Indoor Localization through Pedestrian Dead Reckoning and Activity Recognition

Bart van Ingen

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Indoor Localization through Pedestrian Dead Reckoning and Activity Recognition

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Bart van Ingen

October 31, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

This is an abstract.

Table of Contents

List of Figures

List of Tables

Preface

According to WikipediA, a preface (pronounced "*preffus*") is an introduction to a book written by the author of the book. In this preface I can discuss the interesting story of how this thesis came into being.

This is document is a part of my Master of Science graduation thesis. The idea of doing my thesis on this subject came after a discussion with my good friends Tweedledum and Tweedledee...

Acknowledgements

I would like to thank my supervisor prof.dr.ir. M.Y. First Reader for his assistance during the writing of this thesis...

By the way, it might make sense to combine the Preface and the Acknowledgements. This is just a matter of taste, of course.

Delft, University of Technology
October 31, 2020

Bart van Ingen

"In the future, airplanes will be flown by a dog and a pilot. And the dog's job will be to make sure that if the pilot tries to touch any of the buttons, the dog bites him."

— *Scott Adams*

Chapter 1

Introduction

1-1 Research Questions

Can indoor localization be achieved with realistic phone placement? Can activity recognition improve the above derived method

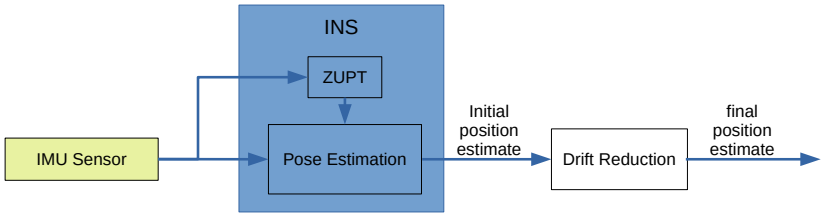
1-2 Thesis Contributions

Chapter 2

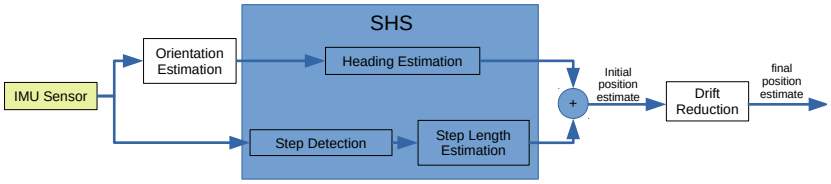
Related Work

2-1 Pedestrian Dead Reckoning

Inertial Pedestrian Dead Reckoning (PDR) is a positioning technique whose advantage lies in that it only requires MEMS IMU sensors, which can easily be carried by a pedestrian. Within PDR there are two methods, namely Inertial Navigation Systems (INSs) and Step and Heading Systems (SHSs).
The INS mechanism integrates the IMU signals and is the implementation used most often within research [7]. With INS the position error grows cubically with time, due to the signal to noise ratio inherent to MEMS IMU sensors[12]. In order to compensate this error it is preferable that the IMU sensor is mounted on the foot, as will be explained in Section 2-1-1. This placement requirement may limit the usability.
SHS is based on integrating the displacement vectors associated with each step taken during pedestrian locomotion. One advantage of the SHS implementation is that the position error is also proportional to the number of steps taken, but without the preference of using foot-based sensors, allowing for other sensor placements.
An overview of the different components required by the two PDR systems can be found in Figure 2-1, which will be explained in more detail later. Many PDR methods accumulate error with time and are therefore not ideal for long-term pedestrian tracking [11]. Drift reduction methods can be used to try and compensate this error in order to allow for long term tracking [21], as also shown in the diagram.



(a) Inertial Navigation System (INS) pedestrian dead reckoning



(b) Step and Heading System (SHS) pedestrian dead reckoning

Figure 2-1: SHS and INS components

2-1-1 Inertial Navigation System

An INS estimates pose, consisting of orientation and position, using sensor fusion algorithms. These methods are directly applied to the signals generated by MEMS IMU sensors [36]. Examples of sensor fusion algorithms are the Extended Kalman Filter (EKF) and Complementary Filter (CF) [17]. Sensor fusion algorithms can estimate position and orientation by integrating acceleration twice and integrating angular velocity once, respectively. This integration, in combination with the effects of noise and bias found in MEMS IMU, causes estimation errors to grow cubically with time for INS [12].

A technique frequently used to compensate the built up error in INS is zero velocity update (ZUPT) and zero angular update (ZARU) [12]. It utilizes the ability to detect time periods in which the sensor is stationary during locomotion. Once detected, ZUPT uses the assumption that speed and angular velocity are zero at that time moment [12, 36]. this a form of pseudo-measurement. This is because the stationary phase is detected, with zero velocity being implied by the assumption, which is not an actual measurement. Comparing the assumption with the output of the sensor fusion algorithm, an error can be calculated and used to compensate the sensor fusion estimate. This process is generally known as a measurement update and would occur every time a stationary period is detected.

Detecting brief stationary time periods in pedestrian IMU data is done easiest when the sensor is placed on the foot. This is because a stationary period is much more pronounced in accelerometer data when the sensor is placed on the foot [36, 37]. When walking a foot periodically returns to a stationary state and stays there for a brief period of time approximately 0.1 to 0.3 seconds [25]. This has lead to most INS research being foot-sensor based [6, 36]. The highest accuracy in PDR has been reached with INS in combination with ZUPT

[11]. Solin et al. [29] overcome this preference for a foot based system with a combination of several pseudo measurements, loop closure and position fixes, opening up new possibilities for implementing INS in realistic smartphone use cases.
Since the majority of research uses foot mounted systems to implement ZUPT for INS [36], it suggest that SHS could be more appropriate for other sensor placements. This is therefore also relevant for PDR in realistic smartphone use cases, since these device are generally not carried in one specific way. Considering this, a focus will be put on implementing a SHS system with the potential to handle different carrying modes.

2-1-2 Step and Heading System

SHS is a form of dead reckoning that identifies steps and their length, and the heading in which the pedestrian is moving. This position estimation can be represented by [20]

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \end{pmatrix} + l_t \begin{pmatrix} \cos(\theta_t) \\ \sin(\theta_t) \end{pmatrix} \tag{2-1}$$

where x_t and y_t represent the position in the x-axis and y-axis at time t, respectively. l_t stands for the step length, while θ_t is the heading of the pedestrian at time t. There are three components need to determine the above parameters. These are walk and step detection, step length estimation, and step heading estimation.

Walk and Step Detection

Walk detection is determining from sensor data whether a walking activity is being performed. Step detection is deciding when during the walking activity a step is taken.
In [1], a variety of algorithms for both walk detection and step detection on unconstrained smartphones are tested and compared. The paper reviews techniques with different levels of complexity from a variety of different papers. This includes techniques such as machine learning, non-linear template matching, frequency domain analysis, and time domain thresholding, as shown in Figure 2-2. The reader is referred to [1] for a detailed explanation of each solution.

In order to compare the different techniques, Brajdic and Harle [1] collected a large dataset from 27 test subjects leading to 130 recordings was made. Each subject held the smartphone in six different carrying modes: in hand in front of user (idle and with device interaction), in a front pocket, in a back pocket and in a handbag or backpack. A ground truth was generated by video recording each session and manually counting the amount of steps taken. Using this dataset a comparison is made between nine algorithms, shown in Figure 2-2 and summarized in Table 2-1.
The paper concludes that for the generated dataset, the best step counting results were obtained using the windowed peak detection, hidden Markov model and continuous wavelet transform. Each had a median error of about 1.3%.
Considering the relative simplicity of the technique, the authors recommend that the windowed peak detection is the most efficient algorithm. The best walk detection algorithms

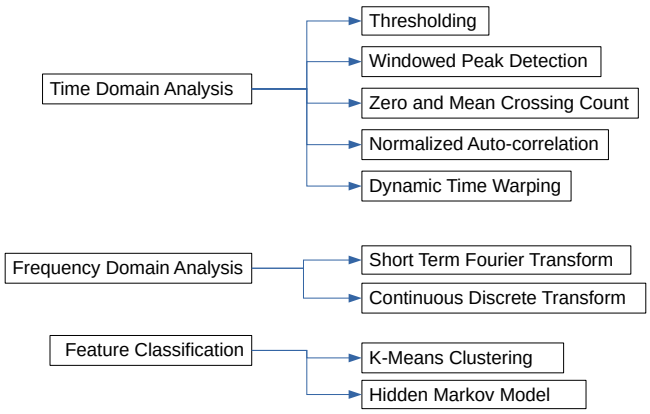


Figure 2-2

were thresholds on either standard deviation or signal energy, short-term Fourier transform, and normalized autocorrelation.
Salvi et al. [27] build upon the conclusions and recommendations of Brajdic and Harle [1], with the aim of further optimizing the windowed peak detection algorithm and its parameters. The algorithm is based on the approach of Palshikar et al. [22] and consists of 5 stages run in series. These are pre-processing, filtering, scoring, detection and post processing, as can be seen in the blue squares of **??** . To optimize the algorithm, different combinations of stage components were compared. A custom made, electronic ground truth device is worn by test subjects for easy comparison with the different combinations. A dataset of 36 recordings was made, where three researchers walking for two to three minutes with the phone held in six different carrying modes. This includes in hand, in a front pocket, in a back pocket, in an armband, in a shoulder purse, and in a neck pouch on a string. Each set contains around 2.5 minutes of accelerometer and ground truth data. An exhaustive grid search across the parameter space was performed to find the optimal set parameters for step detection. An overview of the optimal parameters can be found in Figure **??**. Using these parameters an average accuracy of $95\% \pm 4.5\%$ was reached for the different carrying modes.

Technique	Explanation	Advantages	Disadvantages
Threshold	Acceleration magnitude is monitored for the passing of certain values or sign changes, in the case of Zero Crossing method [4, 12]. Typically determines when foot is on the floor [12], but has also been applied to pitch angle of upper leg, where the sensor has been placed [5]. In some instances the thresholds are altered adaptively, for example through different motion mode detection or information gathered from the previous step [36].	• Often uses acceleration norm, making it robust to sensor orientation [4] • Simple to implement	• The optimal threshold can vary between users, surfaces and shoes [1]. • It is possible that certain motions, aside from the desired motion, can cross the threshold and generate false positives.
(Windowed) Peak Detection	Recognizes local maximum or minimum on acceleration magnitude caused by foot impact on the floor, often within a sliding window [30]. Generally combined with thresholding, which is one of the simplest combination used with SHS [4].		
Auto-correlation (Template Matching)	Finds correlation of a signal with a time shifted copy of itself. It leverages the strong cyclic nature of bipedal locomotion [12]. Since the lag for highest correlation is not known beforehand, an interval is often swept and correlation values compared.	• Periodicity of locomotion is often invariant of sensor placement [12].	• Can only detect periodic motion.
Dynamic Time Warping	Similar to autocorrelation, it measures the similarity between two waveforms from accelerometer data [4], These waveforms can both come from the data or one could be a stride template generated offline. A non linear mapping between the two methods is made, resulting in a DTW distance. A smaller distance indicates higher similarity [4].	• Minimizes the effects of shifting and distortion [28], detect similar shapes with different phases. • Better performance compared to autocorrelation [4].	• Computational overhead. [4, 12]
Short Term Fourier Transform	Transforms acceleration signal of successive windows of data into the frequency domain. For a spectral analysis, a subset containing a stride (two steps) is required to determine the frequency [12].	•	• Requires a stride to determine frequency • Windowing introduces resolution issues [12]
Wavelet Decomposition	Acceleration magnitude is split into low and high frequency components, from which the dominant frequency is assumed to be the walking frequency. Iterating this process on the resulting low frequency signal approximation, a smoother shaped dominant low-frequency signal is generated. The frequency of this signal corresponds to walking cadence [4].		
Hidden Markov Model	Uses gait cycles segmented into different states of a state machine, to determine if a step has been taken [25]. Thresholds can be used to induce a state change. If a full state machine cycle is achieved, a step is detected.	• Can be constructed by hand, through manual step signal analysis [25].	• May require training
K Nearest Neighbours	Uses labeled data containing features from successive time windows and compares a new time window with its features. It finds the labeled data whose features are most similar. This new set recieves then recieves its label.		• Requires training • May be person dependent

Table 2-1: Overview of different step detection methods

Step Length Estimation

In order to generate a displacement vector from step detection the step length must be estimated. Collins and Kuo [2] found that increasing step speed leads to larger step lengths, while step speed can have slow and spontaneous fluctuations depending on the motion mode. In addition, step size depends on the physical characteristics of the user and on their walk strategy, which can be different per individual [6]. These discrepancies indicate that using a simple average step length for every pedestrian could result in quick accumulation of error.

