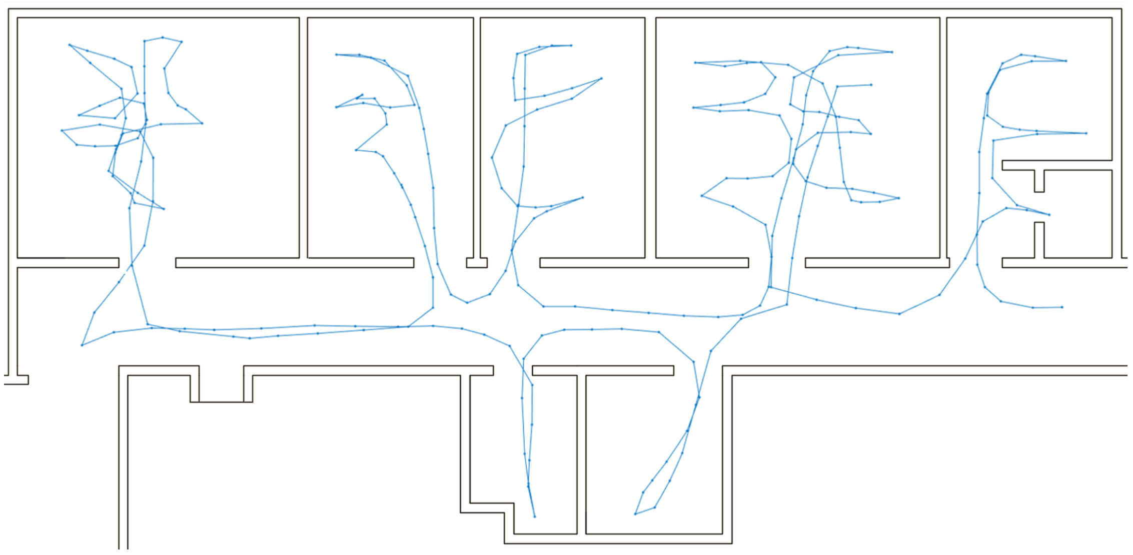
CHALMERS
UNIVERSITY OF TECHNOLOGY



Localisation and Mapping of Received Signal Strength in a Hospital Environment

Master's thesis in Biomedical Engineering and Systems, Control and Mechatronics

ALBIN HJALMARSSON
HENRIK JOHANSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:EX003

Localisation and Mapping of Received Signal Strength in a Hospital Environment

ALBIN HJALMARSSON
HENRIK JOHANSSON

Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

Localisation and Mapping of Received Signal Strength
in a Hospital Environment
ALBIN HJALMARSSON
HENRIK JOHANSSON

Examiner: Prof. Lennart Svensson, Department of Signals and Systems

Master's Thesis 2017:EX003
Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A map of the test environment with an estimated trajectory.

Localisation and Mapping of Received Signal Strength in a Hospital Environment.
ALBIN HJALMARSSON
HENRIK JOHANSSON
Department of Signals and Systems
Chalmers University of Technology

Abstract

Indoor positioning systems can be based on observed Bluetooth signals, but due to the challenges in modelling the received signal strength, using signal strength maps can be a more reliable approach. This thesis proposes a method for automatic mapping of Bluetooth low energy signals. The signal strength maps are intended to be useful for indoor positioning in a healthcare environment. The system consists of Bluetooth low energy beacons and a user who walks around the environment with a handheld mobile smartphone. A method to detect steps is developed using the accelerometer data from the smartphone. Acceleration and inertial measurements with the magnetometer readings is used in an extended Kalman filter to estimate the heading direction. This information is used in a Rao-Blackwellized particle filter along with knowledge of wall layout and beacon positions to estimate the user trajectory. The trajectory and collected Bluetooth signals creates a signal strength map. It is concluded in this thesis that the estimation of the trajectory works well overall. The method correctly estimates the trajectory through the correct rooms. A mean positioning error of down to 1.55 meters is achieved when comparing the constructed maps to manual signal measurements in 17 static positions, a good result considering related work and the amount of data used for testing.

Keywords: Simultaneous Localisation and Mapping, Bayesian state estimation, Indoor positioning, Healthcare, Bluetooth low energy

Acknowledgements

The authors would like thank all of those who have been supportive in the creation of this thesis. First, we would like to thank the people at Trueflow (Tomas, Jonathan & Martin), who have provided us with support, equipment and above all, good company at the office. We would also like to thank the fantastic people at Östra Hospital, especially ward 351, for showing great interest in our work and for providing us with a good test location. Lastly, we would like to give an extra thank to our examiner, Professor Lennart Svensson for the motivation and support he has given us, and interest he has shown in our project.

Albin Hjalmarsson and Henrik Johansson
Gothenburg, January 2017

Contents

List of Figures

List of Tables

List of Acronyms

BLE	Bluetooth Low Energy.
EKF	Extended Kalman Filter.
ENU	East-North-Up Coordinate System.
FIR	Finite Impulse Response filter.
GPS	Global Positioning System.
IPS	Indoor Positioning System.
LTE	Long-Term Evolution - a high speed wireless communications standard for mobile phones.
ME	Mean Error.
PDR	Pedestrian Dead Reckoning.
PF	Particle Filter.
RE	Room Error.
RSS	Received Signal Strength.
SLAM	Simultaneous Localisation And Mapping.

1
Introduction

This is a Master's Thesis created during the latter half of 2016 and the beginning of 2017 by the authors, Albin and Henrik, at Chalmers University of Technology in Gothenburg, Sweden. The work has been done in cooperation with *Trueflow*, a small company located in Gothenburg working with smartphone applications using indoor positioning. The topic of this thesis is indoor positioning, specifically in a healthcare environment. It describes a method of automatically creating a signal strength map, using known Bluetooth signal transmitting positions and on-board sensors from a modern mobile smartphone.

This chapter first describes why this topic is interesting, and moves on by explaining different solutions and earlier work done by others on the topic. Finally, the purpose of the thesis is clarified, and a solution overview is presented.

1.1 Background

The number of applications which use mobile devices' positions has increased during recent years. Outdoor positioning is possible because many of today's mobile devices has an integrated Global Positioning System (GPS) which can give a good estimate of the position. However, the GPS signal indoors is too weak to be able to give a good estimate of the position. Here another system is needed which can give us the position of the device. Such system is called an indoor positioning system (IPS) and there are many different techniques to obtain an indoor position estimate.

1.1.1 IPS in Healthcare Settings

As within many other areas, an indoor positioning system can introduce possibilities for a hospital environment. It could be used to locate personnel and patients as well as medical equipment, optimising the work flow and potentially increasing the quality of healthcare. For example, patients with severe dementia could move more freely within the hospital without the need to be locked away [3]. An IPS can be utilised in a nurse calling system, only calling the nearest nurses available.

Furthermore, an IPS could enable fast localisation of medical equipment, decreasing the time searching for said equipment as well as reducing the need for equipment redundancy [4].

As many of the potential benefits of the technology include that individuals should be tracked, it may introduce ethical issues as well. As for example described by Tucker and Spear [4], an IPS can be perceived as a 'big brother' constantly watching an individual's movements and tasks. Tucker and Spear also describe a hospital scenario in London 2001, where an implementation of an IPS failed due to the staff refusing to use the system.

The performance need for an IPS will vary depending on the environment and its use. Van Haute et al. [3] describes different evaluation metrics that are especially important for an IPS in a hospital environment:

- *room accuracy*: the ability to position the device in the correct room
- *latency*: the time passed from when the position of a device is requested until delivered
- *installation time/cost*
- *energy consumption*

Andrén [5] says the hospital environment in Sweden is going towards one patient per room when building new hospitals and renovating old ones. To minimise transports of patients and nurses the rooms shall be in clusters with the nurses in the centre. The traditional hospital has long corridors with patient's room along the corridor, often with more than one patient per room. This means that indoor navigation systems for this environment should be able to work in locations with many small rooms and long corridors.

A common device to use in indoor positioning systems is a mobile smartphone. Sahlgrenska University Hospital [6] says that mobile telephones are allowed at the hospitals in general. However, they are not allowed in some places where it can interfere with sensitive equipment. Sahlgrenska University Hospital consist of Sahlgrenska Hospital, Östra Hospital and Mölndal Hospital [7].

1.1.2 IPS Solutions

One possible way to position something is to use previous positions and estimate its trajectory. This can be done with for instance pedestrian dead reckoning (PDR) where the device calculates its position based on a previous position and motion data (with for example accelerometers and magnetometers). One problem with PDR is that due to the errors in the motion data (sensor noise) the positioning error will potentially grow over time if no absolute position measurement is available.

To obtain an absolute position measurement indoors a variety of techniques can be used. That may for example include proximity detection or triangulation via estimated distances to known points. The former is based upon the device being located with some certainty within the range of a transmitter that is at a known location. By using several transmitters, a rough position of the device can be obtained throughout the area of interest [3].

The second group of techniques for IPS rely on some way to estimate distances to known points (i.e. transmitters). The common distance based IPSs use either time of arrival or received signal strength (RSS) to calculate a distance to the transmitter node. This RSS approach works well in theory, if the signal path is clear. However a real environment is inhomogeneous, meanings walls and objects will attenuate the signal and negatively affect the position estimation process [8].

A popular approach to solve inhomogeneous location problems using RSS is to instead create signal strength maps. The maps contain information regarding the signal strengths received throughout different positions in the area of interest, also called a *fingerprint*. Fingerprinting is divided into two stages, training and positioning. During training measurements are collected using a known position (also known as a ground truth position) to create a signal strength map. During the positioning stage the measurements are used to estimate a position on the previously created map. The signals used when fingerprinting can be any signal with a spatial dependency, for example WiFi signals [9].

Fingerprinting can be a good solution to indoor positioning, but it is not without drawbacks. A problem with fingerprinting is temporal differences in fingerprints when humans deflect signals or when other objects are moved and thus change the signal map. The orientation of the device when creating the map is also something that could affect the fingerprint due to signals being deflected and attenuated by the body of the user [9, 10]. Another important practical drawback is the often time consuming training stage, which require precise ground truth locations throughout the whole process.

A much faster way of creating fingerprints is to collect signal strength data without actually knowing the true position, or pose, of the device. This has been a major research area in the last decades, and a popular approach is called simultaneous localisation and mapping (SLAM). During SLAM the device is moved throughout the environment collecting signal information. Meanwhile, motion sensor data is collected. All the data is used together with probability models for device movement. These models could incorporate for example a blueprint of the area, proximity to certain locations during certain time steps, and pedestrian dead reckoning. The SLAM algorithm then optimises the variables, which are often positions during data collection and signal strength maps, such that they best fit the collected data and provided motion models [11].

There are several ways to perform SLAM. Grisetti et al. [11] classifies them as either on-line SLAM or full SLAM. On-line SLAM, or filter approaches, estimate the variables iteratively after each time step, using only previously data. Full SLAM or smoothing approaches, on the other hand, uses the complete data set to estimate all variables.

1.2 Related Work

Faragher and Harle [10] has analysed the usability of Bluetooth low energy (BLE) in an IPS. Amongst other conclusions they describe how BLE is susceptible to fast fading, causing large RSS fluctuations. They suggest methods that mitigates these characteristics, including batch filtering of multiple measurements while in movement. Furthermore, they discover that positioning accuracy increases with the number of reachable BLE beacons, up to approximately 6-8 beacons. In their later work [12], they use fingerprinting (with a very accurate IPS to provide ground truth) of BLE-RSS for indoor positioning with promising results. They achieve a mean error of approximately 1 meter.

Ferris et al. [8] proposed a SLAM method called Gaussian Process Latent Variable Models. It stems from the use of Gaussian processes to describe the signal strength map. They treat the unknown variables, which are often positions during data collection and signal strength maps, as *latent* (i.e. hidden). By introducing amongst other a motion model, they maximise the likelihood of all the measurements given all known data and models with regards to the latent variables. This algorithm is used by others, but its complexity has a cubic dependence on the number of time steps [13].

Another way of solving the SLAM problem is using a graph representation, called GraphSLAM [11]. In the graph, each node represents a pose at a time step of the device, and edges represent constraints between the poses. The constraints stem from measurements and models. The process involves optimising the graph, thus finding the set of poses and maps that best matches the measurements. This is commonly done when a loop is closed, i.e. when the device visits the same spot twice. The algorithm offers an intuitive representation of the problem, and potentially fast computation times even for relatively large problems.

Mirowski et al. [13] has implemented a SLAM algorithm they call *SignalSLAM*. This is a SLAM algorithm which uses BLE, WiFi, wireless mobile communication (LTE) and magnetic signals to build a signal strength map. To generate a map they use GraphSLAM, explained in [11], which uses the motion of the smartphone and landmarks with known positions. When in proximity of a landmark or near a position it has visited previously it updates the map or "closes the loop". Their approach is to let people move around in the building which has several Bluetooth and WiFi access points. By using GraphSLAM and dead reckoning they build a map of the location. The dead reckoning algorithm is invariant of the smartphone orientation and they claim they map the environment "from the pocket" with limited or no user input. They use the landmarks previously stated to calibrate the dead reckoning. One of the conclusions with the experiment is that different models of smartphones have better or worse sensors. Mirowski et al. state that their map works, but it had been better if it was used with more smartphones over several days.

1.3 Purpose of Thesis

The main goal of the method presented in this thesis is to have a signal strength map created in an automatic manner, which in itself is useful for indoor positioning. The signal strength map shall describe signals from Bluetooth low energy beacons, which will be run on a low broadcast rate and power in order for longevity of the beacons. The map creation is to be performed using an off-the-shelf smartphone, and its on-board sensors, by having a user holding the device while performing walks in the area of interest.

This thesis proposes a method for said map creation and shows its feasibility. The method involves orientation estimation and step detection, or odometry, using on-board sensors of the smartphone. The odometry is used in a modified particle filter, which also uses a known blueprint layout of the area, and knowledge of beacon positions to estimate the recording positions of the Bluetooth data and thus creating a map.

1.4 Delimitations

The thesis does not investigate the uses and possible issues of IPSs in a healthcare environment, other than that covered in the introduction, and will focus on the technical aspects of the problem. The method proposed uses a known layout and known beacon positions, and thus does not cover mapping of completely unknown environments.

While describing and testing the method, very little regard is dedicated to matters of a live implementation. For many particle filter applications, automatic initialisation and re-initialisation can be important, but these are not covered in this thesis. Very little regard to computational complexity has been shown, and the methods have only been tested in an offline environment. This fact is exploited to some extent in the methods, meaning that some changes have to be made for all methods to be run online.

Only one specific mobile device is used for testing the method. Different mobile smartphones may possess different sensor characteristics. For instance, it has been noted that different devices may observe different signal strengths in similar situations, a problem which is not investigated in this thesis.

1.5 Solution Overview

This section briefly explains the process used in the proposed method. First a user walks around with the handheld smartphone in the area of interest, which is fitted with Bluetooth low energy beacons. The beacons are set to a low broadcast rate, in order for the beacons to have a long lifespan. While the user is walking the Bluetooth signals are collected, along with measurements from the phone's accelerometer, gyrometer, and magnetometer. The latter signals are then used for orientation estimation and step detection.

The orientation (or rotation) estimation is performed with the help of an extended Kalman filter, and uses accelerometer, gyrometer, and magnetometer data. Step detection is achieved through analysing the accelerometer data, thanks to certain patterns that occur when the user holds the device in its hand while walking. Together the orientation and step detection is called odometry.

The odometry is then input to a modified particle filter. The aim of the particle filter is to estimate the position where each detected foot step took place, also called recording positions. This is performed while simultaneously creating a map describing received Bluetooth signal strength at different locations. The estimated trajectory is improved by the use of a particle smoother. The algorithm also makes use of a blueprint matching algorithm which can be used to tell if a transition between two points is possible given the known layout of the area. This further increases the performance by removing infeasible trajectories.

1.6 Thesis Layout

The thesis is split up into five parts after the introduction. First is the theory chapter, describing important background theory that has been obtained from other related work. After that the method is explained, which is mostly developed for this thesis. Then the result chapter follows, showing odometry accuracy, recording position estimation accuracy, and signal strength map performance. This is all followed by a discussion and conclusion.

2

Theory

This chapter contains much background theory that is the foundation of a lot of the methods used in this thesis. It first describes different ways of representing a rotation, which is an important part of the orientation estimation. After that Bayesian state estimation is introduced, along with a few different useful methods, which is the foundation of many parts of this thesis. Following, a description of simultaneous localisation and mapping is presented. After that information regarding the Bluetooth beacons is given, followed by some theoretical background regarding step detection and blueprint matching.

2.1 Rotation Representations

The orientation of a coordinate system, or a rigid body, must be expressed with respect to a reference frame or coordinate system. One system used to the describe the coordinate system used in the real world, called the world frame, is the East-North-Up Coordinate System (ENU) [14]. In this system, the x-axis points to the east, y-axis to the north and z-axis upwards in the opposite direction of the gravity.

The following text is based on B. Siciliano and K. Kahtib's book [15] where they describe a way to use a rotational matrix, **R**, to describe a three-dimensional coordinate system with respect to the world frame:

$$\mathbf{R} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix} \tag{2.1}$$

. **R** is a 3x3 matrix where the column vectors represent unit vectors of an orthogonal frame and thereby the column vectors are mutually orthogonal:

$$\mathbf{x}^T\mathbf{y} = 0 \qquad \mathbf{y}^T\mathbf{z} = 0 \qquad \mathbf{z}^T\mathbf{x} = 0 \tag{2.2}$$

Because the column vectors are unit vectors they also have unit norm,

$$\mathbf{x}^T\mathbf{x} = 1 \qquad \mathbf{y}^T\mathbf{y} = 1 \qquad \mathbf{z}^T\mathbf{z} = 1 \tag{2.3}$$

which means the rotation matrix $\mathbf{R}$ also is orthogonal and has unit norm,

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_3 \tag{2.4}$$

where I_3 is the 3x3 identity matrix. Because the matrix is orthogonal the inverse is equal to it's transpose:

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.5}$$

Hereby the superscript of a vector denotes the frame in which its components is expressed. The expression ${}^A\mathbf{v}$ denotes that the vector $\mathbf{v}$ has its components expressed in frame A. The same vector $\mathbf{v}$ expressed in frame B has the notation ${}^B\mathbf{v}$.

The subscript of a rotation matrix denotes which frame that is rotated and the superscript is with respect to which frame the rotation is done. For example, ${}^A_B\mathbf{R}$ denotes that the rotation matrix of frame B is rotated with respect to frame A.

When expressing the vector $\mathbf{v}$ in frame B in a different frame (A), the rotation matrix ${}^A_B\mathbf{R}$ is multiplied with the vector in frame B:

$${}^A\mathbf{v} = {}^A_B\mathbf{R}\,{}^B\mathbf{v} \tag{2.6}$$

By knowing the rotation matrix from frame A to B, ${}^A_B\mathbf{R}$, and the rotation matrix from frame B to C, ${}^B_C\mathbf{R}$, it is possible to calculate the rotation matrix from frame A to C, ${}^A_C\mathbf{R}$:

$${}^A_C\mathbf{R} = {}^A_B\mathbf{R}\,{}^B_C\mathbf{R} \tag{2.7}$$

When rotating a frame θ around the z-axis the rotated frame's unit vector components become:

$$\mathbf{x} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

This leads to the rotation matrix

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

The rotation matrix is defined in the same way when the frame is rotated around the x- and y-axles. The rotation matrix when rotating around the y-axis is

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.10}$$

and around the x-axis

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.11}$$

These simple rotations in equations (2.9), (2.10) and (2.11) used in in equation (2.7) can describe more advanced orientations of a frame in another reference frame.

2.1.1 Euler Angles

B. Siciliano and O. Khatib [15] describe another way to describe a rotation of a coordinate system, or a rigid body, in three-dimensional space relative a reference coordinate system. The system is called *Euler angles*, which uses a combination of the three rotation matrices in equations (2.9) (2.10) and (2.11). The system uses three angles, φ, θ and ψ to describe the orientation. When the rotations of two consecutive rotations are not around the same axis, it is possible to express any orientation of a frame using these angles.

The order of the sequence of rotations is divided into two categories, six sequences each. *Euler angles* is one of these categories and it is where the first and third rotation is around the same axis, for example $Z - Y - Z$:

$$\mathbf{R}_{ZXZ} = \mathbf{R}_z(\varphi)\mathbf{R}_x(\theta)\mathbf{R}_z(\psi) \tag{2.12}$$

The other category has one rotation around each of the axles and $X - Y - Z$ is one of them:

$$\mathbf{R}_{XYZ} = \mathbf{R}_x(\varphi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \tag{2.13}$$

In equation (2.12) the *Euler angle* is obtained by the *Z-X-Z* rotations where the rotation is first φ around the z-axis, then θ around the x'-axis and lastly a rotation of ψ around the z”-axis. Note that the ’ and ” notations is for the new axis after the first and second rotation respectively. This mean the first rotation is around the reference frame’s z-axis. The second is around the new frame’s x-axis. The last rotation is around the z-axis of the new frame which has already been rotated twice.

2.1.2 Quaternions

Using quaternions is another way to describe the rotation of a coordinate system or a rigid body in a three-dimensional space. A quaternion is a four-dimensional vector which has no problems with singularity, which the *Euler angles* suffer from [16]. The theory about quaternions in this section can be found in [17, 18, 19, 20, 21].

The quaternion vector is calculated as follows:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \mathbf{u}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ u_x\sin\left(\frac{\theta}{2}\right) \\ u_y\sin\left(\frac{\theta}{2}\right) \\ u_z\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.14}$$

The quaternion ${}^{A}_{B}\mathbf{q}$ uses the same notations as the rotation matrix in (2.6) and the expression denotes that frame B is rotated with respect to frame A. A way to visualise the quaternion is that the rotation θ is done around the elements of the unit vector $\mathbf{u}$; u_x, u_y and u_z [19]. If the vector $\mathbf{u}$ is a unit vector the quaternion is a unit quaternion which means it has length of 1 [15],

$$\|\mathbf{q}\| = 1 \tag{2.15}$$

and if the quaternion is a unit quaternion the conjugate of the quaternion is

$$\mathbf{q}* = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ -\mathbf{u}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.16}$$

The inverse of the unit quaternion is then the conjugate of the same,

$$\mathbf{q}^{-1} = \mathbf{q}^* \tag{2.17}$$

which implies that the conjugate of a quaternion is its inverse which changes the transformation:

$${}^B_A\mathbf{q}^* = {}^A_B\,\mathbf{q} \tag{2.18}$$

To transform a vector $\mathbf{r}$ in B frame (${}^B\mathbf{r}$) to a vector in the A frame (${}^A\mathbf{r}$) the following calculation can be used:

$${}^A\mathbf{r} = {}^A_B\,\mathbf{q}\,{}^B\mathbf{r}\,{}^A_B\mathbf{q}^* \tag{2.19}$$

and to transform a vector in A frame to the B frame:

$${}^B\mathbf{r} = {}^A_B\,\mathbf{q}^*\;{}^A\mathbf{r}\,{}^A_B\mathbf{q} \tag{2.20}$$

In some problems, it is convenient to convert the unit quaternion to a rotation matrix to do calculations and then convert it back to a unit quaternion. To transform a quaternion to a rotation matrix, $\mathbf{R}(\mathbf{q})$, the calculations in equation (2.21) is used [17]:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix} \tag{2.21}$$

Equation (2.22) transforms the rotation matrix back to unit quaternion [17]:

$$\mathbf{q}(\mathbf{R}) = \mathbf{R}(\mathbf{q})^{-1} = \begin{bmatrix} \frac{1}{2}\sqrt{R_{11} + R_{22} + R_{33} + 1} \\ (R_{23} - R_{32})/4q_0 \\ (R_{31} - R_{13})/4q_0 \\ (R_{12} - R_{21})/4q_0 \end{bmatrix} \tag{2.22}$$

2.2 Bayesian State Estimation

State estimation is a process that estimates a state s or sequence of states based on measurements z, inputs u and known relations. State estimation can be divided into three main categories [22], *filtering*, *smoothing*, and *prediction*. The theory in this section can be found in [22, 23, 24, 11, 25].

