

# Institutionen för systemteknik

## Department of Electrical Engineering

**Examensarbete**

## **Map Aided Indoor Positioning**

Examensarbete utfört i Reglerteknik  
vid Tekniska högskolan vid Linköpings universitet  
av

**Johan Kihlberg, Simon Tegelid**

LiTH-ISY-EX--12/4572--SE

Linköping 2012



**Linköpings universitet**  
**TEKNISKA HÖGSKOLAN**

Department of Electrical Engineering  
Linköpings universitet  
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola  
Linköpings universitet  
581 83 Linköping



# Map Aided Indoor Positioning

Examensarbete utfört i Reglerteknik  
vid Tekniska högskolan i Linköping  
av

**Johan Kihlberg, Simon Tegelid**


LiTH-ISY-EX--12/4572--SE

Handledare: **Manon Kok**  
isy, Linköpings universitet  
**Johan Östensson**  
Xdin, Linköping

Examinator: **Thomas Schön**  
isy, Linköpings universitet

Linköping, 25 May, 2012



	<b>Avdelning, Institution</b> Division, Department  Division of Automatic Control Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	<b>Datum</b> Date  2012-05-25
	<b>Språk</b> Language  <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English  <input type="checkbox"/> _____	<b>Rapporttyp</b> Report category  <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____
<b>URL för elektronisk version</b> <a href="http://www.control.isy.liu.se/">http://www.control.isy.liu.se/</a> <a href="http://www.ep.liu.se">http://www.ep.liu.se</a>		
<b>Titel</b> Title  <b>Författare</b> Author	Kartstödd inomhuspositionering Map Aided Indoor Positioning  Johan Kihlberg, Simon Tegelid Simon Tegelid	
<b>Sammanfattning</b> Abstract  <p>The popularity of wireless sensor networks is constantly increasing, both for use in static machine to machine environments as well as dynamic environments where the sensor nodes are carried by humans. Higher demands are put on real-time tracking algorithms of the sensor nodes, both in terms of accuracy and speed.</p> <p>This thesis addresses the issue of tracking persons wearing small sensor nodes within a radio network. Focus lies on fusing sensor data in an efficient way with consideration to the computationally constrained sensor nodes. Different sensors are stochastically modelled, evaluated, and fused to form an estimate of the person's position.</p> <p>The central approach to solve the problem is to use a dead reckoning method by detecting steps taken by the wearer combined with an Inertial Measurement Unit to calculate the heading of the person wearing the sensor node. To decrease the unavoidable drift which is associated with a dead reckoning algorithm, a map is successfully fused with the dead reckoning algorithm. The information from the map can to a large extent remove drift.</p> <p>The developed system can successfully track a person wearing a sensor node in an office environment across multiple floors. This is done with only minor knowledge about the initial conditions for the user. The system can recover from divergence situations which increases the long term reliability.</p>		
<b>Nyckelord</b> Keywords kalman filter, particle filter, indoor navigation, step detection, map, dead reckoning, 2.5D, time of flight, 802.15.4, sensor fusion		



# Abstract

The popularity of wireless sensor networks is constantly increasing, both for use in static machine to machine environments as well as dynamic environments where the sensor nodes are carried by humans. Higher demands are put on real-time tracking algorithms of the sensor nodes, both in terms of accuracy and speed.

This thesis addresses the issue of tracking persons wearing small sensor nodes within a radio network. Focus lies on fusing sensor data in an efficient way with consideration to the computationally constrained sensor nodes. Different sensors are stochastically modelled, evaluated, and fused to form an estimate of the person's position.

The central approach to solve the problem is to use a dead reckoning method by detecting steps taken by the wearer combined with an Inertial Measurement Unit to calculate the heading of the person wearing the sensor node. To decrease the unavoidable drift which is associated with a dead reckoning algorithm, a map is successfully fused with the dead reckoning algorithm. The information from the map can to a large extent remove drift.

The developed system can successfully track a person wearing a sensor node in an office environment across multiple floors. This is done with only minor knowledge about the initial conditions for the user. The system can recover from divergence situations which increases the long term reliability.





# Sammanfattning

Intresset för trådlösa sensornätverk ökar konstant, såväl för statiska maskintill-maskintillämpningar som för dynamiska miljöer där sensornoderna är burna av människor. Allt högre krav ställs på positioneringsalgoritmer för sensornätverken, där både hög precision och låg beräkningstid ofta är krav.

Denna rapport behandlar problemet med att bestämma positionen av personburna sensornoder. Rapportens fokus är att effektivt kombinera sensordata med hänsyn till sensornodernas begränsade beräkningskapacitet. Olika sensorer modelleras stokastiskt, utvärderas och kombineras för att forma en skattning av sensornodens position.

Den huvudsakliga metoden för att lösa problemet är att dödräkna sensornodbärarens steg kombinerat med kompass och tröghetssensorer för att skatta stegets riktning. En karta över byggnaden används för att reducera den annars oundvikliga drift som härrör från dödräkning. Informationen från kartan visar sig i stor utsträckning kunna reducera den här driften.

Det utvecklade systemet kan följa en person genom en kontorsmiljö som sträcker sig över flera våningsplan. Detta med enbart lite information om personens initiala position. Systemet kan även återhämta sig från situationer där algoritmen divergerar vilket ökar systemets pålitlighet på lång sikt.



# Acknowledgments

We would like to thank Thomas Schön, our examiner, for listening, giving inspiration, and with great enthusiasm and dedication helped us along through each step of the thesis.

A big thank to our supervisors, Manon Kok and Johan Östensson, for giving us good advice and help to achieve the goals of the thesis.

We would like to thank Arwid Komulainen for being our opponent for this thesis. His comments and thoughts have been an aid in the finalization of the report and have given us some new, valuable perspectives on our work.

Finally a thank to Xdin Linköping for providing us with the necessary means to perform our thesis, and a pinball machine at their office in Linköping.

This thesis has been a great conclusion to our studies at Linköpings University thanks to all people involved. With this report we conclude five years of electrical engineering studies.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Related work . . . . .	1
1.3	Purpose and scope . . . . .	2
1.4	Available hardware . . . . .	2
1.5	Xdin . . . . .	2
<b>2</b>	<b>Probabilistic modeling</b>	<b>5</b>
2.1	Coordinate frames . . . . .	5
2.2	Radio . . . . .	6
2.2.1	Time of Flight . . . . .	6
2.2.2	Received signal strength . . . . .	8
2.3	Inertial sensors . . . . .	9
2.3.1	Accelerometer . . . . .	9
2.3.2	Gyroscope . . . . .	9
2.4	Magnetometer . . . . .	10
2.5	Map . . . . .	11
2.5.1	Representation . . . . .	12
2.5.2	Map structure . . . . .	12
2.5.3	Map interpretation . . . . .	13
2.5.4	Map interface . . . . .	16
2.6	Pedestrian motion model . . . . .	17
2.6.1	Step model . . . . .	17
<b>3</b>	<b>State estimation</b>	<b>19</b>
3.1	Kalman filter . . . . .	19
3.1.1	Extended Kalman filter . . . . .	21
3.2	Particle filter . . . . .	22
3.2.1	Resampling . . . . .	22
3.3	Marginalized particle filter . . . . .	22
<b>4</b>	<b>Approach</b>	<b>27</b>
4.1	Subsystems . . . . .	27
4.1.1	Inertial Measurement Unit . . . . .	27

4.1.2	Step detection . . . . .	29
4.2	Linear map insertion through additive forces . . . . .	33
4.3	Particle filter based approaches . . . . .	36
4.3.1	Particle Filter – Light . . . . .	36
4.3.2	Particle Filter – Heavy . . . . .	40
4.4	Evaluation . . . . .	43
<b>5</b>	<b>Implementation</b>	<b>49</b>
5.1	Wireless client . . . . .	49
5.1.1	Inertial Measurement Unit . . . . .	50
5.1.2	Step detection . . . . .	50
5.1.3	Scan results . . . . .	50
5.1.4	Communication format . . . . .	50
5.2	Server . . . . .	51
5.2.1	Random number generation . . . . .	51
5.2.2	Particle filter utilities . . . . .	53
5.2.3	Map . . . . .	55
5.2.4	Quaternion library . . . . .	55
<b>6</b>	<b>Concluding remarks</b>	<b>57</b>
6.1	Conclusions . . . . .	57
6.2	Future work . . . . .	58
6.2.1	Particle filter implementation on GPUs . . . . .	58
6.2.2	Navigation . . . . .	58
6.2.3	RSSI fingerprinting . . . . .	58
6.2.4	Multi channel time of flight . . . . .	58
6.2.5	SLAM . . . . .	59
6.2.6	Automatic map generation . . . . .	59
6.2.7	Improved map system . . . . .	59
<b>A</b>	<b>Quaternion Algebra</b>	<b>61</b>
A.1	Basic notation . . . . .	61
A.2	Spatial rotation . . . . .	61
<b>B</b>	<b>Hardware</b>	<b>63</b>
B.1	Beebadge . . . . .	63
B.1.1	Accelerometer . . . . .	63
B.1.2	Gyroscope . . . . .	64
B.1.3	Magnetometer . . . . .	64
B.1.4	Pressure sensor . . . . .	64
<b>C</b>	<b>Time of flight</b>	<b>65</b>
C.1	Calibration . . . . .	65
C.2	Initiation . . . . .	65
C.3	Performance . . . . .	66

<b>D RSSI</b>	<b>71</b>
D.1 Identification of measurement model . . . . .	71
D.2 Performance . . . . .	71
<b>E Step length estimation</b>	<b>75</b>
E.1 Step length from step rate . . . . .	75
E.2 Step length estimation in particle filter . . . . .	75
<b>Bibliography</b>	<b>77</b>





# Glossary

## Coordinate frames

<i>b</i>	Body coordinate frame
<i>p</i>	Plane coordinate frame
<i>h</i>	Section coordinate frame
<i>s</i>	Sensor coordinate frame
<i>w</i>	World coordinate frame

## Operators

$\odot$	Quaternion multiplication
$\ \cdot\ _2$	Second norm

## Terminology

<b>beebadge</b>	The product name of the wireless, battery driven device
<b>coordinator</b>	The radio base station. Mains powered and Ethernet connected
<b>endpoint</b>	Referring to a Beebadge in radio contexts
<b>IEEE 802.15.4</b>	Low-cost IEEE radio standard



# Acronyms

<b>2.5D</b>	Two dimension representation with limited altitude
<b>EKF</b>	Extended Kalman filter
<b>GPS</b>	Global Positioning System
<b>GPU</b>	Graphics Processing Unit
<b>IMU</b>	Inertial Measurement Unit
<b>KF</b>	Kalman filter
<b>KLD</b>	Kullback-Liebler distance
<b>LOS</b>	Line of sight
<b>MAP</b>	Maximum a posteriori
<b>MEMS</b>	Microelectromechanical system
<b>ML</b>	Maximum likelihood
<b>MMSE</b>	Minimum mean squared error
<b>MPF</b>	Marginalized particle filter
<b>NLOS</b>	Non line of sight
<b>PF</b>	Particle filter
<b>RSSI</b>	Received signal strength indicator
<b>SLAM</b>	Simultaneous location and mapping
<b>TOF</b>	Time of flight



# Chapter 1

## Introduction

### 1.1 Background

Global Positioning System (GPS) provides an excellent method for positioning of wireless devices, but areas with limited or no coverage may have severe impact on applications which require continuous knowledge of the position. For the purpose of indoor navigation, GPS is not a viable option due to poor coverage. Instead, inertial sensors are often used in dead reckoning algorithms. While such algorithms can give good short term results and can improve GPS positioning when GPS is available, the position tends to drift in the long term. This is caused for example by double integrating linear acceleration, which is often the case when an inertial measurement unit is used as input to a dead reckoning algorithm.

To overcome these problems, a map of a building or a road can greatly reduce or even remove the drifting problems completely caused by dead reckoning. By using a map one can start a dead reckoning algorithm with an unknown initial position and quickly find the absolute position after some unique movement patterns [1, 2, 3].

### 1.2 Related work

Time of Flight measurements for range estimation are becoming increasingly popular in the literature [4, 5, 6, 7, 8, 9]. Most previous work is focused on high precision in terms of position, but lightweight algorithms are becoming more popular for implementation in computationally constrained devices [9]. In this thesis focus will partly be on reducing the needed computations and reducing the energy consumption. Related work for fusing various sensor data with a map to improve the position estimate includes [1, 2, 3, 10, 11, 12]. A big challenge compared to the previous work is trying to reduce the computational complexity. Some work on real-time map-matching applications have been done, often with too tight constraints for the application being useful in practice [13]. Previous

work on step detection includes [14, 15].

### 1.3 Purpose and scope

The purpose of this thesis is to develop and evaluate algorithms for sensor node positioning with map aid in a sensor network, the target precision is to locate the user within a few meters and more precisely which room the user is currently in. The sensor network includes sensor nodes, radio coordinators and some centralized computer that controls the network. For the algorithms to be useful in practice, the developed positioning system must be able to run in real-time and the energy and performance constraints of the sensor nodes must be considered. A task of the thesis is to find a division of algorithms between the different parts of the network to improve overall performance and reduce energy costs of the sensor nodes.

The sensor nodes, running a lightweight operating system, have the ability to collect sensor data and communicate with the centralized computer over the radio.

An Inertial Measurement Unit (IMU) algorithm is already available in the embedded software, which is why only a brief description thereof is in the scope of this thesis.

### 1.4 Available hardware

The sensor nodes are called *Beebadges* and one example is depicted in Figure 1.1(a). In radio contexts the Beebadges are referred to as endpoints.

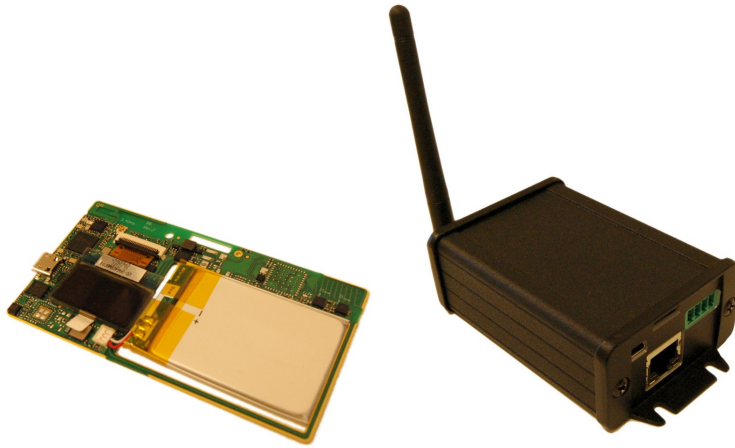
In similarity with a mobile telephony network, a base station is responsible for routing traffic to and from Beebadges. The base station is referred to as a *Coordinator* and is shown in Figure 1.1(b). The current implementation of the coordinators does not allow for performing calculations on them, but does only route traffic between Beebadges and the centralized computer.

The Beebadges are thought of being carried in a belt clip by pedestrians. In Figure 1.2 a Beebadge is shown while worn by a person.

### 1.5 Xdin

Through an acquisition of Enea Experts, Xdin employs over 1100 people in the energy, telecom, manufacturing, and automotive industries. With offices in Stockholm, Gothenburg, Linköping, Lund, and Västerås, Xdin is a large actor on the Swedish consultancy market.

Apart from the industries already mentioned, the skills that Enea Experts' consultants possess makes Xdin offer solutions for a larger set of electronic fields, including machine to machine (M2M) communication, embedded Linux and software quality solutions.



(a) A Beebadge, carrying a number of sensors and a IEEE 802.15.4 radio with a radio chip and an Ethernet chip.  
(b) A coordinator, equipped both with a radio chip and an Ethernet port, serving as a base station for the Beebadges.

**Figure 1.1.** The two main components of the radio network.



**Figure 1.2.** Beebadge worn by a man.





## Chapter 2

# Probabilistic modeling

This chapter describes the available sensors and how they relate to the positioning algorithms. The embedded platform is equipped with one 3-axis accelerometer, one 3-axis gyroscope, one 3-axis magnetometer, and one combined pressure and temperature sensor. Hardware description of the sensors can be found in Appendix B. Further, the hardware supports the possibility of both measuring the time of flight for radio waves between radio units and measuring the received signal strength. This can be used to estimate the relative position of the involved embedded devices. A similar system has been described in [4] which uses a similar type of hardware for positioning.

### 2.1 Coordinate frames

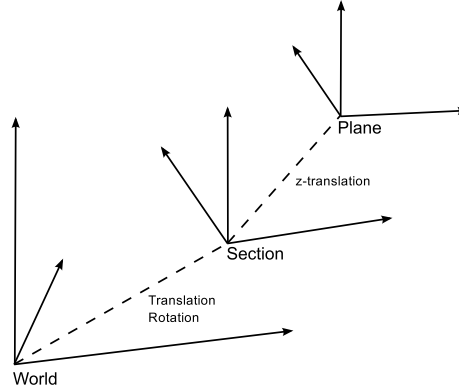
Different coordinate frames are used in this thesis to simplify notation when representing vectors in different coordinate systems. The frames and their relationships are illustrated in Figure 2.1.

**World frame ( $w$  - frame)** The world frame is the main frame, the  $x$ -axis is aligned so that it points towards magnetic north, and the  $z$ -axis is aligned to the gravitational pull of the earth.

**Section frame ( $h$  - frame)** The difference between the world frame and a section frame is that it is rotated along the gravitational axis and can be given a translation compared to the origin of the world frame. The section frame is used to represent buildings.

**Plane frame ( $p$  - frame)** The plane frame is aligned to its parent section frame, the only difference is that it can be translated to represent a height difference between different floors. The plane frame is used to represent floors, or parts of a floor, in a building.

**Body frame ( $b$  - frame)** The body frame is fixed to the user carrying the Beebadge, the  $z$ -axis is aligned to the gravitational pull from the earth, in



**Figure 2.1.** The relationship between the different coordinate frames used.

other words it will be aligned to the  $z$ -axis of the world frame. The  $x$ -axis is aligned to the direction of travel of the user.

**Sensor frame ( $s$  - frame)** The sensor frame is fixed to the circuit board of the Beebadge. All inertial sensors are measured in this frame.

## 2.2 Radio

The on board radio follows the IEEE 802.15.4 low rate and low economical cost radio standard. The standard specifies the physical and the medium access control layers and operates in the 2.4 GHz region. Using 16 different channels, offset quadrature phase shift keying, and direct sequence spread spectrum a data rate of 250 kbit/s is standardized. The IEEE 802.15.4 standard is designed to be used in networks where low cost and energy consumption are prioritized over reliability [16]. Apart from the wireless communication capability, the onboard radio hardware can perform distance related measurements, which will be described in the following sections.

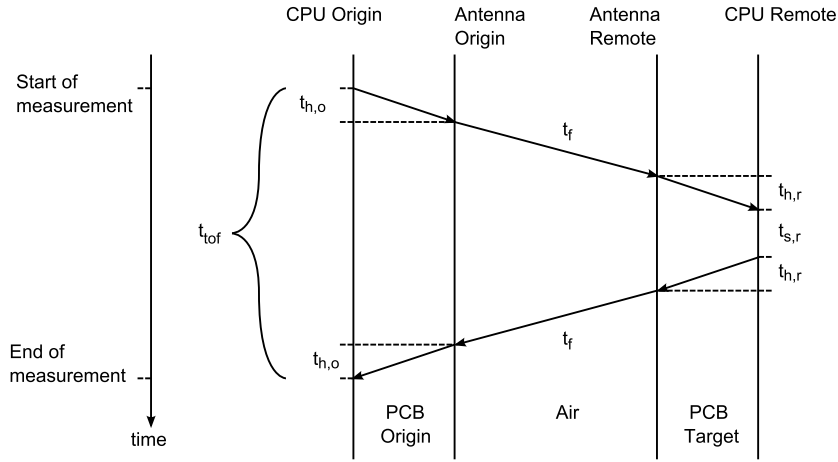
### 2.2.1 Time of Flight

The hardware that is used in this project has the capability of measuring the travel time of radio waves between two radio equipped units. Figure 2.2 illustrates how a time of flight measurement is performed including the different delays that are introduced as the scan progresses, Table 2.1 shows a list of symbols used for time of flight.