Diez et al. [6] categorizes step length estimation methods into integration based and model based methods.
Theoretically, the double integration of the IMU acceleration signal is the best approach to step size estimation, using the INS approaches outlined in chapter Section 2-1-1. This would give a direct measurement of displacement. It does not require any modeling, assumptions, or person specific calibration [6]. However, since it would be an INS approach, it suffers from the same drift problems and would require the same solutions. Therefore this approach benefits from having the sensor to be located on the foot.
Analytical models can be made of human mobility based on geometrical relationships of body composition, angles and displacement of body parts. One of the largest disadvantages of a model based approach is that human proportions are not uniform, requiring approximations and/or some form of calibration for the model to be accurate.

Vezocnik and Juric [33] compared different existing step length estimation algorithms. The review focuses on the methods applicable to smartphone use. This means methods that do not require training and do not require the sensor to be placed on the foot. This therefore excludes machine learning and INS systems. The models used are either based on an inverted pendulum model or relate predictors to step length. Example of step length predictors are step frequency and acceleration range within a step. The robustness of the methods were tested by having the smartphone in different carrying modes. This includes front pants pocket, in-hand while reading, in-hand while swinging carrying arm; and in a bag. Furthermore, a comparison was made between the use of global variables, which are variables constant for every user, and personalized variables, where they are tuned per user. The comparison concludes that for global variables the method from [31] was best, defined as

$$\text{step size} = K \cdot h \cdot \sqrt{F}. \tag{2-2}$$

Here K is a tunable parameter, h is the height of the user and F is the step frequency. This method reported an average error of 4.59 % for personalized variables and 6.96 % for global ones. For personally tuned variables the method of [34] was best. This is a n inverted pendulum model in which the human center of mass is used, located approximately at the pelvis. The center of mass rotates as an inverted pendulum when taking a step. This is followed by a forward horizontal displacement when both feet are on the ground [6]. The model is defined as

$$\text{step size} = K \sqrt[4]{A_{\max} - A_{\min}}. \tag{2-3}$$

Here $A_{\max}$ is the largest measured acceleration measured within a step interval, while $A_{\min}$

is the smallest. K is a calibration variable [6, 34]. The model had an average error of 10.64 % for global variables and 3.60 % for personalized [33].

Step Heading Estimation

Step heading determines the direction of a detected step. It requires the orientation of the sensor in the navigation frame and determining in what direction the sensor is moving in the sensor frame. This provides an estimate in which direction the sensor is moving in the navigation frame. Step heading estimation is currently the component within SHS whose performance is the most limiting for positioning purposes [3, 7, 23]. Even though the phone orientation may be known accurately, the direction in which the user is moving is not instantly clear. There are two approaches to determining the heading. The first is knowing beforehand what the orientation of the phone is with respect to heading. This would constrain the carrying mode that can be used. With the use of motion classification, this method could be expanded, where different carrying modes can be sensed,which changes the heading accordingly. Heading per carrying mode would need to be derivedbeforehand, through the training of the necessary model.

2-2 Pedestrian Dead Reckoning using Activity Recognition

Research has been done previously on using activity recognition in order to improve indoor localization. Hardegger et al. [11] created ActionSLAM which uses Simultaneous Localization And Mapping (SLAM) in combination with body mounted error to determine indoor location. It uses a foot mounted sensor to estimate how the user is moving in combination with location-related actions to compensate

Chapter 3

Method

3-1 Step Detection

The method used for step detection is similar to the one presented by [27], introduced in Section 2-1-2. It differs from the referenced paper in that the data is being handled offline, not having to wait for data buffers to fill. It also differs in that it adds the walk detection method from [1]. The values used for the different parameters are based on those found by [27]. The implemented method consists of five steps:

1. **Pre-processing**
   The IMU used in most smartphones provide acceleration information over three orthogonal axes. For the step detection algorithm the magnitude of the combined signal from the three orthogonal axes is used. Since MEMS IMUs are subject to potential noise from different sources the signal is put through a lowpass filter with a cutoff frequency of 5 hertz.

2. **Walk detection**
   Since a person is not walking continuously, the regions in which it does occur need to be indicated. [1] determined that thresholding on the standard deviation of the norm the accelerometer signal was sufficient. This is done using the parameters found by the paper, which is determining the standard deviation over a moving frame of 0.8 seconds, with all standard deviation magnitude levels above $0.6 m/s^2$ being considered walking.

3. **Filtering**
   A gaussian window is applied with a window size of 13 frames and a standard deviation of 0.35.

4. **Scoring**
   With the scoring stage the "peakiness" of a given sample is evaluated. The result of this stage should increase the magnitude of any peaks. This makes it easier for the

subsequent peak detection. The scoring used is that of mean difference

$$p_i = \frac{\sum_{k=-N,k\neq i}^{N} (x_i - x_{i+k})}{2N} \tag{3-1}$$

where p is the score given to the sample, i is the index of the sample, N is window size, and x is the sample value.

5. **Step detection**
   Here outliers are statistically detected. The algorithm processes the signal by calculating a running mean and standard deviation. These two measures determine whether a sample is an outlier or not. If the difference between sample and mean is over 1.2 times the standard deviation, then it is marked as a potential step.

6. **Post-processing**
   The final stage of step detection is determining local maxima of the outliers detected in the previous stage. Here local maxima are found that have a minimum separation duration of 0.2 seconds.

An example of this step detection method and its different stages can be found in Figure 3-1.

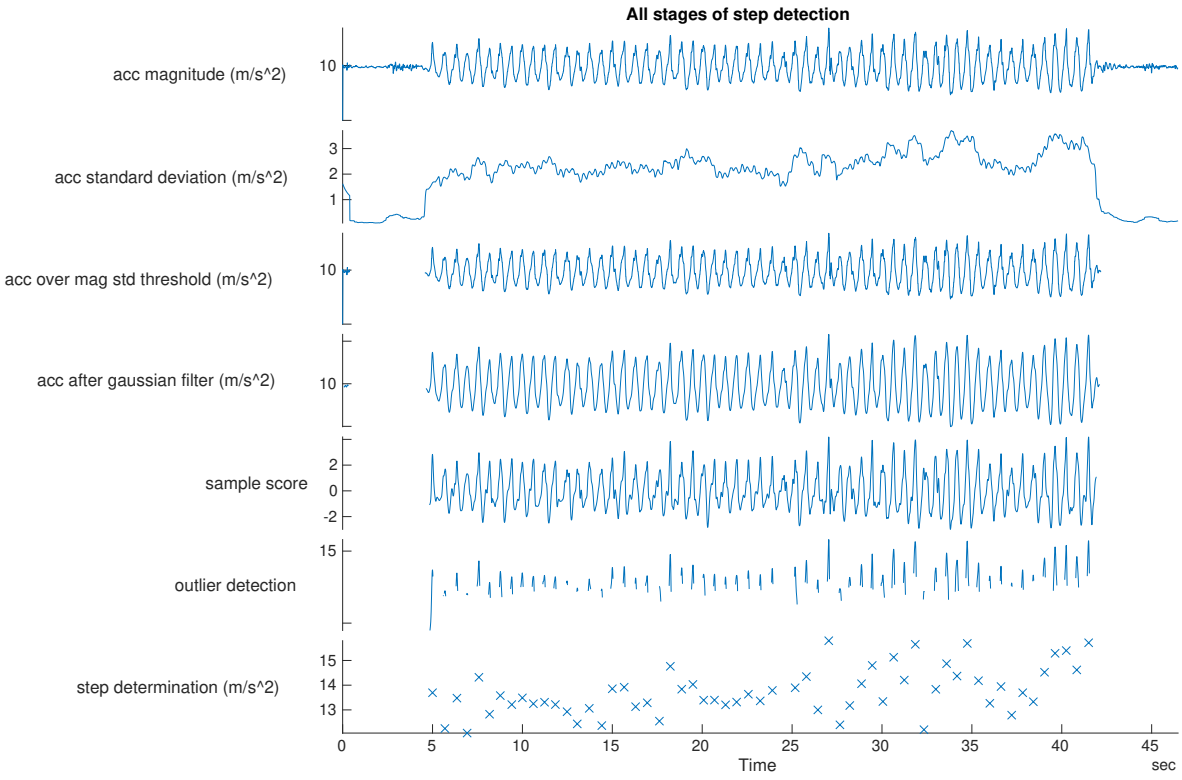


Figure 3-1: All stages of step detection from recorded accelerometer data, with the x's in the last graph indicating the steps taken

3-2 Step Length Estimation

3-3 Orientation Estimation

3-3-1 Coordinate Frames

Within orientation estimation there are different coordinate frames to be considered. There are two frames that are of importance for localization: the body frame and the navigation frame.
The body frame (b) is the coordinate frame of the IMU, the origin of which is at the center of the triaxial accelerometers found in the device [17]. The navigation frame (n) is the geographical frame in which the user is moving. It is also the frame in which we want to determine the pose, consisting of orientation and position, of the body frame. For a localization application it is considered stationary [17]. A superscript on a vector is used to indicate in which coordinate frame it is expressed. For example, x^n is the vector x expressed in the navigation frame. A double superscript on any orientation mapping from one frame to another indicates from which frame to which frame the rotation is occurring. For example, R^{nb} is a rotation matrix that maps from the body frame to the navigation frame.

3-3-2 Orientation Parametrization

Rotation matrices $R \in \mathbb{R}^{3\times 3}$ have the following properties:

$$RR^\top = R^\top R = I_3, \quad \det R = 1. \tag{3-2}$$

These matrices can be used to express a vector x in frame v to frame u as

$$x^{\mathrm{u}} = R^{\mathrm{uv}} x^{\mathrm{v}}. \tag{3-3}$$

Transposing a rotation matrix represent a rotation back to it's original coordinate frame:

$$x^{\mathrm{v}} = \left(R^{\mathrm{uv}}\right)^\top x^{\mathrm{u}}, \tag{3-4a}$$
$$= R^{\mathrm{vu}} x^{\mathrm{u}}. \tag{3-4b}$$

A quarternion is a common parametrization of orientation frequently used by attitude estimation algorithms. This is because it is free of non-singularity in attitude representation [13], also known as wrapping. A unit quaternion can be described by

$$q = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^\top = \begin{pmatrix} q_0 \\ q_v \end{pmatrix}, \quad q \in \mathbb{R}^4, \quad \|q\|_2 = 1. \tag{3-5}$$

A rotation of vector x using quaternions between two frames, from v to u, is indicated as

$$\bar{x}^{\mathrm{u}} = q^{\mathrm{uv}} \odot \bar{x}^{\mathrm{v}} \odot q^{\mathrm{vu}}, \tag{3-6}$$

where $q^{\mathrm{vu}} = (q^{\mathrm{uv}})^{\mathrm{c}}$, with the latter representing the quaternion conjugate, defined by

$$q^{\mathrm{c}} = \begin{pmatrix} q_0 \\ -q_v \end{pmatrix}. \tag{3-7}$$

$\bar{x}^u$ represents the quaternion version of the vector $x^u \in \mathbb{R}^3$, as

$$\bar{x}^u = \begin{pmatrix} 0 \\ x^u \end{pmatrix}. \tag{3-8}$$

The $\odot$ operator describes quaternion multiplication, defined by:

$$p \odot q = \begin{pmatrix} p_0 q_0 - p_v \cdot q_v \\ p_0 q_v + q_0 p_v + p_v \times q_v \end{pmatrix} \tag{3-9}$$

3-3-3 Motion and Measurement Models

The signals generated by an IMU can be formatted in such a way that orientation can be deduced. For many sensor fusion algorithms this is generally done by defining a motion and measurement model, together forming a state space representation. This state space can be defined as [17]

$$q_{t+1}^{\mathrm{nb}} = q_t^{\mathrm{nb}} \odot \exp_{\mathrm{q}} \left(\frac{T}{2} \left(y_{\omega,t} + e_{\omega,t} \right) \right), \tag{3-10a}$$

$$y_{\mathrm{a},t} = -R_t^{\mathrm{bn}} g^{\mathrm{n}} + e_{\mathrm{a},t}, \tag{3-10b}$$

$$y_{\mathrm{m},t} = R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t}, \tag{3-10c}$$

$$e_{\omega,t} \sim \mathcal{N}\left(0, \sigma_\omega^2 \mathcal{I}_3\right), \quad e_{\mathrm{a},t} \sim \mathcal{N}\left(0, \sigma_{\mathrm{a}}^2 \mathcal{I}_3\right), \quad e_{\mathrm{m},t} \sim \mathcal{N}\left(0, \sigma_{\mathrm{m}}^2 \mathcal{I}_3\right). \tag{3-10d}$$

Here Equation (3-10a) is the motion model and Equations (3-10b) and (3-10c) are the measurement models.
For the motion model, q^{nb} represents unit quaternion from body (b) to navigation frame (n). T is the time period between two samples. $y_{\omega,t}$ is the gyroscope measurement. The $\odot$ operator describes quaternion multiplication, as in Equation (3-9). The $\exp_{\mathrm{q}}$ operator is defined as

$$\exp_{\mathrm{q}}(\eta) = \begin{pmatrix} \cos \|\eta\|_2 \\ \frac{\eta}{\|\eta\|_2} \sin \|\eta\|_2 \end{pmatrix}. \tag{3-11}$$

For the measurement model, $y_{\mathrm{a},t} \in \mathbb{R}^3$ is the accelerometer measurement and R_t^{nb} is the rotation matrix mapped from the orientation quaternion generated in (3-10a). $g^n \in \mathbb{R}^3$ is the gravity vector in the navigation frame. Similarly, $y_{\mathrm{m},t} \in \mathbb{R}^3$ is the accelerometer measurement and m^n is the magnetic field in the navigation frame.