Filtering is the task of using all data collected up to time k to obtain a *posterior distribution*. It is often denoted as:

$$p(s_k|z_{1:k}, u_{1:k}) \tag{2.23}$$

Smoothing is another type of state estimation that is closely related to filtering, but also uses data after time k to estimate the state at time k. The sought distribution is

$$p(s_k|z_{1:K}, u_{1:K}) \quad \text{or} \quad p(s_k|z_{1:k+m}, u_{1:k+m}) \tag{2.24}$$

where $m > 0$. Another distribution that is important in Bayesian filtering is the *prediction*,

$$p(s_k|z_{1:k-m}, u_{1:k-m}) \tag{2.25}$$

where $m = 1$ is often used in calculations. A way to visualise the dynamic systems used in this thesis is by the Bayesian network in figure 2.1. The network shows the Markov property, meaning for instance that given the state s_{k-1}, s_k is independent of all other earlier states in the network [11].

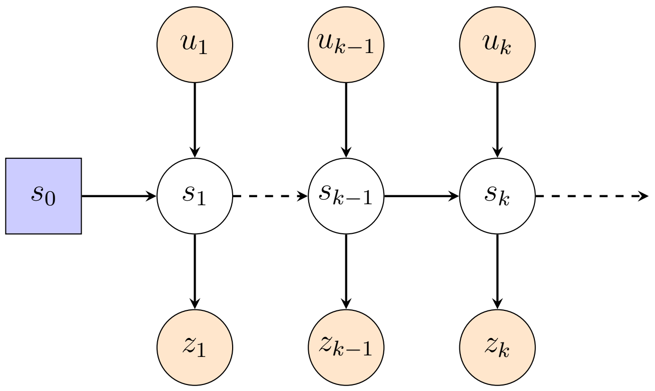


Figure 2.1: Dynamic Bayesian network. The prior s_0 can be known or unknown. The control (or input) u and z are known or measured, while the states s are unknown.

A common way to describe the evolution of these systems are:

$$s_k = f(s_{k-1}, u_k, v_{k-1}), \qquad z_k = h(s_k, u_k, r_k) \tag{2.26}$$

where s_k is the state at time k, u is some form of control or input signal, v is motion noise, z is a measurement, and r is measurement noise. f and h are commonly referred to as *motion model* and *measurement model*, and may be linear or non-linear. Not included in the model above is a possible time dependence. If the noise is additive, equation (2.26) becomes:

$$s_k = f(s_{k-1}, u_k) + v_{k-1}, \qquad z_k = h(s_k, u_k) + r_k \tag{2.27}$$

As mentioned, f and h may be linear or non-linear. In the case of both models being linear the system becomes:

$$s_k = As_{k-1} + Bu_k + v_{k-1}, \qquad z_k = Cs_k + Du_k + r_k \tag{2.28}$$

Due to the stochastic nature of equations (2.26) to (2.28), it is common and in many cases practical to rewrite them as probability density functions [22]. The motion model becomes:

$$p(s_k|s_{k-1}, u_k) \tag{2.29}$$

and the measurement model becomes:

$$p(z_k|s_k) \tag{2.30}$$

The following parts focus of filtering, and a smoothing method will be returned to later in this section. Consider Bayes' rule [26]:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{2.31}$$

Use (2.31) with the posterior distribution in equation (2.23), and use the Markov property of the system to simplify the result:

$$\begin{aligned} p(s_k|z_{1:k}, u_{1:k}) &= \frac{p(z_k|s_k, z_{1:k-1}, u_{1:k})p(s_k|z_{1:k-1}, u_{1:k})}{p(z_k|z_{1:k-1}, u_{1:k})} \\ &= \frac{p(z_k|s_k)p(s_k|z_{1:k-1}, u_{1:k})}{p(z_k|z_{1:k-1}, u_{1:k})} \\ &\propto p(z_k|s_k)p(s_k|z_{1:k-1}, u_{1:k}) \end{aligned} \tag{2.32}$$

The first term, $p(z_k|s_k)$ (measurement model), is often called the *likelihood function* in this context. The second term, $p(s_k|z_{1:k-1}, u_{1:k})$, is called the *prediction*. Since (2.32) is considered a function of s_k the denominator is a normalisation constant that is often neglected. The prediction can be rewritten with the law of total probability, by involving the previous state s_{k-1} in the equation:

$$\begin{aligned} p(s_k|z_{1:k-1}, u_{1:k}) &= \int p(s_k|s_{k-1}, z_{1:k-1}, u_{1:k})p(s_{k-1}|z_{1:k-1}, u_{1:k})\mathrm{d}s_k \\ &= \int p(s_k|s_{k-1}, u_k)p(s_{k-1}|z_{1:k-1}, u_{1:k-1})\mathrm{d}s_k \end{aligned} \tag{2.33}$$

Note that the first part in the integral is the motion model, and the second part, called *prior* at time k, is the posterior distribution from time $k-1$. The result is that given knowledge of the prior distribution, the motion model, and the measurement model, equations (2.32) and (2.33) can be used to obtain the posterior distribution at time k. In what is called Bayesian filtering this is often used recursively to filter a sequence of data. How easily these equations can be solved analytically depends on the models, and different solutions for different cases are shown in the following subsections.

2.2.1 Kalman Filter

Consider the fully linear system in (2.28), with zero mean Gaussian additive noise:

$$v_k \sim \mathcal{N}(0, V), \quad r_k \sim \mathcal{N}(0, R), \quad \forall k \tag{2.34}$$

Also define a prior:

$$p(s_0) = \mathcal{N}(s_0; \hat{s_0}, P_0) \tag{2.35}$$

With the linear system, and all variables being defined as Gaussian, it is possible to recursively obtain the analytically correct posterior distribution, which is also Gaussian. This method is called a Kalman filter. In the *prediction step*, a Gaussian prediction of the state at time k is obtained by using the state at time $k-1$:

$$\hat{s}_{k|k-1} = A\hat{s}_{k-1|k-1} + Bu_k \tag{2.36}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + V \tag{2.37}$$

This prediction is used together with the current measurement to create the posterior distribution in what is often referred to as the *update step*:

$$\tilde{z}_k = z_k - C\hat{s}_{k|k-1} \tag{2.38}$$

$$S_k = CP_{k|k-1}C^T + R \tag{2.39}$$

$$K_k = P_{k|k-1}C^T S_k^{-1} \tag{2.40}$$

$$\hat{s}_{k|k} = \hat{s}_{k|k-1} + K_k\tilde{z}_k \tag{2.41}$$

$$P_{k|k} = (I - K_kC)P_{k|k-1} \tag{2.42}$$

The posterior distribution is then:

$$p(s_k|z_{1:k}, u_{1:k}) = \mathcal{N}(s_k; \hat{s}_{k|k}, P_{k|k}) \tag{2.43}$$

The prediction step is often instead called the *time update*, and the update step is often called *measurement update*.

2.2.2 Extended Kalman Filter

As described earlier, there exists an analytically correct approach to calculate the posterior distribution, given that all parts of the system are linear and all distributions are Gaussian. This may not be the case in every application. Gustafsson [22] explains that for non-linear and/or non-Gaussian problems there are different approaches for state estimation that are still based on Bayesian inference. Note however, that they are based on approximations and will therefore no longer produce optimal posterior distributions.

One approach described by [22] is the extended Kalman filter (EKF). In every iteration of the filter the models are linearised. The additive noise is assumed to be Gaussian, and the posterior distribution is also considered Gaussian. Due to the latter assumption the EKF is not well suited for problems where the true posterior distribution is multimodal or very unlike a Gaussian distribution.

The models $f(s_{k-1}, u_k)$ and $h(s_k, u_k)$ are linearised to create Jacobian matrices, assuming additive noise:

$$F(\hat{s}, u) = \left.\frac{\partial f}{\partial s}\right|_{(\hat{s},u)}, \quad H(\hat{s}, u_k) = \left.\frac{\partial h}{\partial s}\right|_{(\hat{s},u)} \tag{2.44}$$

The prediction step then becomes:

$$\hat{s}_{k|k-1} = f(\hat{s}_{k-1|k-1}, u_k) \tag{2.45}$$

$$P_{k|k-1} = F P_{k-1|k-1} F^T + V \tag{2.46}$$

and the update step is:

$$\tilde{z}_k = z_k - h(\hat{s}_{k|k-1}) \tag{2.47}$$

$$S_k = H P_{k|k-1} H^T + R \tag{2.48}$$

$$K_k = P_{k|k-1} H^T S_k^{-1} \tag{2.49}$$

$$\hat{s}_{k|k} = \hat{s}_{k|k-1} + K_k \tilde{z}_k \tag{2.50}$$

$$P_{k|k} = (I - K_k H) P_{k|k-1} \tag{2.51}$$

2.2.3 Particle Filter

The particle filter (PF) is a non-parametric method of estimating a posterior distribution, that uses a dynamic set of samples to represent the distribution. The particle filter can estimate arbitrarily shaped posterior distributions, and may use highly non-linear motion and measurement models with any type of noise [22]. One important requirement however is that involved functions are point wise evaluable.

Consider N independent samples $X = [x^{(1)}, x^{(2)}, \dots, x^{(N)}]$ that are drawn from the distribution $p(x)$. With these samples, it is possible to approximate the same distribution:

$$p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(x - x^{(i)}) \tag{2.52}$$

The following equation can be used to calculate for instance mean value or covariance

$$\mathbf{E}[g(x)] \approx \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)}) \tag{2.53}$$

by replacing $g(x)$ with a suitable function. However, in many cases it is hard to sample from $p(x)$, so a *proposal density* $q(x)$ is introduced. This density has to be easy to sample from, and must contain the support of $p(x)$. To compensate for the difference between p and q, importance weights are introduced. Let $S = [s^{(1)}, s^{(2)}, \dots, s^{(i)}, \dots, s^{(N_p)}]$ be samples that are called particles, drawn from $q(s)$. The target distribution can be approximated:

$$p(s) \approx \sum_{i=1}^{N} \tilde{w}^{(i)} \delta(s - s^{(i)}) \tag{2.54}$$

The importance weights are defined as:

$$\tilde{w}^{(i)} = \frac{p(s^{(i)})}{q(s^{(i)})}, \qquad w^{(i)} = \frac{\tilde{w}^{(i)}}{\sum_{j=1}^{N} \tilde{w}^{(j)}} \tag{2.55}$$

The importance weights are, due to the normalisation factor, relative weights. This means a particle with a high corresponding weight is not necessarily a good fit to the target distribution, only that it is a better fit than most other particles.

Now the particle filter is introduced. In addition to a target distribution $p(s_k|s_{k-1}, z_k, u_k)$, the proposal density $q(s_k|s_{k-1}, z_k, u_k)$ is used (the choice of proposal density is briefly discussed later). The particle filter maintains an adaptive set of particles that represent possible states, each with a corresponding importance weight. In each iteration every particle is redrawn in the state space based on the proposal density and the previous state of that particle:

$$s_k^{(i)} \sim q(s_k|s_{k-1}^{(i)}, z_k, u_k) \tag{2.56}$$

The importance weights from the previous iteration $w_{k-1}^{(i)}$ are updated (with the same normalisation as used in (2.55), but with simplified notation using "$\propto$"):

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(z_k|s_k^{(i)}) p(s_k^{(i)}|s_{k-1}^{(i)}, u_k)}{q(s_k^{(i)}|s_{k-1}^{(i)}, z_k, u_k)} \tag{2.57}$$

An important part of a particle filter is the resampling. Resampling removes particles that are of low interest (i.e. does not contribute very much to the estimation of the posterior distribution) and replaces them with particles in high probability regions.

How and at what rate the resampling occurs can vary, but a common resampling strategy is to at some or every iteration create a new set of particles, that are randomly chosen based on the importance weights of the previous particle set. The importance weights are then reset to $1/N_p$.

A common choice of proposal density is to use the motion model (which simplifies equation (2.57)). There are other choices however, which may affect performance and particle efficiency.

2.2.4 Fixed Lag Particle Smoother

As briefly described, smoothing is a state estimation method where data obtained *after* time k is used to estimate the state at time k. Particle smoothing may be performed in several different ways. This subsection briefly covers a simple method for approximating the smoothed particle filter output, called fixed-lag smoothing [23].

Remember the sought posterior for the full smoothing problem (in this subsection the control input is left out to simplify notation):

$$p(s_{1:k}|z_{1:K}) \tag{2.58}$$

One simple and efficient way to obtain a smoothed distribution is to for all k set:

$$p(s_k|z_{1:K}) \approx \sum_{i=1}^{N} w_K^i \delta(x_k - x_k^i) \tag{2.59}$$

This is by some called the backtracking particle filter [27]. It uses the particle weights from the last filter time step and the obtained particle trajectories to create a smoothed posterior for each time step. The idea is that if a particle has a high weight at time K, its trajectory (or history) probably provides a good guess of the earlier states. This approach however suffers from particle history degeneracy. In the resampling step of the filter old trajectories are constantly removed, and for $k << K$ the smoothed posterior in (2.59) may be built solely on one trajectory.

To mitigate this problem, (2.59) may be modified:

$$p(s_k|z_{1:K}) \approx \sum_{i=1}^{N} w_{min(k+L,K)}^i \delta(x_k - x_k^i) \tag{2.60}$$

The approach is similar to before, but instead of using the last weights of the sequence it uses a lag L to select weights. This is still an approximation, and relies on two assumptions:

- L is large enough such that the information gathered after $k + L$ does not contribute significantly to the estimate at time k.

- L is small enough such that particle history degeneracy does not become too big of an issue.

The fixed-lag smoothing algorithm is simple to implement, but the problem is that the optimal lag L is typically not known. There is also no known method to automatically select a good enough lag.

2.3 Simultaneous Localisation and Mapping

In ordinary life, a map is often viewed as a representation of 3D space, describing for example locations of buildings and roads. In engineering a map can be described in a more general way as a representation of the environment as seen by exteroceptive sensor(s). Mathematically a map can be represented by a set of *map objects*, for example positions of obstacles or the received signal strength distribution at different locations. Maps can be dynamic (i.e. change with time) or static. In this section the maps described are assumed to be static.

Creating a map, or *mapping*, can be done in different ways. For instance, one can use previous knowledge and models to create a map. For some areas, a common approach is however to use said exteroceptive sensors to build and update the map. That is to use the sensor data and known (or estimated) recording positions, which in the case of received signal strength mapping is often referred to as *fingerprinting*. A more practical and advanced approach to mapping is to automatically determine the position of the device carrying the sensor(s) and then create the map, a process which can be called *positioning and mapping*. The most promising method is however *simultaneous localisation and mapping*, a process that simultaneously determines the recording positions and the map. The difference is that the positioning and mapping can aid each other, creating a better result overall.

In a way to describe the problem, the dynamic Bayesian network in figure 2.1 may be extended to include map elements, as in figure 2.2 (inspired by [24]). The idea is that the measurement z_k is affected by both the state s_k and possibly one or more map objects m_j. The measurement model can then be written as $p(z_k|s_k, M)$, where $M = [m_1, m_2, ...m_j]$ is the map containing the map objects. The map objects may be of several different types, and the set of map objects may be pre-determined or dynamic (for example estimating positions of known objects, versus discovering objects as a robot moves along an environment). The measurement z_k may not always be available or affected by the map.

Formally the problem can be described as acquiring the posterior distribution of the states and the map, given all measurements and control inputs:

$$p(s_{1:k}, M|z_{1:k}, u_{1:k}) \tag{2.61}$$

One important assumption for SLAM is that given the state sequence $s_{1:k}$, map object estimations are independent of each other (see figure 2.2). This leads to the

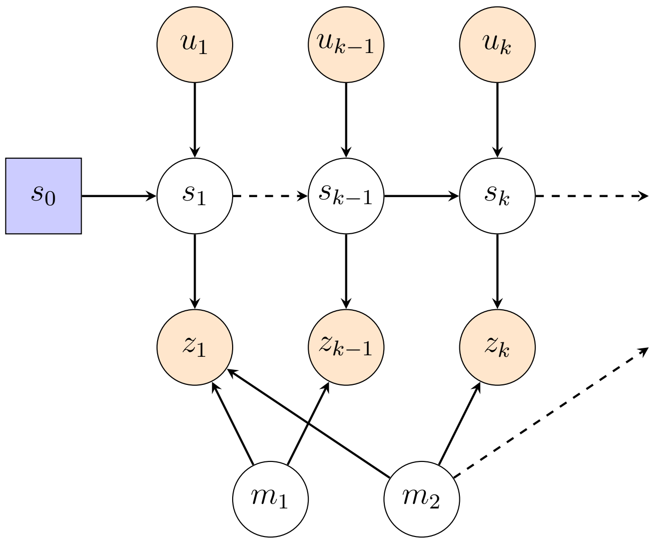


Figure 2.2: Dynamic Bayesian network, with map elements. The prior s_0 can be known or unknown. The control (or input) u and z are known or measured, while the states s and map objects m are unknown. A set of measurements z_k may be affected by one or several *map objects*, which in the SLAM-problem are to be estimated.

possibility to factorise equation (2.61) [24]:

$$\begin{aligned} p(s_{1:k}, M|z_{1:k}, u_{1:k}) &= p(s_{1:k}|z_{1:k}, u_{1:k})p(M|s_{1:k}, z_{1:k}, u_{1:k}) \\ &= p(s_{1:k}|z_{1:k}, u_{1:k}) \prod_{\forall j} p(m_j|s_{1:k}, z_{1:k}, u_{1:k}) \end{aligned} \tag{2.62}$$

The problem is now factorised into $1 + J$ estimation problems, one where the state trajectory is estimated plus J estimations of the map objects conditioned on the trajectory, where J is the number of map objects.

The SLAM problem may be solved using several different approaches. [11] describes two types of SLAM solutions; filtering (online) SLAM, where states and the map are estimated iteratively (for example a filter approach), and smoothing (offline) SLAM, where the entire state trajectory and map is estimated from all available data. A popular smoothing SLAM approach is called GraphSLAM [11], where the problem is expressed as a graph network and optimised with regards to all data.

2.3.1 FastSLAM

In 2002 a SLAM solution called FastSLAM was introduced [24]. FastSLAM estimates a robot trajectory and the position of obstacles (i.e. the map objects are obstacles' positions and position uncertainties), given some odometry and (noisy) distance measurements to said obstacles. It is built on the use of a particle filter that estimates the robot trajectory and the map iteratively, where each particle carries an estimated map of the obstacles. A particle is thus:

$$S_k = [s_{1:k}, M], \quad M = [\mu_1, \sigma_1^2, \dots, \mu_J, \sigma_J^2] \tag{2.63}$$

where μ and σ is mean and covariance of the obstacles' positions.

However, directly applying a particle filter to this state vector would be computationally expensive (since the number of obstacles could possibly be high). Therefore, the result in equation (2.62) is used to create an instance of the Rao-Blackwellized particle filter. The posterior of the trajectory $p(s_{1:k}|z_{1:k}, u_{1:k})$ is estimated with the particle filter, while the map object estimate $p(m_j|s_{1:k}, z_{1:k}, u_{1:k})$ for each object is conditioned on that particles trajectory. This is implemented iteratively:

1. Perform time update of motion states in each particle.
2. Calculate importance weights based on measurements and each particle's map.
3. Resample particles.
4. For each particle and where applicable update the posterior over each map object using EKF.

2.4 Bluetooth Low Energy

In this section, some characteristics of Bluetooth low energy broadcasts will be explained. This includes information that is sent, some frequency band characteristics and a model for signal path loss that is an important part of the methods used in this thesis.

2.4.1 BLE Beacons

A BLE beacon is a small device which broadcasts BLE signals with information for other devices to receive and work with. Beacons in broadcast mode only send signals one-way, but during setup and development it can be configured with two way communication. The Estimote Proximity Beacon, see figure 2.3, is one of the beacons on the market. These have a broadcast power from -30 dBm to 4 dBm which corresponds to -91 dBm to -60 dBm in transmission power. The transmission power is the received signal strength at one meter from the beacon, which is an important parameter used in this thesis. Estimote estimates the range of the signal to 1.5 m - 70 m and an advertising interval from 100 ms to 2000 ms. The broadcasting message is containing information about transmission power, a unique id and two user defined messages called major and minor all of which can be changed in the Estimote mobile application [28]. A lower advertising interval and higher transmission power has a negative effect on the battery life span. [29]

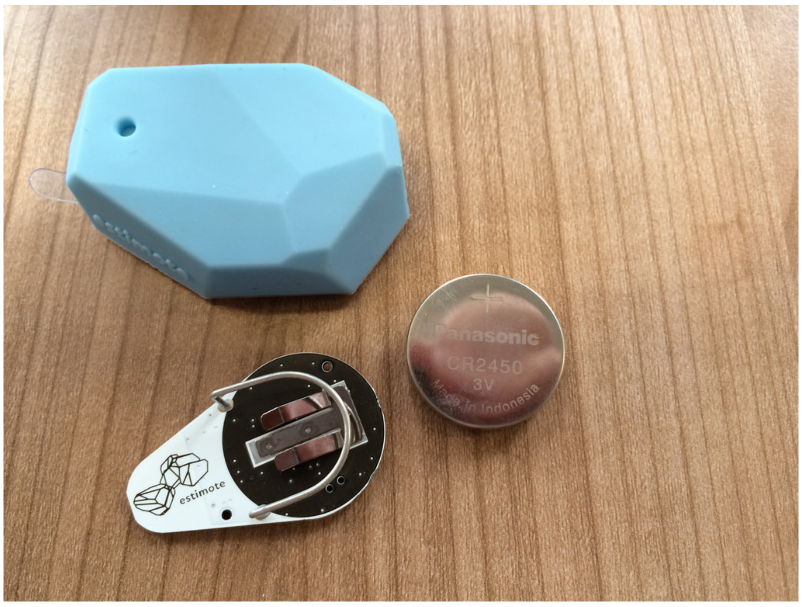


Figure 2.3: Image from [1] (CC BY 2.0), of an Estimote Proximity Beacon and its internal circuit board with battery.

2.4.2 BLE Frequency Bands

As explained by Faragher and Harle [10, 12], Bluetooth low energy operates in the 2.4 GHz band and specifically uses three different broadcast channels at 2402 MHz, 2426 MHz, and 2480 MHz. Each channel is 2 MHz wide and a BLE beacon broadcasts at one channel at a time, switching between them. Furthermore, in accordance with the BLE standard, a receiving device does not necessarily pass on information regarding which channel a broadcast was received on.

The use of three different channels may have large effects on the received signal strength [10], due to the following:

- *Channel gain*: Transmitters as well as receivers may have different gains for the three different channels.
- *Signal propagation*: Even though the channels are relatively close in signal space they may inhibit very different signal propagation properties, mainly multipathing, as shown by Faragher and Harle.

Multipathing is the phenomena when a transmitted signal is propagated via multiple paths in the physical space due to reflections and obstructing objects before reaching a receiver. This may affect the RSS due to constructive and destructive interference. Multipathing behaves differently at different broadcast frequencies, which is why it is relevant for BLE applications using RSS. Faragher and Harle discusses this problem and investigates different multipath mitigation techniques, and suggests that a sliding window filter (mean, median, minimum or maximum) is a good solution. The window length should cover at least three measurements, which will mitigate the different channel characteristics. This is assuming that the broadcast device switches channel after each broadcast.

2.4.3 Bluetooth Received Signal Strength Distribution

Modeling the RSS is not an easy task, and one that this thesis is trying to work around by creating fingerprints for navigation. It is however still useful to have access to some idea regarding expected RSS at different locations.

A common and simple model for RSS from Bluetooth is the path loss model [30, 25]. The RSS at distance d depends on a reference RSS at distance d_0 and a path loss parameter n. The latter is environment dependent, and is often chosen by fitting the model to available data. The received signal strength model is:

$$RSS(d) = RSS(d_0) + 10n\log_{10}(\frac{d}{d_0}) + r \tag{2.64}$$

where r is measurement noise. As stated in the previous subsection a common, and broadcast included, parameter is the transmission power T_x which is the expected RSS one meter from the transmitting device. The model then becomes:

$$RSS(d) = T_x + 10n\log_{10}(d) + r \tag{2.65}$$

This model fairly well describes the line of sight signal propagation (i.e. clear line of sight between receiver and transmitter). It does not however model multipathing characteristics or what happens with the RSS when the line of sight is obstructed. Regarding the noise r, one usually assumes that it is independent and identically distributed.

2.5 Step Detection

When estimating the position of a mobile phone and user, the need to know if and how it moves is of importance. One way to do this is to detect the steps the user is taking and by knowing in which direction, the movement can be estimated [31].