The time units  $t_{h,o}$  and  $t_{h,r}$ , are delays introduced between the antenna and the CPU for the origin and remote node. The length of  $t_{s,r}$  arises from the time the remote CPU takes to process the request and create a response.

Symbol	Description
$t_{\text{tof}}$	Total time of flight duration
$t_f$	Propagation delay for time of flight
$t_{h,o}$	Hardware delay origin node
$t_{h,r}$	Hardware delay remote node
$t_{s,r}$	Software delay remote node
$\mathbf{p}_o$	Position origin node
$\mathbf{p}_r$	Position remote node

**Table 2.1.** List of symbols used for time of flight



**Figure 2.2.** Time of flight flow between the origin and remote node. A time of flight measurement is started from the origin, different delays are added together to form the total measurement time.

### Measurement equation

The time unit  $t_{s,r}$  is assumed to remain constant and is therefore removed underneath the software interface. The actual measurement received is on the form

$$\begin{aligned}
 y_{\text{tof}} &= 2ct_f + \Delta d_c + e_{\text{tof}} \\
 &= 2\|\mathbf{p}_o - \mathbf{p}_r\|_2 + \Delta d_c + e_{\text{tof}}, \quad e_{\text{tof}} \sim \mathcal{N}(0, \sigma_{\text{tof}}^2), \quad (2.1)
 \end{aligned}$$

where  $c$  is the speed of light,  $\mathbf{p}_o$  and  $\mathbf{p}_r$  are the positions of the origin and remote nodes, and  $\Delta d_c = 2(t_{h,o} + t_{h,r})$ .

The equality  $ct_f = \|\mathbf{p}_o - \mathbf{p}_r\|_2$  holds only if strict Line of sight (LOS) conditions are present. This is not the case in an indoor environment. Indoor environments instead give rise to multipath effects in which the radio waves travels a longer distance than the direct path between the two radio devices. It

is hard to stochastically model the multipath environment, and it is therefore hard to tell if the measurements are accurate or not. Measuring the time of flight on several radio channels at a time and using the lowest time may reduce the multipath effects [17, 8], but that is not investigated in this thesis due to restrictions in the radio implementation on the used hardware. The results that are shown in Appendix C exemplifies the multipath effects and thereby invalidates the model (2.1). Due to this fact, we will not use the time of flight measurements in the sequel.

### 2.2.2 Received signal strength

Received signal strength indicator (RSSI) is a commonly available measurement on the IEEE 802.15.4 low power and lossy network radios. The RSSI depends heavily on the path loss in the radio environment and the transmitted signal power and will vary much over time due to time varying channels [18, 5].

With free space propagation the path loss increases logarithmically with the distance according to the log-distance path loss model

$$P_{r,\text{dBm}} = P_{0,\text{dBm}} - 10n \log_{10} \left( \frac{d}{d_0} \right), \quad (2.2)$$

where  $P_r$  is the received power,  $d$  the distance between the transmitter and the receiver,  $P_0$  the power received at a small distance  $d_0$  from the transmitting antenna, and  $n$  is a path loss exponent, which typically ranges between two and four in urban environments [19, pp. 69].

#### Measurement equation

The radio hardware delivers the RSSI measurement on the form as mentioned above. The measurement equation can therefore be stated as

$$y_{\text{RSSI}} = P_{0,\text{dBm}} - 10n \log_{10} \left( \frac{d}{d_0} \right) + e_{\text{RSSI}}, \quad (2.3)$$

where  $e_{\text{RSSI}}$  is noise mainly caused by shadowing. If shadowing had been the only noise source,  $e_{\text{RSSI}}$  had been Gaussian distributed with zero mean [20]. However, for the distribution to be Gaussian distributed with zero mean  $P_{0,\text{dBm}}$  and  $n$  must be exactly known which is not the case in practical scenarios where the environment can change rapidly [21]. Our experiments, described in Appendix D, shows that  $e_{\text{RSSI}}$  has some unknown distribution. Due to that uncertainty in the model, we will not use the RSSI measurements in the sequel.

#### Using the presence of a coordinator as a measurement

The radio performs periodic scans for coordinators on the available radio channels. The main purpose of this scan is to detect if a different coordinator has better radio conditions than the one currently connected to and therefore a handover procedure should be initiated.

The fact that a coordinator responds to endpoint requests is a rough measure of the range. The range can be interpreted to be uniformly distributed within some maximum possible spherical range from a coordinator. The measurement equation is expressed as

$$y_{\text{scan}} = \begin{cases} \frac{1}{\frac{4}{3}\pi r_{\text{range,max}}^3} & , \|\mathbf{p}_o - \mathbf{p}_r\|_2 < r_{\text{range,max}} \\ 0 & , \text{otherwise} \end{cases}, \quad (2.4)$$

where  $r_{\text{range,max}}$  is the maximum range of coverage for a coordinator.

The position estimate might be very uncertain, but it still reduces the uncertainty from covering a whole building or even a larger area to just the maximum coverage area for a coordinator.

## 2.3 Inertial sensors

The embedded platform used in this thesis is equipped with multiple inertial sensors. The inertial sensors are manufactured using Microelectromechanical system (MEMS) technology, traditionally inertial sensors were bulky and expensive. This new technology allows for cheap and somewhat accurate inertial sensors. All inertial measurements are resolved in the sensor frame. More detailed information about the sensors can be found in Appendix B.

### 2.3.1 Accelerometer

The device is equipped with a 3-axis accelerometer that measures acceleration in the range of  $\pm 8g$ .

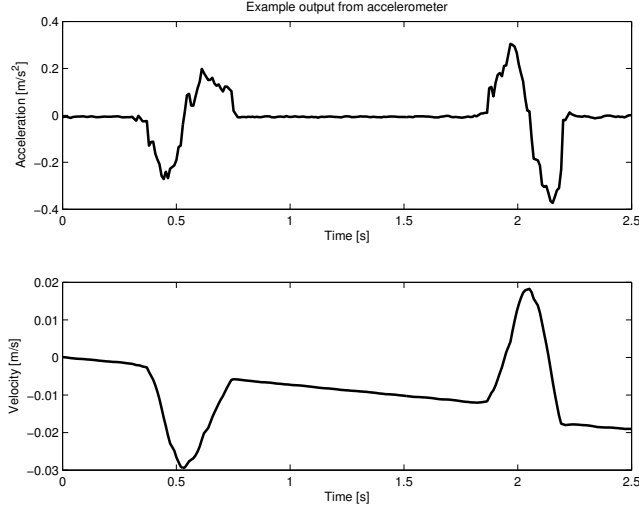
$$\mathbf{y}_a^s = \mathbf{f}^s + \mathbf{g}^s + \delta^s + \mathbf{e}_a^s, \quad \mathbf{e}_a^s \sim \mathcal{N}(0, \Sigma_a), \quad (2.5)$$

where  $\mathbf{f}^s$  is the net acceleration of the endpoint,  $\mathbf{g}^s$  is the gravitational pull from the earth,  $\mathbf{e}_a^s$  is independent Gaussian noise in each direction, and  $\delta^s$  is a varying bias that could depend on multiple external factor, such as the temperature. This bias is assumed to be negligible and is therefore ignored. In Figure 2.3 an example of the accelerometer output can be seen. The unit is moved a distance along the negative  $x$ -axis, then back to the original position. The figure is divided into two parts, the upper shows the acceleration and the lower shows the velocity. The sensor is either not properly calibrated or the  $x$ -axis is not completely orthogonal to the gravity which causes a drift.

### 2.3.2 Gyroscope

The gyroscope used is a 3-axis gyroscope that measures angular velocity in the range of  $\pm 2000$  degrees per second. The measurement equation for the gyroscope is

$$\mathbf{y}_\omega^s = \omega^s + \delta_\omega^s + \mathbf{e}_\omega^s, \quad \mathbf{e}_\omega^s \sim \mathcal{N}(0, \Sigma_\omega), \quad (2.6)$$



**Figure 2.3.** Example data from accelerometer, the endpoint is moved along the negative  $x$ -axis, then back the same distance

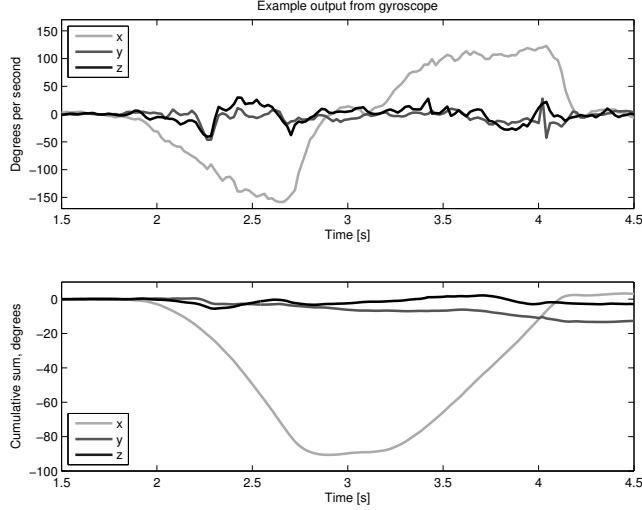
where  $\omega^s$  is the angular velocity of the endpoint,  $\delta_\omega^s$  is the present bias, and  $\mathbf{e}_\omega^s$  is independent Gaussian noise in each direction. The bias is not constant and depends on multiple external factors such as temperature. But for this thesis the bias is assumed to be constant, but different for individual endpoints due to natural variations in the manufacturing process. Thus calibration had to be performed to achieve a good result. Calibration is performed by holding the endpoint still and taking a long running average. The offset is compensated for by simply subtracting it from the measurement. In Figure 2.4 is a sample output from the gyroscope when the endpoint is rotated 90 degrees around the  $x$ -axis, then immediately rotated back to its initial position.

## 2.4 Magnetometer

The magnetometer used is a 3-axis magnetometer that measures magnetic flux in the range of  $\pm 1.2Ga$ . The measurement equation for the magnetometer is

$$\mathbf{y}_m^s = R\mathbf{m}^s + \delta_m^s + \mathbf{e}_m^s, \quad \mathbf{e}_m \sim \mathcal{N}(0, \Sigma_m), \quad (2.7)$$

where  $\mathbf{m}^s$  is the magnetic field without any disturbances,  $\mathbf{e}_m^s$  is independent Gaussian noise in each direction.  $R$  and  $\delta_m^s$  corresponds to soft and hard iron effects effects which is further described in [4]. Soft iron effects turned out to be minimal on the endpoints used for this thesis, which means that the matrix  $R$  is roughly a scaled identity matrix. To get the orientation of the magnetic field only the direction and not the strength is important, we can therefore ignore the effects of  $R$ . The hard iron effect is quite severe, so calibration must be



**Figure 2.4.** Example data from gyroscope, the endpoint is rotated 90 degrees along the  $x$ -axis, then immediately back to its initial position

performed by collecting a lot data points from the magnetometer, making sure that all axes of the magnetometer is aligned with and against the magnetic field during the data collection. The calibration term is then calculated by

$$\hat{\delta}_{m,n}^s = \frac{\max(m_{k,n}) + \min(m_{k,n})}{2} \quad \forall n \in x, y, z, \quad (2.8)$$

where  $k$  are all different measurements. This yields  $\hat{\delta}_m^s = (\hat{\delta}_{m,x}^s, \hat{\delta}_{m,y}^s, \hat{\delta}_{m,z}^s)^T$  which is subtracted from the measurement to compensate for the hard iron effects.

## 2.5 Map

A map covering a building or an area can give substantial information of the possibility to be located at a certain position or making certain movements. This section describes how a map can be represented for giving such information to a positioning algorithm in an efficient way. Since potentially many thousands of queries to the map will be performed every second caused by a large number of users each requiring frequent evaluations of position hypotheses, these queries must be computationally efficient. Since the area covered by a map may be very large, a lot of information must be stored within it. To avoid unnecessary memory consumption, the map should advantageously support partitioning into smaller pieces.

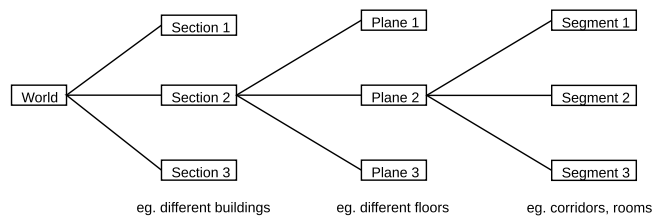
### 2.5.1 Representation

The map is advantageously represented in Two dimension representation with limited altitude (2.5D), which means that the environment is divided into planes and that all motion is always restricted to lie within these planes [15]. The restriction that all motion lies within a plane can be done without hesitation, since human movement is typically constrained to the floor of a building. The advantage is that the complexity of the state estimation algorithms can be reduced with this assumption, essentially reducing the dimension of the state vector by one. The vertical dimension for the user will instead be represented by the vertical translation of the current plane relative to the world frame.

### 2.5.2 Map structure

The world is split into *sections*, typically a section corresponds to a building. Each section is in turn split into *planes*, where each floor in a building or a flat area is represented by a plane. Therefore, each plane has its own z-translation. If deemed necessary each floor can be split into smaller planes, allowing only parts of a floor to be fetched at a given time. This is to decrease the amount of transferred data and storage space needed at a given time on the endpoint.

Figure 2.5 illustrates the overall structure of the map.



**Figure 2.5.** Structure of the map how buildings and floors are constructed from segments

### World

The world's coordinate system is oriented in such a manner that the gravitational pull from the earth is aligned with the  $z$ -axis and the magnetic north is aligned with the  $x$ -axis. The world also has a few nodes that serve as entrance and exit points between planes.

### Section

A section contains one or more *planes*. Each section will have a point in the world's coordinate system which is the origin of the section's own coordinate system. The  $z$ -axis is parallel to the world's  $z$ -axis, while the  $x$  and  $y$  axes can be rotated to better represent rotated buildings and environment. This will ease



the development of maps since it would be easy to align the coordinate systems with the walls of a building.

Each section also contains a list of locations to the coordinators that are mounted in the building. This information can be used as reference positions for time of flight measurements or scan result measurements as discussed in Section 2.2.

### Plane

The coordinate system for the plane is aligned with the section's coordinate system to which it belongs. The difference is that it can have an additional translation in the  $z$ -direction relative to its parent section. Planes consist of *segments* that lie within the plane. Segments have a start location, end location, and a given width. Segments are connected to neighboring segments which represent the possibility of moving between different areas of the plane. The area spanned between the start and end with the given width represents an area in which the user can be located. This is opposed to representing walls which the user cannot pass through. The reason that areas the user can be located in was chosen instead of representing walls is that it is easier to transverse a graph of interconnected segments. This means that at a given segment it is easy to know the neighboring areas that a user can move to. If instead walls were represented, the system must potentially search through a large collection of walls to be able to fully grasp the possibility of movement in a certain direction.

Figure 2.6 shows parts of the office at Xdin in Mjärdevi Science Park, where the segments can be seen as dashed lines on top of the actual room layout. The bullets highlight the start and end positions of different sections.

### Segment

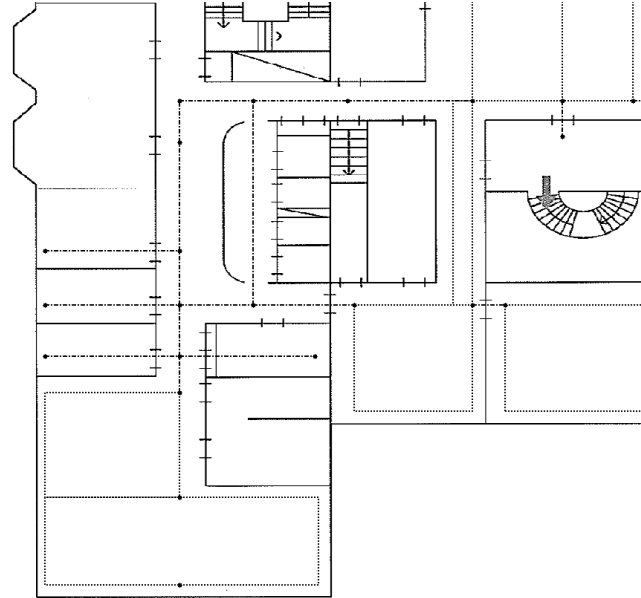
Each segment contains a start and end and has the possibility of having an additional width. Furthermore the segment contains a step length modifier, this modifier is used to help represent areas where the step length is for some reason shorter or longer than a normal step. An example of this is stairs where the step length is approximately half of a normal step.

### 2.5.3 Map interpretation

In this section we describe two different ways of interpreting the information contained in the map. One is handling it as a relative probability density and another way is as a force field pushing the user away from walls.

#### Probability density

One way of interpreting the map is to see it as a two dimensional relative probability density function. This function should be interpreted as the relative probability that a target can be positioned at some given position compared to being in the center of a room or hallway. The function can span higher



**Figure 2.6.** Parts of Xdin's office in Mjärdevi. Bullets represents start and end positions for different segments, dot-dashed lines represents segments with width set to zero and dotted squares represents segments with a given width.

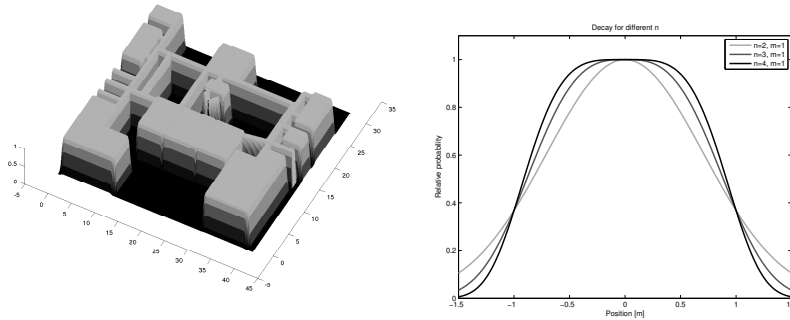
dimensions as well, if for example altitude, radio coverage or magnetic fields are included. Figure 2.7(a) shows the probability density function representation for parts of Xdin's office. The bright areas are rooms and the bright lines are corridors that interconnect the rooms.

Given a segment it is most likely that a person is located within the area spanned by the start, end, and the width. Probability surrounding the center area of the segment will have a decay on the form  $e^{-\frac{|d|^n}{m}}$  where  $d$  is the distance from the edge,  $n$  is how fast the drop off should be, where a small  $n$  will have a slow drop off and a faster drop off as  $n$  increases.  $m$  can be seen as the spread of the segment. If the line has been given a width, the area in the center region of the segment is set at its maximum value of one. Figure 2.7(b) shows a cross section for a segment with zero width.

### Additive forces

A different interpretation of the map besides a probability density is that the map forces targets to be located only where it is possible for them to be. When moving towards a wall, a growing force from the wall affects the momentum of the target so that target cannot move through the wall. Such a model requires that the force becomes infinitely large when the target's distance to the wall tends to zero and close to zero when the distance is sufficiently large.