This state space model is simplified using certain assumptions. For both motion and measurement models, the noise terms (e) is assumed to be normally distributed, independent and have the same noise levels for the three sensor axis of all three sensors, as indicated in Equation (3-10d). The zero mean in these noise definitions also indicate the assumption that the sensors have been calibrated properly and therefore do not contain any bias.
Another assumption is that the sensor does not travel over significant distances in comparison to the size of the earth [17]. Additionally the magnitude of the earth rotation and coriolis acceleration are discarded. Furthermore, it is assumed that the acceleration signal is dominated by the gravity vector, making external acceleration negligible. Additionally, the magnetic field is assumed to be constant. These different assumptions will be needed to taken into account when constructing orientation estimation while walking indoors.

3-3-4 Calibration

For proper orientation estimation . Moder et al. [19] indicates that it is beneficial for PDR systems to have the gyroscope and magnetometer bias calibrated, while the accelerometer bias and scale factor is optional. In addition it is advised to estimate the gyroscope bias frequently before testing.

Gyroscope Calibration

Gyroscope measurements from MEMS IMUs are generally offset by a bias (o_ω) resulting in

$$y_{\omega,t} = \omega_t + o_\omega + e_{\omega,t}, \tag{3-12}$$

where $e_{\omega,t}$ is assumed to have the same noise properties as in Equation (3-10d). The gyroscope bias is slowly time varying but for relatively short experiments can be assumed to be constant [16]. It can be measured by placing the gyroscope on a flat surface for some time and recording data from the sensor. Since the sensor is not moving, the biases are the means of the data over the three axis and the covariance is the square of the standard deviation of the data to the previously mentioned means.

Accelerometer and Magnetometer Calibration

In outdoor environments, mn is equal to the local earth magnetic field and is accurately known from geophysical studies, see e.g. [29]. In indoor environments, however, the local magnetic field can differ quite significantly from the local earth magnetic field. Because of that, we treat mn as an unknown constant.

For magnetometer and accelerometer calibration the method outlined by Kok and Schon [16] can be used. This method indicates that for a perfect calibration a magnetometer measures the local magnetic field, no matter the orientation. Any measurement will then lie on a sphere with a radius equal to the local magnetic field. The same applies for a perfect calibration with an accelerometer, replacing the local magnetic field with the gravity vector. Any sensor calibration should strive to achieve this sphere as best as possible.
MEMS sensor are imperfect sensors, with sensor specific errors that can differ per sensor.

These errors, including non orthogonality of sensor axis, the presence of zero bias, and difference in sensitivity on different axis [16] can be combined into a distortion matrix D and offset vector o. This results in the following uncalibrated measurement model,

$$y_{\mu,t} = D_\mu R_t^{\mathrm{bn}} \mu^{\mathrm{n}} + o_\mu + e_{\mu,t}, \tag{3-13}$$

where μ is a placeholder for either the acceleration vector or local magnetic field vector with the superscript indicating in what coordinate frame it is expressed. Each have their own respective distortion matrix and offset vector. $y_{\mu,t}$ represents the measurement associated with either gyroscope or magnetometer. Due to the distortion matrix and offset vector, the measurements lie on a translate ellipsoid instead of on a sphere as previously stated. Determining these parameters, the sensor errors can be compensated through

$$y_{\mu,t}^{cal} = D_\mu^{-1} \left(y_{\mu,t} - o_\mu \right) \tag{3-14}$$

Without loss of generality the desired sphere radius can be normalized and written as followed to expressed the sphere characteristic

$$\begin{aligned} \left\| \mu^{\mathrm{b}} \right\|_2^2 - 1 &= \left\| R_t^{\mathrm{bn}} \mu^{\mathrm{n}} \right\|_2^2 - 1 \\ &= \left\| D^{-1} \left(y_{\mu,t} - o - e_{\mu,t} \right) \right\|_2^2 - 1 = 0. \end{aligned} \tag{3-15}$$

Since real calibration measurements are still corrupted by noise, the above equality does not hold exactly. The ellipsoid fitting problem can be rewritten as

$$y_{\mu,t}^\top A y_{\mu,t} + b^\top y_{\mu,t} + c \approx 0, \tag{3-16}$$

where

$$A \triangleq D_\mu^{-\top} D_\mu^{-1}, \tag{3-17a}$$
$$b \triangleq -2 o_\mu^\top D_\mu^{-\top} D_\mu^{-1}, \tag{3-17b}$$
$$c \triangleq o_\mu^\top D_\mu^{-\top} D_\mu^{-1} o_\mu. \tag{3-17c}$$

Assuming that A is positive definite, this defines an ellipsoid with parameters A, b and c. This can be rewritten as a linear relation as

$$M\xi \approx 0 \tag{3-18}$$

with

$$M = \begin{pmatrix} y_{\mu,1} \otimes y_{\mu,1} & y_{\mu,1} & 1 \\ y_{\mu,2} \otimes y_{\mu,2} & y_{\mu,2} & 1 \\ \vdots & \vdots & \vdots \\ y_{\mu,N} \otimes y_{\mu,N} & y_{\mu,N} & 1 \end{pmatrix}, \quad \xi = \begin{pmatrix} \mathrm{vec}A \\ b \\ c \end{pmatrix}, \tag{3-19}$$

where $\otimes$ is the kronecker product and vec the vectorization operator. The ellipsoid fitting problem can be written as the following semidefinite program [16]

$$\begin{aligned} \min_{A,b,c} \quad & \tfrac{1}{2}\|M \begin{pmatrix} \operatorname{vec} A \\ b \\ c \end{pmatrix} \|_2^2 \\ \text{s.t.} \quad & \operatorname{Tr} A = 1, \quad A \in S_{++}^{3\times 3} \end{aligned}, \tag{3-20}$$

where $S_{++}^{3\times 3}$ is the set of 3×3 positive definite symmetric matrices. This is a convex optimization problem with a globally optimal solution that can be solved using software packages such as CVX [8] and YALMIP [18]. Initial estimates of the calibration matrix D_μ and offset vector o_μ can be obtained from the estimated $\widehat{A}, \widehat{b}, \widehat{c}$ defined as

$$\beta = \left(\frac{1}{4}\hat{b}^\top \widehat{A}^{-1}\widehat{b} - \widehat{c}\right)^{-1} \tag{3-21a}$$

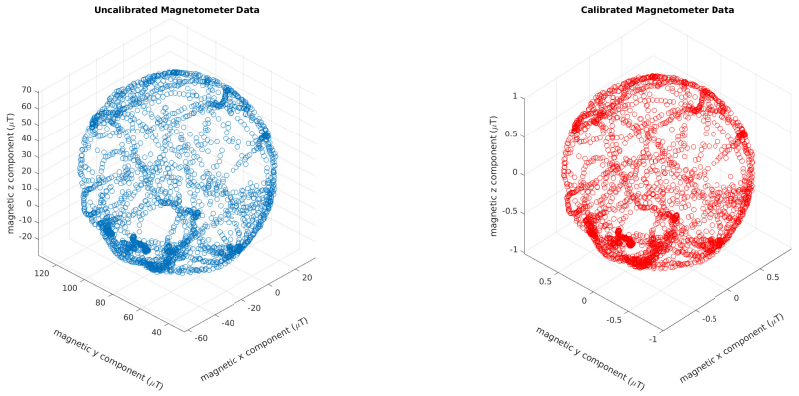
$$\widetilde{D}_{\mu 0}^\top \widetilde{D}_0 = \beta \widehat{A}^{-1} \tag{3-21b}$$

$$\widehat{o}_{\mu 0} = \frac{1}{2}\widehat{A}^{-1}\widehat{b} \tag{3-21c}$$

An example of this calibration can be found in Figure 3-2 where data was recorded by recording a smartphone in as many orientations as possible, in a stationary setting. It clearly shows how the uncalibrated data is mapped to a unit sphere. The distortion matrix and offset vector for this calibration are

$$\widehat{D}_{m0} = \begin{pmatrix} 0.0203 & -0.0001 & -0.0009 \\ 0 & 0.0191 & 0.0000 \\ 0 & 0 & 0.0199 \end{pmatrix}, \tag{3-22a}$$

$$\widehat{o}_{m0} = \begin{pmatrix} -13.3918 \\ 81.0407 \\ 20.9336 \end{pmatrix}. \tag{3-22b}$$



(a) Uncalibrated magnetometer data **(b)** Calibrated magnetometer data

Figure 3-2: Magnetometer calibration for smart MEMS IMU in one location

3-3-5 Extended Kalman Filter

The Kalman Filter (KF) uses a linear state space model in combination with measurements and its characteristics to make an unbiased minimum-variance estimator [32]. The KF assumes that the noise affecting the state space model is additive and that both process and measurement noise are Gaussian and zero mean. The Kalman Filter consists of three steps, with the last two steps performed recursively. The first is stating an initial estimate and the variance of the process and measurement noise. The second is the time update in which the state is propagated through a motion model, which may or may not have an external input, resulting in a one step ahead estimate. The third step is the measurement update in which the estimate generated by the time update is compared with actual measurements related to the state being estimated. Discrepancies between the two will lead to an error measurement. This error can then be used to correct the estimate. The Extended Kalman Filter (EKF) is an adaptation of the Kalman Filter that estimates for non linear models.

For orientation estimation the motion and measurement model presented in Equation (3-10) can be used. As stated in Section 3-3-3, this model has assumptions that need to be accommodated. This will have to be incorporated in the EKF. Additionally sensor data does not all arrive at the same time. This means that the measurement update will potentially need to only update with either magnetometer or accelerometer data.

