Indoor Localization through Pedestrian Dead Reckoning and Activity Recognition

A smartphone based proof of concept

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

A smartphone based proof of concept

Master of Science Thesis

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

Bart van Ingen

November 10, 2020

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

Abstract

Abstract still needs to be written

Table of Contents

List of Figures

List of Tables

Preface

Preface still needs to be written

Acknowledgements

Acknowledgements still needs to be written

Delft, University of Technology
November 10, 2020

Bart van Ingen

Dedication still needs to be written

Chapter 1

Introduction

Indoor localization is an ongoing research field where there is yet a solution to be found that is widely accepted as the state of the art [9]. For outdoor localization, the de facto solution is the Global Positioning System (GPS) [25]. It is a constellation of satellites circling the globe, emitting radio signals at specific frequencies. Once received, these signals can be used to triangulate a position.

The GPS solution can not be applied reliably in an indoor environment. This is because of the difficulty of radio waves in crossing solid structures reliably. Additionally, if a signal is received indoors, it may have been rebounded off of radio reflective surfaces, causing erroneous positioning estimates [25].

As with outdoor localization, indoor localization has a large range of possible applications. It can be used for navigation in unknown environments, such as firefighters in an emergency situation or personal navigation through large public buildings [8, 25]. Another application is the tracking of individuals, such as within elderly homes or workers in a factory setting [8]. For wide applicability it is preferable that an indoor localization method provide accurate position estimations, is easily scalable, and has low system infrastructure cost [8].

In line with these applicability wishes, the smartphone has proven to be the perfect platform for implementation. It is an untethered device widely used in the developed world [8], containing many different sensors combined with computing and communication capabilities. Creating a system that works well on this device will be crucial to launching indoor localization on a global scale [17]. Focusing on this platform brings additional restrictions to what any eventual solution can use. This includes robustness in carrying modes and computational requirements. For example, continuous use of a camera is not beneficial for battery life, limiting the period in which this method can be used [45, 55]. Due to the advantages that smartphones bring, they have been used often in indoor localization research [8, 25, 37, 55].

Within the research field of indoor localization, the approaches are either infrastructure dependent or not, both using sensors on the target. Combined solutions have attempted to use the advantages of each to improve performance [8, 17].

The infrastructure dependent methods imitate the GPS setup on a local scale. They often replace the GPS constellation with other signal-producing devices available within the indoor environment. Some of these systems, such as wireless fidelity(Wifi) and Bluetooth, are detectable through smartphone sensors, while wireless sensor networks (WSN), and ultra-wideband (UWB) require dedicated devices or attachments [9, 25, 54]. For many of these solutions detailed information on the position of the external devices may be needed for estimation, including a signal strength map or device location [quote needed]. This information may not be easily available. Furthermore, these solutions naturally only work in buildings with the installed equipment. This requires either that they are already present or some initial investment and setup. In some cases this can not be expected. For example within subway systems, where legal or investment issues and potential vandalism can be a hindrance [49].

Infrastructure independent positioning solutions do not need to receive signals from external devices. Since no external infrastructure is required, these methods can have low infrastructure cost. Many such methods make use of cameras or inertial sensors placed on the target; sensors often found in smartphones. Camera-based systems uses unique visual features tracked in subsequent frames to determine displacement [17]. With such an approach, privacy concerns, computing complexity and device placement need to be considered. Inertial sensor based methods use sensors that measure linear acceleration and angular velocity. They are micro-electromechanical systems (MEMS), consisting of triaxial orthogonal accelerometers and gyroscopes, and can include a triaxial magnetometer [55]. These sensors are relatively small, cheap and have low power consumption [35]. Low power consumption allows for long periods of use, and therefore long term localization.

While the smartphone market has matured, the market for wearable devices has been growing steadily [26], with the smartwatch being the most prominent. These devices offer similar sensors and capabilities as smartphones, with the same advantages and disadvantages. One form of additional information that they can present is the detection of interactions with the indoor environment not detectable through smartphone sensors alone [43]. Indoor interactions can be position dependent. Examples include interacting with doors, stairs, and furniture. By knowing the location of these structure within the indoor environment, a position measurement can be made if a relevant interaction is detected. The information from these devices can be combined with the data from smartphone sensors to potentially improve localization. This leads to the overall goal of this thesis:

How can IMU sensors in a smartwatch be used to improve smartphone based indoor localization?

1-1 Research Questions

ATM I am not very sure on these research questions. I would like to discuss with you better options!

In order to tackle the subject, two main research questions can be defined, with further subquestions and explanation below them.

- **How can indoor localization be achieved with realistic smartphone placement, while taking computation capabilities into account?**

  - What indoor localization methods consider realistic smartphone carrying modes?