Since the map is represented as lines and planes any convex function around



(a) Relative probability density for parts of Xdin's office, the bright areas are rooms and the bright lines are corridors that interconnect the rooms  
 (b) Cross section of the relative probability function for a line with different  $n$

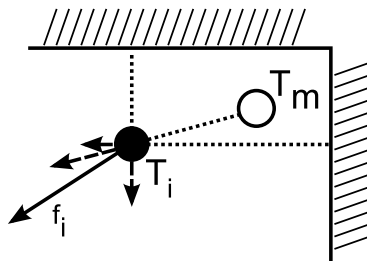
**Figure 2.7.** Probability interpretation of the map.

those would suffice to give a magnitude of the force. The force is intuitively directed orthogonally from the wall towards the target and multiple forces can be added together to get a resulting force affecting the momentum of the target.

Equation (2.9) describes how the force is constructed. The function  $\mathbf{wall}_j(\mathbf{p})$  is a convex function giving the magnitude and direction of the force given the position of the target,  $\mathbf{p}$ .

$$\mathbf{f}_i = \sum_{j \in \mathcal{W}} \mathbf{wall}_j(\mathbf{p}_i), \text{ where } \mathcal{W} \text{ is the set of walls.} \quad (2.9)$$

If positions from other targets are available, repellent forces from them can be modeled as well, which is thoroughly discussed in [22]. The concept is visualized in Figure 2.8 where the target  $T_i$  is affected by two walls and another target  $T_m$ , resulting in the force  $\mathbf{f}_i$ .



**Figure 2.8.** Force vectors illustrating the resulting force affecting a pedestrian.

The map is, as already mentioned, represented as areas where it is possible to be rather than as walls. To arrive at a convex function for the magnitude of the forces, we say that the magnitude of the force is  $f_{i,j} = e^{d_j} - 1$ , where  $d_j$  is the smallest distance to wall  $j$ . The direction of the forces are simply orthogonal to the wall which they stems from.

#### 2.5.4 Map interface

The map has two main tasks, first return the correct probability density value or force vector for a given point, secondly be able to handle swaps between different planes when walking in stairs and entering or leaving different buildings, etc.

##### Map output

The output from the map will be the relative probability or the force vector for a given segment and its neighbors, along with the segment from which the output was attained. Furthermore it must output the step length modifier,  $s_{sl}$ , for a given segment. This modifier is used to modify the step length in certain areas where the step length can vary greatly, an example of this is stairs where the step length is approximately half of a normal step.

##### Traversing the map

For an object to find the force vector or relative probability from the map at a given position, a query is sent to the map with the location and the segment used the last time by the same object. The computational burden can be lowered if only the last segment and its neighbors are processed since it is likely that the predicted position is close to the previous one. An identifier of the segment yielding the highest probability or smallest force is returned so that it is possible to update the record of current segment. Experiments show that only the closest neighbors and not segments further away need to be checked with retained results. If a search is performed deeper than one step, some minimum spanning tree algorithm could be incorporated to improve the efficiency of the map traversal. Another advantage by just traversing the closest neighbors besides lowered computational burden is that rooms or halls that lies on the other side of a wall for example, is to a large extent excluded in the computations.

##### Moving between sections

A few segments will have the extra property that they are linked to a different plane. The destination plane can be in the same section, a different section or can be linked to the world, where map support will be disabled and only dead reckoning is used. These segments are called link segments. A query can be sent to the map regarding a certain segment is linked somewhere else, if it is linked the response will also contain the destination segment and plane.

## 2.6 Pedestrian motion model

Models of pedestrians moving through an environment of some kind can take many more factors into account than just the current estimated position and speed. Common simple models of pedestrians include constant velocity or constant acceleration, but people tend to have deeper interests than just keeping the same speed in a certain direction. Studies show that the movement of pedestrians can be accurately modeled if the surroundings are taken into account [23, 22]. We will later in the thesis use these ideas to insert the information from a map into the motion model. Keeping computational efficiency in mind, we will below describe the simple constant velocity model.

A commonly used motion model is the constant velocity model. It is mainly used for its simplicity and is often good enough. However, navigation systems that use this kind of simple model rely on more accurate measurements which correct for the insufficient motion model.

The constant velocity motion model is defined as

$$x_{k+1} = \begin{pmatrix} I_n & TI_n \\ 0 & I_n \end{pmatrix} x_k + \begin{pmatrix} \frac{T^2}{2} I_n \\ TI_n \end{pmatrix} w_k, \quad (2.10)$$

where the state vector  $\mathbf{x} = (\mathbf{p}^T, \mathbf{v}^T)^T$ ,  $n = \dim(\mathbf{x})$ ,  $k$  is the discrete time index, and  $w_k$  is process noise, representing the model error.

### 2.6.1 Step model

If a step detection algorithm is used for finding the velocity of the user, the equation would simply be

$$v = \frac{d_{\text{step}}}{\Delta T}, \quad (2.11)$$

where  $\Delta T$  is the estimated or measured time between two steps and  $d_{\text{step}}$  is the step length of the pedestrian. This gives a velocity which should be integrated over the duration of the step,  $\Delta T$ . This gives us the following integral assuming the step is taken at time  $\alpha$  with a duration of  $\Delta T$

$$\int_{\alpha}^{\alpha+\Delta T} \frac{d_{\text{step}}}{\Delta T} dt = d_{\text{step}}. \quad (2.12)$$

This means that the middle step when calculating the velocity and integrating is just a computational burden to get back to the initial and already known distance. This relation will be used later in this thesis for reducing the required computations.



## Chapter 3

# State estimation

Extracting the information from a signal covered in noise can be accomplished to a great extent given a stochastic model of the signal. If multiple such models are available for the input signals to a system, and in addition a model for the system dynamics exist, the valuable information of the system state can be estimated. This chapter describes different algorithms for fusing such stochastic models in a recursive manner.

Each algorithm have in common that they arrive at an optimal estimate of the state, but are distinguished in that they are suitable in different cases depending on the stochastic models. What is optimal depend on how the problem is formulated.

If the likelihood of a set of measurements  $\mathbf{y}$  given the set of parameters  $\mathbf{x}$  is defined as  $p(\mathbf{y}|\mathbf{x})$ , the Maximum likelihood (ML) estimate is defined as

$$\hat{\mathbf{x}}^{ML} = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{x}). \quad (3.1)$$

An alternative point estimate is provided by the Maximum a posteriori (MAP) estimate, which also requires a prior density  $p(\mathbf{x})$  in the problem formulation

$$\hat{\mathbf{x}}^{MAP} = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{x}|\mathbf{y}) = \underset{\mathbf{x}}{\operatorname{argmax}} \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \quad (3.2)$$

If  $p(\mathbf{x})$  is equally probable for all  $\mathbf{x}$ , MAP and ML coincide with each other.

We will in the sequel represent the current discrete time with the index  $k$ .

### 3.1 Kalman filter

Given a linear state space model with additive Gaussian noise

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + v_k, \quad w_k \sim \mathcal{N}(0, Q) \quad (3.3a)$$

$$\mathbf{y}_k = H\mathbf{x}_k + e_k, \quad e_k \sim \mathcal{N}(0, R) \quad (3.3b)$$

the Kalman filter (KF) finds the best possible linear estimate,  $\hat{x}_{k|k}$  given measurements  $y_{1:k}$  as input [24]. The best possible linear estimate is defined as the Minimum mean squared error (MMSE) estimator. It can be shown that the MMSE estimator coincides with the ML estimator in the case of additive Gaussian noise.

The Kalman filter is provided in Algorithm 1, and it can be noted that the measurement updates can be performed at any time regardless of how often the time update is performed. This fact is often overlooked in the literature, but is crucial when multiple sample rates are used in a system.

The measurement update step in the Kalman filter can intuitively be described as giving some information related to the unknown states, which reduces the covariance for the involved states. The time update, on the other hand, predicts the next state of the system and causes the covariance to increase since the future is yet unknown.

---

**Algorithm 1** Kalman filter

Given the linear model (3.3) initialized with  $\hat{x}_{1|0} = x_0$  and  $P_{1|0} = P_0$ , the following recursive updates gives the best possible linear estimate:

- **Measurement update**

If we define the auxiliary variables

$$\begin{aligned} \varepsilon_k &= \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1}, \\ S_k &= H_k P_{k|k-1} H_k^T + R_k, \\ K_k &= P_{k|k-1} H_k^T S_k^{-1}, \end{aligned}$$

we can express the measurement update as

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \varepsilon_k, \quad (3.4a)$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T. \quad (3.4b)$$

- **Time update**

$$\hat{\mathbf{x}}_{k+1|k} = F_k \hat{\mathbf{x}}_{k|k}, \quad (3.5a)$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + G_{v,k} Q_k G_{v,k}^T. \quad (3.5b)$$


---



### 3.1.1 Extended Kalman filter

The Kalman filter requires a linear system, but in practice one or more nonlinearities often occur. If the nonlinear function can be linearized around some point  $\tilde{x}$ , the Kalman filter may still be an option, even though non-optimal. The nonlinear system is expressed as

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, v_k), \quad (3.6a)$$

$$\mathbf{y}_k = h(\mathbf{x}_k, e_k). \quad (3.6b)$$

The noise processes  $v_k$  and  $e_k$  are assumed still to be Gaussian, but propagated through the nonlinear functions  $h$  and  $f$ . In the original Extended Kalman filter (EKF) the nonlinear functions are expanded in a Taylor series around a given point  $\tilde{x}$ , retaining only the first-order terms [25],

$$g(x) \approx g(\tilde{x}) + g'(\tilde{x})(x - \tilde{x}), \quad (3.7)$$

where  $g'$  is the Jacobian of the nonlinear function  $g(x)$ . The Kalman filter described in Algorithm 1 is modified with the linearized functions and can be described as Algorithm 2.

---

**Algorithm 2** Extended Kalman filter
 

---

Given a linearized model (3.6) initialized with  $\hat{x}_{1|0} = x_0$  and  $P_{1|0} = P_0$ , the following recursive updates gives the best possible linear estimate if the higher order rest terms in the linearization are disregarded:

- **Measurement update**

Given the auxiliary variables

$$\begin{aligned} \varepsilon_k &= \mathbf{y}_k - h_k(\hat{\mathbf{x}}_{k|k-1}), \\ S_k &= h'_x(\hat{\mathbf{x}}_{k|k-1})P_{k|k-1}h'_x(\hat{\mathbf{x}}_{k|k-1})^T + h'_e(\hat{\mathbf{x}}_{k|k-1})R_k h'_e(\hat{\mathbf{x}}_{k|k-1})^T, \\ K_k &= P_{k|k-1}h'_x(\hat{\mathbf{x}}_{k|k-1})^T S_k^{-1}, \end{aligned}$$

the measurement update becomes

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \varepsilon_k, \quad (3.8a)$$

$$P_{k|k} = P_{k|k-1} - K_k h'_x(\hat{\mathbf{x}}_{k|k-1}) P_{k|k-1}. \quad (3.8b)$$

- **Time update**

$$\hat{\mathbf{x}}_{k+1|k} = f_x(\hat{\mathbf{x}}_{k|k}), \quad (3.9a)$$

$$P_{k+1|k} = f'_x(\hat{\mathbf{x}}_{k|k})P_{k|k}f'_x(\hat{\mathbf{x}}_{k|k})^T + f'_v(\hat{\mathbf{x}}_{k|k})Q_k f'_v(\hat{\mathbf{x}}_{k|k})^T. \quad (3.9b)$$


---

The Extended Kalman filter does not give an optimal estimator in the sense that the Kalman filter does if the rest terms in the Taylor expansion are non-zero. Neither is it guaranteed that the algorithm converges if, for example, a bad choice of initial state or linearization point is chosen.

## 3.2 Particle filter

When highly nonlinear functions come into the model it may be hard to linearize them around some point, especially if the linearization point is unknown. The Particle filter (PF) allows for multiple hypotheses of the state vector which are all weighted samples of the objective function, which is often a probability distribution. The concept is to represent an objective function with sufficiently many hypotheses so that they become dense enough to represent the function. The system model is still the one described in (3.6) but the noise can have an arbitrary density. We describe the core of the particle filter steps in Algorithm 3 without derivation. More in depth about the particle filter can be found in [11, 24, 26].

### 3.2.1 Resampling

In order to continuously represent the objective function as it varies with time, the set of particles has to be updated continuously. The particles are said to be resampled. The purpose of the resampling algorithm is to redistribute the particles to better represent the underlying density. This can be achieved by placing particles where there already exist particles with relatively high weight. In this way, the empirical density is updated to represent the most likely states.

Because of the relatively computationally demanding algorithm, the resampling step does not need to be executed after every reweighting of the particle set. The effective number of particles can be used as an indication of when a resampling have to be performed. A measure of the number of effective particles is given by

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^N (w^i)^2}, \quad (3.16)$$

where  $1 \leq N_{\text{eff}} \leq N$ .  $N_{\text{eff}}$  attains the lower bound when all the weight is placed on one particle and the upper bound when the weight is uniformly distributed over all particles. This measure can be used to detect when a resampling of the particle set is needed, when  $N_{\text{eff}} < N_{\text{th}}$  for some  $1 \leq N_{\text{th}} \leq N$  a resampling of the particle set should be performed. [11] proposes the threshold to be chosen as  $N_{\text{th}} = 2N/3$ .

## 3.3 Marginalized particle filter

For real-time applications, the particle filter quickly becomes infeasible as the state dimension grows. The Marginalized particle filter (MPF) allows different

**Algorithm 3** Particle filter

Given a nonlinear model (3.6), the following steps combined propagate particles to represent the relevant domain of an objective function given measurements  $y_{1:k}$ .

- **Initialization**

Generate  $N$  particles from the initial probability density function  $p(\mathbf{x}_0)$ ,

$$\mathbf{x}_0^i \sim p(\mathbf{x}_0), \quad i = 1, \dots, N, \quad (3.10)$$

and set the weights of the particles to  $w_0^i = 1/N$ ,  $i = 1, \dots, N$ .

- **Measurement update**

Update the weights of the particles according to

$$w_{k|k}^i = \tilde{w}_{k|k}^i p(\mathbf{y}_k | \mathbf{x}_k^i), \quad i = 1, \dots, N, \quad (3.11)$$

and normalize the weights

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^N \tilde{w}_k^j}, \quad i = 1, \dots, N. \quad (3.12)$$

- **Time update**

Propagate the particles through the process dynamics

$$\mathbf{x}_{k+1}^i = f_k(\mathbf{x}_k^i, v_k^i), \quad i = 1, \dots, N. \quad (3.13)$$

- **Estimation**

$$\hat{\mathbf{x}}_{k|k} = \sum_{i=1}^N w_k^i \mathbf{x}_k^i \quad (3.14a)$$

$$P_{k|k} = \sum_{i=1}^N w_k^i \left( \mathbf{x}_{k|k}^i - \hat{\mathbf{x}}_{k|k} \right) \left( \mathbf{x}_{k|k}^i - \hat{\mathbf{x}}_{k|k} \right)^T \quad (3.14b)$$

- **Resampling**

Replace the set of particles  $\{x_k^i\}_{i=1}^N$  with particles drawn at random an identical set  $\{x_k^j\}_{j=1}^N$ , where the probability to draw particle  $x^j$  is  $w^j$ ,

$$\Pr \left( \mathbf{x}_{k|k}^i = \mathbf{x}_{k|k}^j \right) = w_k^j \quad (3.15)$$

After replacing the set of particles, set the weights of the particles to  $w_k^i = 1/N$ ,  $i = 1, \dots, N$ .

optimization methods to be used on the same state vector. A Kalman filter may be used on the linear subset of states while the particle filter is only used on the most crucial nonlinear states. The particle filter performs well for two or three dimensions, but for higher dimensions the number of particles required to represent a posterior distribution grows out of hand [24].

The general model for a system where the linear substructures have been separated from the nonlinear ones is given by

$$\mathbf{x}_{k+1}^n = f_k^n(\mathbf{x}_k^n) + F_k^n(\mathbf{x}_k^n)\mathbf{x}_k^l + G_k^n(\mathbf{x}_k^n)v_k^n, \quad (3.17a)$$

$$\mathbf{x}_{k+1}^l = f_k^l(\mathbf{x}_k^n) + F_k^l(\mathbf{x}_k^n)\mathbf{x}_k^l + G_k^l(\mathbf{x}_k^n)v_k^l, \quad (3.17b)$$

$$\mathbf{y}_k = h_k(\mathbf{x}_k^n) + H_k(\mathbf{x}_k^n)\mathbf{x}_k^l + e_k. \quad (3.17c)$$

where  $\mathbf{x}_k = (\mathbf{x}_k^n, \mathbf{x}_k^l)^T$  is the complete state vector.

In a simplified description of the MPF, the particles are updated and re-sampled in the state space for the nonlinear states, treating the linear states as measurement noise. Then, for each particle a Kalman filter measurement update (or similar) is performed, resulting in the optimal set of linear states given the state of the particle.

The time updates are performed in a similar fashion, where the particle filter time update is performed first, and then the Kalman filter time update (or similar) is performed for each particle.

---

**Algorithm 4** Marginalized particle filter
 

---

- **Initialization**

Initialize  $N$  particles,  $\mathbf{x}_0^{n,i}$ , according to (3.10) and initialize equally many linear state vectors  $\{\mathbf{x}_0^{l,i}, P_0^{l,i}\} = \{\mathbf{x}_0^l, \mathbf{P}_0^l\}$ .

- **Measurement update**

1. Update and normalize the particle weights according to (3.11) and (3.12).
2. Perform a Kalman filter measurement update for each particle. This update becomes slightly different from to (3.4), depending on how the system model is constructed. See [27] for further details.

- **Resampling**

Optionally: Resample according to (3.15).

- **Time update**

1. Propagate the particles through the process dynamics according to (3.13).
2. Perform a Kalman filter time update for each particle. As with the measurement update above, this becomes modified. See [27] for further details.

- **Estimation**

Estimate  $\hat{\mathbf{x}}$  according to (3.14), having  $\mathbf{x}_k^i = (\mathbf{x}_k^{n,i}, \mathbf{x}_k^{l,i})$ .

---



# Chapter 4

## Approach

This chapter describes a number of considered filtering approaches all using the methods and relations described in Chapter 2, and Chapter 3. A number of so called subsystems are also described which will function as an intermediate signal processing step between the sensors and the position estimation filter. The subsystems are differentiated from the position estimation filter in the sense that they do not get any feedback from it.

The different approaches use different ideas of how the filtering system should be constructed. Estimating the position on board the Beebadges results in a system which scales well with the number of users. Such a system will however suffer from the computational constraints of the Beebadges. If, on the other hand, no computations are performed on board the Beebadges raw sensor have to be transmitted from the Beebadges to some receiver. All computations are then instead performed on a centralized computer. While the Beebadges will not perform any computations, they will instead have to transmit high amounts of sensor data over the radio which is power consuming. Such a system will not scale well with added users since much computations are required for each user. Trade-offs between those two variants will be discussed in this and the following chapters.

### 4.1 Subsystems

This section describes filters and subsystems processing the sensor data from the sensors described in Chapter 2. These subsystems are seen as separate systems which produce input to, and are thus not directly included in, the positioning algorithms.

#### 4.1.1 Inertial Measurement Unit

Gyroscope, accelerometer and magnetometer data is combined to form an IMU. More information about the sensors can be found in Appendix B. The IMU

is used to find the orientation of the sensor frame relative to magnetic north and the gravitational pull from the earth. Since the sensor frame is fixed to the body frame, if the orientation of the sensor frame is known it straightforward to calculate the orientation of the body frame. The output from the IMU is a four dimensional vector which represents the orientation on quaternion form

$$q^{ws} = \left( \cos \frac{\theta}{2}, \mathbf{r} \sin \frac{\theta}{2} \right) \in Q_l, \quad (4.1)$$

where  $\mathbf{r}$  is the normalized axis of rotation and  $\theta$  is the angle of counter-clockwise rotation along the axis. Details on quaternion algebra is presented in Appendix A.

