoritme duidelijk van een website gekopieerd, maar omdat jullie hem nu moeten onderhouden is het een goed idee om hem wat leesbaarder te maken.

Daarnaast hebben jullie nog een aantal methodes die je nog zou kunnen opsplitsen, al hebben we het dan niet over methodes van 100 regels (dat kan ook niet, want dan zouden jullie niet zo hoog zitten qua score). Een voorbeeld hiervan is JsonFileWriter.writeToFile(). De makkelijkste manier om deze op te lossen is de hele implementatie vervangen door een library als Google Guava of Apache Commons IO. Dit soort low level functionaliteit zelf schrijven kan, maar leidt al snel tot relatief veel code om basale taken uit te voeren.

Uit jullie eerder genoemde test coverage rapport kunnen we zien dat jullie momenteel 86% test coverage hebben. Dat is keurig, hopelijk lukt het om dat niveau tijdens de rest van het project vast te houden. Momenteel wordt de user interface het minste met unit tests afgedekt, maar dat is vrij normaal als je naar andere systemen kijkt. Unit tests zijn vooral goed in het aftesten van logica, geautomatiseerde tests voor de user interface zijn meestal gebaseerd op end-to-end tests (voor web interfaces vaak met tools als Selenium of Protractor). Uit jullie upload kan ik niet aflezen of jullie ook al van dat soort tools inzetten, maar als je tijdens de rest van het project nog tijd hebt zou dat een mooie vervolgstap zijn om nog uit te zoeken.

D. Second feedback SIG

In de tweede upload zien we dat het codevolume flink is gegroeid, terwijl de score voor onderhoudbaarheid ongeveer gelijk is gebleven. Wel zien we een duidelijke verbetering op de gebieden die in de feedback op de eerste upload, en de mailwisseling daarna, als suggesties voor verbetering werden genoemd. Die voorbeelden zijn inmiddels weggewerkt, en we zien dan ook een verbetering op de deelscores voor Unit Size en Unit Complexity. Die verbetering is echter net niet genoeg om de totaalscore naar de 5 sterren te laten stijgen.

Daarnaast hebben jullie naast veel nieuwe code ook veel nieuwe tests toegevoegd, waardoor de verhouding tussen test- en productiecode er net als bij de eerste upload goed uitziet.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject.

E. Project Infosheet

Title: Indoor Location Estimation
Client: bunq
Date: June 17, 2016

Description

bunq is an innovative banking company that competes with other banks by providing the best services to their costumers by making banking easier. bunq's NearPay feature could be improved if bunq's application would be able to more accurately localize users indoors. During this project, an application was developed that locates the user indoors using BLE beacons. Furthermore, a system was designed that is able to detect when a user is close to an object. Wouldn't it be great to get a link to trailer of a movie pushed to your phone when you are looking at a billboard about it? The methods used in both systems can be integrated in bunq's application in the future to improve.

Challenges: As we did not have any experience with indoor localization, a lot of research had to be done in order to design a robust indoor localization method.

Research: The first two weeks were dedicated to research. When we decided to change the method that we were going to use to locate the user, more research was required. At the end of the project we learned about indoor localization and how to deal with inaccurate sensor measurements.

Process: We have used three git repositories: one for the Android application, one for the server's code and one for the report. The report has been written in LaTeX. At the end of each day we send a status report to our client supervisor, Wessel Van. Each Friday we held a meeting with our TU Delft Supervisor, Marco Zúñiga, where we showed our progress.

Products: We developed a web service and an Android application. The android application is used for two things: To locate the user in a building and to send the beacons in range to the server. The web service is used to detect which user is close to which object. The Android application is extensively tested using JUnit and Mockito. The server has been tested with Golang's build-in testing framework and postman.

Members

Sven Boor sven.boor@gmail.com
interests: embedded systems, scaling software, transport, aerospace
responsibilities: the server-side of both applications, the motion model for the particle filter.
Matthijs den Toom m.denToom@student.tudelft.nl
interests: Embedded software, Robotics, Internet security
responsibilities: the trilateration algorithm, the particle filter, except for the motion model, the client side of the closest object detection system.

Both team members contributed to the general android code, the report and the final presentation.

Client Supervisor: Wessel Van - Lead Android developer at bunq - wessel@bunq.com
TU Delft Coach: Marco Zúñiga - EEMCS - Embedded Software - M.A.ZunigaZamalloa@tudelft.nl
The final report of this project can be found at: `http://repository.tudelft.nl`