The coordinate system of the phone is defined as in the figure 2.4 with the x-axis pointing to the right when looking at the display, y pointing up and the z-axis is point out from the display.

One way to detect steps using a mobile phone is to use the accelerometer and try to detect when the user's heel strikes with the ground. This event results in a peak in the acceleration, which can be detected. When measuring the acceleration of the mobile phone, there is a negative acceleration in the vertical plane when the heel

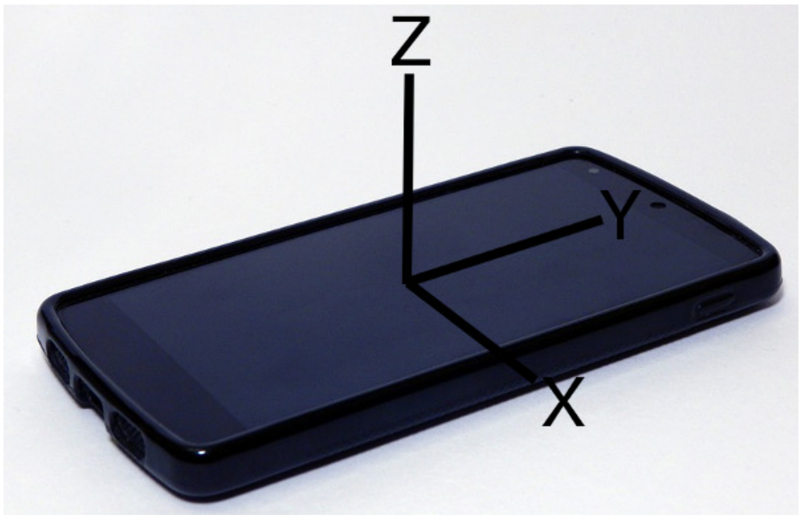


Figure 2.4: The coordinate system of the smartphone. Image from Max Pixel [2] (CC0) modified by the authors.

strikes the ground followed by an acceleration in the opposite direction and then a negative to plan out [14]. The method to detect steps is good for healthy users and not for an impaired user who does not have a distinct heel strike [32].

2.6 Blueprints

A blueprint is a technical drawing describing a mechanical or architectural design, which could be used to reproduce the objects it describes. Floor plans is a type of blueprints describing the layout of a floor and it is used to help people locate themselves in the area. Blueprints could be of different types, some more detailed than others. They could describe the structural layout of the area such as walls, windows and doors. The floor plans could also describe the general layout of furnishing such as chairs and tables in the area.

A way to represent blueprints instead of an image is to have them vectorised. This means that the lines and objects are defined by mathematical expressions. Using this, solid blocks of an object could be defined by lines along the block's edges. Working with vectorised blueprints has the advantage that it is more computational effective and the blueprint is easily scalable, but on the other hand images may be easier to construct.

The blueprints could be used to determine if a position is located in the area of interest. It could then be used to determine if it is in a reachable position or not. It is also possible to check if it is possible to move from one position to another. Even if two positions are valid and both are in reachable positions does not mean it is possible to move from one to another. It could be a wall or other objects in the way making the trajectory invalid.

2.6.1 Bitmap Blueprints

Blueprints can be digitally represented in different ways where one way is in bitmap image format. This is a matrix where each element is mapped to the area and has a value representing different objects. In a binary representation, elements could be 1 for unreachable areas (walls, outdoors etc.) and 0 for reachable or the other way around.

To check if a trajectory is valid, when using the blueprint with representation of logical values, a straight line between the two positions must be interpolated. This line consists of all points in a straight line between the two positions. All points on the line then must to be evaluated to check for intersections with unreachable points in the blueprint.

2.6.2 Vectorised Blueprints

To check if a position is valid when using vectorised representations of blueprints, a check is done to see if the position is inside a rectangle or not. The rectangle could describe reachable or unreachable objects. One way is to define unreachable objects as rectangles and the outer bound of the blueprint as rectangle. A position is valid if it is outside all object rectangles and inside the bounding rectangle.

A way to represent rectangles in a vectorised blueprint is to define its corner positions:

$$\text{rectangle} = (x_{11}, y_{11}), (x_{12}, y_{12}), (x_{13}, y_{13}), (x_{14}, y_{14}) \tag{2.66}$$

Assuming the rectangle is aligned with the coordinate system and the first coordinate is the bottom left corner followed by the corner coordinates of the rectangle clockwise from it, then:

$$x_{11} = x_{12}, \quad x_{13} = x_{14}, \quad y_{11} = y_{14}, \quad y_{12} = y_{13} \tag{2.67}$$

Using substitution

$$x_{11} = x_1, \quad x_{13} = x_2, \quad y_{11} = y_1, \quad y_{12} = y_2 \tag{2.68}$$

gives the rectangle representation

$$\text{rectangle} = (x_1, y_1), (x_1, y_2), (x_2, y_2), (x_2, y_1) \tag{2.69}$$

When using only horizontally aligned rectangles the corner positions share x and y elements with each other as seen in equation (2.69). To check if a position is inside a rectangle the following inequalities are checked:

$$x_1 \leq x \leq x_2 \tag{2.70}$$

$$y_1 \leq y \leq y_2 \tag{2.71}$$

In (2.70) and (2.71), x_1, x_2, y_1, y_2 is the coordinates in the rectangles as stated in (2.69) while x and y is the coordinate of the position which is evaluated. This mean that if both equation (2.70) and (2.71) is satisfied the position is inside the bounds of the rectangle.

One way to check if the trajectory is valid between two points using the vectorised representation is to first define the line using the line equation:

$$y = k \cdot x + m \tag{2.72}$$

where the constants k and m can be calculated using the two points.

To check the validity of the trajectory the line is evaluated to check if it is intersecting a rectangle. When a line is intersecting a line this indicates that the trajectory is trying to move through an object. One way to check if a line intersects a line of a rectangle, the y's ($y_{l1} = y_{l2}$) of both line equations is set equal, which results in:

$$k_1 \cdot x_{l1} + m_1 = k_2 \cdot x_{l2} + m_2 \tag{2.73}$$

Because $x_{l1} = x_{l2}$ the equation becomes

$$x_{intersection} = \frac{m_2 - m_1}{k_1 - k_2} \tag{2.74}$$

To get $y_{intersection}$, in equation (2.72) set $x = x_{intersection}$. If $(x_{intersection}, y_{intersection})$ is inside the range of the trajectory and the rectangle edge there is an intersection which results in a invalid trajectory.

3

Methods

This chapter follows the same structure as the solution overview is written in the introduction. It starts with odometry estimation, and is followed by how blueprint matching is done. Later the model of the received signal strength is returned to. Finally, the method of automatic signal strength mapping, which in its foundation is a positioning algorithm, is explained.

3.1 Estimating Odometry

There are several ways to estimate odometry. One way to obtain a motion is to measure the phone's sensor data and then use it to calculate the movement. To obtain the position from the motion data, integration is performed. That method is good for short distance measurements, but not for long-term due to error accumulation in the integration. The approach chosen for this project is to calculate the orientation of the mobile phone at each detected step, instead of calculating the absolute motion of the phone. In this way, each step is detected using sensor data in a step detection algorithm. The orientation at each step is then an input to a particle filter which, among other things, approximates the step length.

3.1.1 Orientation Filter

To get the orientation of the mobile phone, sensor data from the accelerometer, gyrometer and magnetometer is used in an extended Kalman filter. Due to gyrometer signals being the most accurate signal with the smallest variances, they were chosen as input to the filter. The accelerometer and magnetometer was used in the measurement update of the filter. The magnetometer data is noisy and to reduce the amount of high frequency noise the signal is low-pass filtered before any other calculations. The signals in figure 3.1 is from a static placed mobile phone and the high frequency noise is quite noticeable. The filtered signal is filtered with a 50th order low-pass finite impulse response (FIR) filter with a Chebyshev window and a cut-off frequency at 1 Hz.

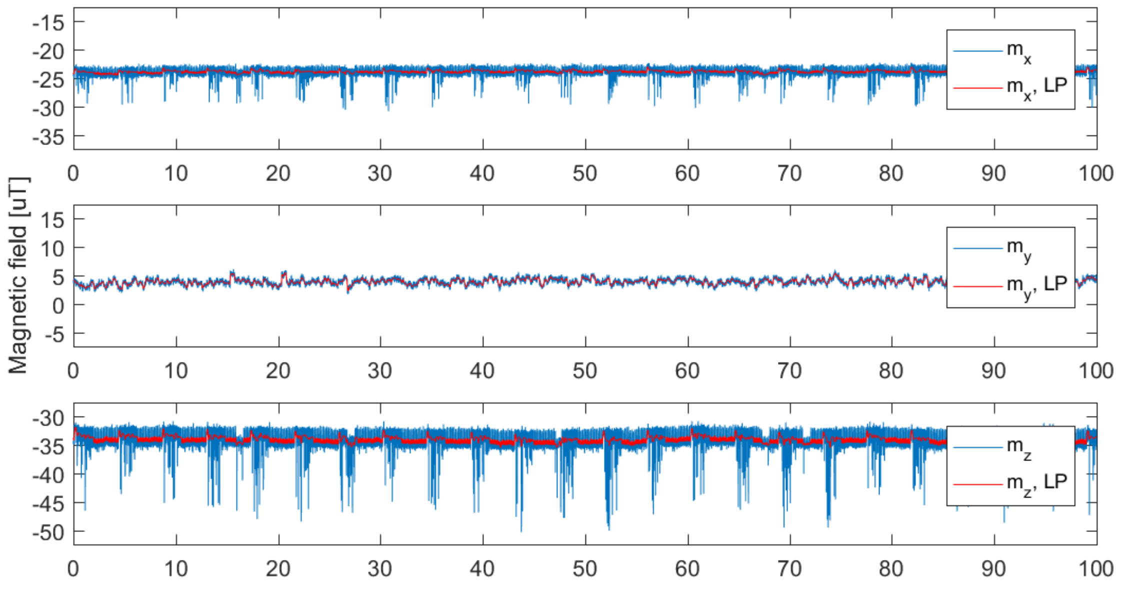


Figure 3.1: Static magnetometer signals before and after low-pass filtering

The following calculations in the subsection uses the same method as implemented in [17]. The EKF orientation filter uses the accelerometer, gyrometer and magnetometer data to calculate the orientation of the mobile phone in the world frame. The state space model of the system is:

$$x_{k+1} = f(x_k, u_k, v_k) \tag{3.1}$$

$$y_k = h(x_k, u_k, v_k) \tag{3.2}$$

Where the state vector x_{k+1} is chosen to be the time varying unit quaternion:

$$x_k = q_k = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3.3}$$

Input vector is chosen to be the 3-dimensional gyrometer data:

$$u_k = \omega_k = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.4}$$

Measurement of the model is chosen to be accelerometer and magnetometer data, both in 3-dimensions:

$$y_k = \begin{bmatrix} y_k^a \\ y_k^m \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \\ m_x \\ m_y \\ m_z \end{bmatrix} \tag{3.5}$$

EKF time update step

For the EKF time update step, the angular velocity is the filter input. Assuming the process noise enters additively on $\omega(t)$ and that $\omega(t) = \omega_{k-1}$ and $v(t) = v_{k-1}$ is piece-wise constant between sampling time t_{k-1} and t_k the continuous time model for the system is:

$$\dot{q}(t) = \frac{1}{2}S(\omega_{k-1} + v_{k-1})q(t) \quad \text{for } t \in [t_{k-1}, t_k), \tag{3.6}$$

where the noise v_{k-1} is assumed to be Gaussian zero mean with the variance $\mathbf{R}_v$. The discretisation of (3.6) becomes,

$$q_k = F(\omega_{k-a})q_{k-1} + G(\hat{q}_{k-1})v_{k-1} \tag{3.7}$$

where

$$F(\omega_{k-1}) = (I + \frac{1}{2}S(\omega_{k-1})T) \tag{3.8}$$

and

$$G(q_{k-1}) = \frac{1}{2}\bar{S}(q_{k-1})T \tag{3.9}$$

In these equations $q_k = q(t_k)$, $q_{k-1} = q(t_{k-1})$, and $G(\hat{q}_{k-1})$ is an approximation of $G(q_{k-1})$. The approximation uses the noise characteristics at the estimated mean instead of the characteristics of the prior. The calculations for q_k, $F(\omega_{k-1})$ and $G(q_{k-1})$ can be found in appendix A. The terms S and $\bar{S}$ are defined as:

$$\bar{S}(q) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{3.10}$$

$$S(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{3.11}$$

This gives the time update equations:

$$x_{k|k-1} = F(\omega_{k-1})x_{k-1|k-1} \tag{3.12}$$

$$P_{k|k-1} = F(\omega_{k-1})P_{k-1|k-1}F^T(\omega_{k-1}) + G(q_{k-1})R_vG^T(q_{k-1}) \tag{3.13}$$

EKF measurement update step

For the EKF measurement update step, the accelerometer and magnetometer data is used. The accelerometer measurement model is

$$y_k^a = R^T(q_k)(g^0 + f_k^a) + e_k^a \tag{3.14}$$

where $R(q_k)$ is the rotational matrix computed from the unit quaternion, g^0 is the nominal gravity vector, f_k^a acceleration in the world frame and e_k^a is the measurement noise.

The expected measurement for the accelerometer $h(\hat{x}_{k|k-1})$ is calculated with the rotation matrix times the gravity vector:

$$h(\hat{x}_{k|k-1}) = R(q)^T g_0 \tag{3.15}$$

The acceleration in the world frame is assumed to be zero ($f_k^a = 0$), because the orientation of the mobile phone is the output of the filter. The measurement update uses an outlier detection which removes measurements which are $3\sigma_a$ larger than the norm of the gravity vector. σ_a is the standard deviation of the accelerometer measurements.

To calculate the Jacobian $h'(\hat{x}_{k|k-1})$ of the EKF for the accelerometer in the update step the following equation is used:

$$h'(\hat{x}_{k|k-1}) = \begin{bmatrix} \frac{dR(q)^T}{dq_0} g^0 & \frac{dR(q)^T}{dq_1} g^0 & \frac{dR(q)^T}{dq_2} g^0 & \frac{dR(q)^T}{dq_3} g^0 \end{bmatrix} \tag{3.16}$$

The partial derivative of $\frac{dR(q)^T}{dq}$ is presented in Appendix A and is described in the equations (A.7) - (A.10).

The magnetometer measurement model is

$$y_k^m = R^T(q_k)(m^0 + f_k^m) + e_k^m \tag{3.17}$$

where $R(q_k)$ is the rotational matrix computed from the unit quaternion, m^0 is the magnetic field strength of the earth in world coordinates, f_k^m is magnetic fields from other sources and e_k^m is the measurement noise.

When the magnetometer lays horizontally still and the direction of north is unknown the earth magnetic field can be calculated as

$$m^0 = \begin{bmatrix} 0 & \sqrt{m_x^2 + m_y^2} & m_z \end{bmatrix}^T \tag{3.18}$$

due to the magnetic field is zero in the east-west direction.

The external magnetic fields is assumed to be zero, and an outlier detection algorithm is implemented which removes outlier magnetometer measurements. Because the magnetic fields cannot be assumed to be constant, an autoregressive filter is used:

$$L_k = (1 - \alpha)L_{k-1} + \alpha\|m_k\| \tag{3.19}$$

where m_k is the magnetometer measurement and α is a small positive number. L_k is used as a threshold in the outlier rejection. The expected measurement $h(\hat{x}_{k|k-1})$ is calculated with the rotation matrix times the magnetic field strength vector:

$$h(\hat{x}_{k|k-1}) = R(q)^T m^0 \tag{3.20}$$

The Jacobian $h'(\hat{x}_{k|k-1})$ for the magnetometer is

$$h'(\hat{x}_{k|k-1}) = \begin{bmatrix} \frac{\mathrm{d}R(q)^T}{\mathrm{d}q_0} m^0 & \frac{\mathrm{d}R(q)^T}{\mathrm{d}q_1} m^0 & \frac{\mathrm{d}R(q)^T}{\mathrm{d}q_2} m^0 & \frac{\mathrm{d}R(q)^T}{\mathrm{d}q_3} m^0 \end{bmatrix} \tag{3.21}$$

The orientation filter consists of a set of functions for the different sensor types. Because the data from the sensors is not read all at the same time the stream of data for the different sensors comes at different time points. Due to this the filter checks which sensor data is available and uses the corresponding function to the data type. If there is gyrometer data available, the filter performs a time update using this data, but if there is no gyro data available at the time, the filter performs a time update using previous gyrometer data.

When accelerometer and/or magnetometer data is available and it is not identified as an outlier, the filter calls a function to do a measurement update using the accelerometer and magnetometer data respectively otherwise it does nothing. After each update the quaternion is normalised to a unit quaternion. A block diagram of the algorithm is in figure 3.2.

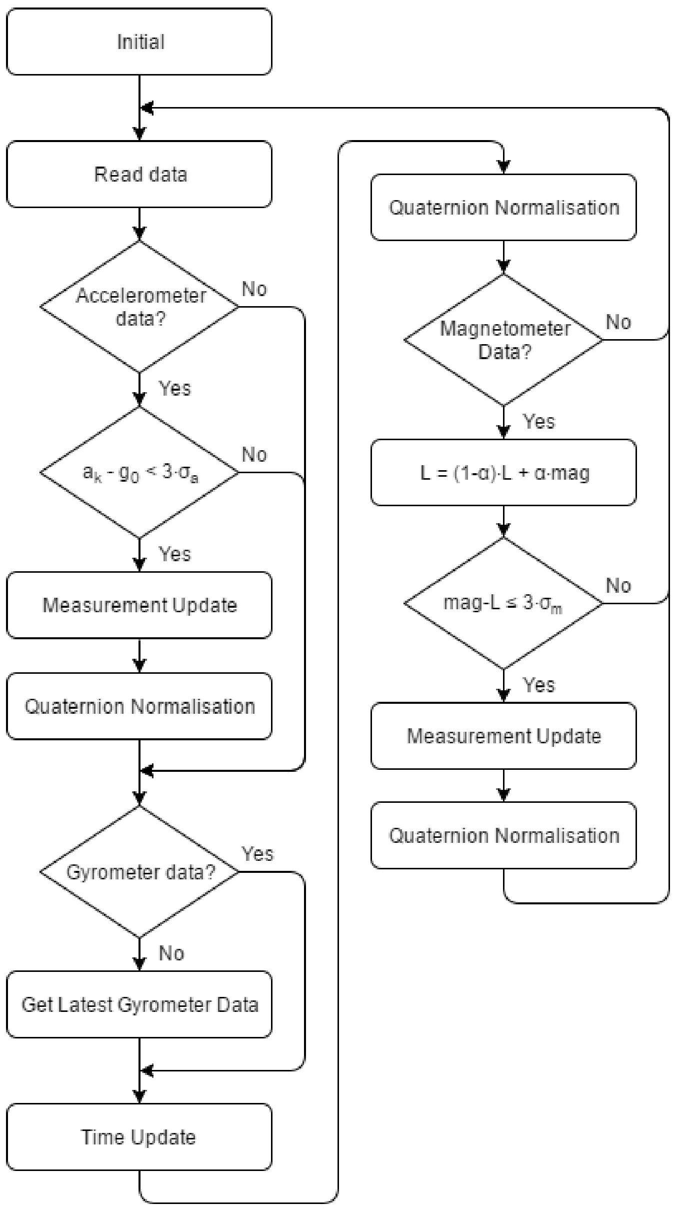
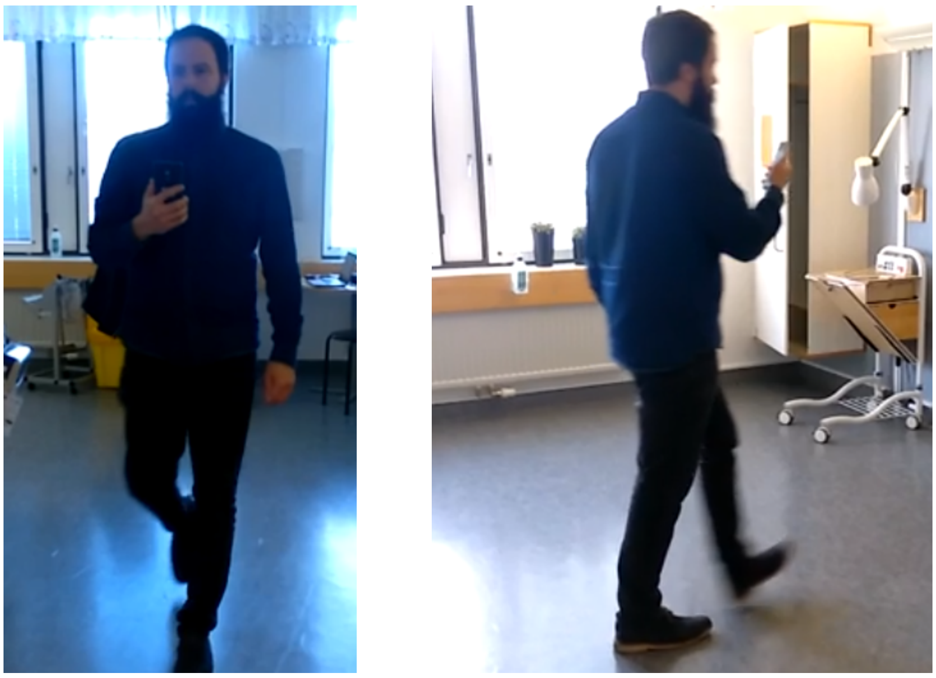


Figure 3.2: The orientation filter algorithm block diagram.

3.1.2 Step Detection

To be able to estimate the odometry using orientation, it is convenient to also know when a step is taken. Each time the user's heel strikes the ground an acceleration spreads throughout the body which the phone can measure. This acceleration can be analysed using different algorithms. The following algorithm is developed assuming the mobile phone is held in front of the user and that the heading of a step is the direction of the phone at the moment a step is detected, see figure 3.3.

It is observed that the acceleration in the y-axis is increasing before a step and decreasing after the step due to a pendulum motion of the hand. This is observed



(a) From the front. (b) From the side

Figure 3.3: Images of the user holding the mobile phone while walking the routes.

for situations when the phone is held in a hand in front of the user looking at the display.

During turns (and turnarounds) there is an acceleration sideways which even if no step has been taken. Because of this the measurements of the acceleration sideways (in the direction of the x-axis, a_x) is set to zero. This can be done because the orientation of the phone is assumed to be with the screen towards the user during the walks and then the steps have no effect on the acceleration x-wise. This leads to:

$$\mathbf{a} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix} \tag{3.22}$$

The norm of the signals is subtracted by the norm of the gravity to get the phones acceleration in the world frame:

$$a_k = \|\mathbf{a}\| - \|\mathbf{g}^0\| \tag{3.23}$$

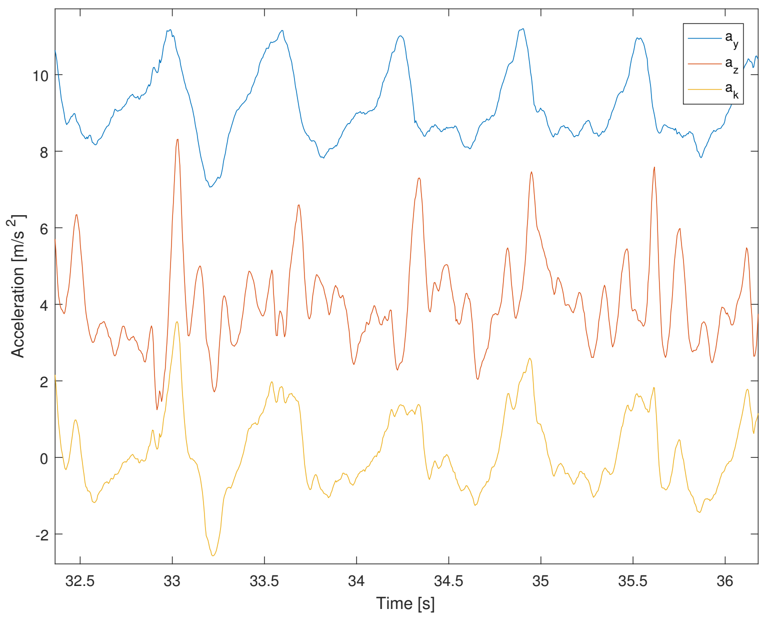


Figure 3.4: The acceleration measured by the acceleration during walk along the y-axis (a_y) and z-axis (a_z) and the norm of the acceleration (a_k).