### Heading

To be able to calculate the direction of travel of the user the orientation of the sensor frame relative to the body frame must be known. Depending on the assumptions made this is either known or unknown. The step direction is defined as the  $x$ -axis in the body frame, given the orientation of the sensor frame in relation to the body frame, we can find the vector in the sensor frame which is parallel to the direction of travel. This vector is called  $\mathbf{v}_s^s$  and expressed in quaternion form  $v_s^s \in \{Q_v\}$ . The direction of travel in the world frame is simply

$$v_s^w = q^{ws} \odot v_s^s \odot (q^{ws})^c, \quad (4.2)$$

expressed in quaternion form. Since the user only moves in the plane spanned by the  $x$ - and  $y$ -axes, these two components can be extracted to form the heading vector as

$$\mathbf{h}_s^w = (v_{s,x}^w, v_{s,y}^w)^T. \quad (4.3)$$

In reality  $\mathbf{v}_s^s$  might not be in the plane of motion, if this is true  $\mathbf{h}_s^w$  will not be of unit length, therefore this vector needs to be normalized before use. If the norm of  $\mathbf{h}_s^w$  is deviating a lot from one it is a sign that the vector  $\mathbf{v}_s^s$  is not in the plane of motion, if this is the case a new  $\mathbf{v}_s^s$  should be found.

When the orientation of the sensor frame relative the body frame is unknown but fixed,  $v_s^s$  must still be found to determine the step direction. One way of solving this problem is estimating the quaternion  $q^{sb}$ , then using this  $q^{sb}$  to find  $v_s^s$  by

$$v_s^s = q^{sb} \odot e_x^b \odot (q^{sb})^c, \quad (4.4)$$

where  $e_x$  is the quaternion representation of the  $x$ -axis in the body frame, which also is the step direction. It would then be possible to use (4.2) to find the step direction in the world frame.

It turns out to be a tricky problem to estimate the  $q^{sb}$ . A simpler approach than estimating  $q^{sb}$  is using known parameters of the movement. Since it is known that a person always moves in the plane spanned by the  $x$ - and  $y$ -axes in the world coordinate frame. When the first step is taken,  $\mathbf{e}_x^w$  is transformed to the sensor frame using the quaternion  $q^{ws}$  and is called  $\mathbf{v}_{sr}^s$ . Transforming this



Symbol	Description
$a_h$	Higher step threshold
$a_l$	Lower step threshold
$a_{\max}$	Maximum acceleration during step
$t_{\max}$	Time at maximum acceleration
$a_{\min}$	Minimum acceleration during step
$t_{\min}$	Time at minimum acceleration
$a_{\text{step}}$	Acceleration difference between peak and trough
$t_{\text{step}}$	Time between peak and trough
$t_{\text{timeout}}$	Time out value between transitions

**Table 4.1.** List of symbols used for step detection

vector similarly as in (4.2),  $\mathbf{v}_{\text{sr}}^w$  is obtained. This is the direction of travel with a fixed angle offset around the  $\mathbf{e}_z^w$  vector. Further  $\mathbf{v}_s^w$  can then be found using

$$\mathbf{v}_s^w = R_3 \mathbf{v}_{\text{sr}}^w, \quad (4.5)$$

where  $R_3 \in \mathbb{R}^{3 \times 3}$  is a rotational matrix around the  $\mathbf{e}_z^w$  vector. Again only looking on the x and y components it is possible to rewrite (4.3) using (4.5) to

$$\mathbf{h}_{\text{sr}}^w = (v_{\text{sr},x}^w, v_{\text{sr},y}^w)^T, \quad (4.6a)$$

$$\mathbf{h}_s^w = R_2 \mathbf{h}_{\text{sr}}^w, \quad (4.6b)$$

where  $R_2 \in \mathbb{R}^{2 \times 2}$  is a rotational matrix.

### 4.1.2 Step detection

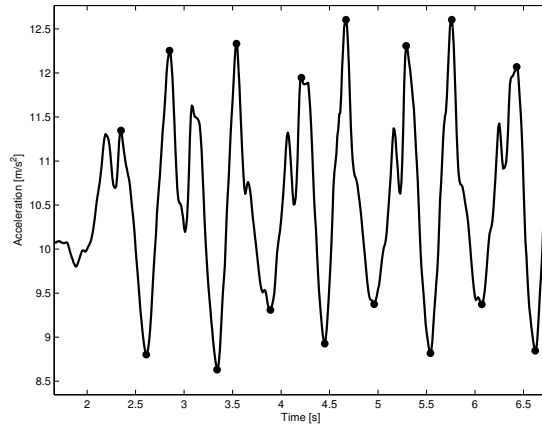
This section describes the step detection algorithm used in this thesis.

#### Motivation

Instead of having to double integrate linear acceleration, step detection detects a change of position which prevents the need of integration. However, an assumption about the step length has to be made since this is not measured. A step detection algorithm can use many different methods. A computationally efficient way is by monitoring the net acceleration and detecting the signature of a step. The reason for using a step counter as opposed to integrating linear acceleration twice to get the position is the problem with compensating for the gravitational pull. The available IMU can be used to compensate for the gravity, but even minor deviation from the truth in the orientation of the IMU causes severe drift in position and velocity. Previous work done in [14] shows that step detection can be implemented with good results on the same hardware that is used for this thesis.

### Step acceleration model

The characteristics of a step can be divided into a horizontal part and a vertical part where both parts have different characteristics. Due to the problem of removing the gravitational component from the accelerometer, only the norm of the acceleration was focused on when developing the step detection. Therefore we cannot look at the individual components, only a combined norm. A step can be divided into two parts, during the first part a sudden increase in acceleration takes place, this is due to the person starts taking the step and accelerates. The second part is characterized by sudden drop in the norm of the acceleration, this is essentially the person leaning or "falling" forward. A more in-depth description can be found in [28]. Figure 4.1 shows an example of the norm of the acceleration for a few steps. The sudden increase and drop can be clearly seen.



**Figure 4.1.** Example of eight steps, circles represent peak and trough values detected

### Algorithm

Before running the data through the algorithm, the norm of the acceleration is low-pass filtered to reduce noise and allow the algorithm to detect longer trends. The low-pass filter used is a first order filter. FIR filters of different orders were tested, but a first order showed to be sufficient and is very computationally efficient. To detect a step a state machine was constructed to detect the different parts of the step. Figure 4.2 shows flow chart of how the state machine is constructed.

The state machine is characterized by two thresholds,  $a_h$  is used as an upper threshold to detect the rising edge of a step and  $a_l$  to detect the falling edge of a step. The state machine starts in the *Default* state, when a norm acceleration above  $a_h$  the state machine transitions to the *Peak* state, during this state the maximum acceleration  $a_{max}$  is saved along with the time  $t_{max}$  when

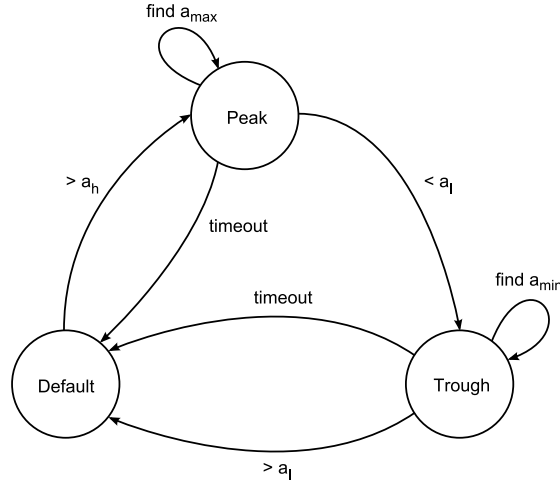


Figure 4.2. State machine for step algorithm

peak acceleration was recorded. When the acceleration falls below  $a_l$  the state machine transitions to the *Trough* state, if no acceleration below  $a_l$  is detected after  $t_{\text{timeout}}$  it is assumed that the detected peak did not belong to a step, the state machine will return to its default state. During the *Trough* state the minimum acceleration  $a_{\min}$  is saved paired with the time  $t_{\min}$ . When the acceleration increases above  $a_l$ ,  $a_{\text{step}}$  and  $t_{\text{step}}$  are calculated by simple subtraction.

$$a_{\text{step}} = a_{\max} - a_{\min}, \quad (4.7)$$

$$t_{\text{step}} = t_{\max} - t_{\min}. \quad (4.8)$$

Depending on values for  $a_{\text{step}}$  and  $t_{\text{step}}$  a step would be noted. In Algorithm 5 pseudocode can be found for the step detection algorithm.

### Using the step details

In addition to just detecting when a step has been taken, the values  $a_{\text{step}}$  and  $t_{\text{step}}$  can be used as information to some classification algorithm to detect certain styles of walking or detection of unnatural walking styles which may give hints about injuries or similar.

The present algorithm does not attach any extra information based on  $a_{\text{step}}$  and  $t_{\text{step}}$  since it is not of interest for the positioning algorithms. The data could, however, be extracted from the algorithm with ease for future applications.

### Performance

The performance of the step detection algorithm was tested on different individuals to be able to evaluate its performance. Table 4.2 lists the results of this

---

**Algorithm 5** Step detection algorithm

---

**First accelerometer sample** Set state to *Default* state

**For each new accelerometer sample**

Calculate the accelerator norm  $a = \sqrt{a_x^2 + a_y^2 + a_z^2}$ .

**if** *state* = *default* **then**

**if**  $a > a_h$  **then**

*state*  $\leftarrow$  *peak*

$a_{\max} \leftarrow a$

$t_{\max} \leftarrow t$

**end if**

**else if** *state* = *peak* **then**

**if**  $a < a_l$  **then**

*state*  $\leftarrow$  *trough*

$a_{\min} \leftarrow a$

$t_{\min} \leftarrow t$

**end if**

**if**  $a > a_{\max}$  **then**

$a_{\max} \leftarrow a$

$t_{\max} \leftarrow t$

**end if**

**if**  $t_{\max} + t_{\text{timeout}} < t$  **then**

*state*  $\leftarrow$  *default*

**end if**

**else if** *state* = *trough* **then**

**if**  $a > a_l$  **then**

$a_{\text{step}} \leftarrow a_{\max} - a_{\min}$

$t_{\text{step}} \leftarrow t_{\max} - t_{\min}$

*state*  $\leftarrow$  *default*

        Output step event, attach  $a_{\text{step}}$  and  $t_{\text{step}}$

**end if**

**if**  $t_{\min} + t_{\text{timeout}} < t$  **then**

*state*  $\leftarrow$  *default*

**end if**

**end if**

---

Individual	Environment	Steps taken	Steps measured	Success rate
1	indoor with shoes	123	123	100
1	indoor without shoes	136	136	100
1	down stairs	73	64	88
1	up stairs	69	58	84
1	outdoor asphalt	138	134	97
1	outdoor soft grass	112	107	96
2	indoor with shoes	110	106	96
2	indoor without shoes	124	114	92
2	down stairs	67	58	87
2	up stairs	63	50	79
2	outdoor asphalt	104	96	92
2	outdoor soft grass	66	62	94
3	indoor with shoes	100	97	97
3	down stairs	69	64	93
3	up stairs	70	68	97

**Table 4.2.** Step detection algorithm was evaluated using multiple individuals on different surfaces

evaluation.

## 4.2 Linear map insertion through additive forces

This algorithm makes use of the ideas of additive forces affecting ones trajectory, as mentioned in Section 2.5.3. This concept allows for using the information from a map in a linear manner, which allows for comparatively lightweight algorithms to be utilized.

Using the constant velocity model (2.10) with the position and velocity in the state vector, together with the additive forces as in (2.9), gives the process dynamics

$$\mathbf{x}_{k+1} = \begin{pmatrix} I_2 & TI_2 \\ 0 & I_2 \end{pmatrix} \mathbf{x}_k + \begin{pmatrix} \frac{T^2}{2} I_2 \\ T I_2 \end{pmatrix} \mathbf{f}_k + \begin{pmatrix} \frac{T^2}{2} I_2 \\ T I_2 \end{pmatrix} w_k, \quad (4.9)$$

where the state vector,  $\mathbf{x}$  is represented as  $(\mathbf{p}^T, \mathbf{v}^T)^T$  and  $\mathbf{f} = (f_x, f_y)^T$  is the repelling force from the surrounding walls given the currently estimated position as described in Section 2.5.3.

For evaluation purposes of this approach, it is assumed that  $\mathbf{p}_0$  is known and that the way the Beebadge is worn is known and static relative to the body.

However, since the velocity is separated in a directional vector and a size, from the IMU output and the step detection algorithm, it is convenient to have the state vector in the same form. We formulate the state vector as

$$x = \begin{pmatrix} \mathbf{p} \\ \mathbf{h} \\ v \end{pmatrix}, \quad (4.10)$$

where  $\mathbf{p}$  is the position,  $\mathbf{h}$  the heading as defined in (4.3), and  $v$  the speed. The measurement equations becomes very simple with this state representation,

$$y_{\text{imu},k} = \mathbf{h}_k + e_k \quad (4.11)$$

$$y_{\text{step},k} = v_k + e_k \quad (4.12)$$

The simple measurement equations yield nonlinearities in the system dynamics, but with a simple Taylor expansion these become reasonably linearized. By adapting (4.9) to the state vector (4.10) we get

$$f(\mathbf{x}_k, \mathbf{f}_k) = \begin{pmatrix} \mathbf{p}_k + T v_k \mathbf{h}_k + \frac{T^2}{2} \mathbf{f}_k \\ \frac{v_k \mathbf{h}_k + T \mathbf{f}_k}{\|v_k \mathbf{h}_k + T \mathbf{f}_k\|} \\ \|v_k \mathbf{h}_k + T \mathbf{f}_k\| \end{pmatrix} \mathbf{x}_k + \begin{pmatrix} \frac{T^2}{2} I_2 \\ T I_2 \\ T \end{pmatrix} w_k \quad (4.13)$$

which together with the Jacobian,  $f'(\mathbf{x}_k, \mathbf{f}_k)$ , can be used in the EKF as described in Algorithm 2.  $f'(\mathbf{x}_k, \mathbf{f}_k)$  is defined as

$$f'(\mathbf{x}_k, \mathbf{f}_k) = \begin{pmatrix} 1 & 0 & T v_{1,k} & 0 & 0 & T h_{k,x} \\ 0 & 1 & 0 & 0 & 0 & T h_{k,y} \\ 0 & 0 & \frac{v_{1,k}(f_y T + h_{k,y} v_{1,k})^2}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} & -\frac{v_{1,k}(f_x T + h_{k,x} v_{1,k})(f_y T + h_{k,y} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} & \frac{T(f_y h_{k,x} - f_x h_{k,y})(f_y T + h_{k,y} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} \\ 0 & 0 & -\frac{v_{1,k}(f_x T + h_{k,x} v_{1,k})(f_y T + h_{k,y} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} & \frac{v_{1,k}(f_x T + h_{k,x} v_{1,k})^2}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} & \frac{T(-f_y h_{k,x} + f_x h_{k,y})(f_x T + h_{k,x} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|^3} \\ 0 & 0 & \frac{v_{1,k}(f_x T + h_{k,x} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|} & \frac{v_{1,k}(f_y T + h_{k,y} v_{1,k})}{\|v \mathbf{h}_k + T \mathbf{f}_k\|} & \frac{v_{1,k}(h_{k,x}^2 + h_{k,y}^2) + T f_x h_{k,x} + T f_y h_{k,y}}{\|v \mathbf{h}_k + T \mathbf{f}_k\|} \end{pmatrix} \quad (4.14)$$

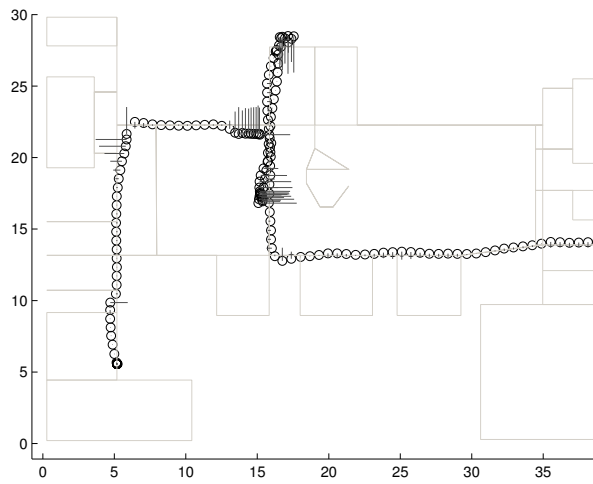
We cannot normalize the direction vector  $\mathbf{h}$  when the norm  $\|v_k \mathbf{h}_k + T \mathbf{f}_k\| = 0$ . In the case when  $\mathbf{f} = \mathbf{0}$  and  $v = 0$ , we get

$$\lim_{v_k \rightarrow 0} f'(\mathbf{x}_k, \mathbf{0}) = \begin{pmatrix} 1 & 0 & 0 & 0 & T h_{k,x} \\ 0 & 1 & 0 & 0 & T h_{k,y} \\ 0 & 0 & h_{k,y}^2 & -h_{k,x} h_{k,y} & 0 \\ 0 & 0 & -h_{k,x} h_{k,y} & h_{k,x}^2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.15)$$

Another problematic case occurs when  $v \mathbf{h} = -T \mathbf{f}$  but due to the performance issues with the approach in total, as described in the following section, this is not taken care of.

### Performance

Using the EKF recursions described in Section 3.1.1 we get the estimated trajectory shown in Figure 4.3. It can be seen that the added forces effectively prevent the trajectory from cutting corners in the map.



**Figure 4.3.** Estimated trajectory with EKF and additive forces from the map. The segments are shown as the underlying grey lines and the added forces for each estimation are shown as lines drawn from the middle of each estimation marker (circles).

Representing the map only with additive forces aids the navigation slightly when moving across hallways, but the momentum that is lost because of repelling forces is never recovered. This effectively makes the velocity to be underestimated and in the long run the filter to diverge.

Due to the nature of the KF, only one estimator of the state vector is kept track of. While this estimator is the most probable one given a model and measurements, our experiments show that it is insufficient to keep track of just one estimator. If the estimator gets off the true trajectory, it is hard to find a way back to the true position using only relative position measurements.

### Remedies

The particle filter evaluates multiple hypotheses as an intermediate step between the time and measurement updates and the computation of the estimator. The PF prevents particles to cross walls with information from the map, but the estimator is unaffected by the map and can therefore cross walls if it is needed. This

often occurs when multiple swarms of particles exist, and this is the expected behaviour since there might exist multiple equally likely trajectories. Once a unique trajectory has been measured, only one swarm of particles should remain and the estimator hopefully converges to the ground truth.

The MPF allows for having multiple hypotheses of the nonlinear states while still keeping the linear(-ized) states optimally chosen, given the nonlinear hypothesis for each particle. However, in this case the linear(-ized) states does not depend on the nonlinear states, i.e. the velocity does not depend on the position in the process dynamics, which leads to the variant of having the velocity as global states for all the particles and just evaluating the belief of the position of every particle.

### 4.3 Particle filter based approaches

The particle filter would with most certainty be the optimal solution in terms of accuracy and simplicity during implementation, but at the same time the particle filter suffers severely from the computational burden of the large amount of particles needed to achieve a good resolution for all dimensions in the state vector. It is therefore crucial to reduce the number of needed dimensions in the state vector and if possible marginalize as much as possible in the state vector.