Time Update

A time update occurs every time a gyroscope measurement (y_ω) is received. It calculates a one step ahead estimate of the state vector and its covariance matrix. The calculations are as followed:

$$\hat{q}^{\text{nb}}_{t|t-1} = \hat{q}^{\text{nb}}_{t-1|t-1} \odot \exp_{\text{q}} \left(\frac{T}{2} y_{\omega,t-1} \right) \tag{3-23a}$$

$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^\top + G_{t-1} Q G_{t-1}^\top \tag{3-23b}$$

with $Q = \Sigma_\omega$ and P the state covariance. Here

$$F_{t-1} = \left(\exp_q \left(\frac{T}{2} y_{\omega,t-1} \right) \right)^R, \tag{3-23c}$$

$$G_{t-1} = -\frac{T}{2} \left(\hat{q}^{\text{nb}}_{t-1|t-1} \right)^L \frac{\partial \exp_q (e_{\omega,t-1})}{\partial e_{\omega,t-1}}, \tag{3-23d}$$

where q^L and q^R are defined as

$$q^L = \begin{pmatrix} q_0 & -q_v^\top \\ q_v & q_0 \mathcal{I}_3 + [q_v \times] \end{pmatrix}, \tag{3-24a}$$

$$q^R = \begin{pmatrix} q_0 & q_v^\top \\ -q_v & q_0 \mathcal{I}_3 + [q_v \times] \end{pmatrix}. \tag{3-24b}$$

Measurement Update

A measurement update occurs every time either an accelerometer or magnetometer reading is received.
To ensure that the assumptions of the gravity vector being dominant in the accelerometer and the homogeneous magnetic field being dominant in the magnetometer, thresholds can be used. This is of importance with pedestrian dead reckoning, as a walking motion will induce additional external forces, affecting the acceleration measured by the IMU. Additionally, when localizing indoors, magnetic disturbance within the built environment will affect the magnetometer readings.
The measurement update for the EKF with quaternions as states is as followed:

$$\tilde{q}^{\mathrm{nb}}_{t|t} = \hat{q}^{\mathrm{nb}}_{t|t-1} + K_t \varepsilon_t, \tag{3-25a}$$
$$\tilde{P}_{t|t} = P_{t|t-1} - K_t S_t K_t^\top \tag{3-25b}$$

, with

$$\varepsilon_t = y_t - \hat{y}_{t|t-1}, \tag{3-25c}$$
$$S_t = H_t P_{t|t-1} H_t^\top + R, \tag{3-25d}$$
$$K_t = P_{t|t-1} H_t^\top S_t^{-1}. \tag{3-25e}$$

If an accelerometer measurement is received and whos magnitude falls in an interval around the magnitude , the variables to use in Equation (3-25) are

$$y_t = y_{\mathrm{a},t}, \tag{3-26a}$$
$$\hat{y}_{t|t-1} = -\hat{R}^{\mathrm{bn}}_{t|t-1} g^{\mathrm{n}}, \tag{3-26b}$$
$$H_t = -\left.\frac{\partial R^{\mathrm{bn}}_{t|t-1}}{\partial q^{\mathrm{nb}}_{t|t-1}}\right|_{q^{\mathrm{nb}}_{t|t-1}=\hat{q}^{\mathrm{nb}}_{t|t-1}} g^{\mathrm{n}}. \tag{3-26c}$$

Likewise if a magnetometer measurement arrives and meets the threshold requirement, the variables are defined as

$$y_t = y_{\mathrm{m},t}, \tag{3-27a}$$
$$\hat{y}_{t|t-1} = \hat{R}^{\mathrm{bn}}_{t|t-1} m^{\mathrm{n}}, \tag{3-27b}$$
$$H_t = \left.\frac{\partial R^{\mathrm{bn}}_{t|t-1}}{\partial q^{\mathrm{nb}}_{t|t-1}}\right|_{q^{\mathrm{nb}}_{t|t-1}=\hat{q}^{\mathrm{nb}}_{t|t-1}} m^{\mathrm{n}}. \tag{3-27c}$$

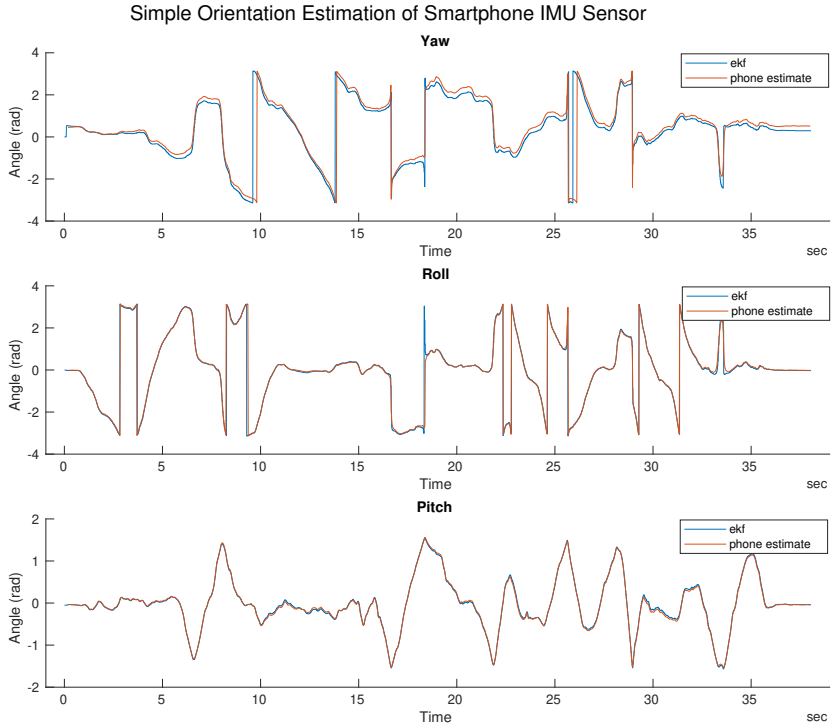


Figure 3-3: simple orientation estimation comparison between own EKF and android orientation estimation

Renormalization

Once a measurement update occurs the resulting quaternion is not necessarily a unit quaternion, which is a requirement for orientation parametrization. To facilitate this a renormalize the quaternion is required, using

$$\hat{q}^{\text{nb}}_{t|t} = \frac{\tilde{q}^{\text{nb}}_{t-1}}{\left\| \tilde{q}^{\text{nb}}_{t|t} \right\|_2}. \tag{3-28}$$

Using steps outlined in [17] and [24], the above outlined EKF was implemented in MATLAB. It was preliminarily tested in a limited case with actual smartphone sensors. Calibration and testing occurred in the same room, where the phone was continuously randomly orientated while recording sensor data. This data was afterwards exported and entered into the EKF. The results can be seen in Figure 3-3, where it is compared with the orientation estimation calculated by the android system. It is clear that the orientation estimation is almost identical to the system derived orientation.

3-4 Particle Filter

The step and heading system,which combines step detection and length estimation with orientation estimation, generates an estimate of the trajectory walked by the pedestrian. Since

there are uncertainties in all aspects of the SHS, it is inevitable that drift will occur within. The estimate then no longer accurately represents reality. In order to counter this drift, one method is to use map information in combination with a Particle Filter. This is a numerical Bayes estimator able to estimate non linear posterior densities, allowing for multimodal distributions [9, 15]. Multimodal distribution refers to distributions with multiple maxima in the probability density function. This allows the particle filter to have multiple hypothesis for the system that it is modeling. In the case of localization, it may therefore indicate that there are two position estimates are equally as likely based on the information that the filter has received so far. This can occur due to symmetry in the indoor environment [35].
The particle filter is known to work well in three dimensional state space [9], with higher dimensions making the particle representation too sparse to be a meaningfully represent the posterior distribution. Since for indoor localization only y position, x position and heading are needed, the particle filter is sufficient.

As the name suggests, the particle filter uses a finite amount of particles to determine a state estimate. These particles are weighted and are propagated through a motion model, augmented with a noise realization on relevant variables. These noise realizations represent uncertainty of certain components of the motion model. The probability density function of these noise realizations are defined beforehand. Taking the weighting of each particle into account a prior estimate can be made. The particle cloud is then, when possible, compared with measurements, adjusting the weightings accordingly. These measurements may come from different sources. This process is familiar to the reader as it closely resembles that of the previously mentioned Kalman Filter, also a Bayes estimator.

For indoor localization on one floor the state space is defined as $\mathbb{R}^2$, and in combination with map information is limited by the outer perimeter of the building the user is located in. The particle is defined by

$$x_k^i = \begin{pmatrix} x\ position \\ y\ position \\ heading \end{pmatrix}, \tag{3-29}$$

where x_k^i is the state of particle i at time k.
The initialization of the particle cloud depends on information known beforehand. If there is a known initial position, all particles can be initialized around that point. Otherwise the particles can be initialized by spreading them evenly within the outer perimeter of the building. The initial weight of once initialized the following four steps are performed recursively [35, 36]:

1. **Measurement update**
   Each particle is weighted against constraints or system evaluation function. The likelier the particle is within the system constraints, the higher the weighting is. Each particle weighted is redefined through

$$\omega_k^i = \frac{1}{c_k}\omega_{k-1}^i p(y_k|x_k^i), \tag{3-30a}$$

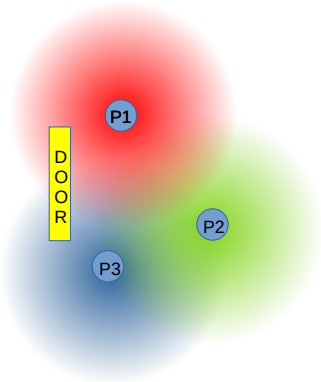
where the normalization weight (c_k) is given by

$$c_k = \sum_{i=1}^{N} w_{k-1}^i p\left(y_k | x_k^i\right) \tag{3-30b}$$

Here ω_k^i is the weighting for particle i at time k, y_k is the measurement at time k, and N is the total number of particles. For localization, the state of the particle is its xy position. With a map, the position of physical structures can be compared with the trajectory of particles as a measurement. For example, if these structures cannot be traversed, as is the case with walls, the trajectory of a particle that crosses this structure is incorrect. Regions in which the user cannot be found have a likelihood of 0, while all indoor accessible regions have a likelihood of 1. This form of measurement turns wall information into a two dimensional probability density function, an example of which can be found in Figure 3-4a.

A map can also be used to compare particle position with landmark position of detectable activities. For example, maps indicate where doors are located. If a door opening activity can be detected a conditional probability can be used to determine the weighting of each particle. This conditional probability can be a multivariate normal probability density function. This is defined by a mean in x and y position, which is the xy position of each particle, and a standard deviation along both axis. This is shown visually by Figure 3-4b. Here there are three particles each with a multivariate normal probability density function represented by the different colored circle, with a gradient in opacity. The more opaque the higher the likelihood is. When compared to the position of the door the weight of particle P3 and P2 will be higher than that of P2. If a door activity is detected the weighting of P1 and P3 will be higher than P2. This increases the likelihood of these two particles being resampled in the next step.

(a) 2D probability density function from map information. black is zero likelihood and white is a likelihood of one.



(b) Particles with a multivariate normal probability density function to determine weighting depending on position of a door

Figure 3-4: Using map information for measurement updates in the particle filter

2. **Estimation**

Using the weightings and positions of the particle cloud, an estimate can be made of the posterior probability density of the process that is being modeled [9]. There a different ways of defining the. The maximum a posteriori (MAP) estimate picks the particle with the highest weight from the posterior probability function [26], while the minimum mean square error (MMSE) estimate calculates a weighted mean [9, 26] as

$$\hat{x}_{k|k} = \sum_{i=1}^{N} w_k^i x_k^i \tag{3-31}$$

3. **Resampling**
Depending on the new weights of the different particles, a subset is taken and resampled to form a new set of particles, which will be used in the next iteration of the particle filter. There a multiple methods that can be used for resampling including multinomial, stratified resampling, systematic resampling, and residual resampling [14]. Hol et al. [14] conclude that from this set systematic resampling is the best considering resampling quality and computational complexity. For systematic resampling

$$x^{i*} = x(F^{-1}(u^i)), \tag{3-32a}$$

$$u^i = \frac{(i-1)+\tilde{u}}{N}, \tilde{u} \sim \mathcal{U}[0,1). \tag{3-32b}$$

Here x^{i*} represents the newly sample particle with index i and F^{-1} represents the generalized inverse of the cumulative probability distribution of the normalized particle weights, N is the total number of particles.

Resampling inevitably destroys information and therefore increases uncertainty by the random sampling [9]. Resampling should therefore occur only when it is needed. Gustafsson [10] proposes the use of *effective number of samples* to trigger a resampling. This can be calculated as followed

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w^i\right)^2} \tag{3-33a}$$

where

$$1 \leq N_{\text{cff}} \leq N \tag{3-33b}$$

The resampling condition can then be defined as $N_{eff} < N_{th}$ [10], where the threshold can be placed at $N_{th} = 2N/3$, with N being the total number of particles.