The lines in figure 3.4 is the acceleration along the y- and z-axis, and the acceleration a_k from (3.23). The resulting acceleration signal a_k is low-pass filtered to reduce the amount of noise in the signal. It also makes it easier to detect local maxima/minima because the number of peaks is reduced when filtered. The type of filter used is a FIR filter with a Chebyshev window. The FIR filter delays the signal and therefore must be compensated. The number of samples the delay is, is half of the filter order:

$$n_{delay} = \frac{n_{order}}{2} \tag{3.24}$$

The delay time in seconds t_{delay} is the number of delayed samples divided with the sample frequency f_s,

$$t_{delay} = \frac{n_{delay}}{f_s} \tag{3.25}$$

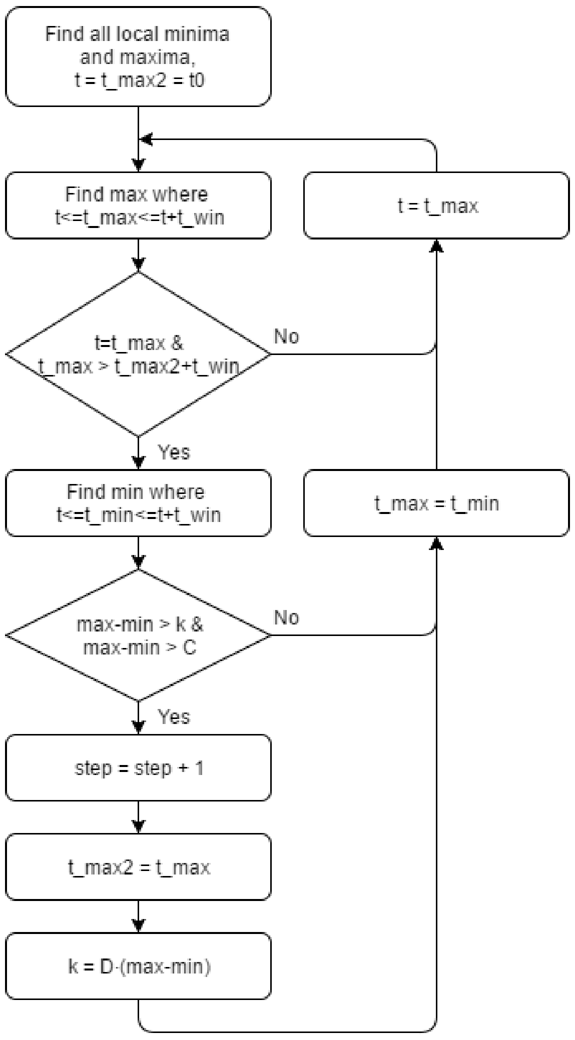

Find all local minima and maxima, t = t_max2 = t0
Find max where t<=t_max<=t+t_win
t = t_max
t=t_max & t_max > t_max2+t_win
No
Yes
Find min where t<=t_min<=t+t_win
t_max = t_min
max-min > k & max-min > C
No
Yes
step = step + 1
t_max2 = t_max
k = D·(max-min)

Figure 3.5: The step detection algorithm block diagram.

The initial phase of the step detection the algorithm finds all local minima and maxima of the filtered accelerometer signal. The algorithm uses a window with length t_{win} to look for the largest amplitude of a local maxima. Initially the window starts at the first local maxima and then it shifts it starting point to the largest local maxima inside the window. The algorithm shifts the starting point until the largest local maxima is the starting point.

After the largest amplitude of a local maxima is found the time where it is located, t_{max}, is compared to the time the last step were detected, t_{max2}. If the difference is smaller than the window size t_{win}, the algorithm start over with the next local maxima as starting point. The reason to do this is to prevent the algorithm from detecting two near peaks as two steps, when it actually only was one step taken.

The next phase, if the difference in time is large enough, the smallest amplitude inside the window after the local maxima is found. The difference in amplitude between the local maxima and the local minima is one factor that decides if there is a step taken or not.

The difference in amplitude is compared to two thresholds working as a lower bound threshold. k is the dynamic threshold and depends on the last step's amplitude:

$$k = D \cdot (a_{max} - a_{min}) \tag{3.26}$$

where $a_{max} - a_{min}$ is the difference in amplitude of the last detected step. D is a positive constant smallar than one. The other threshold is a predefined constant and is the smallest amplitude a step is allowed to have, which prevents k from becoming to small.

3.2 Blueprint Matching

A vectorised blueprint representation consists of a set of rectangles. One bounding rectangle to state the outer bounds of the blueprint and a number of rectangles stating the object edges in the blueprint. Figure 3.6 is an example a blueprint of the test environment, where the red rectangle is the bounding rectangle and the blue is the objects or walls in the area.

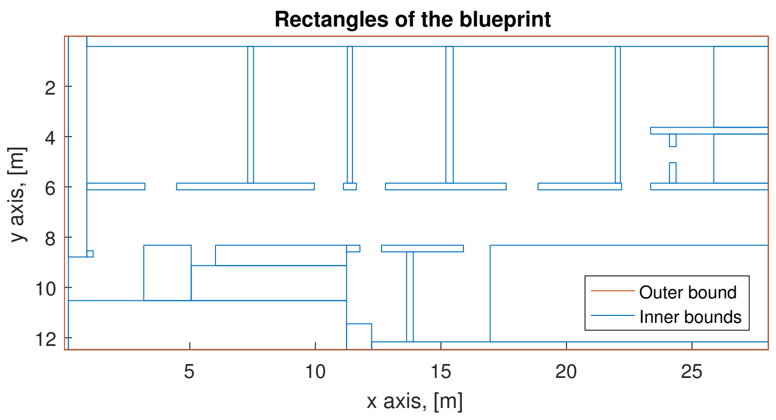


Figure 3.6: The blueprint converted to a set of rectangles, with both inner and outer bound.

The algorithm, to check if a position or a trajectory between two positions is valid, uses the prior and posterior position as inputs. The rectangles from the vectorised representation of the blueprints is also an input to the algorithm. The algorithm assumes all rectangles are aligned with the coordinate system of the map.

The first thing the algorithm checks is if the input position is inside the bounding rectangle. If the prior and posterior position is valid the trajectory between the positions is calculated with the line equation, 2.72. The algorithm uses the advantage that the rectangles are assumed to only have either x or y components. Thereby the coordinates of the corners of the rectangle directly states their equations.

Each rectangle from the vectorised blueprint then gives four line equations each, two with x- and two with y-components. All components for all rectangles is then used in the line equation of the trajectory to obtain the intersection of the rectangle lines and the trajectory. Because the rectangle lines and the line equation of the trajectory isn't bounded the intersection coordinate must be checked if it is inside the bound of both the rectangle and the trajectory. If it is inside both the bounds there is an intersection and the trajectory is invalid.

Figure 3.7 shows four example cases of trajectories:

1. The trajectory intersects a rectangle, and the posterior position is inside the rectangle and therefore it is invalid.

2. The extended line of the trajectory intersects the rectangle, but it is not inside the trajectory and therefore it is valid.

3. The trajectory intersects the rectangle twice which makes it invalid, even if the posterior position is outside the rectangle.

4. Both the trajectory and the extended line of it intersects the extended lines of the rectangle. The line is valid because the intersections are outside the rectangle boundaries.

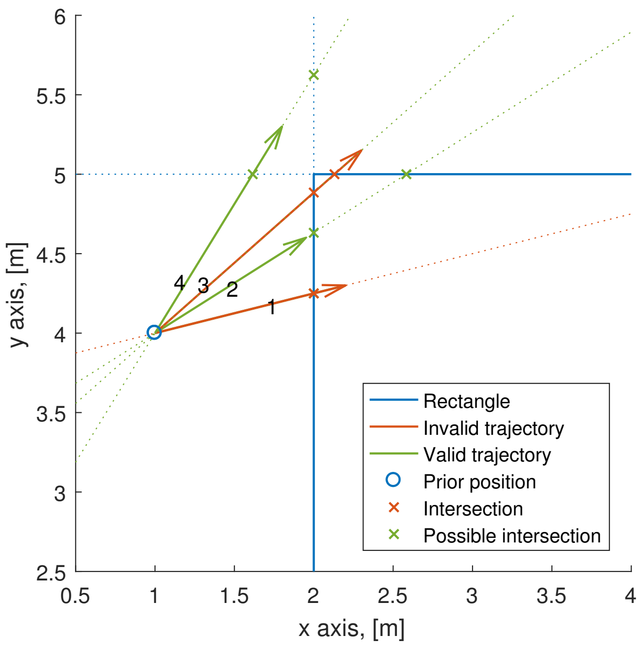


Figure 3.7: Example of four different trajectories and their validity.

3.3 Selecting Parameters for RSS Model

In this thesis, the RSS model in (2.65) will be used as a basis for a measurement likelihood function. A way to rewrite it is:

$$z^b = T_b + 10 n_b \log_{10}(d) + r \tag{3.27}$$

for beacon b, distance d between that beacon and the current position, and measurement z^b. The question is how to choose T_b, n_b and the measurement noise. Equation (3.27) will first be considered without the measurement noise r. Given a sufficient set of N measurements and measurement distances, one can find the remaining parameters by linear least squares. Express the problem as:

$$\underbrace{\begin{bmatrix} 1 & 10\log_{10}(d_1) \\ \vdots & \vdots \\ 1 & 10\log_{10}(d_N) \end{bmatrix}}_{\text{A}} \cdot \begin{bmatrix} T_b \\ n_b \end{bmatrix} = \underbrace{\begin{bmatrix} z_1^b \\ \vdots \\ z_N^b \end{bmatrix}}_{\text{B}} \tag{3.28}$$

Finding the least squares solution can be done as:

$$\begin{bmatrix} \hat{T}^b \\ \hat{n}^b \end{bmatrix} = (A^T A)^{-1} A^T B \tag{3.29}$$

In order to use the above approach, the said set of measurements and distances are needed. In this case, an approximated route at the test environment along with corresponding measurements will be used. The optimal pairs of T_b and n_b for all beacons from this route are shown in figure 3.8. Also illustrated is the linear least squares fit for a relationship between optimal T_b and optimal n_b. A relationship is noted between the parameters. As explained later, the SLAM algorithm in this thesis estimates the transmission power iteratively. With this in mind, n can be chosen accordingly:

$$z^b = T_b + 10(\alpha T_b + \beta)\log_{10}(d) + r \tag{3.30}$$

where α and β are the parameters from the linear relation fit. Note that the linear relationship between n_b and T_b only makes sense within a certain range. If T_b is too large n will be positive, meaning that the model states that the received signal strength actually increases with distance. This is an issue to keep in order during the implementation.

The question remains how to choose the noise r. To do this the same approximate route will be used, along with the model in (3.30). The transmission power will be the optimal parameter found for each beacon. In figure 3.9 the histogram of the deviation from actual measurements and the expected measurements using the model. It appears that a normal distribution fits the data points well. Therefore the parameter r could be assumed to be normally distributed. However, for reasons explained in a later section a t-distribution will be used instead.

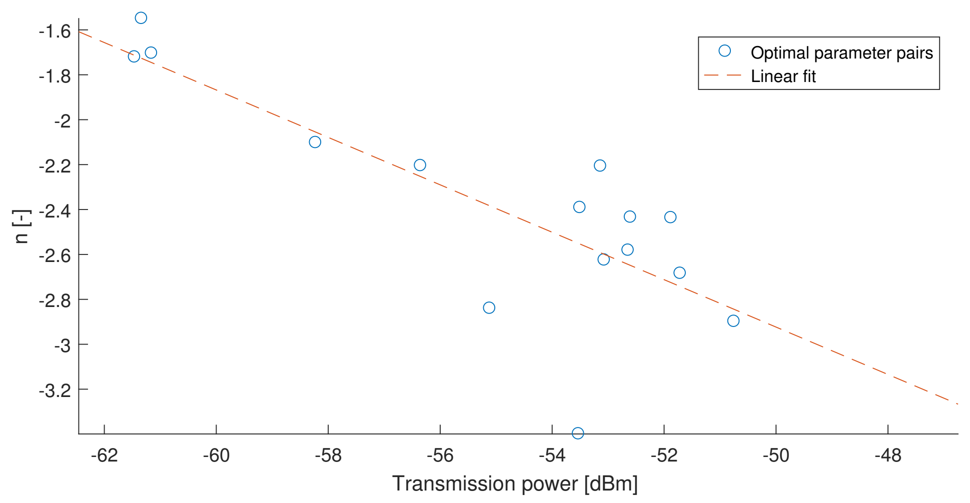


Figure 3.8: Illustration of the optimal parameter pairs for different beacons, using a "Simple" walk (see figure 4.2) as data source and approximate positions to calculate the distance. Also shown is a linear fit to the relationship between the two parameters.

Note that the noise parameter will not only represent measurement noise, but also model error. The noise is assumed to be independent and identically distributed, even though the model error likely depends on receiving position.

To summarise this section, the model for received signal strength is:

$$z^b = T_b + 10(\alpha T_b + \beta)\log_{10}(d) + r, \qquad r \sim \text{t}(0, R, 1) \tag{3.31}$$

3.4 Positioning and Mapping

In this section the algorithm that is used to estimate the recording positions and create the map is presented. First the pre-processing of signals will be shown, then the map itself is introduced, followed by some design choices in the FastSLAM-adaptation. Finally, the map updating will be explained.

The algorithm uses the odometry and received Bluetooth signals to estimate the recording positions and the map described below. The Bluetooth signals are filtered before used, using an average window filter of 2 seconds. This window length is a good compromise with regards to the number of measurements in the window and the smearing effect, and should catch at least three measurements.

The particle filter runs with 600 particles. To obtain a final map (as opposed to 600 maps, one for each particle) the trajectory is smoothed according to the description in the theory section. This smoothed trajectory is then used to re-create the map, in the same way as described below.

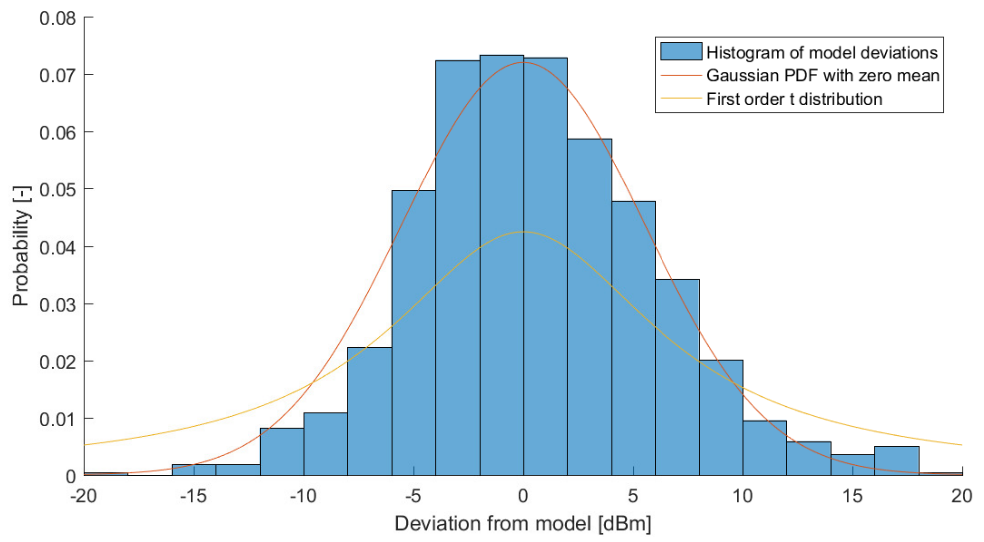


Figure 3.9: Histogram of deviation between measurement and model prediction, for all beacons together. Also included is a normal distribution with zero mean and matched variance, as well as a first order t distribution.

3.4.1 Signal Strength Map

The map will consist of two types of objects:

- T_b: The RSS at a distance of 1 meter from beacon b, also called *transmission power*.
- $P_{b,i,j}$: The distribution of received signal strengths from beacon b in the grid cell with coordinates (i, j).

The grid map consists of equally sized squares. A map is then made up of $N_b + N_b \cdot N_i \cdot N_j$ objects, where N_b is the number of beacons, and $N_j \cdot N_j$ are the number of grid cells.

The first object type is the transmission power. Even though this is a setting on the beacons, it has been noted during manual measurements that the actual RSS at one meter can vary from the specified value. For this reason, this parameter will be estimated with the FastSLAM-algorithm described later. The second element, even though a part of the map, will not be a part of the SLAM algorithm.

3.4.2 FastSLAM Adaptation

The chosen state vector is presented in equation (3.32). This representation includes 2D coordinates x and y, step length l and an orientation bias b:

$$s_k = [x_k, y_k, l_k, b_k]^T \tag{3.32}$$

A bare minimum would be to include only the 2D coordinates, but due to unknown step length it is favourable to include it too. Also, due to the slowly varying nature of the orientation estimation error an orientation bias is estimated as well. The true orientation is defined as: $\phi = u - b$, where u is the measured orientation, and b is the bias.

An important design choice in any filter is the motion model, $p_m(s_k|s_{k-1}, u_k)$. The motion model must fit the chosen state vector, as well as the problem. It is chosen as described follows, inspired by [25]:

$$p_m(s_k|s_{k-1}, u_k) = \mathcal{N}\left(\begin{bmatrix} x_k \\ y_k \end{bmatrix}; \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + l_k \cdot \begin{bmatrix} \cos(u_k - b_k) \\ \sin(u_k - b_k) \end{bmatrix}, \begin{bmatrix} \sigma_p^2 & 0 \\ 0 & \sigma_p^2 \end{bmatrix}\right) \\ \cdot \mathcal{N}(b_k; b_{k-1}, \sigma_b^2) \cdot \mathcal{N}^*(l_k; l_{k-1}, \sigma_l^2) \tag{3.33}$$

where σ_p^2, σ_b^2 and σ_l^2 are motion noise variance parameters, and $\mathcal{N}(...)$ is the Gaussian probability density function. The last term is noted $\mathcal{N}^*(...)$ which is defined as:

$$\mathcal{N}^*(l_k; l_{k-1}, \sigma_l^2) = \begin{cases} \alpha_l \cdot \mathcal{N}(l_k; l_{k-1}, \sigma_l^2) & l^{min} \leq l_k \leq l^{max} \\ 0 & \text{otherwise} \end{cases} \tag{3.34}$$

This step length motion model will restrict samples of the step length to between the two limits. This is needed to keep the step length from being unreasonably large or small. In this equation α_l is a normalisation constant such that the integration from $-\infty$ to ∞ with regards to l_k is equal to one. In the implementation of sampling from this model, this constant is not needed and is therefore left unknown.

In this SLAM implementation, the proposal density is chosen to be the motion model in (3.33). This is believed to fit the problem reasonably well, and simplifies the calculation of the importance weights.

3.4.3 Calculating the Importance Weights

Equation (3.35) is used when calculating the importance weights for the particles:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(z_k|s_k^{(i)})p(s_k^{(i)}|s_{k-1}^{(i)}, u_k)}{q(s_k^{(i)}|s_{k-1}^{(i)}, u_k, z_k)} \tag{3.35}$$

To include the blueprint matching in the particle weighing, the dynamic model will be:

$$p(s_k^{(i)}|s_{k-1}^{(i)}, u_k) = p_{BP}(s_k^{(i)}|s_{k-1}^{(i)})p_m(s_k^{(i)}|s_{k-1}^{(i)}, u_k) \tag{3.36}$$

where p_{BP} is the blueprint checking model, equal to one or zero depending on if the transition is valid or not. By setting the proposal density q to the motion model, (3.35) becomes:

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(z_k|s_k^{(i)}) p_{BP}(s_k^{(i)}|s_{k-1}^{(i)}) \tag{3.37}$$

The likelihood function $p(z_k|s_k^{(i)})$ will be the (rewritten) model for received signal strength in (3.31). Also, if the transmission power is estimated the likelihood function can be written as:

$$p(z_k|s_k^{(i)}) = \int p(z_k|s_k^{(i)}, T_b)p(T_b)\mathrm{d}T_b \tag{3.38}$$

If the transmission power is assumed to be precisely known, the likelihood function becomes $p(z_k|s_k^{(i)}, T_b)$. If not, and both terms are Gaussian, the integral can be solved easily [24]. However, it has been noted for this thesis that a Gaussian distribution for the likelihood function does not work well enough, and that a first order t-distribution with similar variance works better:

$$p(z_k|s_k^{(i)}, T_b) = t(z_k|\bar{z}_k, 7.5, 1) \tag{3.39}$$

where $\bar{z}_k$ is the mean value of equation (3.31). To avoid solving (3.38) in closed form for a t-distribution and a (possibly) Gaussian distribution, the distribution of T_b is assumed to be $p(T_b) = \delta(T_b - \hat{T}_b^{(i)})$, i.e. the uncertainty is disregarded. Intuitively this means replacing the previously known transmission power with its estimate in equation (3.39).

3.4.4 Mapping

Creating the map is a matter of using the recording positions and corresponding received signal strengths, and storing the information in the correct way. In this thesis this is done recursively. Note that when running the positioning particle filter described above, each particle has its own possibly unique trajectory, and will thereby create its own (possibly) unique map.

To update the estimated transmission power for each beacon, the RSS model in (3.31) is used. Despite the t-distribution used for the likelihood function above, in this part a Gaussian distribution is assumed. The following dynamic model is constructed:

$$T_k^b = T_{k-1}^b + v_{k-1}, \quad v_{k-1} \sim \mathcal{N}(0, V_T) \tag{3.40}$$

$$z_k^b = T_k^b + 10(A \cdot T_k^b + B)\log_{10}(d) + r_k, \quad r_k \sim \mathcal{N}(0, R_T) \tag{3.41}$$

In this model V_T is the motion noise variance, R_T is the measurement noise variance and d is the distance between the current position and the beacon b. The motion noise variance is tuned to create a good filter behaviour, and the measurement noise is chosen as in section 3.3. The equations (3.40) and (3.41) are used in an extended Kalman filter to update the estimate $\hat{T}^b$ for each particle.

The second map element, the received signal strength distribution for a cell, is represented by all measurements received in that cell. Updating this element is a simple matter of storing the current measurement in the cell belonging to the current position estimate.

4

Results

This chapter is describing the results using the different algorithms. First the step detection algorithm results is presented, then the orientation filter. After that the results with these two combined with a particle filter is presented. Finally, the created signal strength maps are evaluated using a set of manual fingerprints.

4.1 Test Environment

During a visit to Östra Sjukhuset, where the tests were carried out, data was collected by walking a set of eight predefined routes at the ward. During the walk, data was collected by logging accelerometer, gyrometer, magnetometer data and received Bluetooth signal strength. Signal strengths were also logged at static points around the ward to be used as fingerprints for localisation tests.

15 Bluetooth low energy beacons from Estimote [29] were mounted on the ceiling throughout the different rooms, as can be seen in figure 4.1. The beacons used were Estimote Proximity Beacons which were set up using the Estimote Android application [28] with the following parameters:

Operating system	2.1.0
Hardware revision	D3.4
Primary packet type	Estimote Default
Transmitting power	-66 dBm

Table 4.1: The parameters used for the Estimote Proximity Beacons.

The mobile smartphone used was a *OnePlus 2* with *Android Marshmallow (6.0.1)* and *OxygenOS (3.1.0)*. The applications *BLE analyser (1.04)* [33] and *Sensor Log (1.0.9)* [34] were used to log BLE-RSS and sensor data respectively.