#### 4.3.1 Particle Filter – Light

A very light weight particle filter is developed where computational requirements are kept as low as possible. The aim of this specific filter is to implement it on the Beebadge which has severe limitations in computational resources. The state vector only includes the position in the plane, the current segment and the current plane the particle belongs to, giving the state vector,

$$\mathbf{x} = (p_x, p_y, s, f)^T, \quad (4.16)$$

where  $s$  is the current segment and  $f$  is the current plane, or floor, for the particle.

This allows keeping a very low particle count yet sufficiently high spatial resolution. A severe limitation to this filter is that the endpoint must be fixed to the wearer, and with a known orientation relative to the wearer.

#### Step update

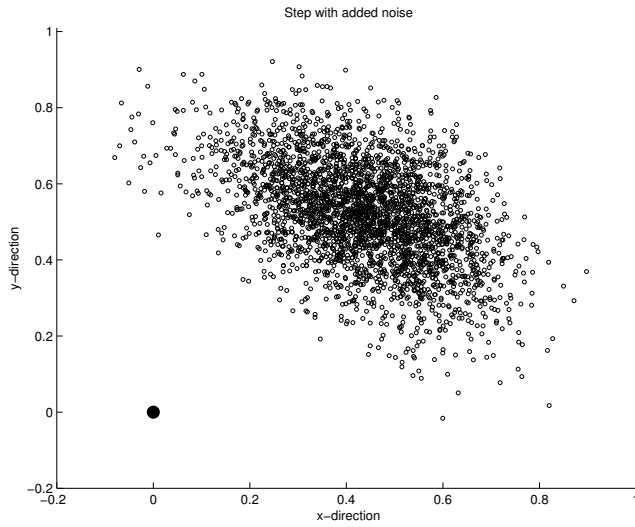
When a step has been taken the product  $s_l s_{\delta l}$  is calculated, where  $s_l$  is the step length and  $s_{\delta l}$  is the step length modifier. The step length is assumed to be constant and the step length modifier is fetched from the current map segment, details about the step length modifier can be found in Section 2.5.2. Each particle is moved  $s_l s_{\delta l}$  in the direction of travel with added noise. A conventional particle filter only changes the state of the particle during the time update according to the system dynamics, in this case we avoid the time update



all together. The advantage of this is that it becomes natural how to introduce the map into the filter, also when no new step is detected no radio traffic is used nor any extra computations are performed, which is a crucial saving. Noise should be added in both the direction and the step length. Since the heading,  $\mathbf{h}_s^w$  is normalized we can rewrite the heading as  $\mathbf{h}_s^w = (\cos(\theta), \sin(\theta))^T$  where  $\theta$  is the heading in radians. This generates the following equations,

$$\begin{pmatrix} d_{\text{step},x} \\ d_{\text{step},y} \end{pmatrix} = (s_l s_{\delta l} + e_l) \begin{pmatrix} \cos(\theta + e_\theta) \\ \sin(\theta + e_\theta) \end{pmatrix}, \quad e_\theta \sim \mathcal{N}(0, \sigma_\theta^2), \quad e_l \sim \mathcal{N}(0, \sigma_l^2). \quad (4.17)$$

This generates a cloud of particles like the shown below in Figure 4.4.



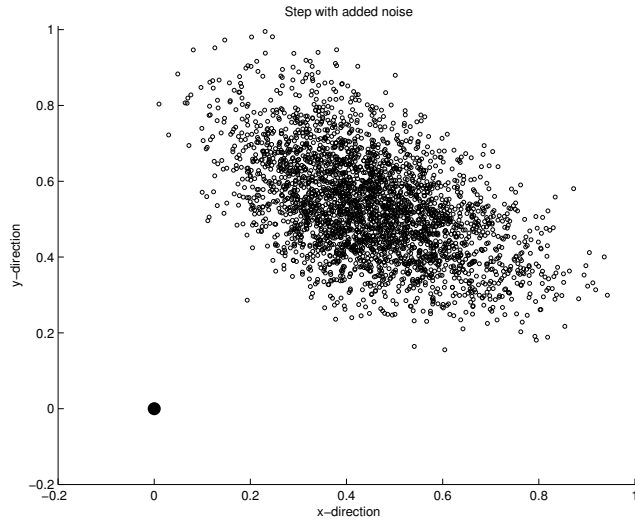
**Figure 4.4.** Step with Gaussian noise added both to the heading and the step length, heading 50 degrees and step length of 0.7m, the noise is over exaggerated to better illustrate the spread

This strategy would generate the best result. A severe limitation with this is that the heading  $\theta$  must be computed since the output from (4.2) is in vector form and also that  $2N$  trigonometric function calls must be done during each measurement update, where  $N$  is the amount of particles present. To avoid this

computational burden, (4.17) can with simple trigonometry be rewritten to

$$\begin{aligned}
\begin{pmatrix} d_{\text{step},x} \\ d_{\text{step},y} \end{pmatrix} &= (s_l s_{\delta l} + e_l) \begin{pmatrix} \cos(\theta) \cos(e_\theta) - \sin(\theta) \sin(e_\theta) \\ \sin(\theta) \cos(e_\theta) + \cos(\theta) \sin(e_\theta) \end{pmatrix} \\
&\approx \left/ e_\theta \text{ small, use Taylor expansion} \right/ \\
&\approx (s_l s_{\delta l} + e_l) \begin{pmatrix} \cos(\theta) - e_\theta \sin(\theta) \\ \sin(\theta) + e_\theta \cos(\theta) \end{pmatrix} \\
&= (s_l s_{\delta l} + e_l) \begin{pmatrix} \mathbf{h}_{s,x}^w - e_\theta \mathbf{h}_{s,y}^w \\ \mathbf{h}_{s,y}^w + e_\theta \mathbf{h}_{s,x}^w \end{pmatrix}, \\
e_\theta &\sim \mathcal{N}(0, \sigma_\theta^2), \quad e_l \sim \mathcal{N}(0, \sigma_l^2)
\end{aligned} \tag{4.18}$$

This generates a particle cloud as in Figure 4.5 with a lot less computations.



**Figure 4.5.** Step with Gaussian noise added using Taylor expansion for the heading, heading 50 degrees and step length of 0.7m, the noise is over exaggerated to better illustrate the spread

The two methods do not generate the same result but evaluating the two different methods shows that the performance is only marginally effected by this simplification. After particles have been moved they are weighted together to form an estimate according to (3.14). A resampling step described in Section 3.2.1 is taken to remove less likely particles with particles that have higher probability, this is to ensure that the particle cloud is continuously updated to represent the actual distribution.

### Filter algorithm

Algorithm 6 describes the lightweight particle filter in detail.

---

**Algorithm 6** Lightweight particle filter
 

---

1. **Measurement update**

Obtain latest  $\mathbf{h}_s^w$  from (4.2). Obtain latest step length  $s_k$ .

**for** each particle in  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^N$

Extract step length modifier from the current segment for the given particle  $s_{\delta l}^i$

Use Equation 4.18 to generate the step

$$\mathbf{d}_{\text{step}} = \begin{pmatrix} d_{\text{step},x} \\ d_{\text{step},y} \end{pmatrix} \quad (4.19)$$

Move the particle and reweigh it according to the map

$$\begin{aligned} \tilde{\mathbf{x}}_k^i &= \tilde{\mathbf{x}}_k^i + \mathbf{d}_{\text{step}}, \\ w_k^i &= w_k^i p_{\text{map}}(\tilde{\mathbf{x}}_k^i), \end{aligned}$$

where  $p_{\text{map}}$  is the relative probability density described in Section 2.5.3.

Calculate the estimate using

$$\hat{\mathbf{x}}_k = \sum_{i=1}^N \frac{w_k^i}{c} \tilde{\mathbf{x}}_k^i, \quad (4.21)$$

where  $c = \sum_{i=1}^N w_k^i$ .

Resample using (3.15)

Constant position used, so prediction is simply

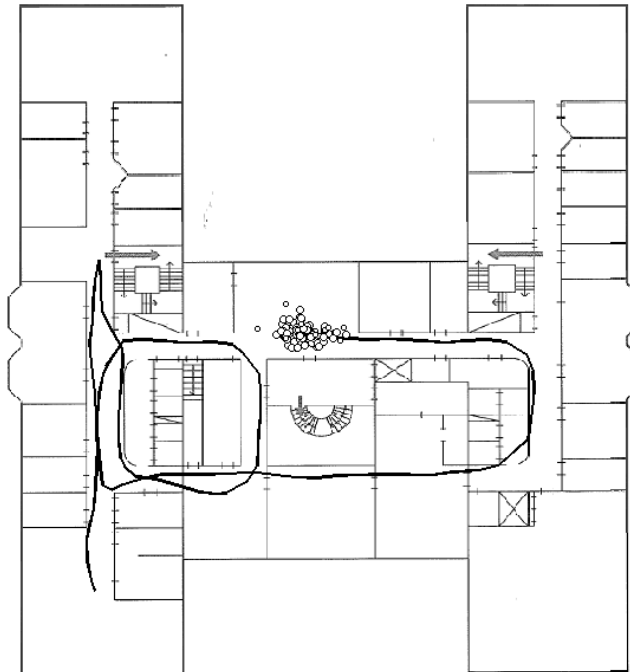
$$\tilde{\mathbf{x}}_{k+1}^i = \tilde{\mathbf{x}}_k^i, \quad (4.22)$$

for each particle in  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^N$

---

### Performance

Matlab evaluation of the lightweight particle filter show that this filter can be run with very few particles with maintained accuracy. Tests were run with as low as 100 particles with good result. The problem with running with very few particles is that the filter does not become very robust. The probability of the filter diverging increases substantially when decreasing the amount of particles. In Figure 4.6 an example trajectory from walking around the office at Xdin. No ground truth is shown but the overall shape is very accurate from the actual path.



**Figure 4.6.** Particle Filter Light output when walking around the office at Xdin, 100 particles represented as circles, size of a circle indicates the weight of the particle

### 4.3.2 Particle Filter – Heavy

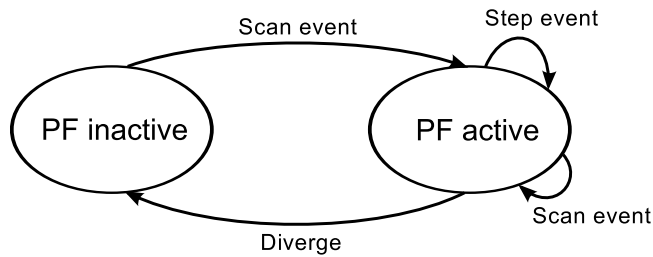
A more computationally heavy particle filter than the one described above is developed without the intent to run it on the Beebadge. The major difference between this filter and the lightweight particle filter is that the Beebadge can be worn in any orientation. The Beebadge must still be firmly attached to the body for the step detection to work. This gives a larger state vector,

$$\mathbf{x} = (p_x, p_y, s, f, \theta)^T, \quad (4.23)$$

where  $s$  is the current segment and  $f$  is the current plane, or floor, for the particle. This filter requires a lot more particles to achieve both a good spatial resolution and a high resolution in  $\theta$ .

### Finite state machine

Figure 4.7 shows the finite state machine for the particle filter. When the program starts for the first time it is set to the inactive state, when the first scan result arrives the filter is started and transitions to the active state. If the filter diverges it will transition to the inactive state and awaits the next scan result.



**Figure 4.7.** Finite State Machine for the filter. The initial state is the inactive state. It transitions to active when a new scan arrives and transitions to inactive when the filter diverges

### Filter start up

When a new scan result from a arrives while the filter is inactive, a start up procedure is run to initiate the filter around the coordinator. Particles are evenly spread out within the range of the coordinator.

### Divergence test

A simple test to determine if the filter has diverged is if

$$\sum_{i=1}^N w_k^i < \epsilon, \quad (4.24)$$

which means if the sum of all weights before the resampling is below a set threshold  $\epsilon$ . When the filter has diverged it is returned to its initial state and will restart as soon as a new scan result arrives. This new scan result provided sufficient information to allow the filter to restart.

### Step update

When a step has been taken each particle is moved one step length in the direction of travel with added noise. The direction of travel is computed in the

same way as in the light particle filter but with the added heading compensation from (4.6b) to account for the heading deviation for each particle. Noise is added in both the direction and the step length which generates a cloud of particles as in Figure 4.4. After each particle is moved, all particles are weighted together to form an estimate. A resampling step is performed to remove less likely particles with particles that have higher probability, this is to ensure that the particle cloud is continuously updated to represent the actual distribution.

### Scan update

When a new scan update arrives and the filter is inactive, a filter start up is performed around the base station from which the scan result arrived. If the filter is active the weight of all particles out of the range of the coordinator are set to zero, according to the modelled uniform distribution in (2.4). After this has been performed a resampling step is performed.

### Reducing the number of particles

Decreasing the amount of particles would decrease the computational resources needed to run the filter. A proposed method of determining the amount of particles to be resampled to is

$$N_{\text{new}} = \min(\alpha N_{\text{old}} + (1 - \alpha)N_{\text{eff}}, \beta N_{\text{old}}, N_{\text{min}}), \quad (4.25)$$

where  $N_{\text{new}}$  is the amount of particles that should be kept after resampling,  $\alpha N_{\text{old}} + (1 - \alpha)N_{\text{eff}}$  is the new particle count which is a compromise between the old particle count and the efficient particle count.  $\alpha \in [0, 1]$  is used to shift focus between either the old particle count or the effective particle count. Where a high  $\alpha$  would lead to a slow and conservative decline and a low alpha would lead a quicker decline. A quicker decline could threaten the diversity of particles during later updates.  $\beta N_{\text{old}}$  where  $\beta \in [0, 1]$  is to guard against a for some reason faulty  $N_{\text{eff}}$  which potentially could ruin the filter. This is also to guard against lost diversity which might be needed during later updates. Finally  $N_{\text{min}}$  is the minimum amount of particles. This is needed because  $N_{\text{eff}} \leq N_{\text{old}}$  which would lead to that the particle count would monotonically decrease to 1 without the lower bound on particles.

The proposed method relies on some intuition on the number of particles needed to represent the underlying density and is only useful for reducing the number of particles. The method have shown to work well during the convergence phase of a particle filter, where the initial belief is highly uncertain and subsequent measurements reduces this uncertainty. There exist methods to determine the absolute number of required particles to represent the underlying density to some precision. The Kullback-Liebler distance (KLD) is a commonly used measure to determine the absolute number of required particles given the true underlying density, or an approximation thereof [29, 30]. The KLD introduces additional complexity in the standard PF algorithm which affects the real-time performance [29]. Our simple method is computationally efficient and

is sufficient for our purposes. Figure 4.8 illustrates how the number of particles decreases as the filter converges. A number of snapshots are included at some time instances to illustrate the particles are spread during the convergence phase.

### Algorithm

Algorithm 7 shows the flow of the particle filter with angle deviation as an extra state.

### Performance

This filter initially requires a high particle count to maintain decently high resolution in all dimensions of the state vector. The filter normally converges to a correct position with the correct heading deviation after only a few turns. In Figure 4.10(a) an example of a simple stroll around the office at Xdin, comparing with Figure 4.6 the heavy filter has a smoother trajectory and it has a lot lower probability of diverging due to the increased amount of particles. When comparing the performance with and without the map support it can be clearly seen that the map to a large extent will remove the drift caused in the dead reckoning system. This is depicted in Figure 4.9

The sensitive part of the heavy particle filter is during the first few steps, when the uncertainty of both the position and  $\theta$  is large. The particles may be spread over several different floors, as shown in Figure 4.10(b), and if the true trajectory pass through narrow passages the hypotheses surrounding it may be resampled to other areas which eventually will turn out to be false. This problem is exemplified in Figure 4.11 where the green trajectory illustrates the particle swarm following the ground truth and the red trajectory illustrates a mirrored swarm which survives due to the mirrored structure of the room. When walking several laps around the block in the green trajectory, particles will more likely be resampled from the green to the red trajectory than the other way around because of punishment by the map when cutting corners. The green swarm will starve and eventually vanish. When the red swarm hits the wall, the filter will diverge.

When running the filter on a modern 3.1 GHz PC it will with a good margin run in real-time. Some simple benchmarking data are shown in Table 4.3. The filter with 800 particles executes in roughly 1000 times real-time at a normal walking pace ( $\approx 2$  steps/second). This means that a single external PC can handle over 1000 active users that continuously walk around, in real situations people would not move constantly so the actual capacity is much higher.

## 4.4 Evaluation

After evaluating the approaches mentioned in this chapter we decided to implement the heavy particle filter. Using an EKF supported by additive forces from walls suffers from the problem of only one hypothesis being evaluated at a

---

**Algorithm 7** Heavyweight particle filter
 

---

**1. Measurement update**

Obtain latest step length  $s_k$ .

**for** each particle in  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^N$

Extract step length modifier from the current segment for the given particle  $s_{\delta l}^i$

Construct the rotation matrix  $R$  using the angular offset state  $\theta$  in the given particle, use  $R$  in Equation 4.6b to get the estimated step direction. Equation 4.18 generates the step with added noise.

$$\mathbf{d}_{\text{step}} = \begin{pmatrix} d_{\text{step},x} \\ d_{\text{step},y} \end{pmatrix} \quad (4.26)$$

Move the particle and reweigh it according to the map

$$\begin{aligned} \tilde{\mathbf{x}}_k^i &= \tilde{\mathbf{x}}_k^i + \mathbf{d}_{\text{step}} \\ w_k^i &= w_k^i p_{\text{map}}(\tilde{\mathbf{x}}_k^i) \end{aligned}$$

where  $p_{\text{map}}$  is the relative probability density described in Section 2.5.3.

Calculate the estimate using

$$\hat{\mathbf{x}}_k = \sum_{i=1}^N \frac{w_k^i}{c} \tilde{\mathbf{x}}_k^i, \quad (4.28)$$

where  $c = \sum_{i=1}^N w_k^i$ .