4. **Time update**
Each time step, every particle updates its own state with a dynamic model. For SHS, the dynamic model used is

$$x_{t+1}^i = \begin{pmatrix} x(0)_t + (l_t + e_l) * \cos(x(2)_t) \\ x(1)_t + (l_t + e_l) * \sin(x(2)_t) \\ x(2)_t + \Delta\theta_t + e_\theta \end{pmatrix}, \quad e_\theta \sim \mathcal{N}\left(0, \sigma_\theta^2\right), \quad e_l \sim \mathcal{N}\left(0, \sigma_l^2\right). \tag{3-34}$$

where $x(*)$ indicates the index of the state vector defined in Equation (3-29). The inputs to this dynamic model are l_t, which is the step length found by step detection and step length estimation, and $\Delta\theta_t$, which is the change in heading between the current and previous time step.

3-5 Activity Recognition

Chapter 4

Results

4-1 Step Detection

Salvi et al. [27] have opensourced the data used for their step detection algorithm. This includes validation data made with a ground truth device and the output of the devised algorithm on an android device. This allows for comparison between the two but also with eventual other techniques. The raw data consists of sampling time, three accelerometer axis signals, when a new step has been detected by the ground truth device, and step detected by the designed algorithm. The validation sets consist of two users carrying the phones in the carrying modes outlined in Section 2-1-2. The device is carried in each carrying mode individually, not simultaneously. An extract of this data can be found in Figure 4-1, showing the accelerometer magnitude and points indicating the ground truth step and the steps detected by the Salvi et al. [27] algorithm. The figure also shows how the carrying mode affects the characteristics of acceleration magnitude. For example, between carrying the smartphone in an armband and frontpocket, the former has a much more gradual change with a clear sinusoidal form, while the latter has a larger range of accelerations with much more abrupt changes.

The matlab algorithm, outlined in Section 3-1, could be applied on the validation data of the researchers. This allows it to be compared with a ground truth and with the solution of the researchers. Figure 4-2 shows how the matlab algorithm compares to the results in the validation datasets. Figure 4-2a indicates the absolute number of steps detected. Figure 4-2b shows the percentage error compared to the ground truth, where positive percent error indicates over counting, while negative under counting.

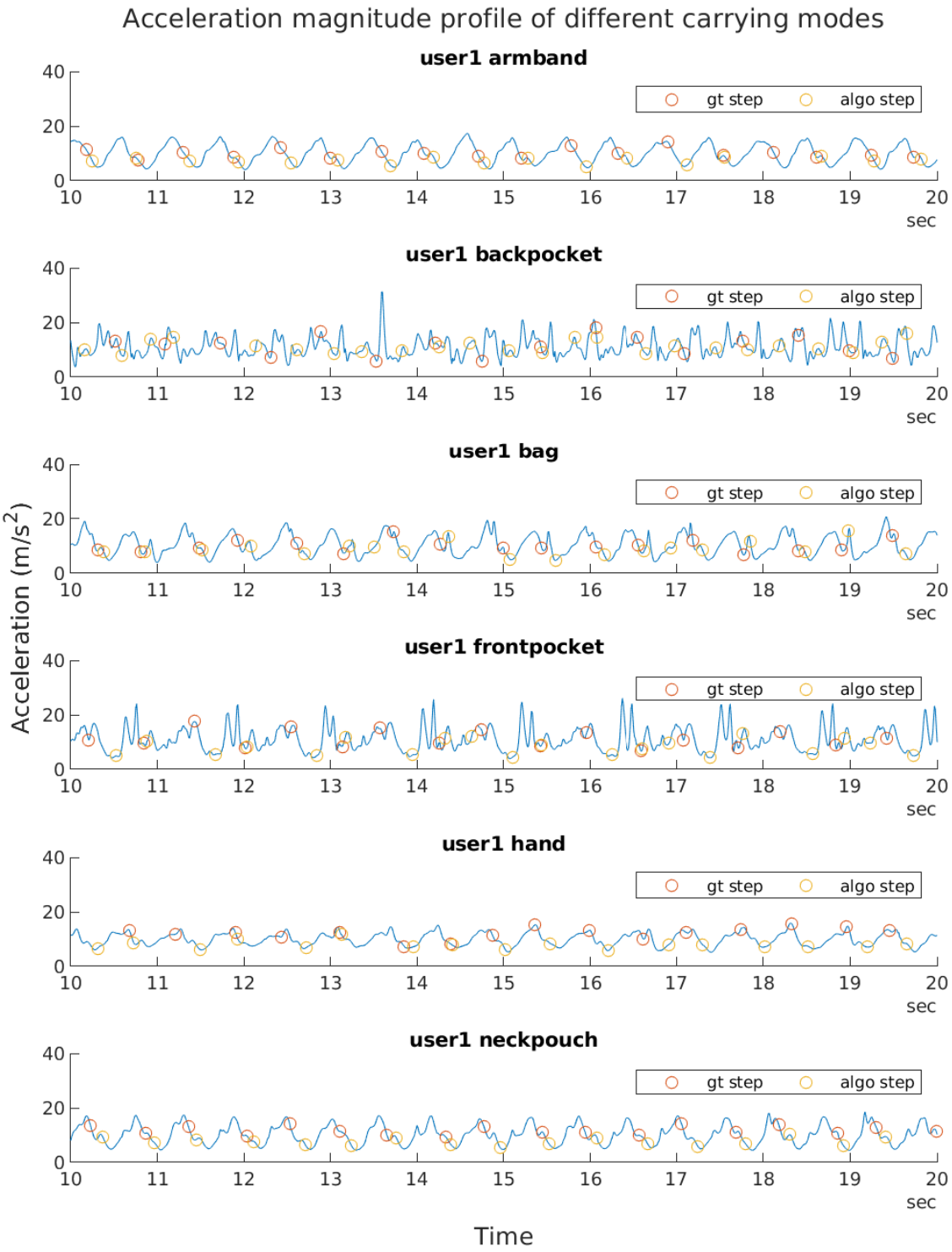
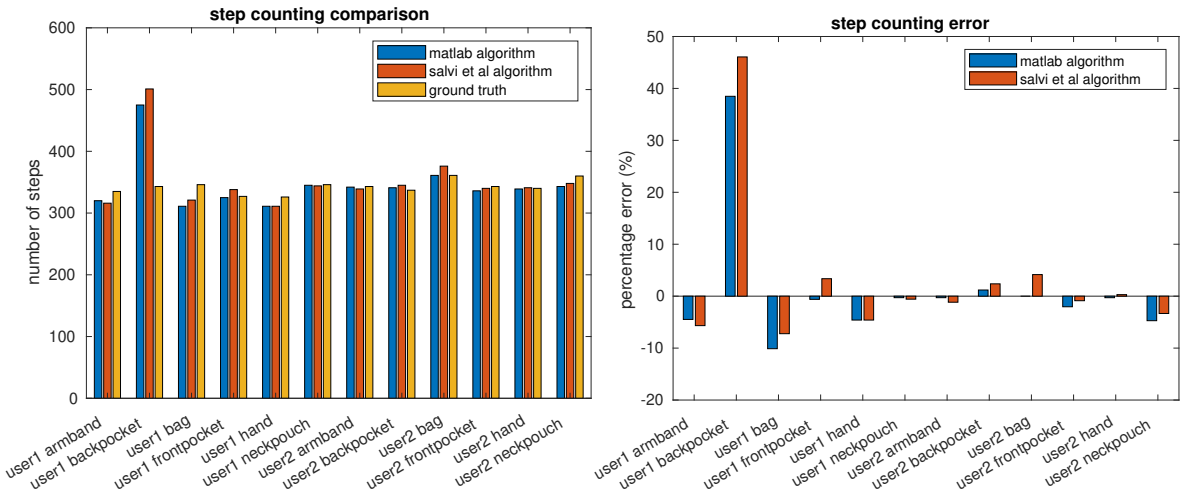


Figure 4-1: extract of validation dataset from Salvi et al. [27], indicating ground truth steps and step detected by their algorithm.



(a) Absolute number of steps counted. **(b)** Percentage error from ground truth number of steps.

Figure 4-2: Comparison between Salvi et al. [27] step detection algorithm, matlab algorithm and ground truth for different carrying modes.

In addition to the data made available online, original data was gathered in which a subject walked exactly 60 steps while having a smartphone in 3 different carrying modes, backpocket, frontpocket and in hand. The percentage error from the ground truth can be seen in Figure 4-3.

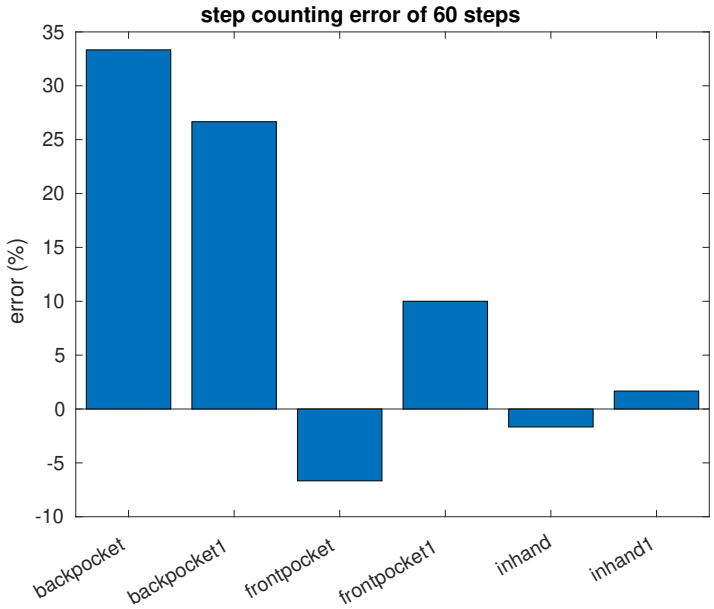


Figure 4-3

For a PDR SHS it is not enough for the amount of steps to be accurate, but also the time when a step actually occurs. A step detection combined with the heading orientation at the moment of occurrence determines the vector in which the position estimate of the pedestrian will move. In order to ascertain how both the Salvi et al. [27] algorithm and the algorithm

from Section 3-1 perform in this respect, true positive step detection was determined. Since it is unrealistic to expect the step detection to be at the exact time an actual step occurs, intervals are defined where if both a step occurs and is detected, the detection counts as a true positive. If in this interval two steps are detected, both detections are not considered true positive, and are considered a true positive double count. The time interval is increase itteratively in attempt to find the highest true positive count for both algorithms. The results can be found in Figure 4-4

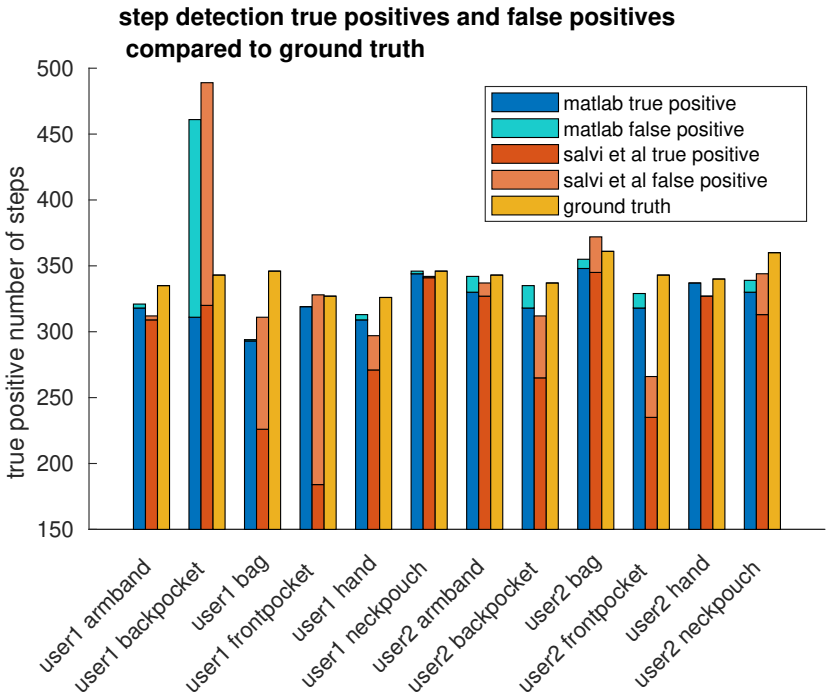


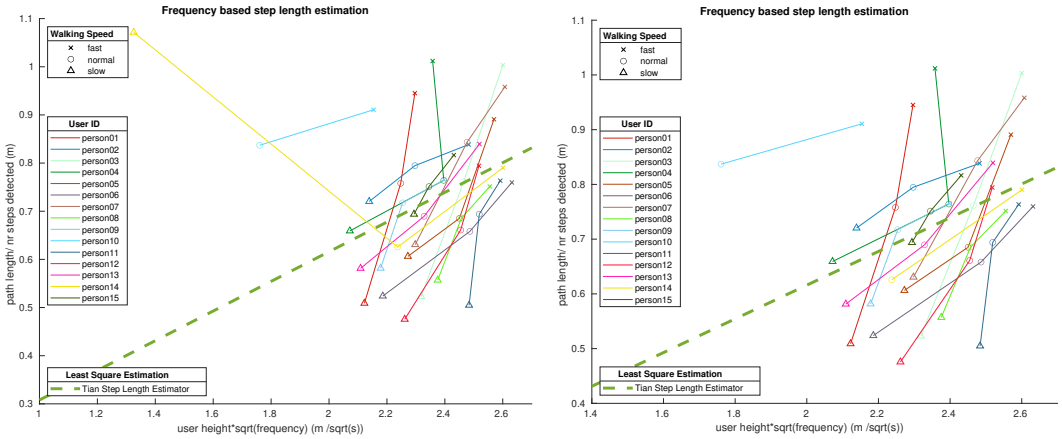
Figure 4-4: Comparison between Salvi et al. [27] step detection algorithm, matlab algorithm and ground truth for different carrying modes.