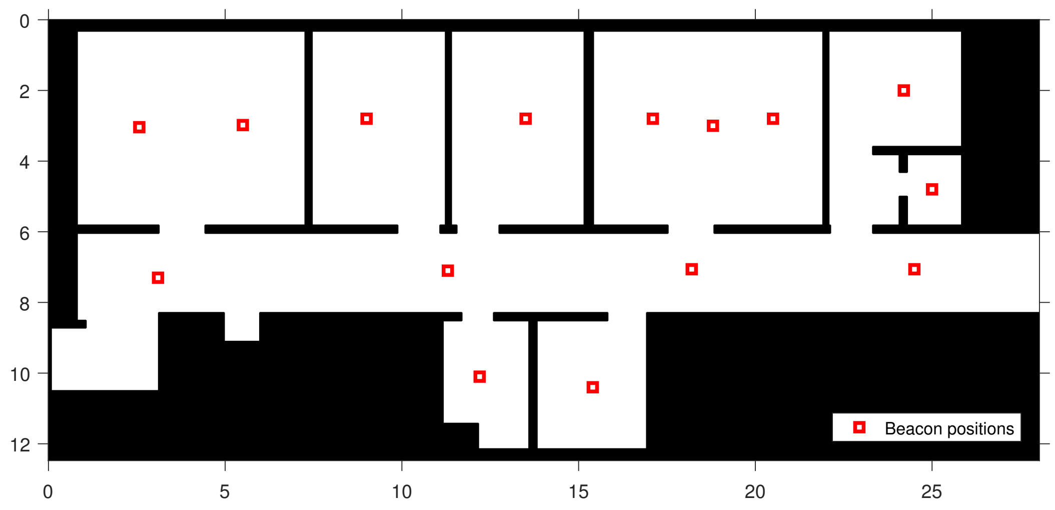


Figure 4.1: The layout of the test environment, as well as beacon positions. The beacons were placed in the ceiling. The axis labels are in meters.

4.1.1 Trajectories

The routes at the hospital ward were planned to test different aspects of the algorithms. They were all performed the same day, with the beacons at the same place. Each of the rooms had beds along the sides and they are not included in the blueprints due to being not stationary beds. Even though they could be moved around they had predefined places in each room. There were 4 beds in the large rooms, 2 in the middle-sized rooms, and one in the smallest in the top right corner. The two rooms at the bottom of the blueprints are toilets.

The following describes how the data collection routes were planned. The corresponding figures represent the routes in a way describing the principle of the routes. This means that the trajectory is an estimate that should be used to compare if the positioning algorithm is guessing the correct room or not for example.

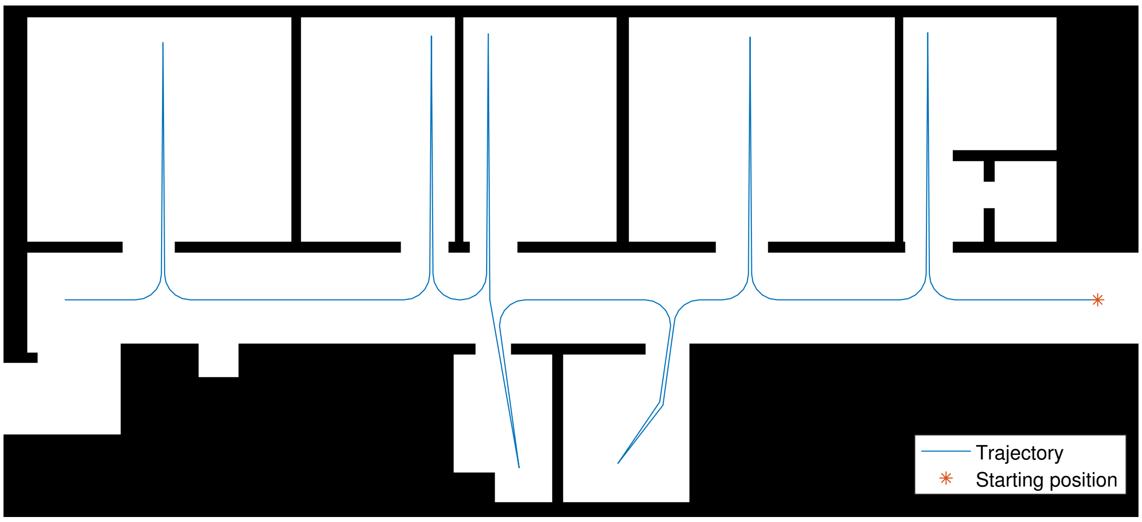


Figure 4.2: A simple trajectory through each of the rooms called "Simple".

The trajectory in figure 4.2 is a simple trajectory into each of the rooms. Each of the rooms are entered and visited by walking in a straight line through to the opposing wall. The user then turned around and left the same way out. This route was walked two times, hereby called route "Simple #1" and "Simple #2".

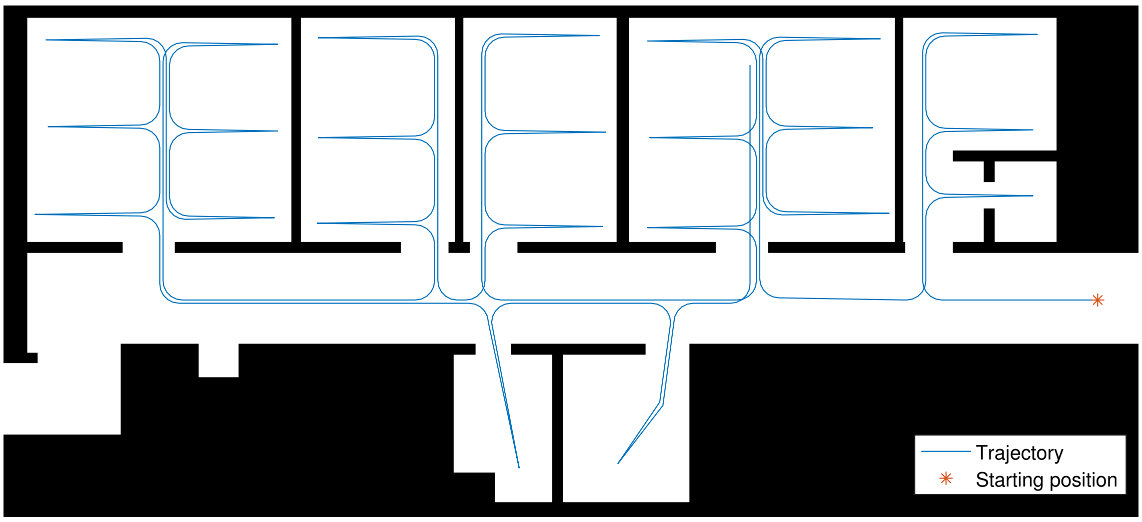


Figure 4.3: A trajectory showing a complete trajectory, meant to map the entire area of interest, called "Complete"

In figure 4.3 the route is planned to cover as much of the different pathways as possible. In each room the route goes by and around each of the beds to cover all the possible pathways. This route is referred to as "Complete". The route has been walked twice.

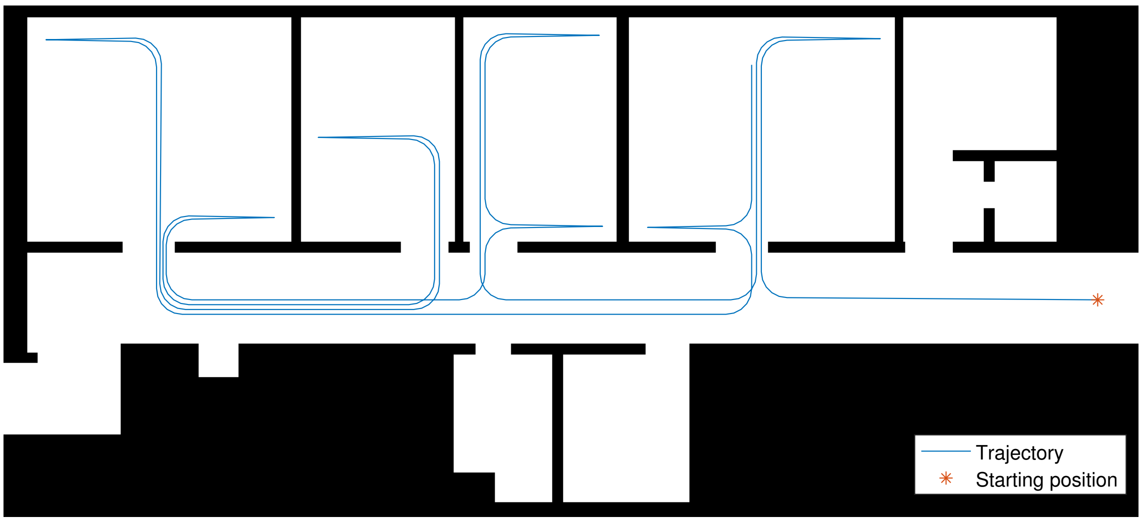


Figure 4.4: A trajectory visiting different bedsides.

The trajectory in figure 4.4, which was walked once, was planned to visit random beds. This has a more natural path which a nurse or a doctor would take when looking after his or her patients. This route is called "Bed route #1".

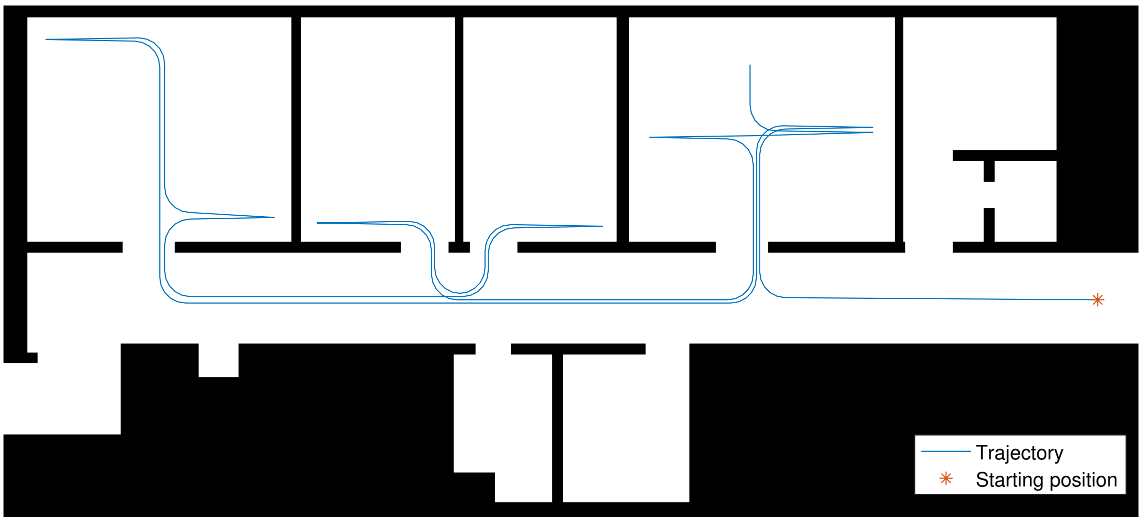


Figure 4.5: Another trajectory visiting different bedsides.

The route in figure 4.5 has the same purpose as in previous figure 4.4, where the route visits beds at random. This was also walked once and is called "Bed route #2".

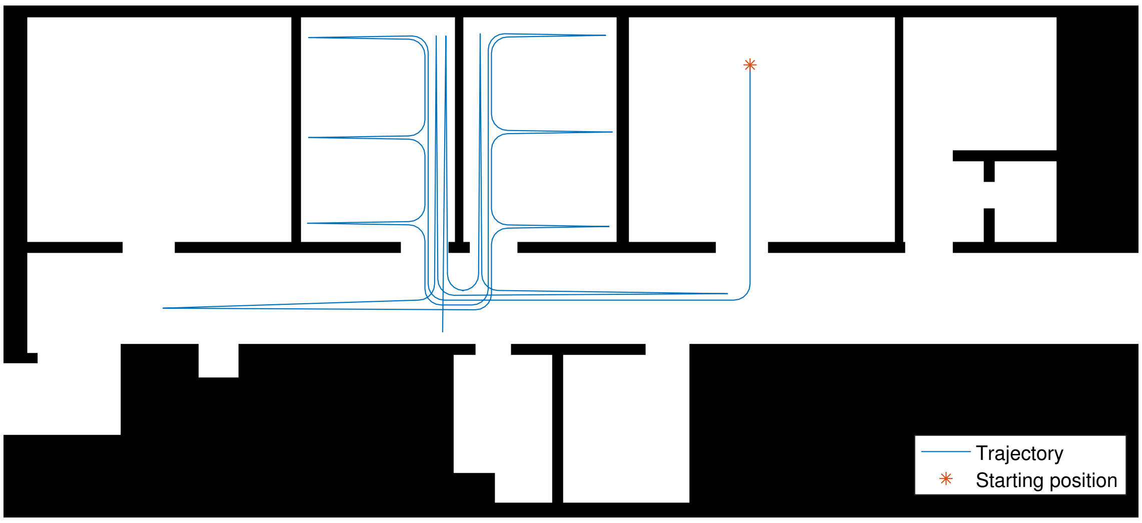


Figure 4.6: A trajectory planned to visit the same room more than once.

The main purpose with the route in figure 4.6 was to test room accuracy. At the beginning of the route the trajectory goes through the small rooms visiting each of the beds. Then it goes out to the corridor to turn around and then enter the left rooms again but without visiting any beds. After that the route is planned to go out of to the corridor again. From this point the trajectory first visits the right room, then the left room to end up in the corridor. The route is called "Slam test".

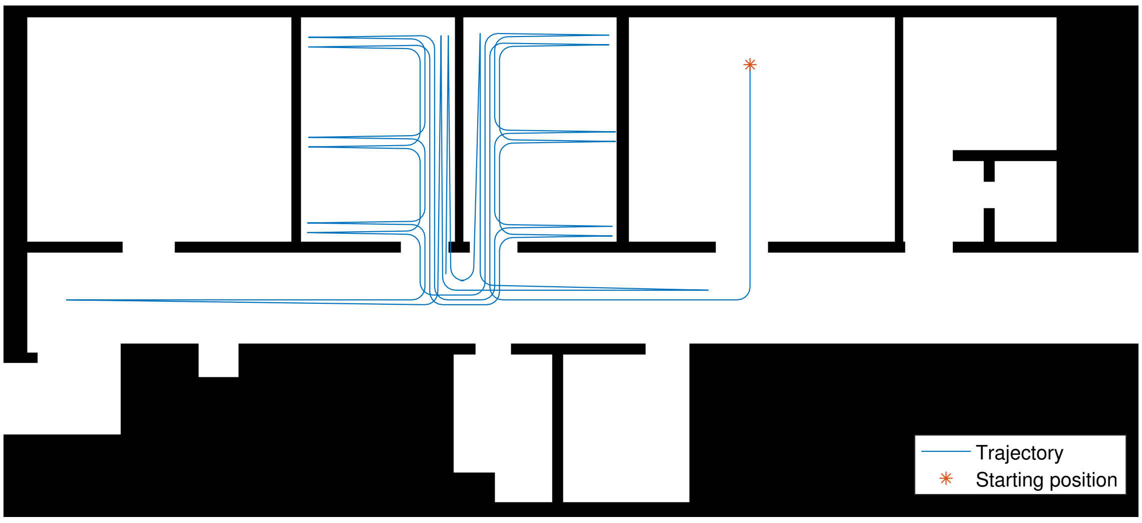


Figure 4.7: Another trajectory where the same rooms is visited more than once.

The route in figure 4.7 is planned in the same way as previous figure 4.6 and it has the same purpose. The difference is that the route takes two turns visiting the beds at the two rooms and then leaves to test the room accuracy.

4.2 Step Detection

During the walks acceleration data was logged for the step detection algorithm. During the walk the mobile was held in front of and with the screen towards the user throughout the whole walk. The parameters used in the step algorithm is presented in table 4.2:

Parameters:

D-constant	0.65
Constant threshold	1.5 m/s^2
Maximum threshold	2.5 m/s^2
FIR filter order	100
Filter window type	Chebychev
Filter window length	101
Window sidelobe attenuation	100 dB
Filter cutoff frequency	8 Hz
Max step frequency	2.5 Hz
Step window	0.4 s

Table 4.2: Table of parameters used in the step detection algorithm.

The parameters in the step detection algorithm were manually iterated and tested using "Simple #1" to obtain good step detection. The parameters are the *D-constant*, which denotes how much lower the next step's amplitude difference can be with respect to the previous. The constant threshold is the lowest amplitude difference that counts as a step. The maximum threshold is the maximum amplitude difference a step can be registered as. If it is a higher amplitude it is treated as it were the maximum threshold. FIR filter order denotes how large window the FIR filter will use when doing the low pass filtering. The filter cut-off frequency is the frequency where the normalised gain of the filter is -6 dB. Max step frequency is the maximum frequency the algorithm can handle without detecting two step in the same window of time. The max step frequency gives the step window which is the window of time where it is expected at maximum one step.

In figure 4.8 the signal analysis can be found. It shows both the norm or the accelerometer data and the norm of the filtered accelerometer data. The steps detected using the filtered data are shown as solid red lines and the dashed red lines are the actual time where the steps occurred. The reason for the difference in time is because of the filter delay.

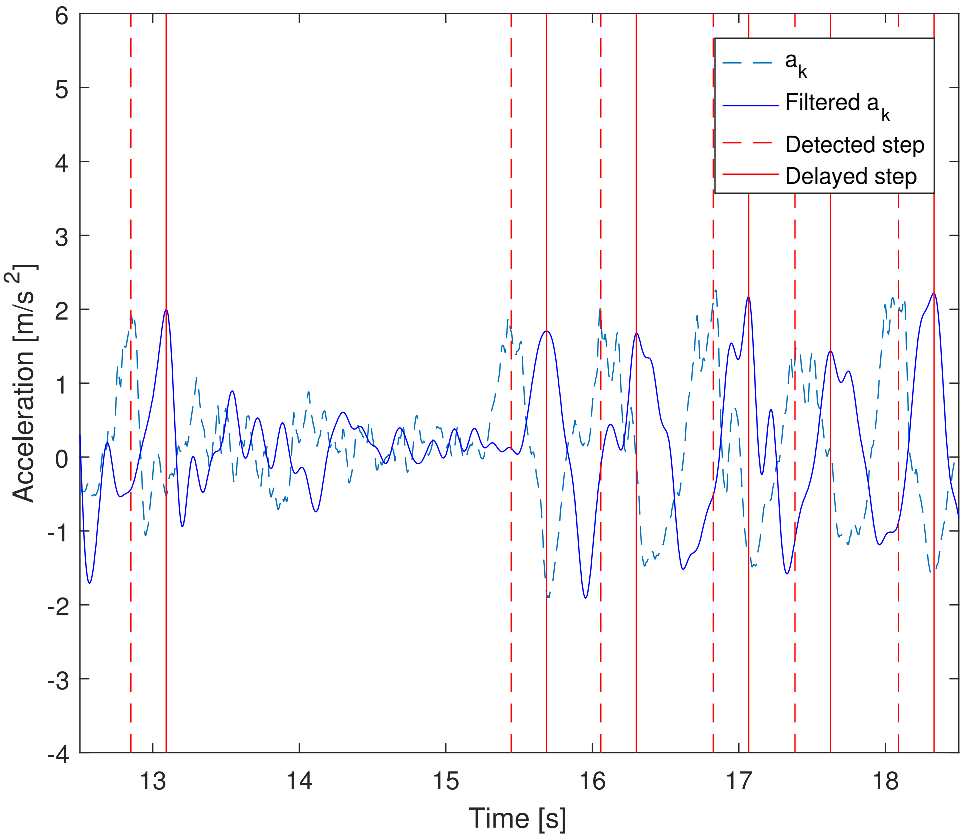


Figure 4.8: The step detection showing normalised and filtered accelerometer data with their detected step respectively.

A reference video was filmed during each route which was used to count the actual steps. In this way, the dependency on any external pedometer was eliminated and the correct number of steps taken was obtained.

The results of the step detection algorithm are shown in table 4.3. In this table, it is assumed that no false-positive detections occur, so the accuracy is the number of detected steps divided by the number of actual steps. During "Complete #1" the reference video was not started until a bit into the walk and the steps cannot be verified, hence the smaller number of steps compared to "Complete #2".

Route	**Detected steps**	**Actual steps**	**Accuracy**
Bed route 1	156	163	95.7%
Bed route 2	120	124	96.8%
Double slam	241	252	95.6%
Simple 1	118	127	92.9%
Simple 2	120	126	95.2%
Complete 1	172	189	89.6%
Complete 2	279	283	98.6%
Slam test	163	181	90.1%
Total	**1375**	**1448**	**94.7%**

Table 4.3: Table of the results from the step detection algorithm compared to the actual number of steps taken during the different routes.

4.3 User Orientation

The orientation filter is drifting a little because it has the gyrometer data as input. This is compensated for by measuring the static drift and subtracting this from the gyrometer signal. The amount of drift measured and identified in the mobile phone used was:

$$b_{gyro} = \begin{bmatrix} -0.044 \\ -0.004 \\ 0.087 \end{bmatrix} \tag{4.1}$$

The small deviations and drift in direction of heading, which still appear despite removing the drift, is taken care of later in the particle filter.

To visualise the result from the orientation filter, the following figures show the heading at each detected step. Each step has the same step length and due to this and to missed step detections the shape of the plots does not fit into the blueprint. They are deformed and are only to visualise the orientation filter output. Despite this the structure of the trajectory resembles the room layouts and it is possible to follow.

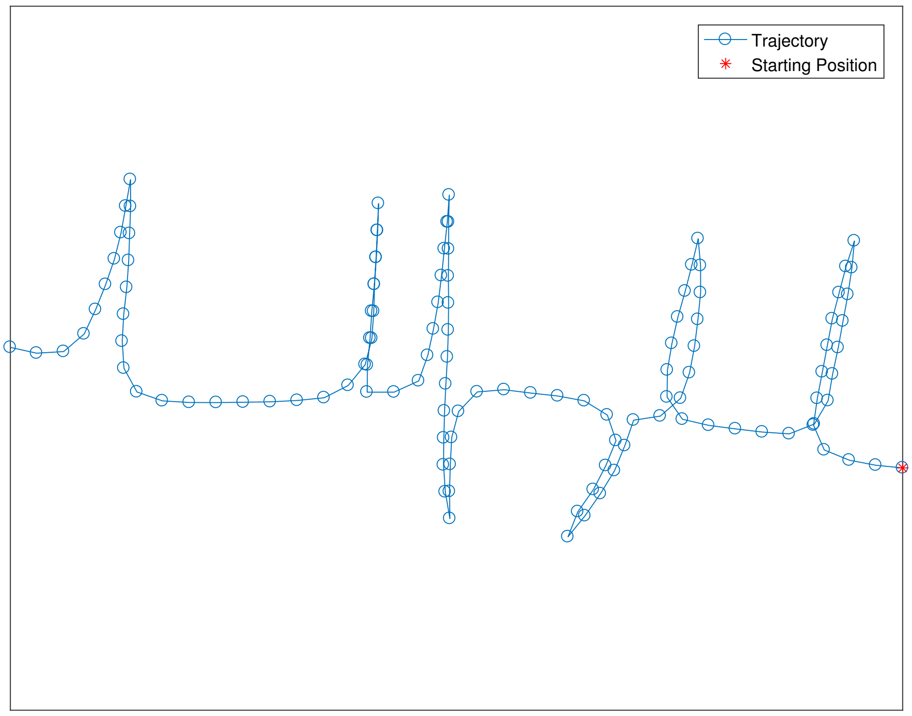


Figure 4.9: The estimated orientation for "Simple #1" with constant step length at each detected step.

In figure 4.9 the trajectory is created using the "Simple #1" walk, and the resemblance to the actual simple trajectory in figure 4.2 in section 4.1.1.

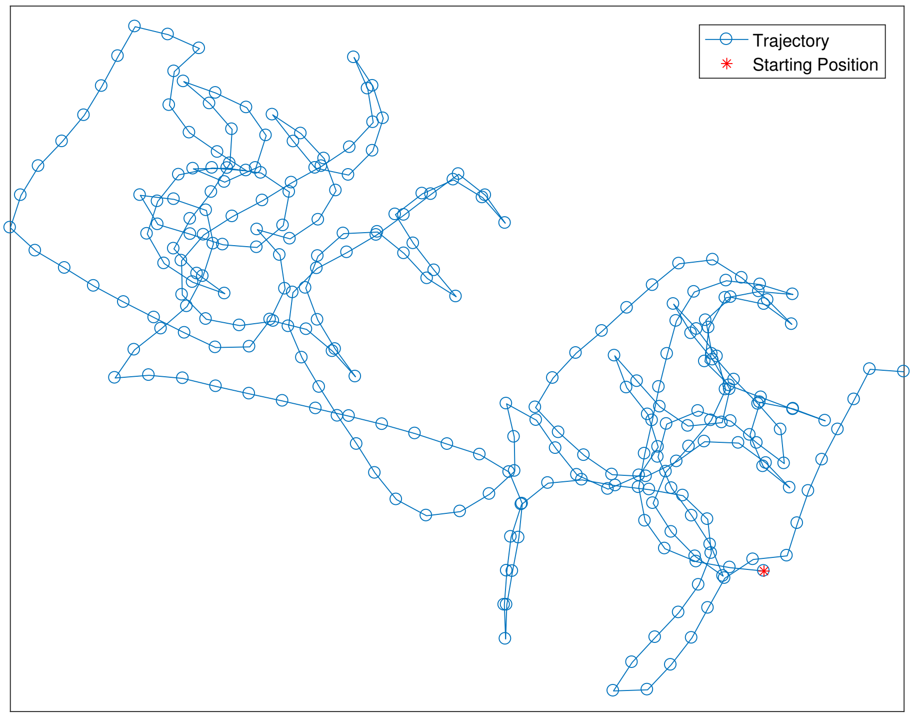


Figure 4.10: The estimated orientation for "Complete 1" with constant step length at each detected step.