Resample using (3.15) with new particle count using (4.25)

Constant position used, so prediction is simply

$$\tilde{\mathbf{x}}_{k+1}^i = \tilde{\mathbf{x}}_k^i, \quad (4.29)$$

**for** each particle in  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^N$

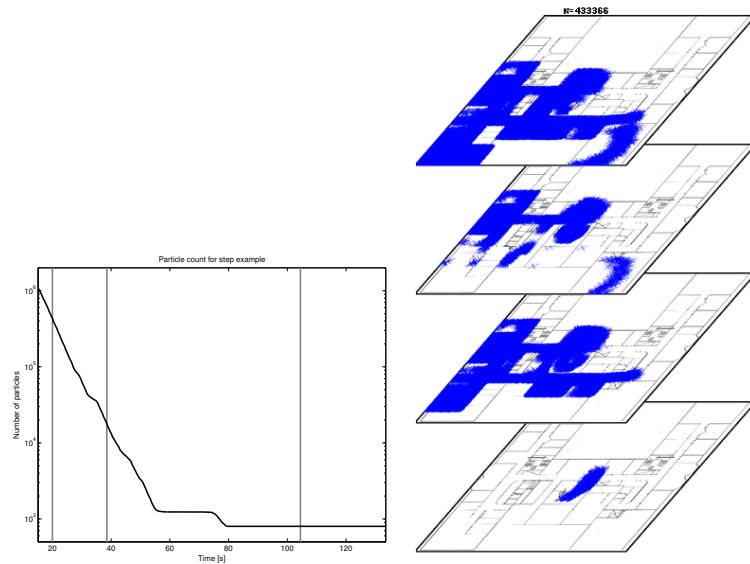
---



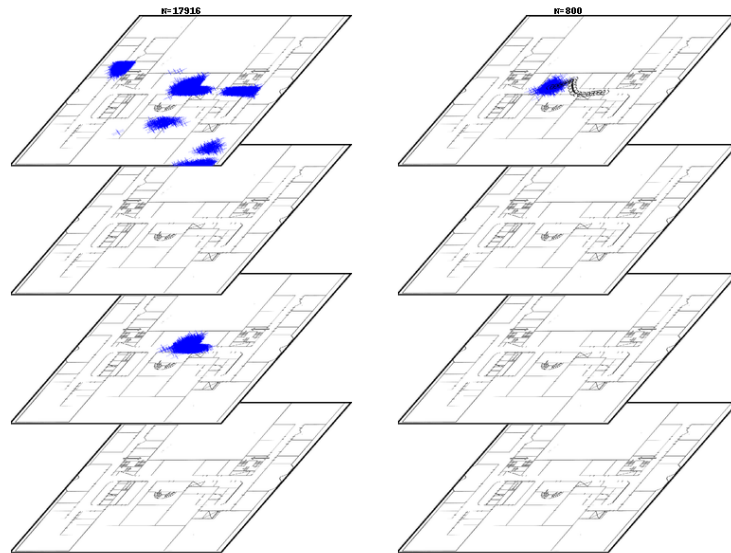
Time instance, first step, $t_1$	17.7s
Time instance, last step, $t_{129}$	84.0s
Number of steps, $n_{\text{steps}}$	129
Total PF execution time, $t_{\text{exec}}$	56003 $\mu\text{s}$
Avg. step rate	$n_{\text{steps}}/(t_{129} - t_1) = 1.94 \text{ Hz}$
x Real-time	$(t_{129} - t_1)/t_{\text{exec}} = 1183.7$

**Table 4.3.** Heavyweight particle filter benchmark based on data from a continuous walk during some minute.

single time. This makes it very vulnerable to errors from other sub systems such as an temporary incorrect heading or missed steps. Furthermore this method requires the Beebadge to be worn in a known way and with a known starting position. The light weight particle filter performed well in simulations, but the requirements that the way the Beebadge is worn must be known, as well as the starting position, makes it unusable in practical situations. The added performance and versatility in terms of initial conditions and robustness outweighs the increase in computational requirements in the heavyweight PF.

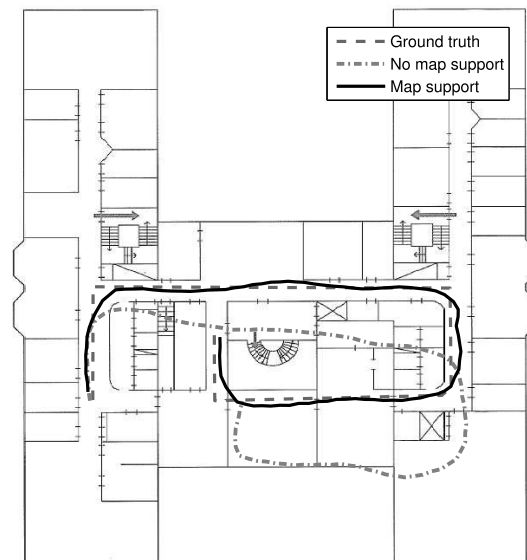


(a) Number of particles over time. (b)  $N=433366$  at highlighted time instance 1. Snapshots at the highlighted time instances are shown in the other three figures.

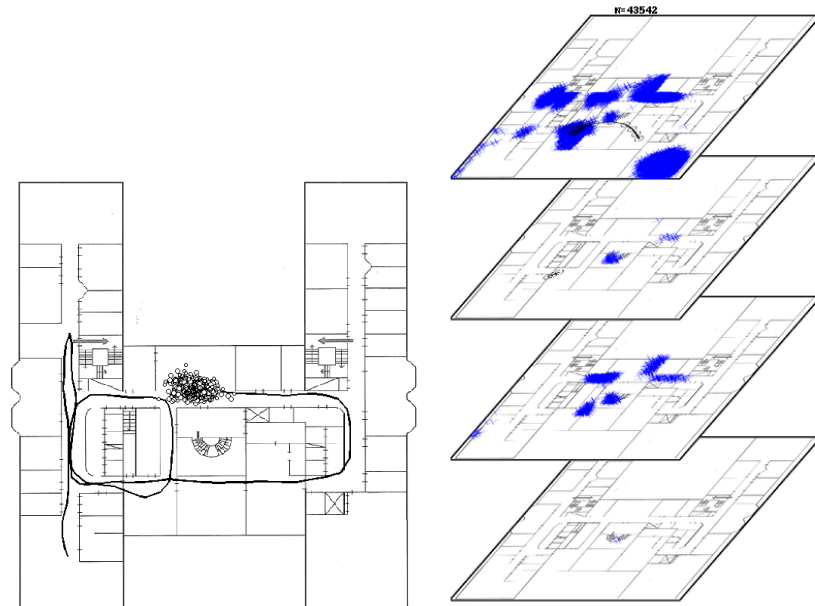


(c)  $N=17916$  at highlighted time instance 2. (d)  $N=800$  at highlighted time instance 3.

**Figure 4.8.** Illustration of decreasing number of particles as the PF converges.

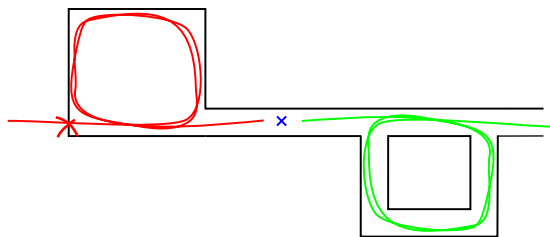


**Figure 4.9.** A stroll around the office at Xdin. Ground truth is shown together with dead reckoning without map support and dead reckoning aided by the map.



(a) An estimated trajectory at Xdin's office, 1000 particles represented as circles, size of a circle indicates the weight of a particle. (b) A scenario where the filter have not converged yet. The spread in hypotheses is caused by a large coverage for a coordinator.

**Figure 4.10.** Output from the particle filter.



**Figure 4.11.** Illustration of a problematic case where a correct trajectory (green) is being starved by an incorrect trajectory (red), causing the filter to potentially diverge.

## Chapter 5

# Implementation

The system is divided into separate parts, each with its defined tasks in the complete system. An overview of the system is shown in Figure 5.1.

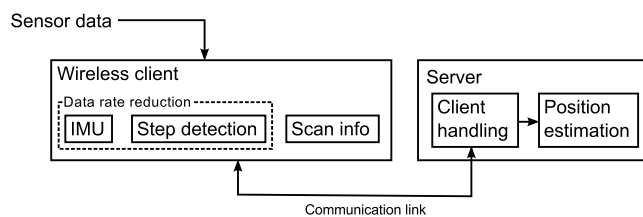


Figure 5.1. Overview of the implemented system.

### 5.1 Wireless client

The task of the battery driven Beebadge is to reduce the amount of data needed to be sent over the radio. In order to achieve such a reduction, the Beebadge keeps track of its attitude by filtering the inertial sensor data in an IMU. It also detects when a step has been taken according to Algorithm 5. By running these two algorithms in parallel the high sample rate inertial data can be reduced to just above some event per second. When a step is detected, the current estimated attitude is sent as an event over the radio channel to a receiver which uses the event as a measurement in the filter. As a measure of the absolute position of the endpoint, the known positions of the responding coordinators are also sent to a receiver. Sending this kind of sparse sensor data does not cost significantly more than sending position estimations over the radio, which had been the case if the Beebadge had estimated the position on its own.

### 5.1.1 Inertial Measurement Unit

The IMU used in this thesis is proposed by [31]. It is a light weight IMU and is therefore well suited for our computationally constrained hardware. The output from the IMU is a four dimensional vector which represents the orientation on quaternion form described in (4.1).

### 5.1.2 Step detection

The step detection algorithm derived in Algorithm 5 runs at the same rate as the accelerometer samples enter the system. The algorithm reduces the relatively high data rate to around 2 Hz depending on the step rate of the person who wears the device.

The current orientation, in quaternion form, is sent along with each step at this lower data rate to a server which is running all particle filters.

### 5.1.3 Scan results

The scan results described by (2.4) are used as indicators of the absolute position of the device. Such scans are performed at 0.1 Hz in the current environment and multiple responses can be returned from the same scan.

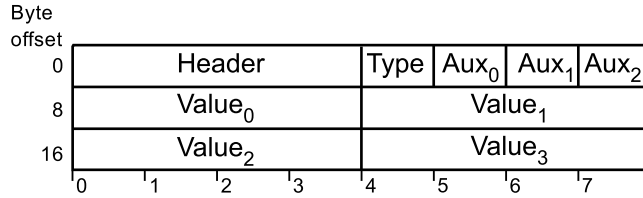
### 5.1.4 Communication format

We are using a simple fixed length packet format for the communication of sensor data between the client and the server. The packet is initiated with a constant multibyte header, used both for indication of the start of a packet and for identification of endianness differences between the client and the server hardware.

The endianness is the ordering of the bytes within a word on a certain platform. Two types of endianness exist, big and little endian. In big endian the bytes are arranged in the same way as they would have been if they were written on a paper, from the most to the least significant byte. Little endian, on the other hand, has the bytes are ordered in reverse order compared to big endian, this representation might seem illogical but it can be shown to simplify arithmetic computations.

The header is defined to the same value for both the client and the server. The client packs its messages starting with the header as defined. When the server unpacks the message it checks for the header in both forward and reversed order. If it is discovered to be reversed, the necessary rearrangement of the data in the packet is done before it is taken care of.

The structure of the packet is shown in Figure 5.2, the length is fixed for simplicity and the contents of the different fields depend on the type field. Further details of the fields are not included in the thesis.



**Figure 5.2.** The datastructure of the packets containing sensor data.

## 5.2 Server

The server is assumed to have much greater computational capacity than the Beebadges and is also assumed to not suffer from any energy constraints. Therefore the particle filter described in Algorithm 7 is implemented on the receiver side to keep track of the users position. By centralizing the filter, the position estimates of other users can be fused to improve the overall quality of the estimates. This requires some model of the behaviour of the users, which is out of the scope for this thesis.

### 5.2.1 Random number generation

The particle filter may require Gaussian distributed random numbers, depending on the models used. This is the case in this thesis, which is why methods for generating such numbers have been evaluated.

Two different methods have been compared. Firstly the two methods are described, then a performance comparison is done. Both the quality of the generated numbers and the execution speed is compared.

#### Box-Müller transform

The *Box-Müller transform* uses properties of the unit circle and the distribution of angles and the squared radius therein to arrive in standard normal variables as described in Algorithm 8.

---

#### Algorithm 8 Box-Müller transform

If  $U_1$  and  $U_2$  are independent and uniformly distributed in the interval  $[0, 1)$ , then

$$X_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \quad (5.1a)$$

$$X_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2) \quad (5.1b)$$

are independent and standard normally distributed [32].

---

The numbers are generated in pairs, and four relatively computationally demanding functions have to be computed for each pair of generated numbers.

### Marsaglia polar method

The *Marsaglia polar method* is directly based on the ideas behind the Box-Müller transform, but assumes that it is more computationally demanding to compute the trigonometric sine and cosine functions than generating uniformly distributed random values. The method is described in Algorithm 9, where similarities with Algorithm 8 can be identified.

---

#### Algorithm 9 Marsaglia polar method

---

Sample  $u_1$  and  $u_2$  from a uniform distribution in the interval  $[-1, 1)$  until  $u_1^2 + u_2^2 < 1$ . Then use  $u_1$  and  $u_2$  for computing

$$x_1 = u_1 \sqrt{\frac{-2 \ln(u_1^2 + u_2)}{u_1^2 + u_2}} \quad (5.2a)$$

$$x_2 = u_2 \sqrt{\frac{-2 \ln(u_1^2 + u_2)}{u_1^2 + u_2}} \quad (5.2b)$$

where  $x_1$  and  $x_2$  are independent and standard normally distributed [33]. With  $1 - \pi/4 \approx 21\%$  chance, the uniformly distributed samples  $u_1$  and  $u_2$  takes values outside the unit circle and will have to be resampled.

---

### Comparison

The quality of the measurements is the same, since the underlying properties of the two algorithms are the same. The distribution of  $10^6$  generated numbers are shown in Figure 5.3 compared with a standard normal probability density function.

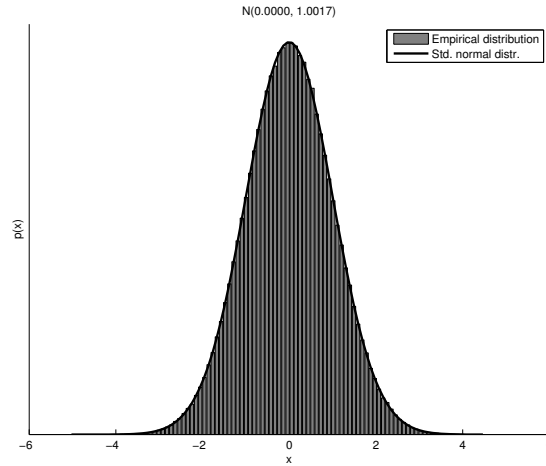
The execution times for the two methods are compared in Table 5.1, it can be seen that the Marsaglia polar method performs slightly better than the Box-Müller transform on the tested platform.

Algorithm	Execution time
Box-Müller transform	18.071 s
Marsaglia polar method	16.572 s

**Table 5.1.** Execution time for generating  $10^8$  standard normal distributed values on a 1.83 GHz x86 processor with uniformly distributed samples generated with glibc.

A more recent algorithm called the *Ziggurat Method* is claimed to execute faster than the two methods given above [34]. It is not included in this thesis since its complex structure makes it fall outside of our scope.





**Figure 5.3.** Quality of  $10^6$  generated standard normal distributed values.

### 5.2.2 Particle filter utilities

Several generic particle filter utilities are implemented to enable some degree of abstraction in the filter implementation. It is common in the literature to arrange the steps described in Algorithm 3 in different ways to fit different applications. We have chosen to make the tasks of the utilities small enough to enable implementation in different filtering applications.

#### Resampling

The most crucial function to optimize is the resampling because it introduces a large computational overhead compared to other recursive optimization algorithms. The algorithm implemented is the *systematic resampling* algorithm [35] due to its comparatively low computational requirements [36]. The algorithm has been modified to include the possibility to change the number of particles in the resampled set from the number in the prior set of particles as described in Section 3.2.1.

The resampling algorithm is described in Algorithm 10, where floating point round-off error avoidance is included for the sake of completeness. Further, the two sets  $\{\mathbf{x}_k^i\}_{i=1}^N$  and  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^M$ , where  $N$  is the prior size of the particle set and  $M$  the size after resampling, are represented by two separate memory allocations. Those allocations are used alternately to avoid double copying between them. The areas are resized when either the new size is larger than the currently allocated or when the new size has shrunk to half the allocated size. We choose to not shrink the size every time it is possible due to the computational cost of memory management.

**Algorithm 10** Systematic resampling

Given the set of particle states  $\{\mathbf{x}_k^i\}_{i=1}^N$ , the set of particle weights  $\{w_k^i\}_{i=1}^N$ , and the number of particles  $N$ , the new number of particles  $M$  we compute the resampled set of particle states  $\{\tilde{\mathbf{x}}_k^i\}_{i=1}^M$ , and the set of particle weights  $\{\tilde{w}_k^i\}_{i=1}^M$ .

```

1:  $\Delta r \leftarrow 1/M$ 
2:  $r \leftarrow v/M$ , where  $v \sim \mathcal{U}(.5, .5)$ 
3:  $j \leftarrow 1$ 
4:  $w_c \leftarrow w_k^1$ 
5: for  $i = 1 \rightarrow M$  do
6:   while  $w_c < r$  do
7:     if  $j \leq N$  then ▷ Avoid floating point round-off errors.
8:        $j = j + 1$ 
9:        $w_c = w_c + w_k^j$ 
10:    end if
11:   end while
12:    $r = r + \Delta r$ 
13:    $\tilde{\mathbf{x}}_k^i \leftarrow \mathbf{x}_k^j$ 
14: end for
15:  $\{\tilde{w}_k^i\}_{i=1}^M \leftarrow 1/M$ 
16:  $N \leftarrow M$ 

```

**Numerical tools**

A function for normalizing the set of weights  $\{w_k^i\}_{i=1}^N$  and at the same time monitor the sum of the weights for divergence is implemented. If the sum of the weights falls below a given threshold, the function indicates that the filter has diverged and that the weights cannot be normalized. Otherwise, the weights are normalized so that the weights sum to one. The function is described in Algorithm 11.

**Algorithm 11** Particle weight normalization

Given the set of particle weights  $\{w_k^i\}_{i=1}^N$ , and a divergence threshold  $w_{\text{th}}$  the normalization procedure is implemented as

```

1:  $s \leftarrow \sum_{i=1}^N w_k^i$ 
2: if  $s < w_{\text{th}}$  then
3:   return False ▷ The filter has diverged
4: else
5:   for  $i = 1 \rightarrow N$  do
6:      $\tilde{w}_k^i \leftarrow w_k^i/s$ 
7:   end for
8: end if
9: return True ▷ Normalization went well

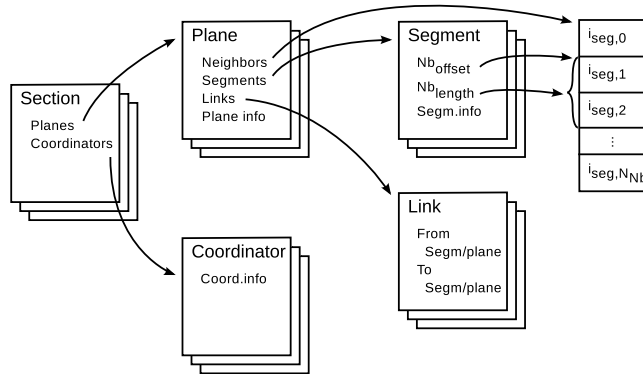
```

A simple function for computing the number of effective particles is also

implemented. This function assumes that the particle weights are normalized and computes (3.16).

### 5.2.3 Map

The map is constructed around a series of structures that holds the information regarding the map. Figure 5.4 illustrates the structure of how the map is represented from a software architectural point of view



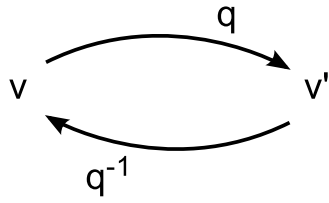
**Figure 5.4.** The software architectural structure of the map.

Each section contains two lists, one list of all planes that are present in the section, the other contains a list of the coordinators that are present in the section. The Coordinator structure contains information about id and location for each coordinator. Each plane contains three lists, one which is used to store the neighbor information for the segments, one list with all segments that are present in the plane, and one list with all links. Furthermore the plane contains basic information about itself, such as height above its parent section.

The map is currently hardcoded into the software. This choice is made to ease development, for a later version the mapping information can with ease be stored on a hard drive and fetched when a certain section is needed.

### 5.2.4 Quaternion library

A simple quaternion library is implemented to allow the programmer to utilize quaternion variables and perform simple arithmetic operations on them. Besides the basic arithmetic operations, a rotation and inverse rotation function is implemented as shown in Figure 5.5,  $v$  is a vector before the transformation and  $v'$  is  $v$  transformed by the quaternion  $q \in Q_l$ . Quaternion algebra is further explained in Appendix A.