- **How can wearable devices improve indoor localization?**
  - What information can be derived from wearable devices?

1-2 Thesis Contributions

This thesis will explore the different possibilities for realistic smartphone-based indoor localization and how wearable devices can aid in improving a position estimate. It will explore a proof of concept with the intention of showing how data from wearable technology can improve localization. This thesis will test the designed system's performance in experiments, highlight the different advantages, disadvantages, and limitations. Future steps are outlined on how to improve the derived method for future research.

1-3 Thesis Outline

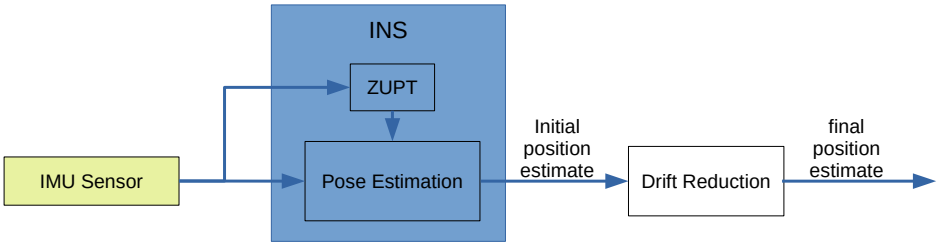
Thesis outline will be completed once thesis structure has solidified

Chapter 2

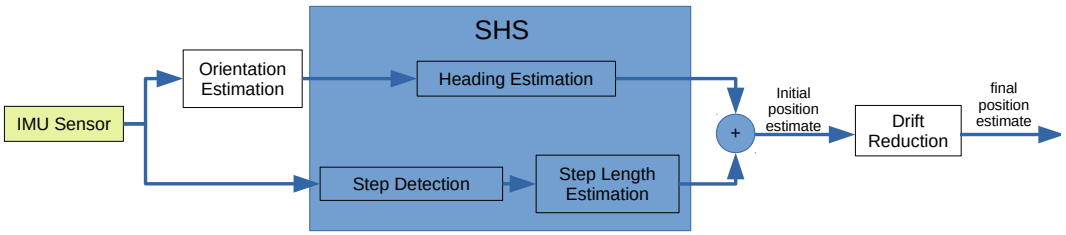
Related Work

2-1 Pedestrian Dead Reckoning

The positioning technique that uses MEMS IMU sensors is called Inertial Pedestrian Dead Reckoning (PDR). Dead reckoning indicates that the output of the system will be the relative position change from a start point [56]. Within PDR there are two methods, namely Inertial Navigation Systems (INSs) and Step and Heading Systems (SHSs). The INS mechanism integrates the IMU signals directly and is the implementation used most often within research [13]. SHS is based on integrating the displacement vectors associated with each step taken during pedestrian locomotion. Many PDR methods accumulate error with time and are therefore not ideal for long-term pedestrian tracking [20]. Drift reduction methods can be used to try and compensate for this error in order to allow for long-term tracking [34]. An overview of the different components required by the two PDR systems can be found in Figure 2-1, with further explanation in the following sections.



(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 Inertial Navigation System estimates pose, consisting of orientation and position, using sensor fusion algorithms. These methods are directly applied to the signals generated by MEMS IMU sensors [54]. Examples of sensor fusion algorithms are the Extended Kalman Filter (EKF) and Complementary Filter (CF) [29]. 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 [22].

A technique frequently used to compensate for the built-up error in INS is zero velocity update (ZUPT) and zero angular update (ZARU) [22]. 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 [22, 54]. this a form of pseudo-measurement. This is because the stationary phase is detected, with zero angular and linear 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 for the sensor fusion estimate. This process is generally known as a measurement update and can occur every time a stationary period is detected. Through this measurement update the error grows linearly with number of steps [14].

Detecting brief stationary time periods in pedestrian IMU data is done easiest when the sensor is placed on the foot [9, 12]. This is because a stationary period is much more pronounced in accelerometer data when the sensor is placed on the foot [54, 57]. 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 [39]. This has lead to most INS research being foot-sensor based [12, 54]. A recent deviation from this trend, Solin et al. [45] overcome this preference for a foot-based system with a combination of several different 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 [54], it suggests that it not yet appropriate for other sensor placements. This is relevant for PDR in realistic smartphone use cases since these devices are generally not carried in one specific way, and if they are, it is doughful if they are placed on the foot.
With this information in mind, there are two tracks that can be followed. The first is recreating the solution in [45]. The other is determining if the other PDR method is more appropriate. In the next section, the second track will be followed.