4-2 Step Length Estimation

Vezocnik and Juric [33] have also made the data they used open source. This data consist of accelerometer data of 15 different people for three walking speeds and in four smartphone carrying modes. Metrics for each test subject are collected, including height, gender and leg length. The walking modes were qualitative, in that they were either slow, normal, or fast walking speed. The carrying modes include the smartphone in pocket, in a bag, in the hand with the phone screen parallel to the floor, and in hand will swinging the carrying arm. Each person has two measurements for each combination, one for a 15 meter long straight path and another for 108 meter long straight path. The test subjects are not forced to walk these distances exactly, allowing for natural termination of their trial.
As done in [33] the smaller length set can be used to determine the parameter and the second to assess the performance.The best performing algorithm for global parameters is reiterated in Equation (4-1) for convenience, where h is the users height and F is the step frequency.

$$\text{step size} = K \cdot h \cdot \sqrt{F}. \tag{4-1}$$

In order to determine the tunable parameter K correctly for the SHS, it needs to be used with the output of the step detection algorithm. This is because step detection will have a direct effect on the step frequency. The step detection algorithm cannot guarantee that all steps are counted, potentially affecting the tunable parameter. From the results of step detection, the most accurate results came from holding a smartphone in the hand. This carrying mode is present in the data from [33], and can therefore be used for analysis. The results are shown in Figure 4-5. Here the parameters for Equation (4-1) of all three walking speeds are plotted. In order to find K as a universal constant a least square estimation is performed, shown by the green striped line in the plot.



(a) step length estimation constant = 0.3116 **(b)** step length estimation constant = 0.3080

Figure 4-5: step length estimation

Using the open source data has not come without problems, as can be seen in Figure 4-5a. While most data seems to have relatively good step detection, there are two samples that do not conform to the general trend suggesting that step detection is not working correctly.

The two potentially faulty data points are the slow walking sample of both person 10 and 14. For the former, no steps have been detected and is therefore not visible in the plots, while for the latter too little steps have been detected. For person 14, 14 steps have been detected for slow walking, which is less steps than when the subject was walking fast. This clearly suggests that a wrong step detection occurred. It is difficult to determine why this is occurring since the exact details. It could be that during this sample, the test subject was holding the phone incorrectly, the person had a very different step strategy when performing at this speed, or the phone was malfunctioning. This outlier affects the eventual tunable parameter. Removing it will change the estimate, as shown in Figure 4-5b. The performance of the step length estimate can be checked by using the validation dataset, where all users walked a longer distance, but with the same experimental setup. Here the accelerometer data can be passed through step detection and step length estimation to estimate the total distance traveled. This can then be compared with the actual distance traveled to determine the error. The data can also be used to determine if the difference in tunable parameter has a significant affect on the estimate. The results are found in Figure 4-6, where the absolute distance error for all walking speeds for all test subjects are shown, indicated by the dataset ID.

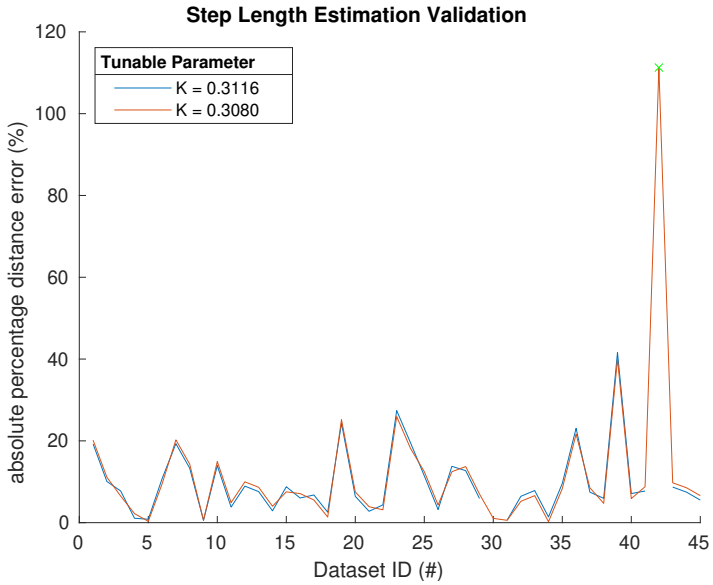


Figure 4-6

What is noticeable from the results is that there is a large outlier. This is with dataset 42, which corresponds to the slow walking speed of test subject 14, which is also the setting in which the tunable parameter estimation had an outlier. This further supports that there is something significantly different when this test subject is performing at this walking speed. With this outlier the mean absolute error is 11.8 percent with a standard deviation of 17.1 percent. Without the outlier, the mean absolute error is 9.7 percent with a standard deviation of 8.0 percent. Both results are worse than those cited by [33], which indicate a mean of 7 percent with a standard deviation of 5 percent. The results also indicate that the different tunable parameters do not make a significant difference in accuracy.

A similar smaller scale experiment was performed with three different people also walking at

three different walking speeds. The results can be found in Figure 4-7.

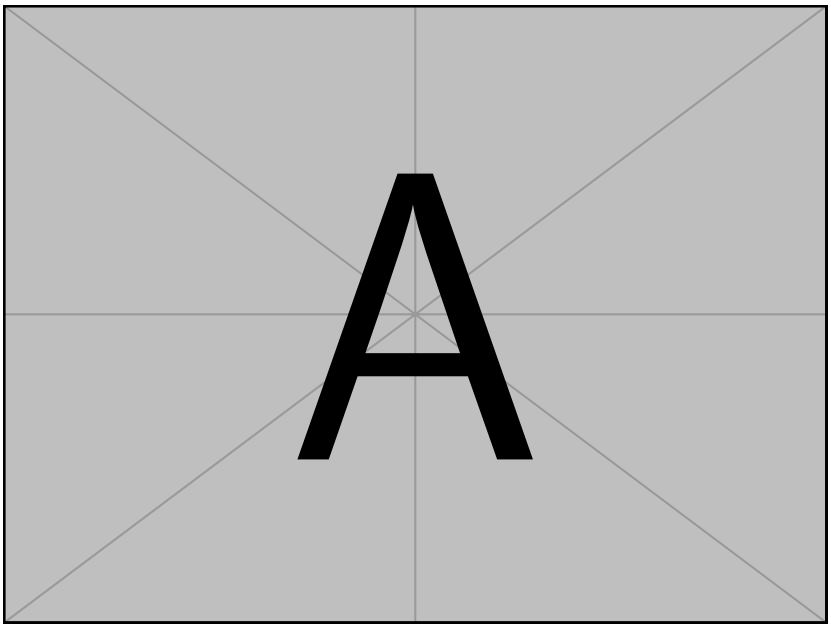


Figure 4-7

still need to add comment on results

4-3 Indoor Experiments

Due to restriction caused by the COVID-19 pandemic, indoor testing was located at the house of a family member. The test consisted of walking around indoors with a smartphone in hand, and recording the IMU signals in the phone through an app, while also filming the path that is being walked. During the experiment a smartwatch was worn around of the not smartphone hand and used to open doors. This opening of doors was also recorded using the app running on the smartphone, which had a button to record timestamps. The full experimental process can be found in Appendix A.

In order to use map information with the particle filter, different sources were combined to generate a map. Rudimentary paper blueprints of the building were available and could be photographed and traced in software to generate a picture of the indoor environment. This image has clear color distinction between the different structures within the building, including walls, doors and furniture. The positioning and size of furniture was estimated. The final image is seen in Figure 4-8a. This image is in pixel coordinates and needs to be transformed into meter coordinates. It then needs to represent a 2D probability density function so that it can be used in the particle filter measurement update.
The OccupancyMap in the MATLAB Robotics Tool box is suitable for this. An occupancy grid is a grid in which each cell has a value representing the probability of the occupancy of that cell. By measuring a known structure in Google Maps using the measurement tool and comparing with its pixel length in the image, a meter to pixel ratio can be defined. The Google Maps measurement can be found in Figure 4-8b. This pixel to meter ratio combined with the image, allows MATLAB to generate an occupancy grid as shown in Figure 4-8c. This same process can be used to get the meter position of all doors.

(a) Image from building blueprints. Black, grey and red represent walls, furniture, and doors, respectively.

(b) Measurement from Google Maps

(c) Particle Filter map result

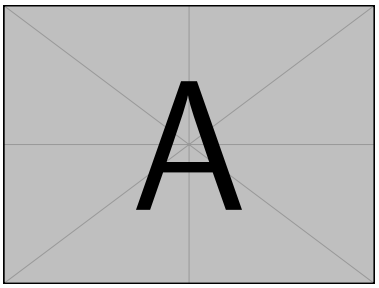
Figure 4-8: particle filter map creation

During the experiment, a total of 8 trials were walked by one test subject, each trying to generate a different trajectory than the previous trials. The video recorded during the ex-

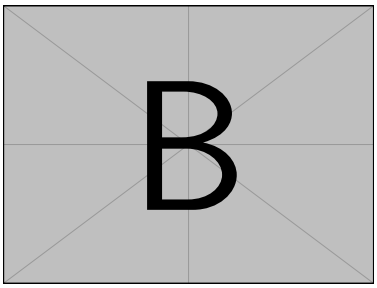
periment can be used in postprocessing to get a ground truth. This was done by replaying the video and at set intervals manually indicating approximately where on the map the test subject was located. The position and time elapsed were recorded. The output of these trials can be used to test certain SHS components individually and the system as a whole.

4-3-1 Indoor Orientation Estimation

The indoor orientation estimation method outlined in Section 3-3-5 can be compared with the output of the android operating system. Two examples can be found in



(a) Orientation estimation of trial 1 compared to android orientation estimation



(b) Orientation estimation of trial 2 compared to android orientation estimation

The error between each trial can be found in Figure 4-10.