**Figure 5.5.** Quaternion functions that was implemented to ease quaternion operations.

# Chapter 6

## Concluding remarks

In this chapter we give our conclusions in Section 6.1 and proposes tasks for future work in Section 6.2 which have the potential of improving our results.

### 6.1 Conclusions

It is fully possible to with simple sensors fuse the sensor data with a map to create a map aided localization system that can locate people to within a few meters in big buildings. The system can handle thousands of users in real-time on a standard desktop computer.

The map system built around the idea of a probability density is a great aid to the tracking system. It is computationally efficient and is easy to traverse to follow the user through the map. To extend the system to help the user to navigate through buildings is a small step from the current system, the structure of the map is well suited for this application as well.

Step detection using a simple 3-axis accelerometer with the proposed step detection algorithm is possible for a range of surfaces and different wearers. Further improvements is to improve its accuracy when walking up and down stairs.

Using time of flight and RSSI indoors proved to be problematic due to the difficulty of accounting for the multipath effects and the varying signal environments. Improvements to this field would greatly increase the accuracy of indoor positioning.

An external particle filter which only receives sparse data from the user can successfully track a pedestrian with very little initial knowledge of the position and orientation of the pedestrian.

Working with single hypothesis approach such as an EKF is vulnerable to even the slightest drift and can easily get stuck at walls. Further this requires a precise knowledge about the initial state for the user, which is rarely known.

Comparisons was made on the energy cost between performing the calculations on board and sending sensor data over the radio for the algorithm to

be run elsewhere, where the computational power is better and the energy constraints are not relevant. The conclusion is that due to the large overhead in the packet transmissions, the difference between sending position and sending selected sensor data to be processed externally is minimal or none. Therefore it is concluded that computational demanding filtering can be performed externally to allow for improved results.

## 6.2 Future work

In this section we describe tasks that can improve the system, but which fall outside of the scope of this thesis.

### 6.2.1 Particle filter implementation on GPUs

Most of the computations in a particle filter are independent and can therefore be done in parallel. Since Graphics Processing Unit (GPU)s are excellent at performing vast amount of smaller computations in parallel implementing the particle filter code on a GPU could improve the performance greatly for a small cost in terms of additional hardware. More about particle filter implementation on GPUs can be found in [37, 38].

### 6.2.2 Navigation

The graph structure of the map is well suited to be used as a navigation aid. If the user desires to move to a certain segment, navigational instructions could be sent to the endpoint to inform in which direction the user should walk to find the desired destination.

### 6.2.3 RSSI fingerprinting

The radio propagation is, as mentioned earlier, very hard to model and predict in indoor conditions. If the RSSI levels are known in and around the vicinity of each coordinator, it would be possible to compare the measured RSSI values with a map. This could be used to greatly increase the accuracy the location for users [39, 40]. The main problem with this is that the whole map must be mapped and surroundings must be static for the RSSI fingerprinting remain valid. Even smaller changes to the surroundings, such as adding a wall, could severely disrupt accuracy.

### 6.2.4 Multi channel time of flight

As mentioned in Section 2.2.1, measuring the time of flight on several frequency channels has showed to both reduce the multipath effect and give an indication of when it is present [17, 8]. Our algorithms relies heavily on dead reckoning and lacks information of the absolute position to a large extent. Our results could

be vastly improved with a reliable time of flight measurement, both during the start up phase of the algorithm and after the filter have converged.

### 6.2.5 SLAM

Simultaneous location and mapping (SLAM) is an increasingly popular algorithm for robotic navigation. The algorithm is out of the scope of this thesis and is therefore not described in any detail. Information about the algorithm can be found in [41, 42, 43]. A straightforward continuation on the studied algorithms is to apply SLAM for mapping signal strength and even the geographical map autonomously. While SLAM may reduce the required manual labor of creating maps, it might require a large amount of data for the maps to converge. Uncertain maps puts higher demands on other sensors for a maintained precision in the position estimates. Studies of those problems are left for the future.

### 6.2.6 Automatic map generation

For this thesis maps are generated by hand using simple drawings over the areas of interest. A possible extension is to extract the topology from building drawings done in [44]. This would greatly decrease the cost associated with generating the maps. This is out of the scope of this thesis and only smaller areas are needed to evaluate the positioning system.

### 6.2.7 Improved map system

The map is currently statically defined in the software. This was done to bypass the tedious programming required to implement a dynamic system to handle the map. A better solution would be to store the map in for example XML files and load them into the software when needed.





# Appendix A

## Quaternion Algebra

A quaternion is a complex number where the imaginary part has been extended to three dimensions.

### A.1 Basic notation

A quaternion can either be denoted as  $q = (q_s, q_i, q_j, q_k)$  or  $q = (q_s, \mathbf{q})$ , where  $\mathbf{q} = (q_i, q_j, q_k)$ . Special quaternions used in this thesis are  $Q_l$  which means it's of unit length and  $Q_v$  which means the  $q_s = 0$

The following basic mathematical operators and was used in this thesis.

$$\text{Addition } p + q = (p_s + q_s, \mathbf{p} + \mathbf{q}) \quad (\text{A.1a})$$

$$\text{Multiplication } p \odot q = (p_s q_s - \mathbf{p} \cdot \mathbf{q}, p_s \mathbf{q} + q_s \mathbf{p} + \mathbf{p} \times \mathbf{q}) \quad (\text{A.1b})$$

$$\text{Conjugation } p^c = (p_s, -\mathbf{p}) \quad (\text{A.1c})$$

$$\text{Norm } \|p\|_2 = \sqrt{p_s^2 + \mathbf{p} \cdot \mathbf{p}} \quad (\text{A.1d})$$

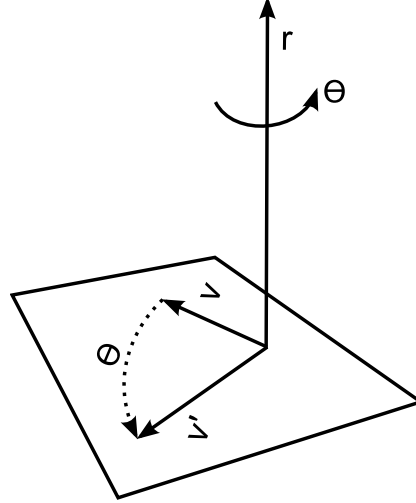
$$\text{Inverse } p^{-1} = \frac{p^c}{\|p\|_2} \quad (\text{A.1e})$$

### A.2 Spatial rotation

Quaternions are very useful as a tool to represent spatial orientation. When using quaternions as aid in spatial rotation the common notation is  $q = (w, x, y, z) \in Q_l$  where  $\mathbf{r} = (x, y, z)$  is the rotational vector and  $w = \cos \frac{\theta}{2}$  where  $\theta$  is the angle of anti clockwise rotation around  $\mathbf{r}$  in radians. Compared to Euler angles they avoid the problem of gimbal lock, but provide a much less intuitive way of representing orientation compared to Euler angles. A rotation of the quaternion  $v = (0, \mathbf{v}) \in Q_v$  where  $\mathbf{v}$  is the desired vector to rotate, with the quaternion  $q \in Q_l$  can be expressed as

$$p = q \odot v \odot q^c, \quad (\text{A.2})$$

where  $p \in Q_v$  is desired rotated vector. Figure A.1 shows how the rotation is performed.



**Figure A.1.** Illustration of how a quaternion rotation is performed

The quaternion rotation can be expanded to a matrix operation shown in (A.3).

$$P = \begin{pmatrix} w^2+x^2+y^2+z^2 & 0 & 0 & 0 \\ 0 & w^2+x^2-y^2-z^2 & 2xy+2wz & 2xz-2wy \\ 0 & 2xy-2wz & w^2-x^2+y^2-z^2 & 2yz+2wx \\ 0 & 2xz+2wy & 2yz-2wx & w^2-x^2-y^2+z^2 \end{pmatrix} \quad (\text{A.3})$$

The first row and first column can be discarded, since the first element of the quaternion is always zero and since only the last three elements of the quaternion is of interest the first row can be discarded. This give a simplified transformation matrix

$$P' = \begin{pmatrix} w^2+x^2-y^2-z^2 & 2xy+2wz & 2xz-2wy \\ 2xy-2wz & w^2-x^2+y^2-z^2 & 2yz+2wx \\ 2xz+2wy & 2yz-2wx & w^2-x^2-y^2+z^2 \end{pmatrix} \quad (\text{A.4})$$

A rotation of a vector  $v \in \mathbb{R}^3$  is then performed as  $v' = P'v$ .

# Appendix B

## Hardware

This appendix will describe the hardware used. The hardware was available to us and assembled at the start of this thesis, so no manufacturing of hardware was needed.

### B.1 Beebadge

The Beebadge is a credit card sized circuit board with a number of sensors as described later. The Beebadge is equipped with a NXP JN5148 chip which contains a 32-bit OpenRISC microcontroller and a IEEE 802.15.4 capable radio. The JN5148 is delivered with a software interface for the IEEE 802.15.4 MAC layer and runs Contiki as operating system. The Beebadge is driven by a 350mAh Li-ion battery which is charged either via USB or by an induction charger. An OLED connector enables simple textual or graphical communication to the user. A buzzer also exists to enable alarms or other monophonic communications to the user.

#### B.1.1 Accelerometer

For simplification, only one accelerometer is mentioned in the report. The Beebadge is in reality equipped with two accelerometers, but only one is used in this thesis. The two accelerometers are MMA8451Q made by Freescale Semiconductor. It has a resolution of 14-bits and the output is in 2's complement. It can work at sampling speeds up to 800 Hz, 100 Hz was used for this thesis. Scaling can be set to  $\pm 2g$ ,  $\pm 4g$ , and  $\pm 8g$ . In this thesis a scaling of  $\pm 8g$  was used. Converting from raw output data to acceleration is done using the following equation

$$\mathbf{a} = \frac{\mathbf{a}_{\text{RAW}} \cdot \text{scale}}{8192}, \quad (\text{B.1})$$

where  $\mathbf{a}_{\text{RAW}}$  is the 3-axis raw output vector from the device and  $\text{scale} \in \{2, 4, 8\}$  is the scaling used, units of  $\mathbf{a}$  is  $\frac{m}{s^2}$ .

### B.1.2 Gyroscope

The gyroscope used is an L3G4200D made by STMicroelectronics. It has a 16-bit resolution and the output is in 2's complement. It can work at sampling speeds up to 800Hz, 100 Hz was used for this thesis. Scaling can be set to  $\pm 250$ ,  $\pm 500$ , and  $\pm 2000$  degrees per second. In this thesis a scaling of  $\pm 2000$  was used. Converting from raw output data to angular velocity is done using the following equation

$$\omega = \frac{\omega_{\text{RAW}} \cdot \text{scale}}{32768} \cdot \frac{\pi}{180}, \quad (\text{B.2})$$

where  $\omega_{\text{RAW}}$  is the 3-axis raw output vector from the device and  $\text{scale} \in \{250, 500, 2000\}$  is the scaling used, units of  $\omega$  is rad/s.

### B.1.3 Magnetometer

The magnetometer used is an HMC5883 made by Honeywell. It has a resolution of 12-bits and the output is in 2's complement. It can work at sampling speeds up to 75 Hz, the maximum sampling speed was used for this thesis. Scaling can be set to eight different levels from  $\pm 0.9$  Gauss up to  $\pm 7.9$  Gauss, in this thesis  $\pm 1.2$  Gauss was used, which gives a convenient scaling of 1024 units per Gauss. Converting from raw output data to acceleration is done using the following equation

$$\mathbf{m} = \frac{\mathbf{m}_{\text{RAW}}}{1024}, \quad (\text{B.3})$$

where  $\mathbf{m}_{\text{RAW}}$  is the 3-axis raw output vector from the device and the units of  $\mathbf{m}$  is Gauss.

### B.1.4 Pressure sensor

The magnetometer used is an BMP085 made by Bosch Sensortec. It has a resolution of 16-bits and the output is in 2's complement. It can work at sampling speeds up to 128 Hz, for this thesis the a sampling rate of 10 Hz was used. The measuring range is 300hPa to 1100hPa. Converting from raw output data to pressure is long and tedious, it is there fore excluded from this thesis but can be found in the data sheet for the BMP085 sensor.

# Appendix C

## Time of flight

The time of flight for radio waves are directly proportional to the distance between two radio devices in strict LOS conditions. When Non line of sight (NLOS) conditions are present the distance will increase due to that radio waves propagate slower through denser materials. Radio waves will also suffer from multipath conditions which can severely disrupt measurements. This appendix illustrates the time of flight experiments that we have performed under different indoor conditions.

### C.1 Calibration

The delays  $t_{h,o}$  and  $t_{h,r}$  will depend on which node is acting as origin and which is acting as remote. Due to natural variations in the manufacturing process, the delays will also be different between different units. Since the measurements in this application are always performed between an endpoint and a coordinator, the sum between the two delays is assumed to be constant for all pairs of nodes. The software interface comes with a parameter used to compensate for the extra delay in the circuit. This parameter is found by performing multiple measurements and finding the average bias in the measurements

$$\Delta \hat{d}_c = 2c (\tilde{t}_{h,o} + \tilde{t}_{h,r}) = \frac{1}{N} \sum_{i=1}^N (c(t_{\text{tof},i} - t_{s,r}) - d_{\text{tof},i}), \quad (\text{C.1})$$

where  $d_{\text{tof},i}$  is the ground truth for the distance at which the measurement was performed and  $c$  is the speed of light. For a complete description of the variables used in this appendix, see Table 2.1

### C.2 Initiation

A time of flight measurement is initiated by an endpoint by selecting which coordinator should be the remote node. The turnaround time,  $t_{s,r}$ , will vary de-

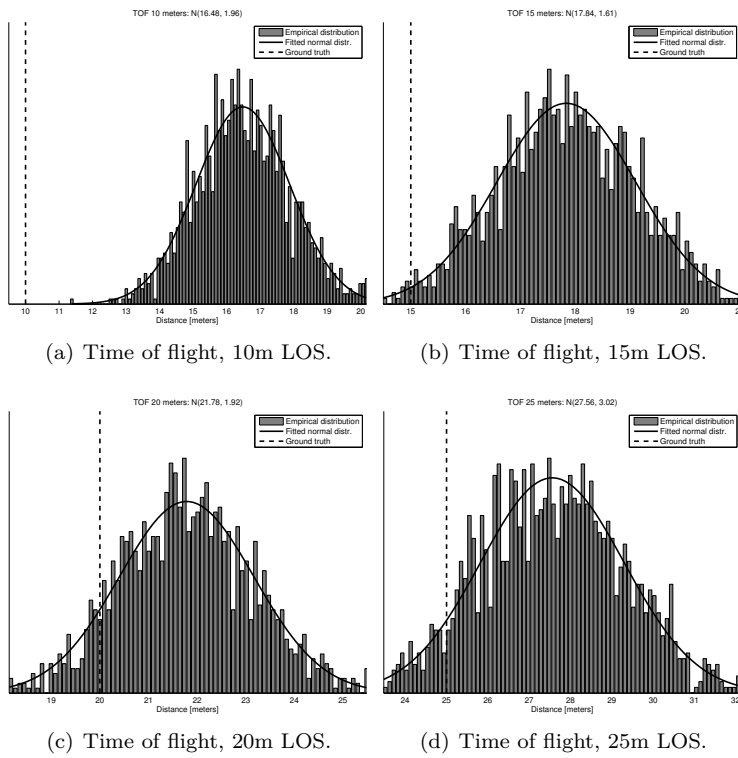
pending on frequency offset of the remote node. This offset is unknown to the transmitting node and will cause a constant offset in the measurement. To reduce this bias, time of flight measurements can be performed in both directions, called forward and reverse direction. This will average out the discrepancies between the nodes and in most cases reduce this bias. When selecting the forward direction the endpoint will act as the origin node and the coordinator as the remote node and vice versa for the reverse direction.

### C.3 Performance

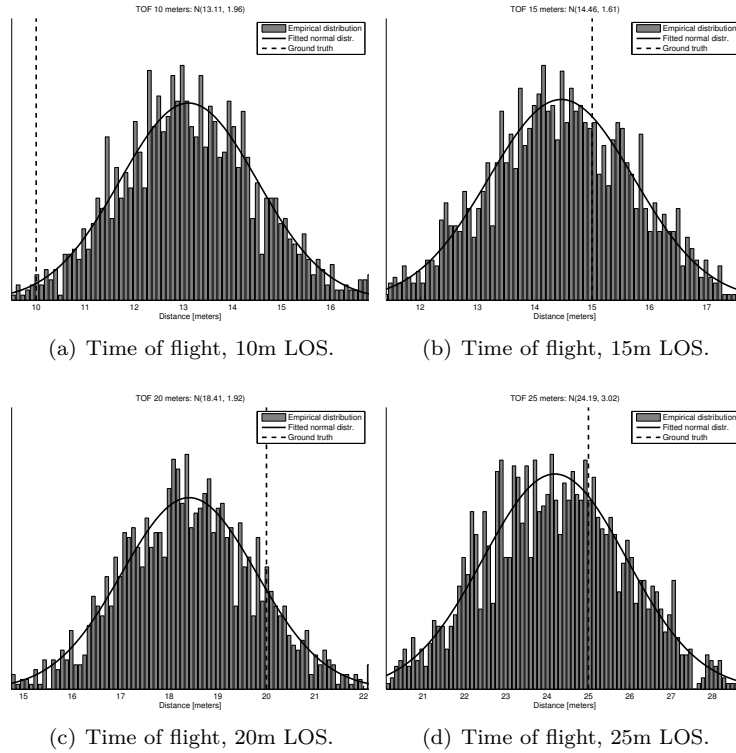
The achievable performance is studied during static data collection configurations. LOS and NLOS conditions are studied separately in order to study the difference in performance them in between. On shorter ranges, the deviations in calibration parameters and frequency have larger impact on the measurements, since the greater part of the measurement consists of delays in the hardware. In Figure C.2 we note that the density is normal distributed but has a non-zero bias compared to the ground truth.  $\Delta\hat{d}_c$  is found from a large set of data collected in LOS at several different distances. The remaining non-zero biases may have their explanation in the small impact of  $\delta$  even in LOS conditions. The short wavelength will potentially cause rapid changes in the multipath fading for small movements in position or attitude of the endpoint.

The same figures as in Figure C.2 with  $\Delta\hat{d}_c = 0$  are shown in Figure C.1 for reference. It can be seen that the absent calibration parameter results in overestimated distances in LOS conditions. It is from those measurements  $\Delta\hat{d}_c$  are found using (C.1).

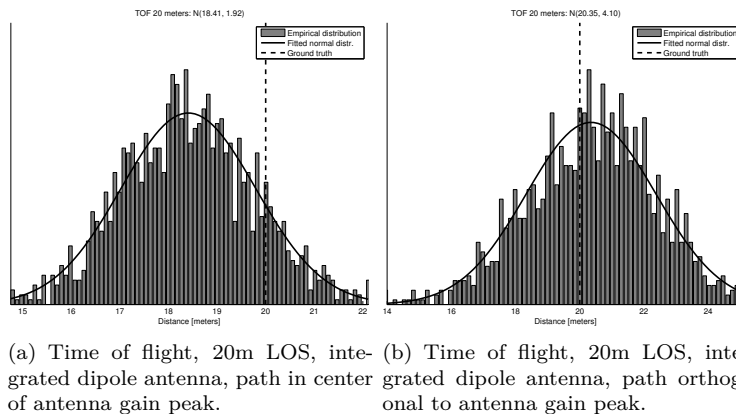
Figure C.3 shows that different attitudes of the endpoint will have a large impact on the quality of the measurements. This is probably related to the donut shaped radiation pattern of the used monopole antennas. If the endpoint's and the coordinator's antennas are placed in parallel to each other the most accurate results are achieved. If the endpoint's antenna is instead pointing towards the antenna of the coordinator, multipath effects occur and result in an overestimated distance. For NLOS conditions shown in Figure C.4 it can be seen how the multipath effects severely disrupts measurements.