2-1-2 Step and Heading System

A step and heading system is a form of dead reckoning that detects steps, estimates step length, and the heading in which the pedestrian is moving. This position estimation can be represented by [33]

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

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 step heading of the pedestrian at time t. There are three components needed to apply this system. These are walk and step detection, step length estimation and step heading estimation. Similarly to INS, SHS position error is proportional to the number of steps taken. The largest difference is that this error growth does not have a preference of using foot-based sensors, allowing for other sensor placements [13]. However, other of sources of error can be introduced due to errors in the detection of steps and their length estimation. Since other sensor placements are possible, SHS is the PDR technique often used with smartphones as the base for sensing [quote needed]. This indicates that this technique is appropriate for further investigation in answering the thesis research questions.
Each component of a step and heading system can be investigated individually in attempt to find the best solution for each. The assumption made is that combining the best solution for each component will lead to the best overall step and heading system. The individual components will be investigated next.

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.
Techniques such as feature classification, frequency domain analysis, and time-domain thresholding can be used to detect the two activities [55]. An overview of techniques is shown in Figure 2-2 and summarized in Table 2-1.

Naturally, each form of analysis for walk and step detection has its advantages and disadvantages.
Time-domain analysis often uses the acceleration norm, making it robust to sensor orientation [9]. Its simplest form, thresholding, is trivially simple to implement. The problem with thresholding is that it is difficult to determine the optimal value, as it can vary between users, surfaces, and even shoes [4].
Similar to time-based methods, the periodicity in the norm of acceleration during locomotion is frequently invariant to sensor placement. This therefore also makes frequency analysis robust to sensor orientation. Frequency methods have the disadvantage that they are only able to detect periodic motion. For both frequency and time domain-based analysis, it is possible that certain motions, aside from the desired motion, can cross the threshold and generate false positives. In addition, frequency analysis and template matching, a more advanced time based method, will have a certain computational overhead to consider [9, 22].
Feature classification methods use features in accelerometer traces to classify steps, using form of machine learning. These method have the advantage that they can determine relations from labeled data automatically, removing a need for humans to define them. This is

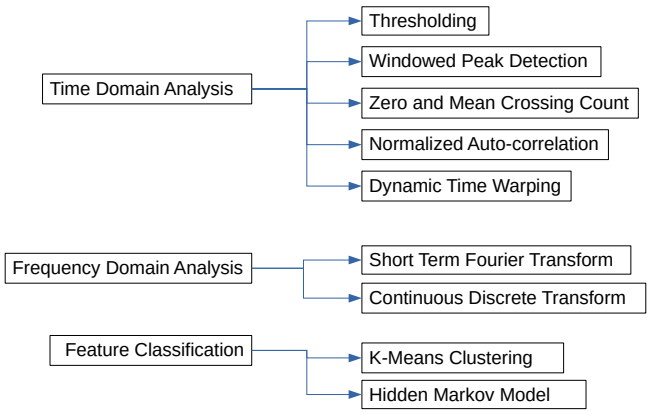


Figure 2-2: Overview of walk and step detection methods

also a disadvantage as it requires labeled data, the gathering of which can be a labor-intensive process [5]. Furthermore, the relationships they determine will depend on the data gathered. This may lead to person dependent performance [quote needed]. In addition, certain classification methods may require significant processing power, limiting the platforms on which it can perform [quote needed].

In [4], 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. In order to compare the different techniques, Brajdic and Harle [4] 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 the nine algorithms.
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 windowed peak detection as the most efficient algorithm for step detection. The best walk detection algorithms were thresholds on either standard deviation or signal energy, short-term Fourier transform, and normalized autocorrelation.
Salvi et al. [41] build upon the conclusions and recommendations of Brajdic and Harle [4], 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. [36] and consists of 5 stages run in series. These are pre-processing, filtering, scoring, detection and post processing. An exhaustive grid search across the parameter space was performed to find the optimal set for step detection. Using these parameters, an average accuracy of $95\% \pm 4.5\%$ was reached for the different carrying modes.

Technique	Explanation
Threshold	Acceleration magnitude is monitored for the passing of certain values or sign changes, in the case of Zero Crossing method [9, 22]. Typically determines when the foot is on the floor [22], but has also been applied to pitch angle of the upper leg, where the sensor has been placed [11]. In some instances, the thresholds are altered adaptively, for example through different motion mode detection or information gathered from the previous step [54].
(Windowed) Peak Detection	Recognizes local maximum or minimum on acceleration magnitude caused by foot impact on the floor, often within a sliding window [46]. Generally combined with thresholding, which is one of the simplest combination used with SHS [9].
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 [22]. Since the lag for the highest correlation is not known beforehand, an interval is often swept and correlation values compared.
Dynamic Time Warping	Similar to autocorrelation, it measures the similarity between two waveforms from accelerometer data [9], 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 [9].
Short Term Fourier Transform	Transforms acceleration signal of successive windows of data into the frequency domain. For spectral analysis, a subset containing a stride (two steps) is required to determine the frequency [22].
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 [9].
Hidden Markov Model	Uses gait cycles segmented into different states of a state machine, to determine if a step has been taken [39]. Thresholds can be used to induce a state change. If a full state machine cycle is achieved, a step is detected.
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 then recieves its label.