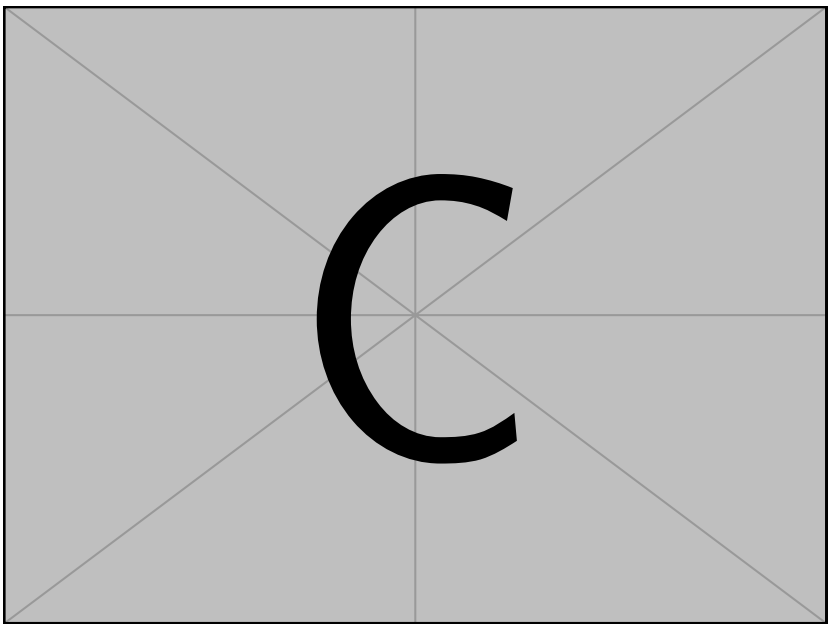
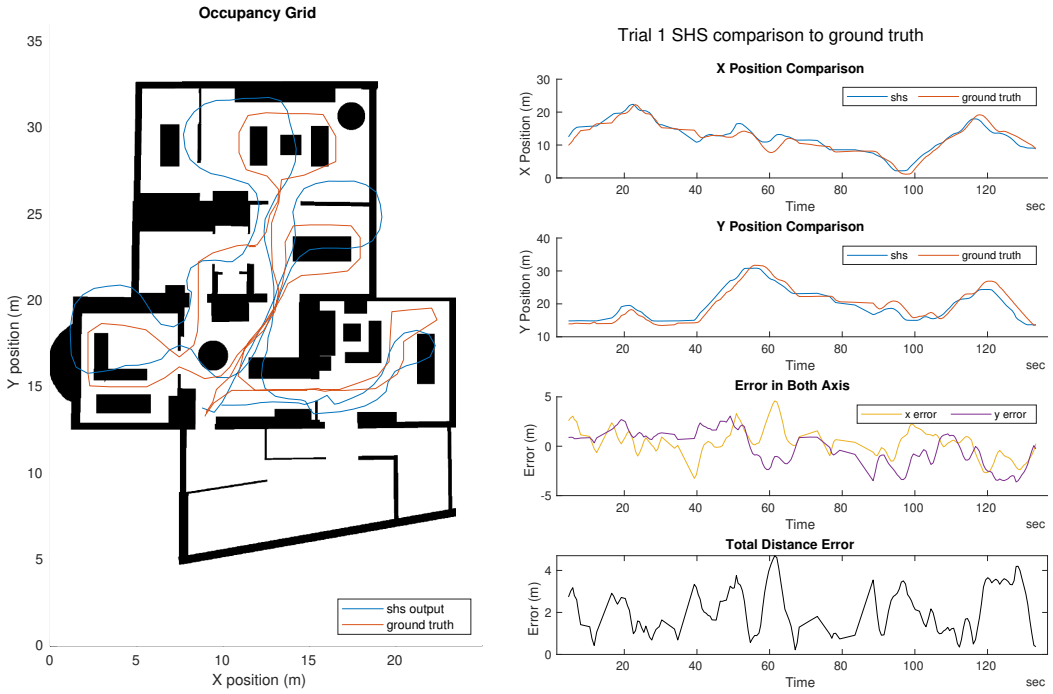


Figure 4-10

From these results it is shown that the error between my own orientation estimation and that calculated by the android system has similar performance.

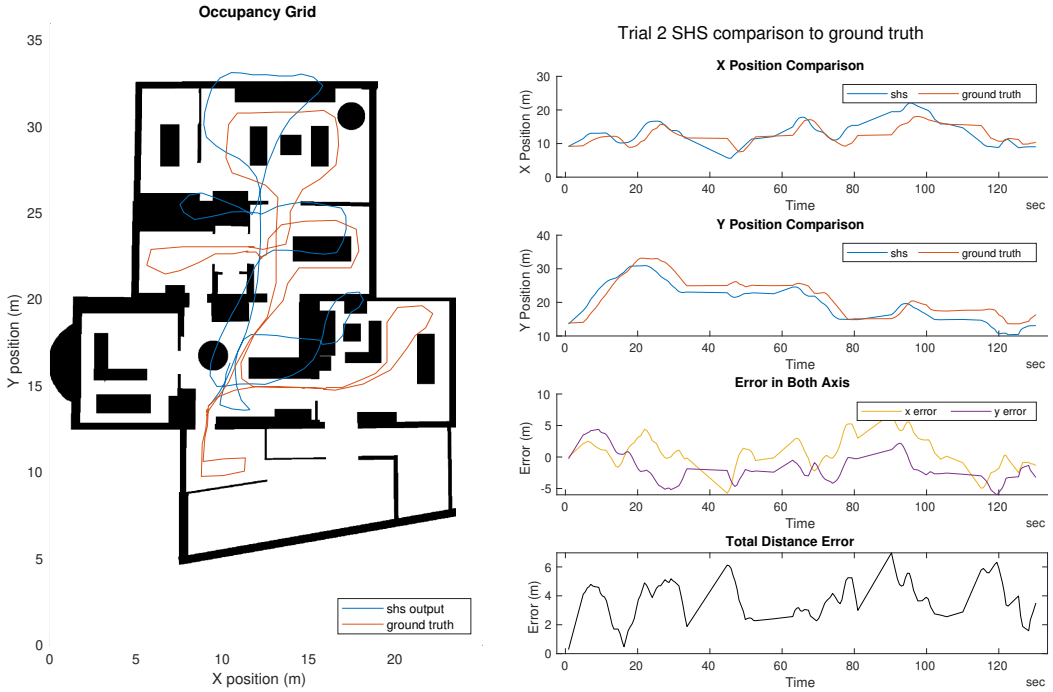
4-3-2 Step and Heading System

All components of the step and heading system have been tested separately, giving an indication of their strengths and limitations. Now the whole system can be combined and tested in an indoor environment to evaluate the whole system performance. The output of the step and heading system will be the relative position change from a start point, known as dead reckoning. For every trial the smartphone imu data is passed through the different components of the SHS outlined in this report. Each trajectory can then be compared to the ground truth generate through post processing as outlined at the beginning of this sections. Two trials with their trajectory projected onto the map and side by side comparison with the ground truth can be found in Figure 4-11 and Figure 4-12. Note that the shs trajectories in these images have been rotated heuristically in order to fit the map as best as possible. This needs to be done since there is no way for the step and heading system output to know how to orient itself with respect to the building. This is because it does not have pre-existing knowledge on the orientation of the magnetic field inside the building.



(a) trajectory comparison **(b)** axis comparison

Figure 4-11: SHS comparison of trial 1 with ground truth



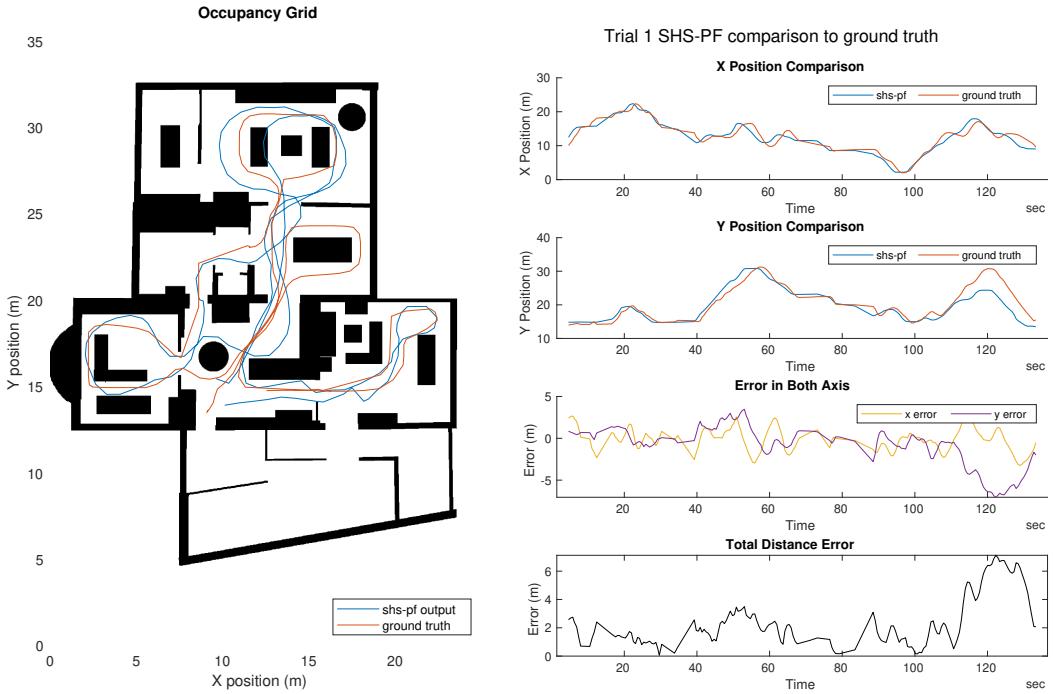
(a) trajectory comparison **(b)** axis comparison

Figure 4-12: SHS comparison of trial 2 with ground truth

4-3-3 Particle Filter

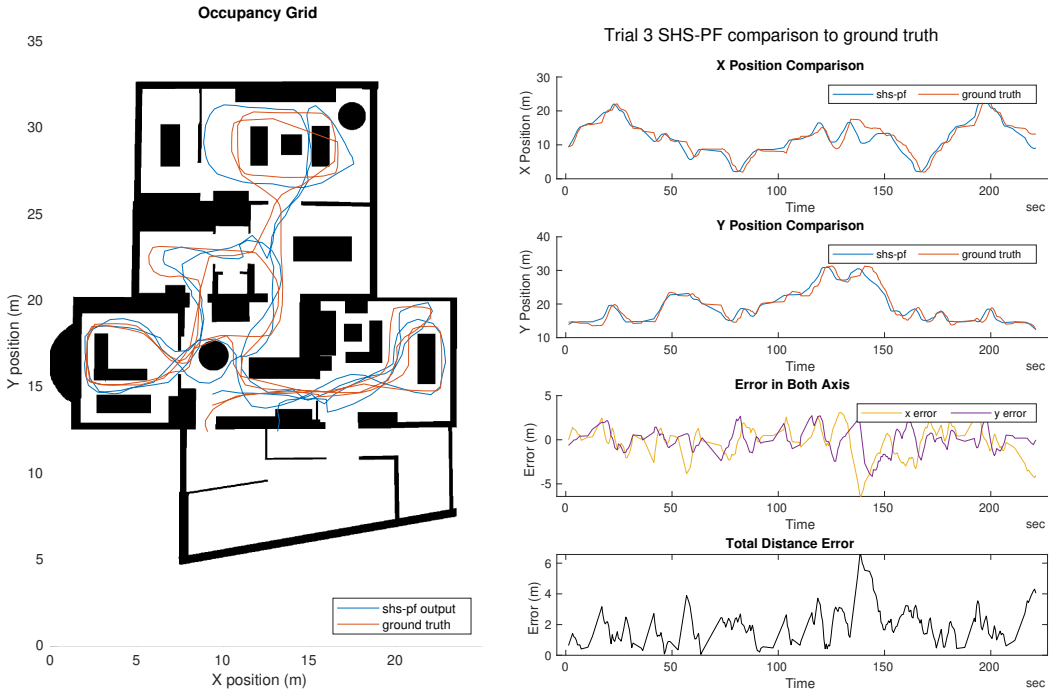
Using the output of the SHS and the method outlined in Section 3-4 for the particle filter, the indoor localization particle filter could be constructed

With activity



(a) trajectory comparison **(b)** axis comparison

Figure 4-13: SHS comparison of trial 2 with ground truth



(a) trajectory comparison **(b)** axis comparison

Figure 4-14: SHS comparison of trial 5 with ground truth

4-4 Activity Recognition

Chapter 5

Discussion

Chapter 6

Conclusion

Appendix A

Indoor Localization Experiment

Apparatus:

Hardware:

- 3 Smartphones
  - Samsung j5 for video recording
  - Iphone for backup orientation estimation and post processing time synchronization
  - One Plus Nord for primary orientation estimation and ground truth activity recorder
- 1 apple smartwatch
- Shirt with two breast pockets
- Indoor test location

Software:

- Iphone and apple watch app (https://apps.apple.com/us/app/sensorlog/id388014573#?platform=appleWatch)
- Android app for primary sensor recording (https://play.google.com/store/apps/details?id=fr.inria.tyrex.senslogs&hl=en&gl=US)

Calibration:

Calibrate primary estimation sensor (One Plus Nord) just before experiment, and perform all calibration at the same location within test location!

- *Magnetometer*:
  - Stand away from any clear magnetic disturbance

  - rotate the phone in as many orientations as possible, infinity sign is often used

- *Accelerometer*:

  - Rotate the phone slowly in as many orientations as possible in attempting to be in as many orientations as possible.