**Figure C.1.** Time of flight densities at several distances with the endpoint having the same attitude relative to the coordinator.  $\Delta \hat{d}_c = 0$ . The ground truth and fitted normal densities are included for reference.

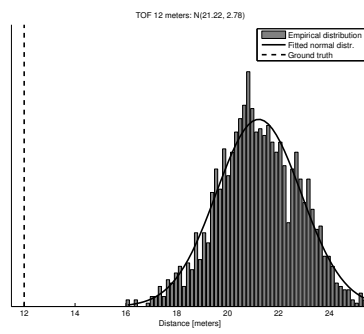


**Figure C.2.** Calibrated time of flight densities at several distances with the endpoint having the same attitude relative to the coordinator. The ground truth and fitted normal densities are included for reference.



**Figure C.3.** Calibrated time of flight densities at a fixed distance with the endpoint different attitudes relative to the coordinator. The ground truth and fitted normal densities are included for reference.





**Figure C.4.** Time of flight measurements for NLOS conditions. A severe disruption from the ground truth can be seen.



# Appendix D

## RSSI

This appendix describes the performed RSSI experiments and illustrates the achieved performance.

### D.1 Identification of measurement model

The manufacturer of the radio chip used in this thesis proposes the RSSI to range conversion as

$$d = 0.02 \cdot 10^{\left(\frac{108 - \text{RSSI}}{20}\right)}. \quad (\text{D.1})$$

Solving the log-distance path loss model (2.2) for  $d$  gives

$$d = d_0 10^{\left(\frac{P_{0,\text{dBm}} - P_{r,\text{dBm}}}{10n}\right)}. \quad (\text{D.2})$$

Identification in (D.1) and (D.2) gives

$$d_0 = 0.02, \quad (\text{D.3a})$$

$$P_{0,\text{dBm}} = 108, \quad (\text{D.3b})$$

$$P_{r,\text{dBm}} = \text{RSSI}, \quad (\text{D.3c})$$

$$n = 2, \quad (\text{D.3d})$$

which gives both the model used and the recommended parameters for the RSSI to range conversion.

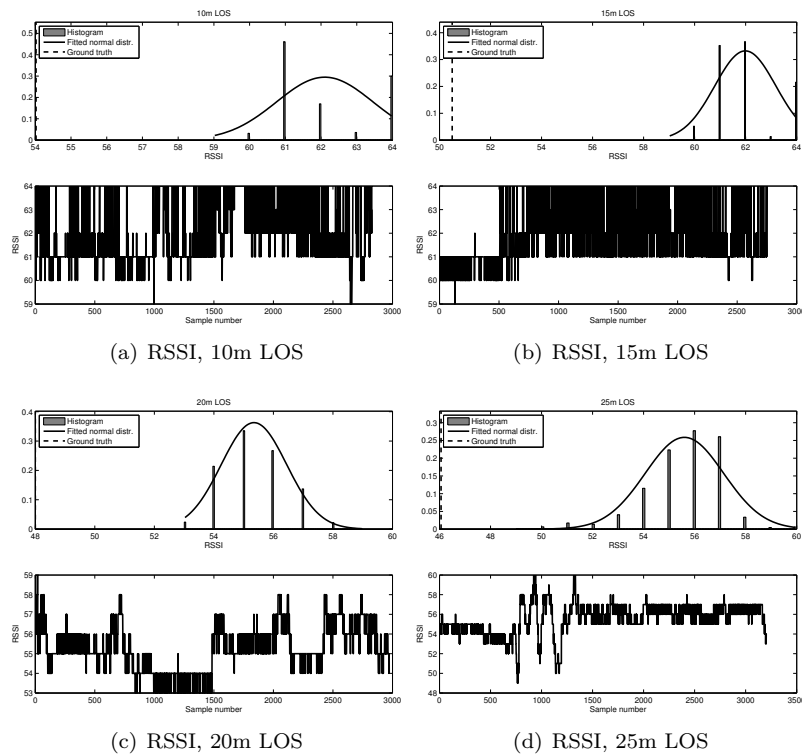
### D.2 Performance

Our experiments, depicted in Figure D.1, show that the model (2.3) is invalid for indoor environments. The empirical densities clearly have a non-zero bias which is different at different distances. The bias indicates that  $P_{0,\text{dBm}}$  and  $n$

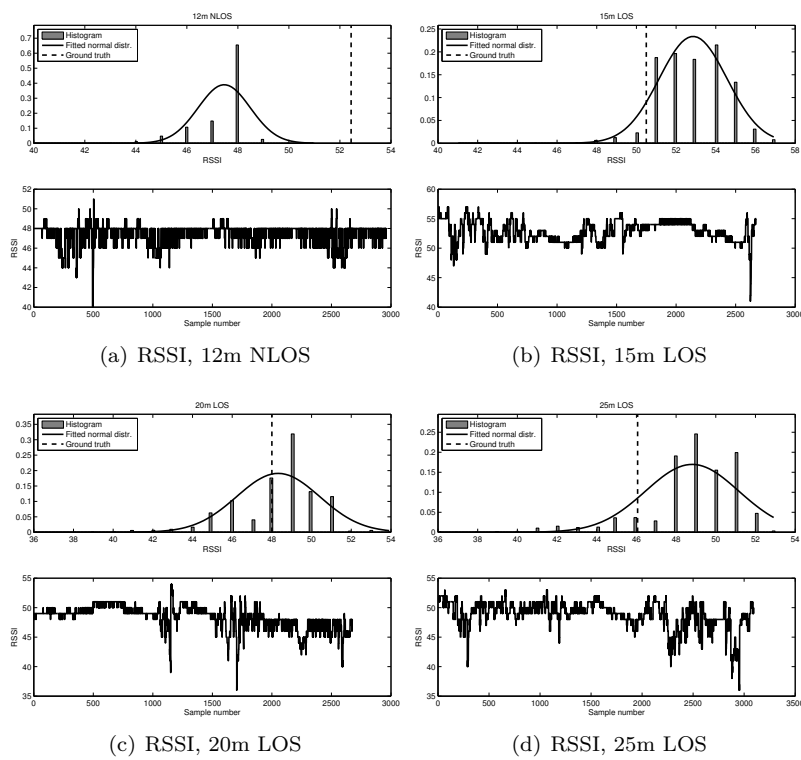
are chosen badly. While those could be calibrated for improved accuracy, we have chosen not to because of the poor performance in NLOS and multipath conditions as shown in Figure D.2.

The measured RSSI values shown in Figure D.3 are converted into distances using (D.1). The deviation from the ground truth is large and unpredictable even for LOS conditions from 20 meters and above. For 10 meters, the measurements are accurate in LOS conditions, but since the NLOS conditions at approximately the same distance are far from correct the model cannot be trusted even at short distances.

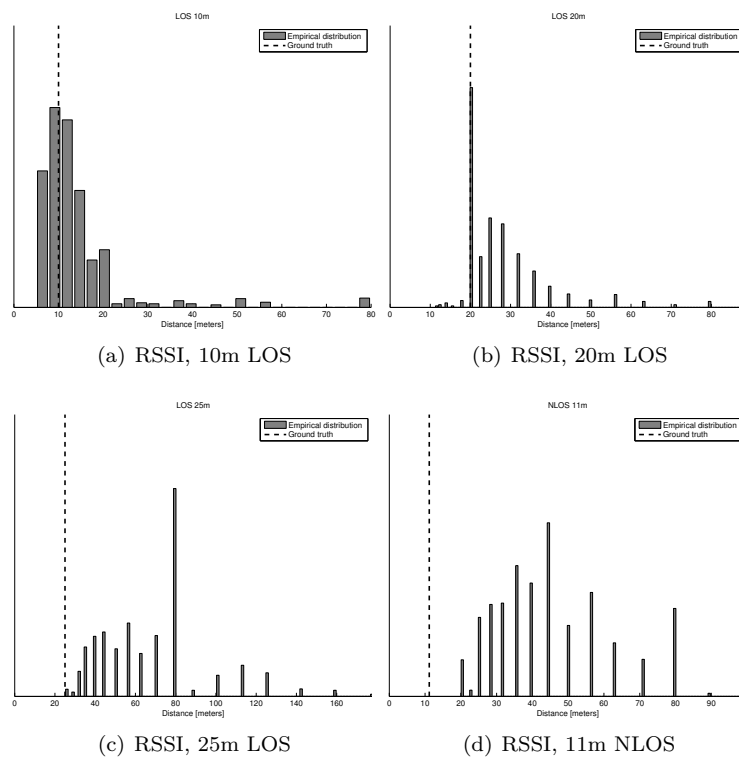
Several other models for the received signal strength at a certain communication distance exist which might be better suited for indoor environments where NLOS conditions are common. The multi-wall model [45] is commonly used and tends to give better results than the log-distance path loss model [46, 39, 40]. However, the multi-wall model puts higher computational demands on the algorithm since it incorporates a map over the building. Experiments performed in [39, 40] show that the multi-wall model gives an improvement compared to the log-distance path loss model but still is too rough for giving a decent range.



**Figure D.1.** Measured RSSI values at several distances in LOS conditions. Fitted normal distributions, ground truth and the RSSI values over time are shown.



**Figure D.2.** Measured RSSI values at several distances in both NLOS and LOS conditions. Fitted normal distributions, ground truth and the RSSI values over time are shown. The LOS figures are measured with the endpoint's antenna pointing towards the coordinator, which induces multipath effects.



**Figure D.3.** Empirical range densities calculated from the RSSI values. The sparse resolution is caused by a low resolution in the samples.

## Appendix E

# Step length estimation

Estimating the step length correctly would increase the accuracy of the position. The step length can vary due to two factors. First a fixed based step length which for the biggest part depends on the length of the person wearing the Beebadge, a longer person will tend to take longer steps. Secondly the walking pace affects the step length [28].

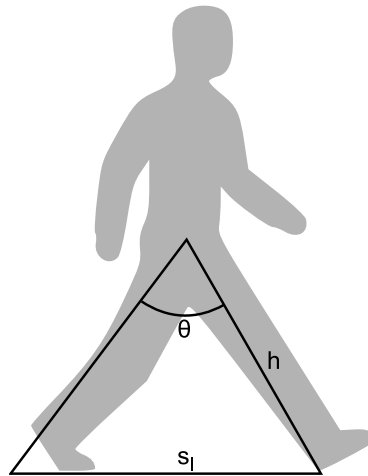
### E.1 Step length from step rate

The step length is proportional to the step rate, which at first might sound unintuitive, but faster steps tend to be longer and vice versa for slower steps. This can easily be tested by the reader, taking fast and short steps or slow and long steps will feel unnatural. Figure E.1 shows the components of a step, when taking step with a high rate  $\theta$  will be greater aswell as the rate of which it changes. This will lead to a longer step length.  $h$  will be directly proportional to the length of the person assuming normal proportions of the body.

Literature describes a clear relationship between the step length and the step rate [28]. This relationship should not be completely clear for users with different lengths and therefore was not included in this thesis.

### E.2 Step length estimation in particle filter

Tests was conducted to introduce the step length as another state in the particle filter. The filter was initiated with particles with different step lengths and as the filter converges particle with incorrect step length should disappear during early resampling steps. This however made the filter very sensitive at an early stage, missed steps or drift could cause particles with the correct step length to be removed. This severely decreased the accuracy and in many situations even cause the filter to diverge all together.



**Figure E.1.** Components that lead up to the step length of human



# Bibliography

- [1] P. Hall, “A Bayesian approach to map-aided vehicle positioning,” Master’s thesis, Linköping University, Department of Electrical Engineering, 2001.
- [2] U. Forsell, P. Hall, S. Ahlqvist, and F. Gustafsson, “Map-aided positioning system,” tech. rep., NIRA Dynamics AB, Linköpings Universitet, 2002.
- [3] N. Svenzen, “Real time implementation of map aided positioning using a Bayesian approach,” Master’s thesis, Linköping University, Department of Electrical Engineering, 2002.
- [4] J. D. Hol, *Sensor Fusion and Calibration using Inertial Sensors, Vision, Ultra-Wideband and GPS*. PhD thesis, Linköping University, Division of Automatic Control, 2011.
- [5] A. Fink, H. Beikirch, M. Voss, and C. Schröder, “RSSI-based indoor positioning using diversity and inertial navigation,” in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pp. 1–7, sept. 2010.
- [6] J. D. Hol, F. Dijkstra, H. Luinge, and T. B. Schön, “Tightly coupled UW-B/IMU pose estimation,” in *ICUWB, IEEE International Conference on Ultra-Wideband*, pp. 688–692, sept. 2009.
- [7] J. D. Hol, T. B. Schön, and F. Gustafsson, “Ultra-wideband calibration for indoor positioning,” in *Ultra-Wideband (ICUWB), 2010 IEEE International Conference on*, vol. 2, pp. 1–4, sept. 2010.
- [8] M. Bedford and G. Kennedy, “Evaluation of Zigbee (IEEE 802.15.4) time-of-flight-based distance measurement for application in emergency underground navigation,” *Antennas and Propagation, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.
- [9] A. Falhi and Z. Guennoun, “Localization estimation in wireless sensor networks based on IEEE 802.15.4 standard,” in *Multimedia Computing and Systems (ICMCS), 2011 International Conference on*, pp. 1–6, april 2011.
- [10] A. Peker, O. Tosun, and T. Acarman, “Particle filter vehicle localization and map-matching using map topology,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 248–253, june 2011.

- 
- [11] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *Signal Processing, IEEE Transactions on*, vol. 50, pp. 425–437, feb 2002.
- [12] I. Miller, M. Campbell, and D. Huttenlocher, "Map-aided localization in sparse global positioning system environments using vision and particle filtering," *Journal of Field Robotics*, vol. 28, no. 5, pp. 619–643, 2011.
- [13] C. Ascher, C. Kessler, M. Wanklerl, and G. Trommer, "Dual IMU indoor navigation with particle filter based map-matching on a smartphone," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pp. 1–5, sept. 2010.
- [14] J. Säll and J. Merkel, "Indoor navigation using accelerometer and magnetometer," Master's thesis, Linköping University, Department of Electrical Engineering, 2011.
- [15] O. Woodman, *Pedestrian localisation for indoor environments*. PhD thesis, University of Cambridge, 2010.
- [16] IEEE, "IEEE standard for local and metropolitan area networks—part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std 802.15.4-2003*, 2003.
- [17] H. Maheshwari and A. Kemp, "On the enhanced ranging performance for IEEE 802.15.4 compliant WSN devices," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pp. 1–5, feb. 2011.
- [18] J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann, "Weighted centroid localization in Zigbee-based sensor networks," in *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pp. 1–6, oct. 2007.
- [19] T. S. Rappaport, *Wireless Communications: Principles and Practice (2nd Edition)*. Prentice Hall, 2002.
- [20] S. Mazuelas, A. Bahillo, R. Lorenzo, P. Fernandez, F. Lago, E. Garcia, J. Blas, and E. Abril, "Robust indoor positioning provided by real-time rssi values in unmodified wlan networks," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, pp. 821–831, oct. 2009.
- [21] J. Prieto, S. Mazuelas, A. Bahillo, P. Fernandez, R. Lorenzo, and E. Abril, "Adaptive data fusion for wireless localization in harsh environments," *Signal Processing, IEEE Transactions on*, vol. 60, pp. 1585–1596, april 2012.
- [22] M. Luber, J. Stork, G. Tipaldi, and K. Arras, "People tracking with human motion predictions from social forces," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 464–469, may 2010.

- 
- [23] S. Pellegrini, A. Ess, M. Tanaskovic, and L. Van Gool, "Wrong turn - no dead end: A stochastic pedestrian motion model," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 15–22, june 2010.
- [24] F. Gustafsson, *Statistical Sensor Fusion*. Studentlitteratur AB, 2010.
- [25] G. L. Smith, S. F. Schmidt, and L. A. McGee, "Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle," Tech. Rep. TR R-135, NASA, 1962.
- [26] T. B. Schön, "Solving nonlinear state estimation problems using particle filters - an engineering perspective," Tech. Rep. LiTH-ISY-R-2953, Linköping University, Department of Electrical Engineering, 2010.
- [27] T. Schon, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *Signal Processing, IEEE Transactions on*, vol. 53, pp. 2279–2289, july 2005.
- [28] H. Leppäkoski, J. Collin, and J. Takala, "Pedestrian navigation based on inertial sensors, indoor map, and WLAN signals," in *Acoustics, Speech, and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012.
- [29] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *International Journal of Robotics Research*, 2003.
- [30] F. Pei, P. Cui, and Y. Chen, "Adaptive mcmc particle filter for nonlinear and non-gaussian state estimation," in *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, p. 494, june 2008.
- [31] S. Madgwick, A. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pp. 1–7, 29 2011-july 1 2011.
- [32] G. Box and M. E. Müller, "A note on the generation of random normal deviates," *Ann. Math. Statist.*, vol. 29, no. 2, pp. 610–611, 1958.
- [33] G. Marsaglia and T. A. Bray, "A convenient method for generating normal variables," *SIAM Review*, vol. 6, no. 3, pp. 260–264, 1964.
- [34] G. Marsaglia and W. W. Tsang, "The ziggurat method for generating random variables," *Journal of Statistical Software*, vol. 5, pp. 1–7, 10 2000.
- [35] G. Kitagawa, "Monte carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, 1996.

- [36] J. D. Hol, T. B. Schön, and F. Gustafsson, “On resampling algorithms for particle filters,” tech. rep., Linköping University, Department of Electrical Engineering, 2006.
- [37] G. Hendeby, J. D. Hol, R. Karlsson, and F. Gustafsson, “Graphics processing unit implementation of the particle filter,” Tech. Rep. LiTH-ISY-R-2749, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Oct. 2006.
- [38] K. Par and O. Tosun, “Parallelization of particle filter based localization and map matching algorithms on multicore/manycore architectures,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 820–826, june 2011.
- [39] Widyawan, M. Klepal, and D. Pesch, “Influence of predicted and measured fingerprint on the accuracy of RSSI-based indoor location systems,” in *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, pp. 145–151, march 2007.
- [40] S. Hossain, S. H. Ariffin, N. Fisal, C. K. Neng, N. A. Hassan, and L. Latiff, “Accuracy enhancement of fingerprint indoor positioning system,” in *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*, pp. 600–605, feb. 2012.
- [41] R. Smith, M. Self, and P. Cheeseman, “A stochastic map for uncertain spatial relationships,” in *Fourth International Symposium of Robotics Research*, p. 467–474, 1987.
- [42] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *Robotics Automation Magazine, IEEE*, vol. 13, pp. 99–110, june 2006.
- [43] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): part ii,” *Robotics Automation Magazine, IEEE*, vol. 13, pp. 108–117, sept. 2006.
- [44] M. Schafer, C. Knapp, and S. Chakraborty, “Automatic generation of topological indoor maps for real-time map-based localization and tracking,” in *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pp. 1–8, sept. 2011.
- [45] J. Lähteenmäki, “Indoor propagation models,” in *COST action 231, Digital mobile radio towards future generation systems final report*, pp. 175–189, 1999.
- [46] A. Borrelli, C. Monti, M. Vari, and F. Mazzenga, “Channel models for IEEE 802.11b indoor system design,” in *Communications, 2004 IEEE International Conference on*, vol. 6, pp. 3701–3705 Vol.6, june 2004.