Figure 4.10 with the "Complete 1" trajectory is not as easy to follow, but the structure can be found if the line is followed. All the orientation filter results can be found in appendix B.

4.4 User Positioning

In this section, different results from the positioning algorithm will be demonstrated. The section goes from showing a simple algorithm to the most complex combination of the parts described in the method chapter. The results will be shown in the form of the estimated trajectory from different data sequences. For this part, there is no comparable performance metric available, and the analysis will be from visual inspection. In a later section the validation of the created maps will be looked into.

The initial position and heading is known in all presented cases. Except for the first figure 4.11, all results are shown as smoothed trajectories. In figures 4.11 and 4.12 it is possible to view the improvement that the fixed lag smoothing accomplishes, where the former is shown without smoothing. A clear improvement is shown as compared to figure 4.2.

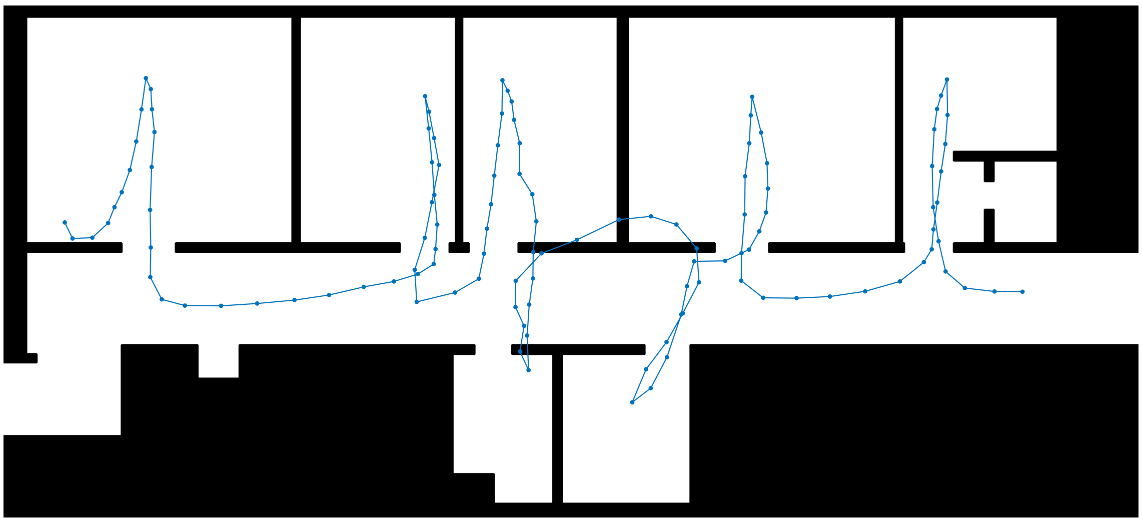


Figure 4.11: Results of the algorithm running with only odometry and blueprints, showing the filtered position estimate from data set "Simple #1". The same sequence but smoothed can be viewed in figure 4.12.

4.4.1 Blueprints Only

The first algorithm shown, runs a particle filter with the odometry as input, and using only the blueprints to calculate importance weights. The result for "Simple #1" is shown in figure 4.12, and for "Complete #1" in figure 4.13. With this relatively simple positioning method (with no external signals involved) a fairly good result is still obtainable in both cases, although with some larger errors for the latter route.

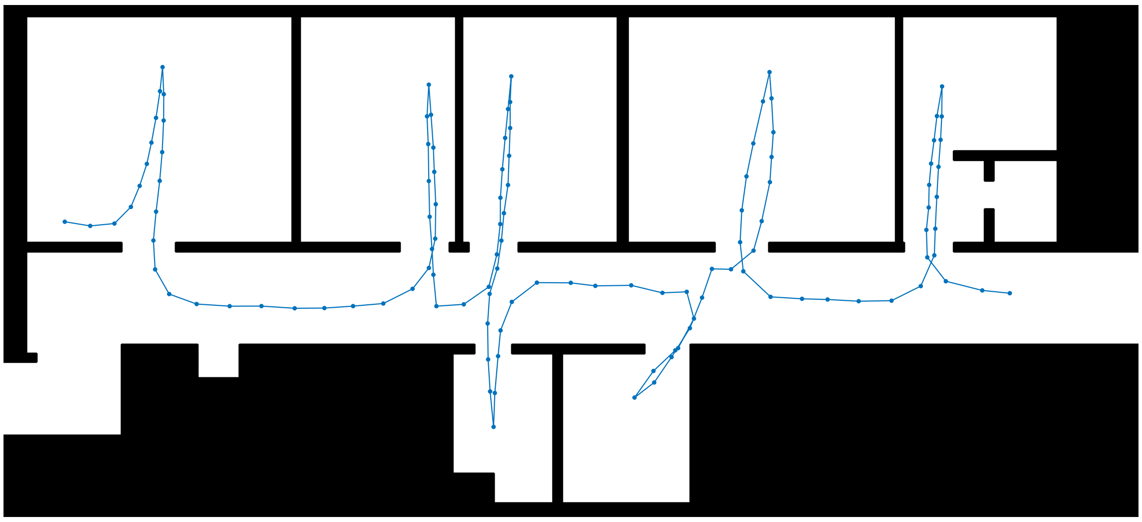


Figure 4.12: Results of the algorithm running with only odometry and blueprints, showing the smoothed position estimate from data set "Simple #1".

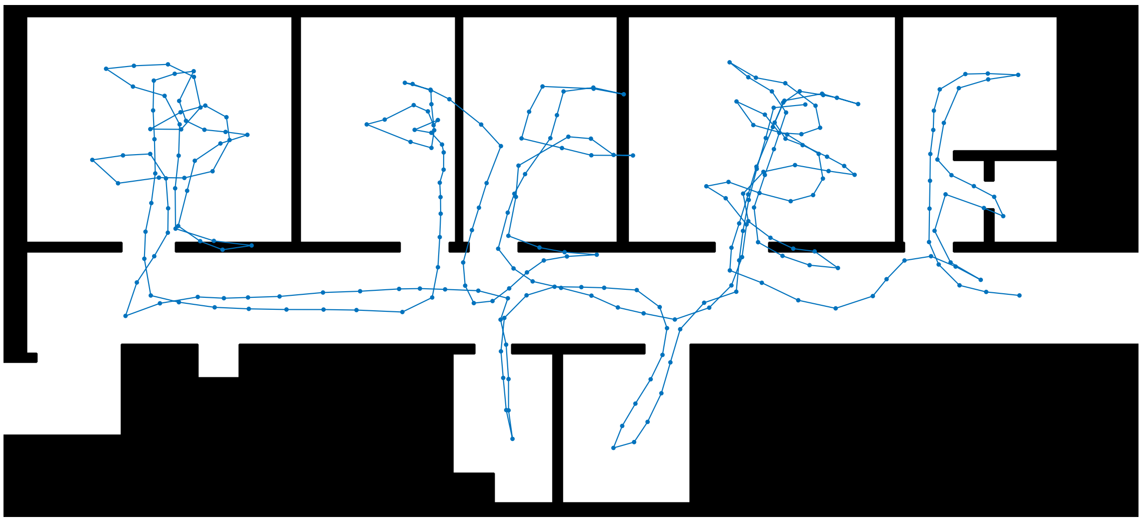


Figure 4.13: Results of the algorithm running with only odometry and blueprints, showing the smoothed position estimate from data set "Complete #1".

4.4.2 Blueprints and Static RSS Model

Next up, the RSS likelihood function is added, using a static predetermined transmission power $T_x = -56dBm$. A clear improvement is shown when adding the Bluetooth readings to the algorithm. The room accuracy is as good as can be determined visually.

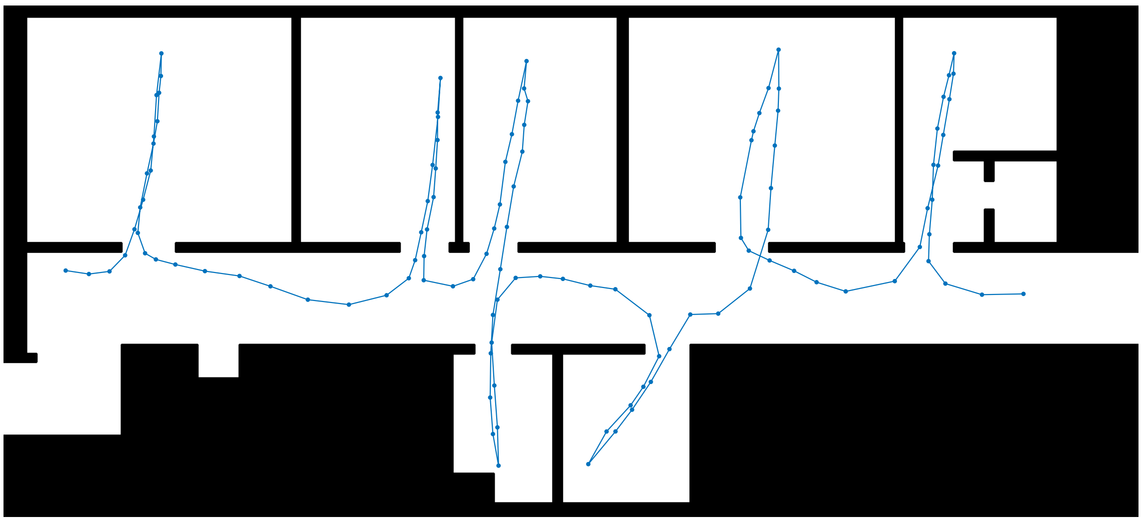


Figure 4.14: Results of the algorithm running with odometry, blueprints, and static bluetooth likelihood model. Showing the smoothed position estimate from data set "Simple #1".

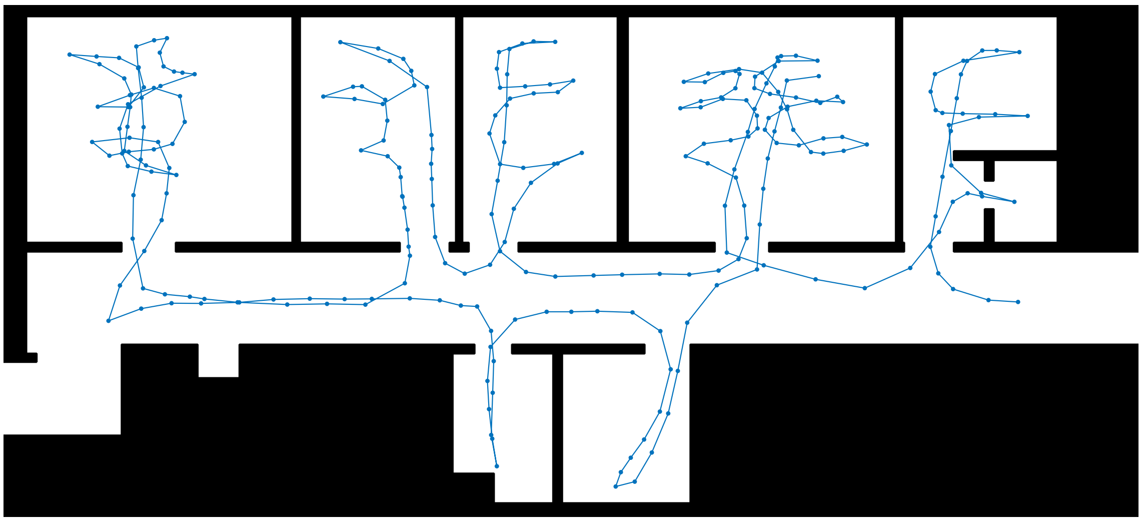


Figure 4.15: Results of the algorithm running with odometry, blueprints, and static bluetooth likelihood model. Showing the smoothed position estimate from data set "Complete #1".

4.4.3 Blueprints and Estimated RSS Model

For this subsection, the transmission power estimation is added to the algorithm, meaning each particle estimates each beacon's transmission power. The end results are similar to the previous subsection with minor improvements and declines in performance. Especially notable is that for the "Complete" route, the turning points at the beds are more distinguished in many cases.

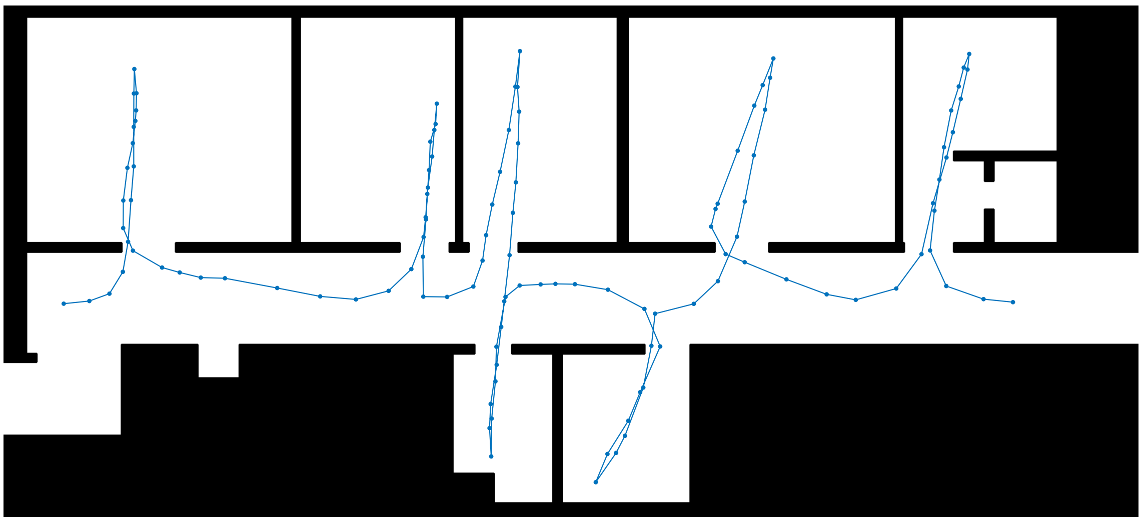


Figure 4.16: Results of the algorithm running with odometry, blueprints, and estimated bluetooth likelihood model. Showing the smoothed position estimate from data set "Simple #1".

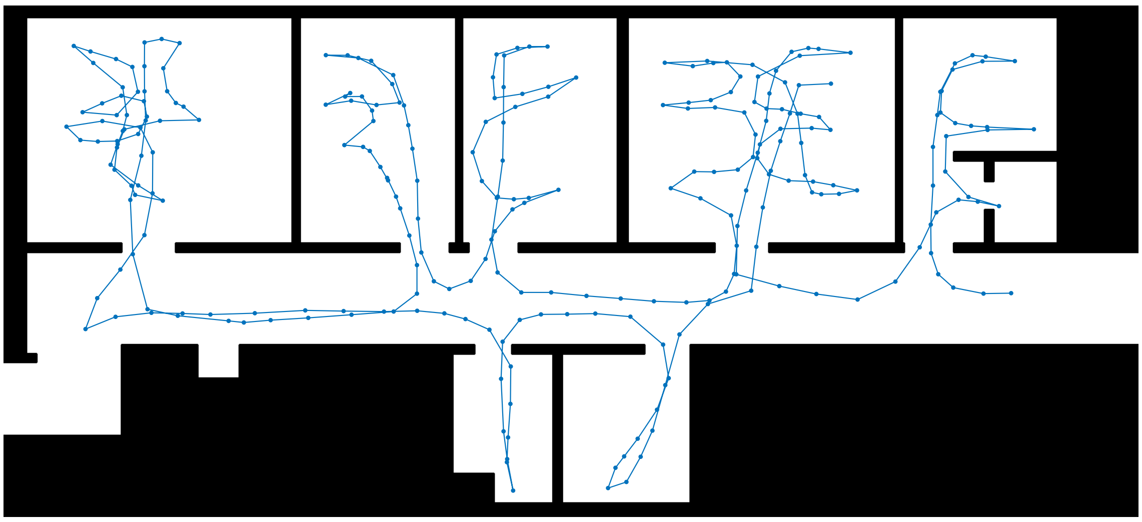


Figure 4.17: Results of the algorithm running with odometry, blueprints, and estimated bluetooth likelihood model. Showing the smoothed position estimate from data set "Complete #1".

4.5 Signal Strength Map Performance

In this section signal strength maps created from trajectories described earlier will be evaluated. In order to evaluate a map, the map will be used solely to estimate the recording position of 17 manual fingerprint locations. In this section the fingerprints will be described first, followed by the method of location estimation and then results will be presented in the form of mean errors.

The fingerprints were recorded by a user standing still with the mobile smartphone in hand, for approximately 20 seconds. Approximately 350 measurements in total were received per fingerprint. The fingerprint locations can be viewed in figure 4.18.

In this section maps are added together in order to create a map with a larger amount of information. In figure 4.18 it is possible to see which cells contain stored measurements from any beacon, as well as the total number of measurements stored in that cell. The figure shows the map which is constructed by adding together maps created from walks "Simple #1", "Simple #2", "Complete #1" and "Complete #2".

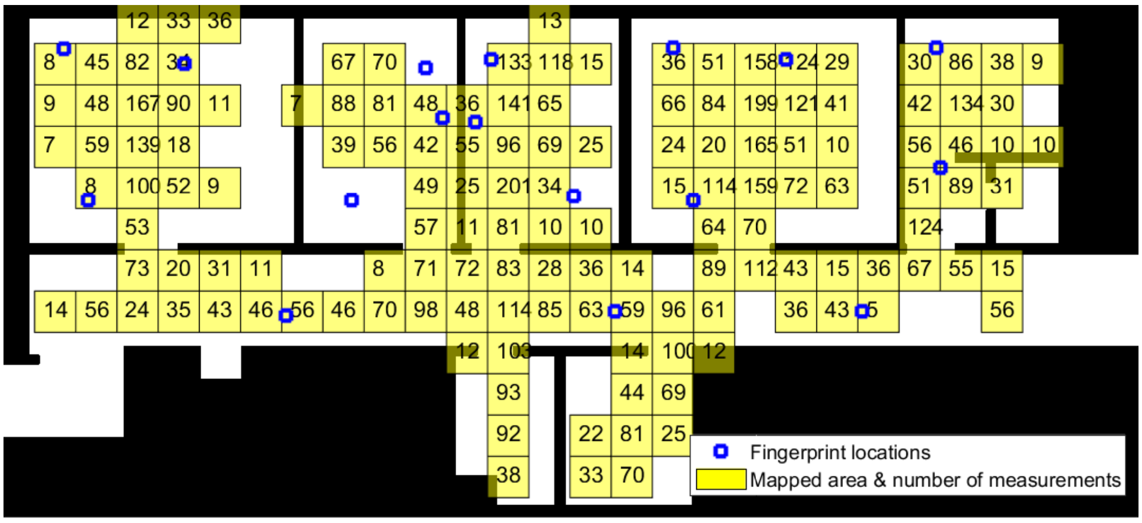


Figure 4.18: True recording positions of 17 fingerprints, and an illustration of which cells that have been mapped, in a map created from four different walks. Also featured is the number of measurements stored in that cell.

4.5.1 Fingerprint Recording Position Estimation

To estimate the position for each fingerprint, each map cell will be assigned a probability value. Likelihood functions for all beacons and cells are obtained by using the t-distribution described earlier, with the mean value depending on a prior guess and previous measurements. The probability values are then calculated as the product of all those likelihood functions, for each obtained measurement. After that the probability values are normalised such that the sum of all values is one. The probability value is

$$p_{x,y} \propto \prod_{\forall b} \prod_{n=1}^{N_m^b} t(z_n^b; \bar{z}_{x,y}^b, \sigma, 1) \tag{4.2}$$

where N_m^b is the number of measurements from beacon b. The mean value $\bar{z}_{x,y}^b$ is the mean value of the expected RSS and all previous measurement in the map cell. The expected RSS is according to (3.31), where either a static transmission power at $-56dBm$ or the estimated transmission power is used. In these experiments the scale parameter is $\sigma = 7.5$. The set of cell positions and corresponding normalised probability values can then be regarded as weighted samples from a posterior distribution over the position. The position estimate is then obtained by selecting the cell with highest value. It could also be selected as the weighted mean value, but in these experiments the difference was minimal.

The result in terms of errors is presented in table 4.4. It shows the mean error (ME) in meters and room error (RE) in number of wrong room estimates, for different test parameters for each row. The parameters are:

- **Map**: stating from which route the map was created, and what algorithm was used to create it. First the used walks are stated, where for example "S1+C1" means that maps from "Simple #1" and "Complete #1" are added together to create a new map. Secondly the elements of the map creation algorithm are stated, where "BP" means that blueprint matching was used and "T_x est." means that the transmission power was estimated.

- **Map checking algorithm**: explaining what algorithm was used to check a recorded fingerprint against the map. In this context "T_x est." means that the estimated transmission power in the map was used in the likelihood function to compare the cell against a measurement. In contrast "T_x stat." means that a static transmission power at $-56dBm$ is used. Weather the stored measurements are used or not is noted by "mean shift", meaning that the mean of the likelihood function is shifted according to above.

Furthermore, the map checking algorithms were run in three different settings deciding which map cells to be regarded:

- **All cells**: A probability value is calculated for all cells, regardless of the map information available for that cell. If no measurements are stored, only the expected RSS is used in the likelihood function.

- **Removing empty map cells**: If a map cell contains no previous measurements at all (not from any beacon), it is considered unreachable and is not used for positioning. This in practice demands that the area of interest is mapped thoroughly enough that all relevant cells have been visited.

- **Removing incomplete map cells**: If a map cell is missing information regarding a beacon from which the fingerprint recording has received a measurement, the map cell is not considered. To put it in another way, if the fingerprint contains a measurement from beacon b, but a map cell does not contain measurements from beacon b, that map cell is not considered further. This puts an even higher demand on thorough mapping.

These three settings are presented in three main columns in table 4.4.

		All		**w/o empty**		**w/o incomp.**	
Map	**Map check alg.**	ME	RE	ME	RE	ME	RE
-	T_x stat.	1.83	1	-	-	-	-
S1, BP	T_x est.	2.06	2	1.8	0	-	-
S1, BP	Mean shift, T_x stat.	1.67	3	1.93	2	-	-
S1, BP	Mean shift, T_x est.	1.94	3	2.01	2	-	-
C1, BP	Mean shift, T_x stat.	1.69	1	1.74	1	2.49	2
C1, BP	Mean shift, T_x est.	2	1	2.04	1	2.49	2
S1+C1, BP	Mean shift, T_x stat.	2.01	1	2.05	1	1.95	1
S1+C1, BP	Mean shift, T_x est.	2	1	1.91	1	1.95	1
S1, BP & T_x est.	Mean shift, T_x stat.	1.73	2	1.71	1	4.71	7
S1, BP & T_x est.	Mean shift, T_x est.	1.72	2	1.83	1	4.71	7
C1, BP & T_x est.	Mean shift, T_x stat.	2.06	1	2.04	1	2.05	1
C1, BP & T_x est.	Mean shift, T_x est.	2.06	1	2.06	1	2.05	1
S1+C1, BP & T_x est.	Mean shift, T_x stat.	1.88	2	1.86	2	1.6	0
S1+C1, BP & T_x est.	Mean shift, T_x est.	2.01	2	1.86	2	1.82	0
S1+C1+S2+C2, BP & T_x est.	Mean shift, T_x stat.	1.9	2	1.84	1	1.55	0
S1+C1+S2+C2, BP & T_x est.	Mean shift, T_x est.	2.26	1	2.12	1	1.55	0

Table 4.4: The mean error and number of room errors for fingerprinting position compared to the estimated position from the signal strength map. The three different scenarios, in different main columns, is where the algorithm is looking at all cells, removing empty cells and removing incomplete cells.