- *Gyroscope / Magnetic North / Noise*

  - Lay the phone horizontally and start recording while stationary

Preparation:

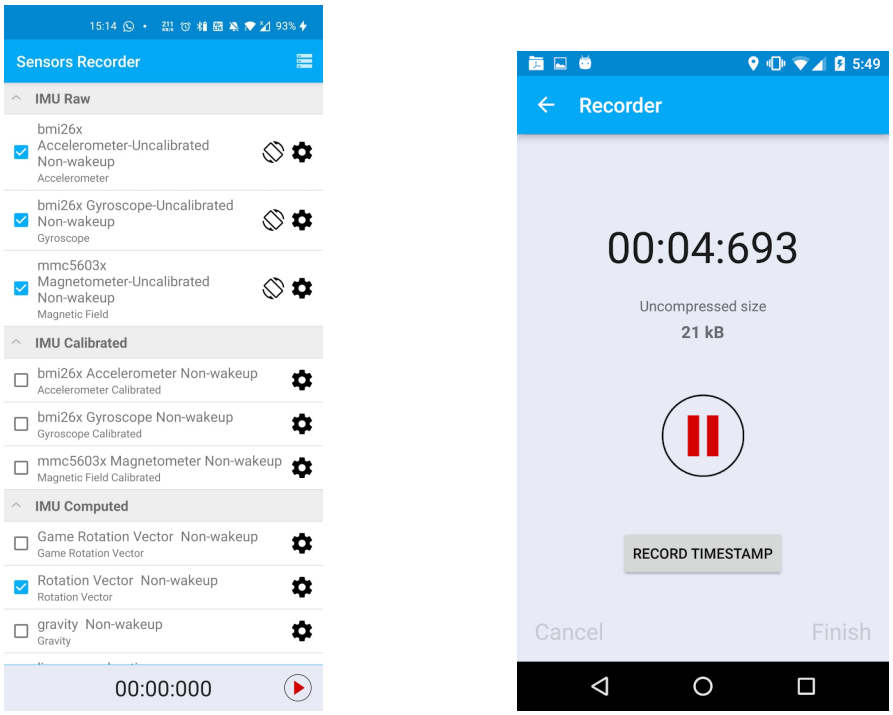
1. Calibrate One Plus Nord with the calibration method outlined above
2. Determine with which hand you will hold the phone, strap the smartwatch to the other hand
3. Ensure that all doors in test location are closed
4. Define start location within test location and record it

Testing:

For *each test run* perform the following steps:

1. Go to start location
2. Start sensor recording on smartwatch and iPhone
3. place iPhone in one of the breast pockets of the shirt
4. Start a video recording on the Samsung phone and place it in the other breast pocket, making sure that the lens is not covered and can see what is in front of the torso of the test subject, hence also what direction is being moved in.
5. Start sensor recording on One Plus Nord, logging the uncalibrated accelerometer, gyroscope, and magnetometer. Also record the rotation vector that the phone records. This is shown in illustration 1.
6. Walk around test location as naturally as possible, holding the One Plus Nord in front of you with the screen point up. Try and keep this phone orientation as best as possible.
7. Walk to doors and open them using the arm that has the smartwatch on it. When touching a door handle click the button in the android app that records the time stamp as shown in illustration 2. Close the door behind you, so that it can potentially be opened again later.
8. After walking for around 3 minutes, return to start location
9. Stop recording of sensor and video on all devices
10. export the data (film and sensor) from all 3 devices to the same folder, making it clear which trail it was

Pictures:



(a) Recording settings

(b) Record timestamp button

Figure A-1

Postprocessing

1. Calibrate walking around imu sensor data using calibration data
2. Run calibrated walking around data through orientation estimation algorithm
3. Compare result with orientation estimation made by the system and maybe even with the iphone
4. Determine steps and subsequent step length
5. Combine orientation information and step information to generate an estimated trajectory
6. Use this estimated trajectory as input for the particle filter
7. Using the video recording made during testing, manually indicate per step where on the blueprint you are. This will be used to determine the performance of the estimate of the SHS system.

Bibliography

[1] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. *UbiComp 2013 - Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 225–234, 2013. doi: 10.1145/2493432.2493449.

[2] Steven H. Collins and Arthur D. Kuo. Two Independent Contributions to Step Variability during Over-Ground Human Walking. *PLoS ONE*, 8(8):1–11, 2013. ISSN 19326203. doi: 10.1371/journal.pone.0073597.

[3] Christophe Combettes and Valerie Renaudin. Walking direction estimation based on statistical modeling of human gait features with handheld MIMU. *IEEE/ASME Transactions on Mechatronics*, 22(6):2502–2511, 2017. ISSN 10834435. doi: 10.1109/TMECH.2017.2765005.

[4] Pavel Davidson and Robert Piché. A Survey of Selected Indoor Positioning Methods for Smartphones, apr 2017. ISSN 1553877X.

[5] Estefania Munoz Diaz, Ana Luz Mendiguchia Gonzalez, and Fabian De Ponte Müller. Standalone inertial pocket navigation system. *Record - IEEE PLANS, Position Location and Navigation Symposium*, pages 241–251, 2014. doi: 10.1109/PLANS.2014.6851382.

[6] Luis Enrique Diez, Alfonso Bahillo, Jon Otegui, and Timothy Otim. Step Length Estimation Methods Based on Inertial Sensors: A Review. *IEEE Sensors Journal*, 18(17):6908–6926, 2018. ISSN 1530437X. doi: 10.1109/JSEN.2018.2857502.

[7] Luis Enrique Diez, Alfonso Bahillo, Jon Otegui, and Timothy Otim. Step Length Estimation Methods Based on Inertial Sensors: A Review. *IEEE Sensors Journal*, 18(17):6908–6926, 2018. ISSN 1530437X. doi: 10.1109/JSEN.2018.2857502.

[8] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[9] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.

[10] Fredrik Gustafsson. *Statistical sensor fusion*. Studentlitteratur, 2010.

[11] Michael Hardegger, Daniel Roggen, Sinziana Mazilu, and Gerhard Troster. ActionSLAM: Using location-related actions as landmarks in pedestrian SLAM. *2012 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2012 - Conference Proceedings*, (April 2014), 2012. doi: 10.1109/IPIN.2012.6418932.

[12] Robert Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys and Tutorials*, 15(3):1281–1293, 2013. ISSN 1553877X. doi: 10.1109/SURV.2012.121912.00075.

[13] Hashim A Hashim. Special Orthogonal Group SO(3), Euler Angles, Angle-axis, Rodriguez Vector and Unit-quaternion: Overview, Mapping and Challenges. (3), 2019.

[14] Jeroen D Hol, Thomas B Schon, and Fredrik Gustafsson. On resampling algorithms for particle filters. In *2006 IEEE nonlinear statistical signal processing workshop*, pages 79–82. IEEE, 2006.

[15] Johan Kihlberg and Simon Tegelid. Map aided indoor positioning, 2012.

[16] Manon Kok and Thomas B. Schon. Magnetometer calibration using inertial sensors. *IEEE Sensors Journal*, 16(14):5679–5689, 2016. ISSN 1530437X. doi: 10.1109/JSEN.2016.2569160.

[17] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. Using inertial sensors for position and orientation estimation. *Foundations and Trends in Signal Processing*, 11(1-2):1–153, 2017. ISSN 19328354. doi: 10.1561/2000000094.

[18] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[19] Thomas Moder, Clemens Reitbauer, Markus Dorn, and Manfred Wieser. Calibration of smartphone sensor data usable for pedestrian dead reckoning. In *2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*, volume 2017-Janua, pages 1–8, 2017. ISBN 9781509062980. doi: 10.1109/IPIN.2017.8115910.

[20] Estefania Munoz Diaz, Dina Bousdar Ahmed, and Susanna Kaiser. *A Review of Indoor Localization Methods Based on Inertial Sensors*. Elsevier Inc., 2019. ISBN 9780128131893. doi: 10.1016/b978-0-12-813189-3.00016-2. URL http://dx.doi.org/10.1016/B978-0-12-813189-3.00016-2.

[21] Estefania Munoz Diaz, Dina Bousdar Ahmed, and Susanna Kaiser. A Review of Indoor Localization Methods Based on Inertial Sensors. *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*, pages 311–333, 2019. doi: 10.1016/b978-0-12-813189-3.00016-2. URL http://dx.doi.org/10.1016/B978-0-12-813189-3.00016-2.

[22] Girish Palshikar et al. Simple algorithms for peak detection in time-series. In *Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*, volume 122, 2009.

[23] Jiuchao Qian, Jiabin Ma, Rendong Ying, Peilin Liu, and Ling Pei. An improved indoor localization method using smartphone inertial sensors. *2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013*, (October):1–7, 2013. doi: 10.1109/IPIN.2013.6817854.

[24] Reglerteknik Linkoping. Statistical Sensor Fusion - Lab 2: Orientation Estimation using Smartphone Sensors, 2013.

[25] Mingrong Ren, Kai Pan, Yanhong Liu, Hongyu Guo, Xiaodong Zhang, and Pu Wang. A novel pedestrian navigation algorithm for a foot-mounted inertial-sensor-based system. *Sensors (Switzerland)*, 16(1):9–11, 2016. ISSN 14248220. doi: 10.3390/s16010139.

[26] S. Saha, Y. Boers, H. Driessen, P. K. Mandal, and A. Bagchi. Particle based MAP state estimation: A comparison. *2009 12th International Conference on Information Fusion, FUSION 2009*, pages 278–283, 2009.

[27] Dario Salvi, Carmelo Velardo, Jamieson Brynes, and Lionel Tarassenko. An Optimised Algorithm for Accurate Steps Counting from Smart-Phone Accelerometry. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2018-July:4423–4427, 2018. ISSN 1557170X. doi: 10.1109/EMBC.2018.8513319.

[28] Pavel Senin. Dynamic Time Warping Algorithm Review. *Science*, 2007(December):1–23, 2008. ISSN 1557170X. URL http://129.173.35.31/{~}pf/Linguistique/Treillis/ReviewDTW.pdf.

[29] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala. Inertial Odometry on Handheld Smartphones. *2018 21st International Conference on Information Fusion, FUSION 2018*, pages 1361–1368, 2018. doi: 10.23919/ICIF.2018.8455482.

[30] Melania Susi, Valérie Renaudin, and Gérard Lachapelle. Motion mode recognition and step detection algorithms for mobile phone users. *Sensors (Switzerland)*, 13(2):1539–1562, 2013. ISSN 14248220. doi: 10.3390/s130201539.

[31] Qinglin Tian, Zoran Salcic, Kevin I.Kai Wang, and Yun Pan. A Multi-Mode Dead Reckoning System for Pedestrian Tracking Using Smartphones. *IEEE Sensors Journal*, 16(7):2079–2093, 2016. ISSN 1530437X. doi: 10.1109/JSEN.2015.2510364.

[32] Michel Verhaegen and Vincent Verdult. *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[33] Melanija Vezocnik and Matjaz B. Juric. Average Step Length Estimation Models' Evaluation Using Inertial Sensors: A Review. *IEEE Sensors Journal*, 19(2):396–403, 2019. ISSN 1530437X. doi: 10.1109/JSEN.2018.2878646.

[34] Harvey Weinberg. Using the ADXL202 in Pedometer and Personal Navigation Applications. *Analog Devices AN-602 application note*, 2(2):1–6, 2002. URL www.BDTIC.com/ADI.

[35] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. *UbiComp 2008 - Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 114–123, 2008. doi: 10.1145/1409635.1409651.

[36] Yuan Wu, Hai Bing Zhu, Qing Xiu Du, and Shu Ming Tang. A Survey of the Research Status of Pedestrian Dead Reckoning Systems Based on Inertial Sensors. *International Journal of Automation and Computing*, 16(1):65–83, 2019. ISSN 17518520. doi: 10.1007/s11633-018-1150-y.

[37] Ning Yu, Yunfei Li, Xiaofeng Ma, Yinfeng Wu, and Renjian Feng. Comparison of Pedestrian Tracking Methods Based on Foot-and Waist-Mounted Inertial Sensors and Handheld Smartphones. *IEEE Sensors Journal*, pages 1–1, may 2019. ISSN 1530-437X. doi: 10.1109/jsen.2019.2919721.

Glossary

List of Acronyms

SHS	Step and Heading System
INS	Inertial Navigation System