As a baseline the first row and first main column (using all map cells) of the table 4.4 is used. This is the case where no specific information regarding the environment is used, except for beacon positions. All other tests seek to add information regarding the BLE-RSS to improve the results. Now the different methods of adding this information are compared.

First it is noted that it is not simply the case that more information equals a smaller mean error. The baseline performs relatively well when keeping in mind the amount of work put into constructing the maps used in the other cases. By calculating the average mean error of different subsets of results it is possible to notice trends among the parameters.

It is noted that in general using the estimated transmission power when estimating the fingerprint location creates a result that is slightly worse than with a static transmission power. A similar result is noted when comparing how the map was created, where the maps created using transmission power estimation perform slightly worse overall. These results are somewhat surprising, since it is noted in section 4.4

that adding the transmission power estimation improved those results.

Next the different ways of excluding cells are compared. In most cases the results between main column 1 and main column 2 are similar. However, there is a slight trend that can be observed. That is that for more sophisticated maps (i.e. further down in the table), excluding map cells in some way becomes favourable. The trend is especially noticed for main column 3, where the simple maps performs very bad, and the most sophisticated map performs best of all tests. In figure 4.19 the positioning results from the best case is shown.

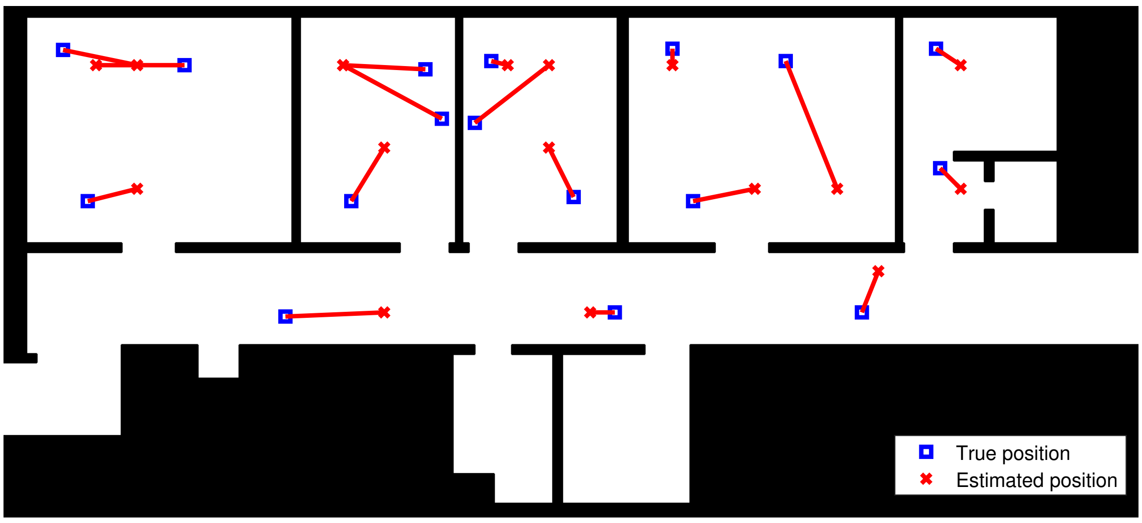


Figure 4.19: Positioning performance of 17 manual fingerprints and a map composed of data from four walks. The mean error is 1.55 meters.

4.5.2 Comparing Map Content to Fingerprints

It is possible to directly compare the manual fingerprint recordings to the map cell contents, of the true recording position. In figure 4.20 four examples of comparisons from one cell with a fingerprint recording is shown. They show a variety of different cases in how the measurements are distributed. For all beacon comparisons for the same fingerprint recording, see appendix C.

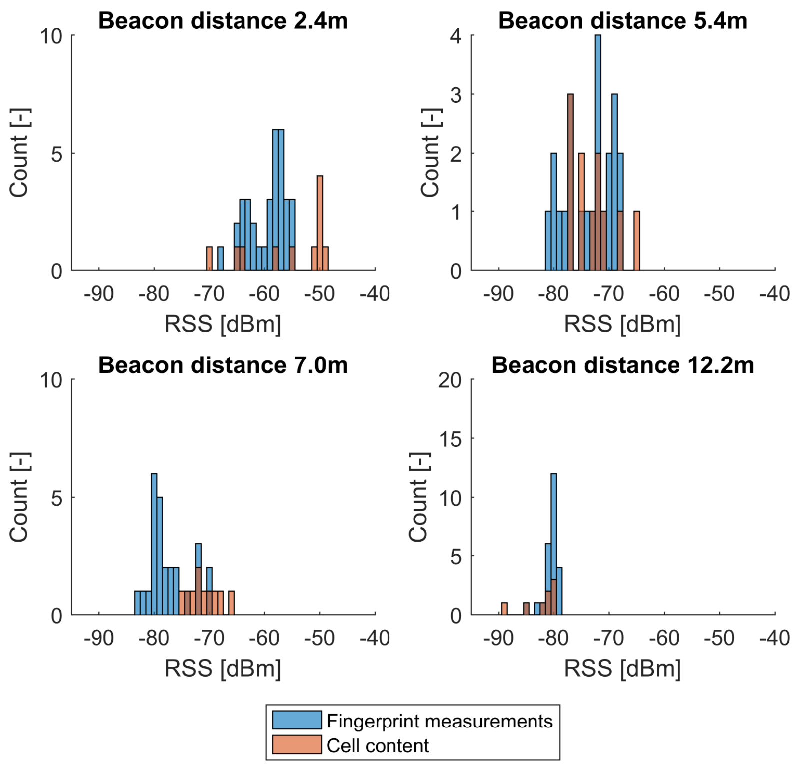


Figure 4.20: Comparison of fingerprint measurements and stored measurements in a map constructed from four walks. Results presented per beacon, here showing 4 out of 15 beacons, comparing the contents of the cell corresponding to the true recording position.

In many of the cases the gathered measurement distribution and the fingerprint matches fairly good when it comes to the mean value. The distributions are however differently shaped in many cases. It is noted that they are not likely t-distributed, as assumed in the mapping methods.

Note however, that the number of values available for both the map and fingerprint are low, making it difficult to accurately estimate the distributions. Another difference is that the fingerprint was recorded in one position, while the map cell represent all measurements collected from (estimated) positions within a one square meter area.

5 Discussion and future work

Overall the results are promising, showing that it is possible to estimate the recording positions from the user route. Note that this in itself is an IPS. It however requires the user to move around so that the system can pinpoint the location. This makes the system unsuitable as a complete solution in a healthcare environment, since that solution should be able to work under different circumstances as well. This is a big reason for constructing the received signal strength maps in the first place.

In the introduction of this thesis, four different evaluation metrics for an IPS in healthcare was introduced. One question may be how the proposed method evaluates against those:

- *Room accuracy*: While estimating the recording positions, the room accuracy is very high. Assuming that good estimation of recording positions leads to good maps, the system performs well on this point.
- *Latency*: This point falls more on the final IPS solution, i.e. the system that should utilise the created signal strength maps. This final solution is not a part of this thesis, but the latency of that system should be low.
- *Installation time/cost*: Installing the beacons is an easy task, as they are battery driven and are glued in place. Part of the installation is the mapping, the topic of this thesis, which is not very time consuming.
- *Energy consumption*: The beacons have an expected durability of at least two years according to Estimote [29] (depending on broadcast rate and power), at which point they need to be replaced. The mapping device uses the full range of on-board sensors and heavy calculations, so depending on the implementation the mapping device can subject to a large energy consumption. This should not be a problem, as it is a one-time task.

With our setup, it is possible to detect which room the person is in, but the accuracy of where in the room the person is positioned is not perfect. The number of beacons per room could possibly be increased to improve the accuracy, but the deployment and setup time would increase.

The environment of the tests has been a closed section of a hospital ward. Different hospitals have different building types. Some could have a lot of concrete walls, while others have more plaster walls. This could affect the signals ability to go through walls and this would lead to different beacon layout strategies, and possibly changed performance of the methods proposed.

A signal propagation environment may differ over time, meaning that the signal strength map becomes outdated. Refurnishing and renovations could result in a change in the map. The developed methods do not take care of this problem with changing signal strength maps. This is something that should be developed in a real system, and several solutions have been proposed for this issue in the literature.

5.1 Odometry

The step detection algorithm has a high overall accuracy when detecting steps. The walks it had to analyse were routes with many turns and turnarounds. This makes it harder for the algorithm because the type of movement and the step pattern change. It is probably during the turnarounds the algorithm misses steps the most. The detection misses could be due to a less distinct heel strike during the turns. It is during the turnarounds that the best precision in step detection is needed. One miss here can have a bigger impact on the position estimate compared to long straight walks.

The orientation filter performs well. It handles fast changes in direction well and does not drift much, even if it does slowly drift a little. It handles relative rotations well, but lack the ability to correct itself to the magnetic north fast enough. This is one improvement that would make the orientation filter better and make it much more suitable for a real-time implementation.

The implemented orientation filter is a filter design implementation for a different environment were the goal is to calculate the orientation of a mobile phone, when the phone is not moving around in the room. For this thesis, much work has been put into analysing signals and finding the correct parameters to make it work on a phone in motion.

The orientation filter design is a simple design and one thing to improve it is to estimate the gyroscope bias and accelerometer bias as suggested by Svensson in [17]. Another improvement would be if the filter could estimate the acceleration in the direction of the walk to help the particle filter to estimate the step length.

One thing to make the algorithms, both orientation filter and step detection, better is to make it able to work regardless how the mobile phone is held by the user. To make them able to detect and adapt to the direction the user is heading. There are solutions proposed to this problem in the literature as well.

5.2 Automatic Mapping

The proposed automatic mapping method is shown to be promising. It can estimate the recording positions with a decent accuracy with the knowledge of starting position, blueprint layout and beacon positions. This knowledge can be easy to incorporate for an end user, by uploading a blueprint (which can be automatically vectorised) and inputting the positions.

A potential drawback of the method is the reliance on particle filters. A particle filter can provide a good approximation for tough problems, but can also be subject to divergence and sample depletion. This problem is not visible in the results in this thesis, but was apparent during the course of creating this thesis. The use of blueprint matching greatly increases the performance of the position estimation, together with a simple smoothing algorithm. The live estimation of the beacons' transmission powers also increases the performance.

However, the results are not perfect. The actual routes taken during testing extends further into each room than the filter estimates. Also, the position estimate when walking the "Complete" routes is not very good when walking from bed to bed in a large room. This example is a hard problem to solve. Minor mistakes in odometry estimation can have large effects in that circumstance, and with the lack of any high-precision reference the filter cannot estimate the position very well. Comparing these results is a bit difficult. There was no good reference system available to compare the estimated recording positions. A third party indoor positioning system with high accuracy would be a key feature to have during the development of these methods. This could give the developers an exact reference point to where the person is located and it would be easier to examine the performance of the proposed methods.

In the results chapter, this thesis demonstrates a comparison between the maps created and manual fingerprint recordings. In these results, one should only consider general trends and not specific mean errors. This is since the number of fingerprints is not very large, which means that some variation in the results is to be expected due to coincidence. Slight trends are visible in the data, trends that are somewhat surprising. It seems that the use of transmission power estimation actually lowers the quality. It may seem that the filter constructed to estimate the transmission power does not fit the actual signal distribution. However, a better positioning result is obtained in the map construction, so how can that be? One possible explanation is that during the particle filtering, the estimated transmission power constantly changes depending on the recording position, and that this yields a good result. When trying to position the fingerprints, the transmission power levels in the map are only a good fit to the last estimated position from the map construction. Therefore, those levels do not fit for fingerprint positioning.

The most comparable results are in table 4.4, where the best result obtained is a mean error of 1.55 meters. A comparable result found in the literature is by Faragher and Harle [12], whom fingerprinted an environment of Bluetooth low energy beacons using a very high broadcast rate (giving more data to work with) and a highly accurate third party indoor positioning system to aid them. The result was a mean error of approximately one meter. In comparison with this thesis it seems reasonable, given the much larger amount of data available and the precise recording positions for Faragher and Harle. It should be noted that it is not clear how the map should be used in an optimal way to improve positioning though. This would be assisted by more knowledge of how Bluetooth signals are actually distributed, so that better likelihood functions can be constructed and used in positioning algorithms.

In several stages of this thesis, a t-distribution has been used as a likelihood function. This worked fairly good for the recording position estimation, with the help of odometry and blueprints. The calculated mean value of this distribution fits the actual received measurements pretty well. However, in other aspects the t-distribution does not fit the actual data collected, which can be seen in the results chapter. There are methods available for estimating a Gaussian probability distribution given samples from it, but for further development in this area it seems that more advanced probability distributions need to be used.

It would be of a high interest to further develop the proposed SLAM-algorithm. In the current method, certain map elements are estimated and used simultaneously. A preferable development in this regard would be to use previous measurements in a map cell, to directly affect the likelihood function of incoming measurements in that area while creating the map. This does however require some knowledge that is not readily available. As mentioned earlier, more knowledge regarding the expected distribution of received signals is needed. Also, it would be necessary to have an idea of how distributions in different areas are correlated. For example, does earlier received measurements in one area tell something about the distribution in other nearby areas? If so, how are they correlated?

On the same subject, the map structure could perhaps be different. Different sizes of cells could affect performance, but it is also possible that another structure would be beneficial. Perhaps each room could be treated as one cell. Another idea worth investigating is if specific areas share some parameters in a model, for example if each room could share parameters for the RSS model in equation 2.64.

The signals from the BLE beacons is fluctuating even if the device is still. The received signals also suffer from interference by multipathing and disturbances by the surroundings. For example, the human body is blocking some of the signal. The latter is a detail that has been disregarded by the mapping methods in this thesis. Given the orientation estimation it would be fully possible to include this information in the map. Exactly how is unclear however, but an example method that greatly increases the map complexity is to create different maps for different recording orientations. Another idea may be to use knowledge of beacon positions and user orientation to mitigate the problem.

To take other type of signals into account could result in better performance of the algorithms. Other signal systems than the BLE, such as Bluetooth transmitting on all channels and WiFi could be used to obtain a good result. It is for example possible to take advantage of the existing WiFi-access points to use them as signal beacons. The magnetic field is another example, where the received signal strength of the magnetic field in a position can be stored as a fingerprint and later used in the positioning algorithm.

6

Conclusion

This thesis shows that using a smartphone's on-board sensors, Bluetooth low energy beacons, and the operating area layout is sufficient for automatic signal strength mapping. A working step detection method is demonstrated with an overall accuracy of 94.7%, along with an orientation estimation method. The blueprint matching is very useful for estimating the recording positions. The method also utilises known Bluetooth beacon positions and a coarse model for received signal strength. It is shown that by estimating received signal strength model parameters and positions simultaneously increases the accuracy.

Overall this thesis demonstrates that this method accurately estimates the recording positions to the correct room, with some positioning error within each room. Furthermore, the created maps are tested against manual fingerprint recordings. In these tests a positioning mean error of down to 1.55 meters is achieved. Moreover, it is noted that a higher degree of knowledge regarding Bluetooth received signal strength distribution would be beneficial, both in further development of the thesis methods and in creating a way to utilise the created maps.

Bibliography

[1] K. Alfrink, “Messing with tupil’s estimote kit,” 2014, accessed 2017-01-10. [Online]. Available: https://www.flickr.com/photos/kaeru/14980173937

[2] M. Pixel, “Smartphone,” 2016, accessed 2017-01-18, original picture from Max Pixel (CC0) and modified by the authors. [Online]. Available: http://maxpixel.freegreatpicture.com/Mobile-Mobile-Phone-Smartphone-Phone-Smart-Phone-1138907

[3] T. Van Haute, E. De Poorter, P. Crombez, F. Lemic, V. Handziski, N. Wirstrm, A. Wolisz, T. Voigt, and I. Moerman, “Performance analysis of multiple indoor positioning systems in a healthcare environment,” *International Journal of Health Geographics*, vol. 15, no. 1, pp. 1–15, 2016. [Online]. Available: http://dx.doi.org/10.1186/s12942-016-0034-z

[4] A. L. Tucker and S. J. Spear, “Operational failures and interruptions in hospital nursing,” *Health Serv Res*, vol. 41, 2006. [Online]. Available: http://dx.doi.org/10.1111/j.1475-6773.2006.00502.x

[5] Y. Andrén, “Fullt flexibelt: Flexibilitet och generalitet i sjukhusbyggnader,” Sveriges Kommuner och Landsting, Tech. Rep., 2008.

[6] SahlgrenskaUniversityHospital, “Telefon/telephone,” (Accessed: 2016-09-08). [Online]. Available: https://www.sahlgrenska.se/om-ditt-besok-hos-oss/infor-din-sjukhusvistelseditt-sjukhusbesok/telefon/

[7] ——, “Sahlgrenska university hospital,” (Accessed: 2016-09-08). [Online]. Available: https://www2.sahlgrenska.se/en/sahlgrenska-university-hospital/in-english/

[8] B. Ferris, D. Fox, and N. D. Lawrence, “Wifi-slam using gaussian process latent variable models.” in *IJCAI*, vol. 7, no. 1, 2007, pp. 2480–2485.

[9] V. Moghtadaiee and A. G. Dempster, “Design protocol and performance analysis of indoor fingerprinting positioning systems,” *Physical Communication*, vol. 13, pp. 17–30, 2014.

[10] R. Faragher and R. Harle, “An analysis of the accuracy of bluetooth low energy for indoor positioning applications,” in *Proceedings of the 27th International*

Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014), Tampa, FL, USA, vol. 812, 2014.

[11] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[12] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, 2015.

[13] P. Mirowski, T. K. Ho, S. Yi, and M. MacDonald, "Signalslam: Simultaneous localization and mapping with mixed wifi, bluetooth, lte and magnetic signals," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, Oct 2013, pp. 1–10.

[14] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on.* IEEE, 2011, pp. 222–230.

[15] B. Siciliano and O. Khatib, *Springer handbook of robotics.* Springer Science & Business Media, 2008.

[16] Z.-Q. Zhang, X.-L. Meng, and J.-K. Wu, "Quaternion-based kalman filter with vector selection for accurate orientation tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 10, pp. 2817–2824, 2012.

[17] L. Svensson, "Project 1: Orientation estimation using smartphone sensors," 2014, unpublished. A project memo in the course SSY320: Sensor Fusion and Nonlinear Filtering at Chalmers University of Technology.

[18] S. Särkkä, V. Tolvanen, J. Kannala, and E. Rahtu, "Adaptive kalman filtering and smoothing for gravitation tracking in mobile systems," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on.* IEEE, 2015, pp. 1–7.

[19] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[20] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," 2010.

[21] R. Munguia and A. Grau, "Attitude and heading system based on ekf total state configuration," in *2011 IEEE International Symposium on Industrial Electronics*, June 2011, pp. 2147–2152.

[22] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[23] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[24] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, 2002, pp. 593–598.

[25] H. Nurminen, A. Ristimäki, S. Ali-Löytty, and R. Piché, "Particle filter and smoother for indoor localization," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on.* IEEE, 2013, pp. 1–10.

[26] S. Särkkä, *Bayesian filtering and smoothing.* Cambridge University Press, 2013, vol. 3.

[27] M. Klepal, S. Beauregard *et al.*, "A backtracking particle filter for fusing building plans with pdr displacement estimates," in *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on.* IEEE, 2008, pp. 207–212.

[28] Estimote, "Estimote (1.9.1)," [Mobile application software]. [Online]. Available: https://play.google.com/store/apps/details?id=com.estimote.apps.main

[29] ——, "Developer docs," (Accessed: 2017-01-07). [Online]. Available: http://developer.estimote.com/

[30] J. Rodas, C. J. Escudero, and D. I. Iglesia, "Bayesian filtering for a bluetooth positioning system," in *2008 IEEE International Symposium on Wireless Communication Systems.* IEEE, 2008, pp. 618–622.

[31] X. He, J. Li, and D. Aloi, "Wifi based indoor localization with adaptive motion model using smartphone motion sensors," in *2014 International Conference on Connected Vehicles and Expo (ICCVE).* IEEE, 2014, pp. 786–791.

[32] M. Marschollek, M. Goevercin, K.-H. Wolf, B. Song, M. Gietzelt, R. Haux, and E. Steinhagen-Thiessen, "A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons," in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.* IEEE, 2008, pp. 1319–1322.

[33] Bluevoid, "Ble analyzer (1.04)," [Mobile application software]. [Online]. Available: https://play.google.com/store/apps/details?id=nl.bluevoid.BLE_Analyzer2

[34] hfalan, "Sensor log (1.0.9)," [Mobile application software]. [Online]. Available: https://play.google.com/store/apps/details?id=com.hfalan.activitylog

A

Appendix: Orientation Filter Derivations

$$\dot{q}(t) = \frac{1}{2}S(\omega_{k-1} + v_{k-1})q(t) = \frac{1}{2}S(w_{k-1})q(t) + \frac{1}{2}S(v_{k-1})q(t) = \\ \frac{1}{2}S(\omega_{k-1})q(t) + \frac{1}{2}\bar{S}(q(t))v_{k-1} \tag{A.1}$$

$$q(t+T) = q(t) + \int_t^{t+T} \frac{1}{2}S(\omega_{k-1})q(t) + \frac{1}{2}\bar{S}(q(t))v_{k-1}d\tau = \\ (I + \frac{1}{2}S(\omega_{k-1})T)q(t) + \frac{1}{2}\bar{S}(q(t))Tv_{k-1} \tag{A.2}$$

With the piecewise constant property, the discrete signals is:

$$q_{k+1} = (1 + \frac{1}{2}S(\omega_k)T)q_k + \frac{1}{2}\bar{S}(q_k)Tv_k \tag{A.3}$$

Since the system is linear we can use superposition:

$$q_k = (I + \frac{1}{2}S(\omega_{k-1})T)q_{k-1} + \frac{1}{2}\bar{S}(q_{k-1})Tv_{k-1} \tag{A.4}$$

Identification gives

$$F(\omega_{k-1}) = (I + \frac{1}{2}S(\omega_{k-1})T) \tag{A.5}$$

and

$$G(q_{k-1}) = \frac{1}{2}\bar{S}(q_{k-1})T \tag{A.6}$$

$$\frac{dR(q)}{dq_0} = 2\begin{bmatrix} 2q_0 & -q_3 & q_2 \\ q_3 & 2q_0 & -q_1 \\ -q_2 & q_1 & 2q_0 \end{bmatrix} \tag{A.7}$$

$$\frac{dR(q)}{dq_1} = 2\begin{bmatrix} 2q_1 & q_2 & q_3 \\ q_2 & 0 & -q_0 \\ q_3 & q_0 & 0 \end{bmatrix} \tag{A.8}$$

$$\frac{dR(q)}{dq_2} = 2 \begin{bmatrix} 0 & q_1 & q_0 \\ q_1 & 2q_2 & q_3 \\ -q_0 & q_3 & 0 \end{bmatrix} \tag{A.9}$$

$$\frac{dR(q)}{dq_3} = 2 \begin{bmatrix} 0 & -q_0 & q_1 \\ q_0 & 0 & q_2 \\ q_1 & q_2 & 2q_3 \end{bmatrix} \tag{A.10}$$

B

Appendix: Orientation Estimation Results

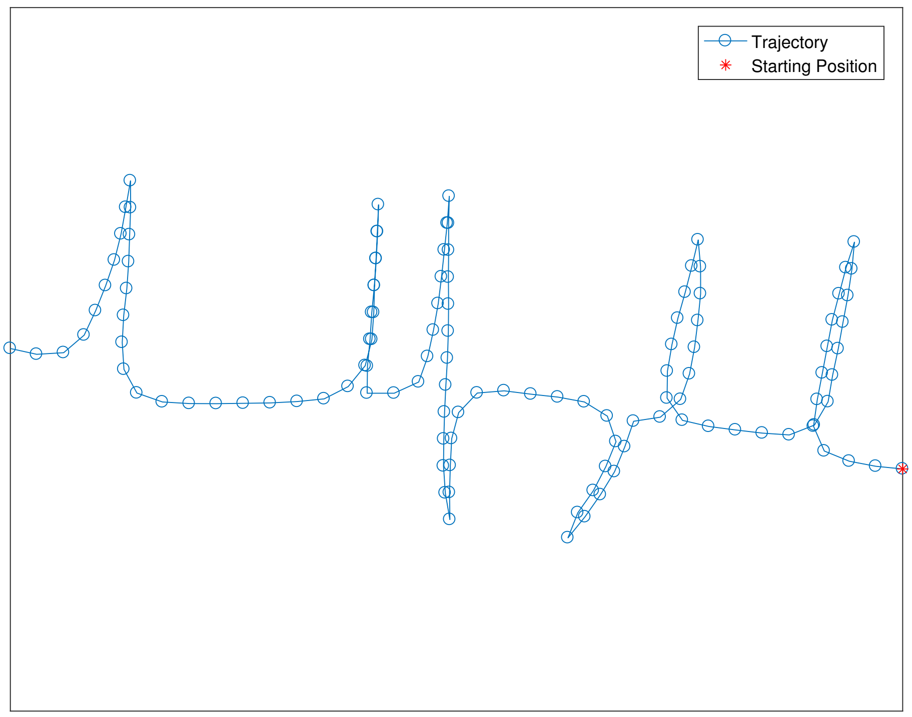


Figure B.1: The estimated orientation for "Simple #1".

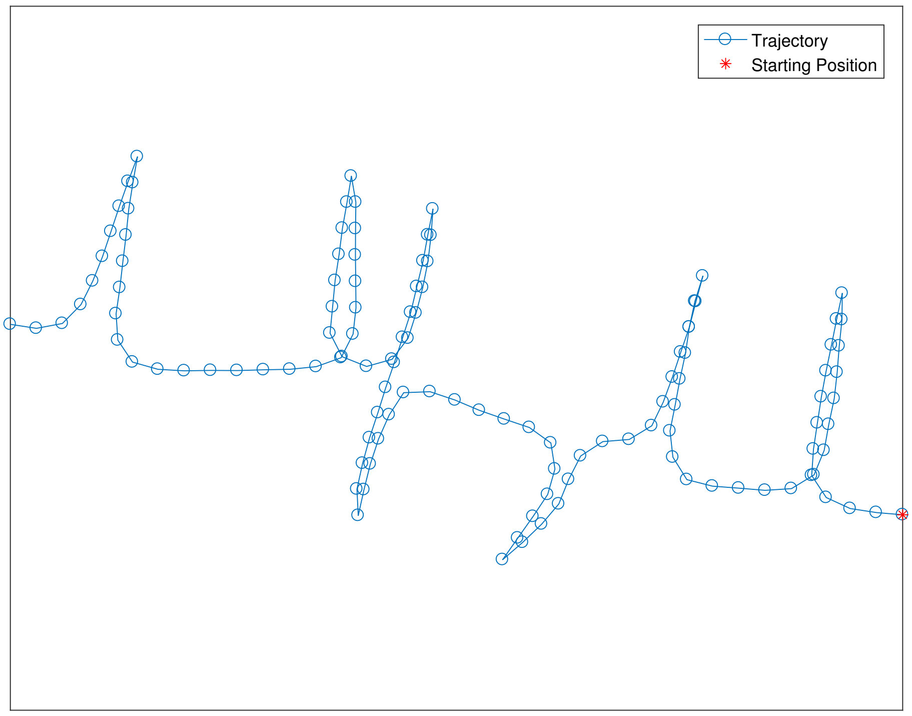


Figure B.2: The estimated orientation for "Simple #2".

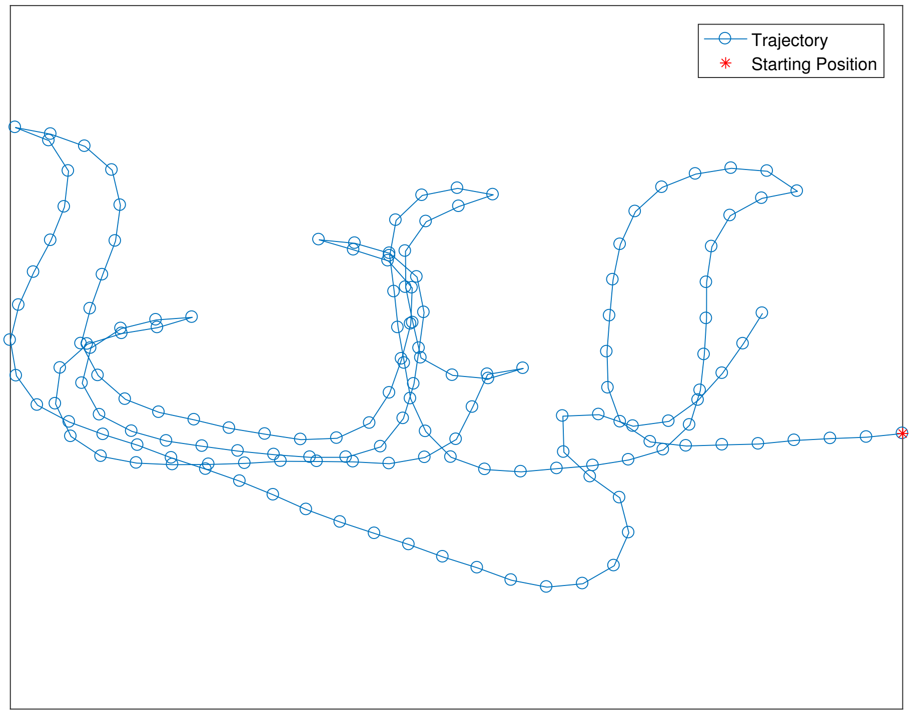


Figure B.3: The estimated orientation for "Bed route #1".

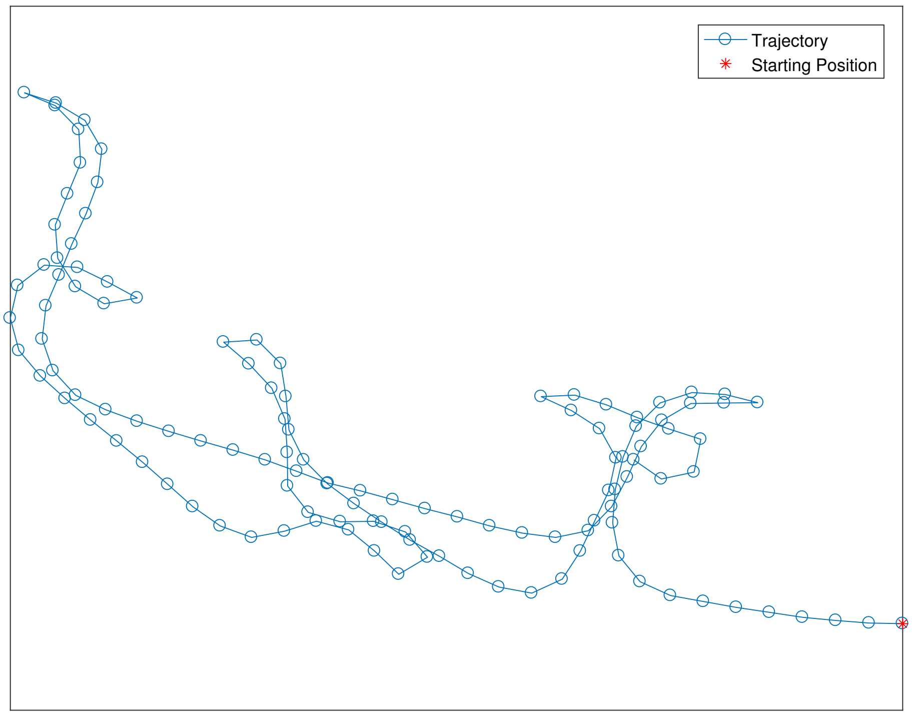


Figure B.4: The estimated orientation for "Bed route #2".

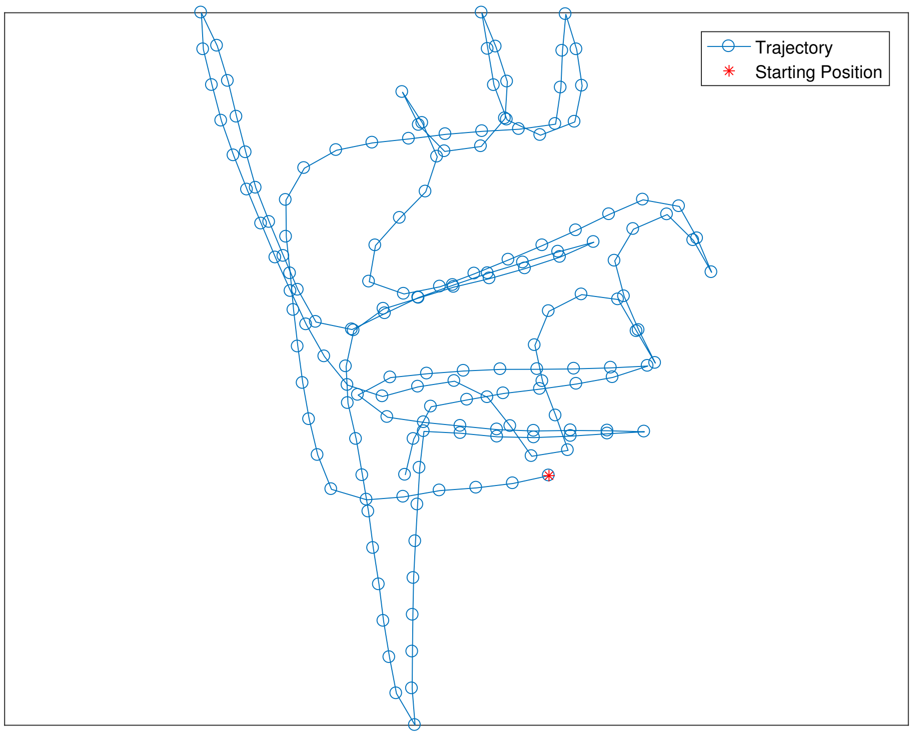


Figure B.5: The estimated orientation for "Slam test".

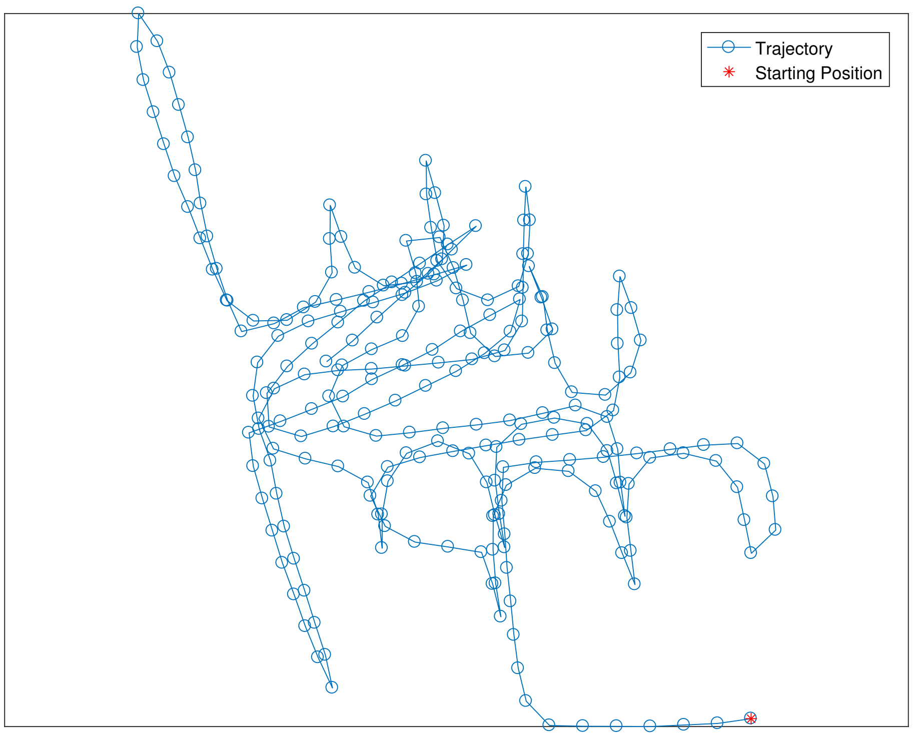


Figure B.6: The estimated orientation for "Double slam".

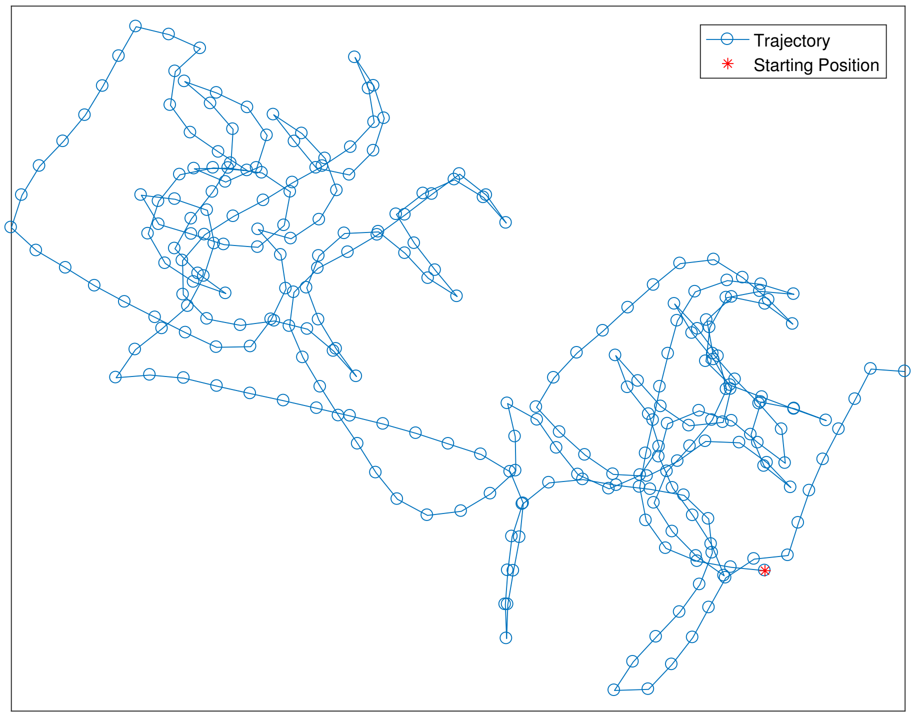


Figure B.7: The estimated orientation for "Complete #1".

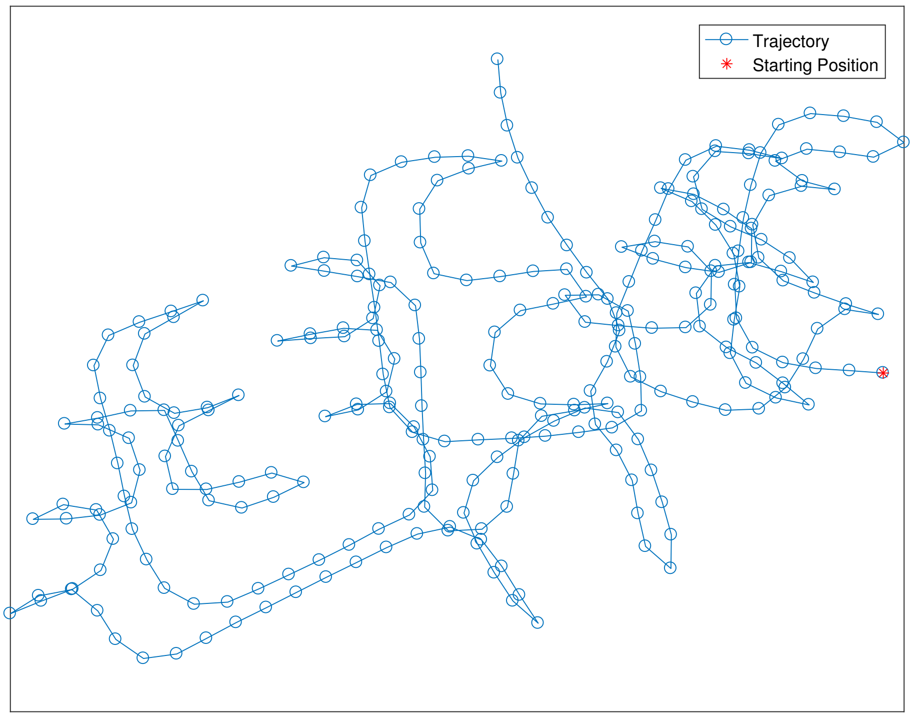


Figure B.8: The estimated orientation for "Complete #2".

C

Appendix: Fingerprint Comparisons

This appendix contains comparisons of a manual fingerprint recording and its corresponding cell in a signal strength map created from four different walks.

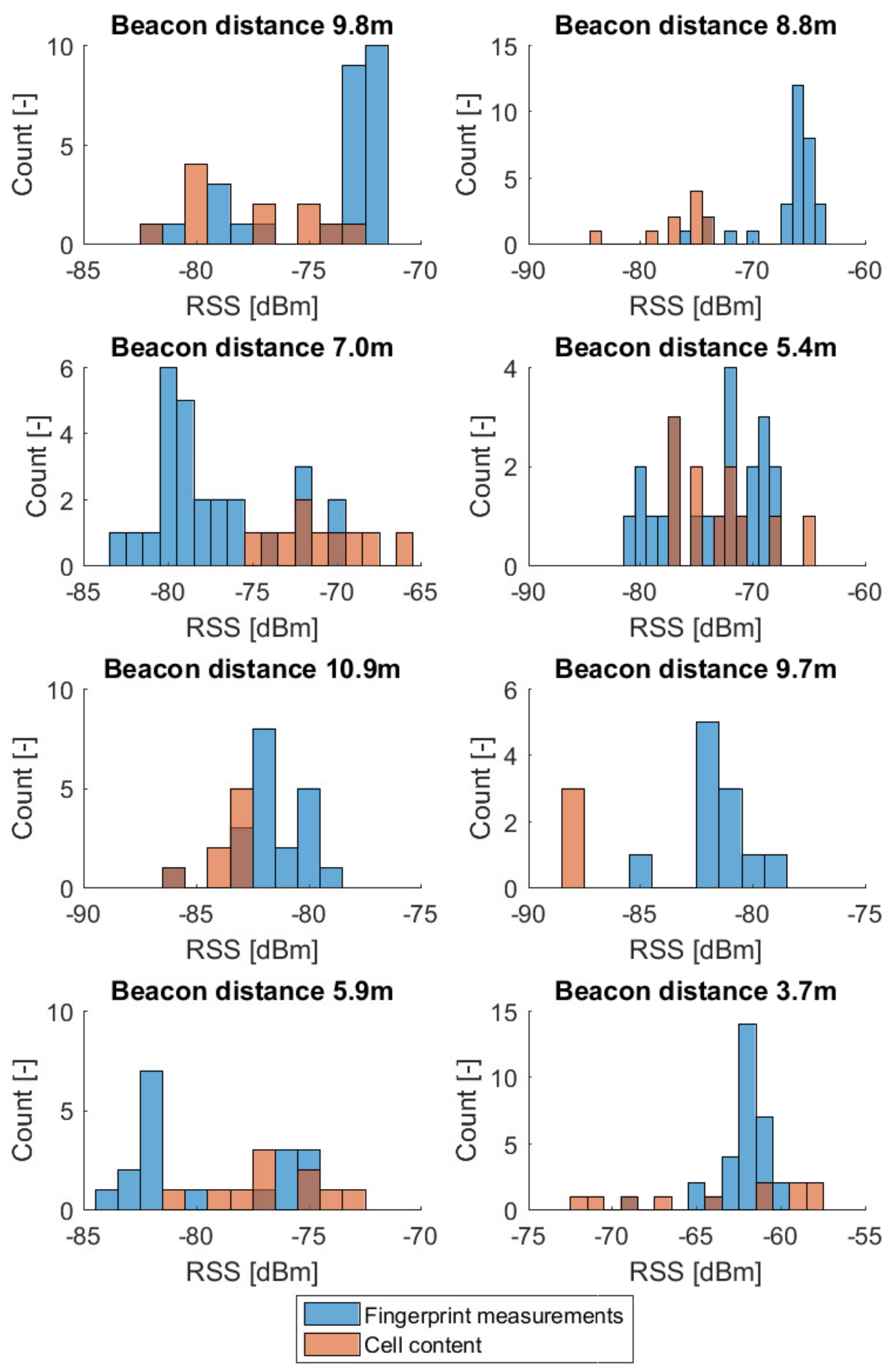


Figure C.1: Comparison of fingerprint measurements and stored measurements in a map constructed from four walks. Results presented per beacon, here showing 8 of 15 beacons. Figure C.2 contains comparisons from the same fingerprint but for the 7 remaining beacons.

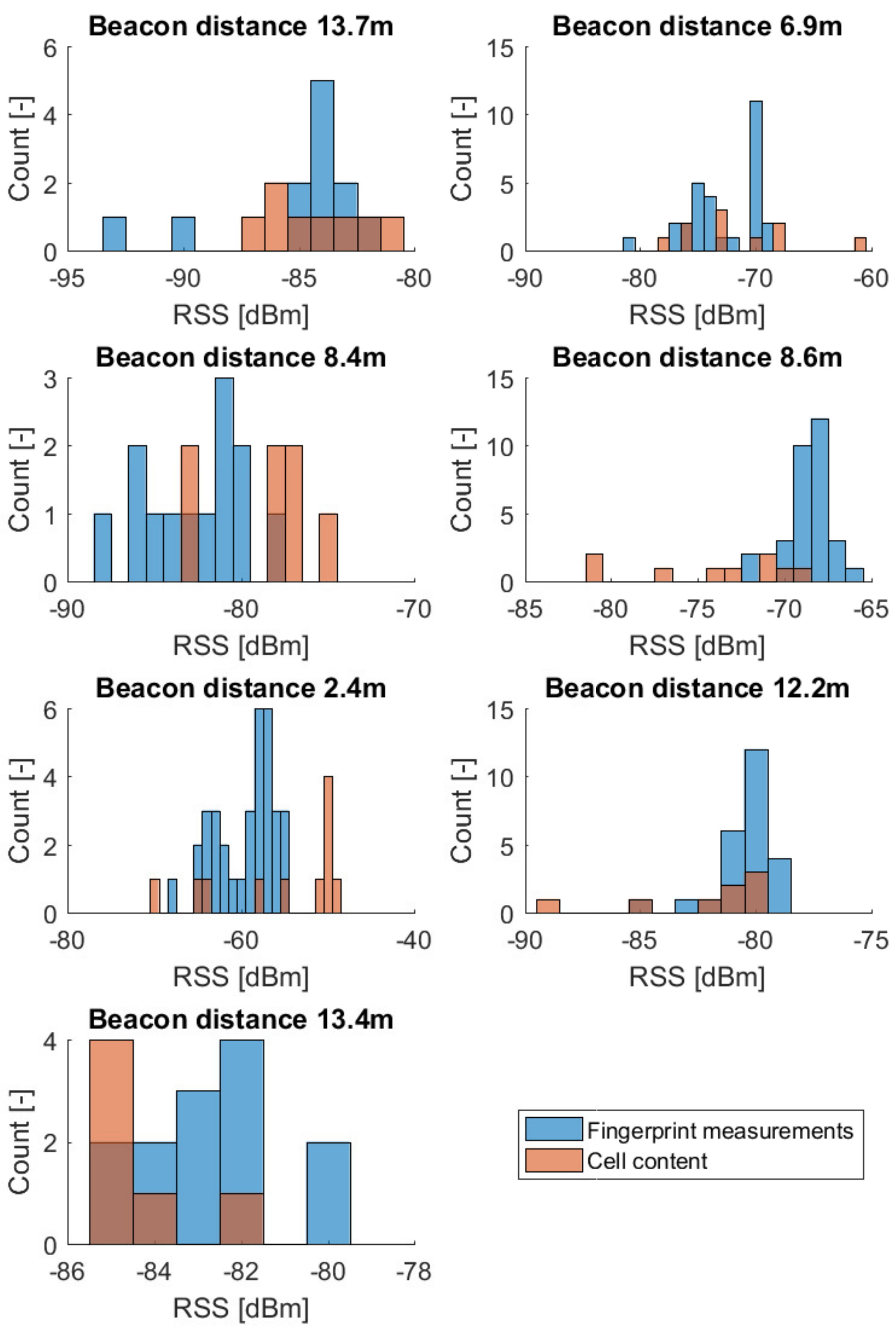


Figure C.2: Comparison of fingerprint measurements and stored measurements in a map constructed from four walks. Results presented per beacon, here showing 7 of 15 beacons. Figure C.1 contains comparisons from the same fingerprint but for the 8 remaining beacons.