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 [6] 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 [12]. These discrepancies indicate that using a simple average step length for every pedestrian could result in quick accumulation of error.

Diez et al. [12] 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 [12]. 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. An overview of different step length estimation methods can be found in Table 2-2.

Table with different step length methods

Table 2-2: Different Step Length Methods

Vezocnik and Juric [51] 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. Examples of step length predictors are step frequency and acceleration range within a step. The robustness of the methods was tested by having the smartphone in different carrying modes. This includes front

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 [52] was best. This is an 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 [12]. The model is defined as

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

Here $A_{\max}$ is the largest measured acceleration measured within a step interval, while $A_{\min}$ is the smallest. K is a calibration variable [12, 52]. The model had an average error of 10.64 % for global variables and 3.60 % for personalized [51].

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 [7, 13, 37].
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 [48]. 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 derived beforehand, through the training of the necessary model.

this section is not done yet, and so will need completing

I do not have a comparison of different orientation estimation methods. How do I explain the use of EKF compared to the different possibilities? Should i just add this comparison section?

2-2 Drift Reduction

introduce subject

Cardinal Heading Aided Heading (CHAH)

CHAH assumes that most buildings have a square or rectangular construction and that there are four possible headings that the user is likely to walk in [1]. These are the *cardinal headings*. Results indicate that the position accuracy can be maintained below 5 meters in periods up to 40 minutes of movement when using a foot based INS [1]. This method clearly does not hold when the building does not have this square or rectangular shape, or if the internal structure is not appropriately aligned with the external structure [9].

Topological Map Matching

Topological map matching is a technique in which the indoor environment is represented by a set of links and nodes, respecting the indoor structure of the building [9]. Using this technique the users position estimate is restricted to only lie on a node or link. This constrains the degrees op freedom an estimate can have. Large open spaces present a challenge since user can walk in what ever direction they please [9].

Landmark Detection

Another approach to drift reduction is through the use of landmark detection. Landmarks are locations with a unique footprint detectable in sensor data. When the user visits a landmark and this is detected, the location of the landmark can be used to calibrate the position estimate of the user, hence compensating drift [10]. In addition drift reduction by position comparison, landmarks can also be used in determining the user specific parameters of SHS, such as step length estimation parameters [17, 42].

Depending on the sensors available, different landmarks become detectable. Examples of landmarks are doors and stairs [10, 17, 49], but also unique magnetic footprints [33], and electromagentic signal footprints [17]. Some landmarks can be easily determined beforehand, through the use of available building blueprints [17]. Others can be detected online [20, 21], not requiring a map representation to be available beforehand, using technique such as Simultaneous Localization and Mapping (SLAM).

One form of landmark detection can be done through activity recognition. In some cases this can be a simple approach, for example applying a threshold on orientation change to determine if a corner has been turned or [17, 25]. Hardegger et al. [20] created ActionSLAM, which uses a foot mounted sensor to estimate displacement in combination with location-related actions to compensate any drift buildup. It does not require map information as it used SLAM, to produce a map itteratively. In the case of ActionSLAM, the activity recognition was performed manually by annotating from video recordings. The paper indicates a tracking error between a ground truth and the calculated path to be around 1.3 meters. This research is augmented in Hardegger et al. [21] have recognized the growing trend of wearable technology. They derived a method that used an INS based system in combination with activity recognition through template matching, particle filters, and SLAM to improve both activity recognition and localization. Examples of detected actions are opening a window, picking up a phone and opening a drawer.
A different approach was used by Grzonka et al. [16], who used a body suit composed of multiple IMU sensors to detect the opening and closing of doors as landmarks, also using a SLAM approach for map generation. Torok et al. [49] have developed DREAR, which is a hidden markov model to detect landmarks using a combination of map information and smartphone sensors, both inertial sensor and magnetometer, to detect when a person is sitting, walking, and taking an escalator.
above section can be smoother

I need to refer to the use of particle filters and that for PDR their use is two fold, landmark detection and absolute position estimate

Activity Recognition

Activity recognition is large research field in which methods span a broad range of complexity, from simple thresholding to the use of deep learning techniques such as neural networks and its many variations [30]. Even the subset of activity recognition that use IMU sensors does not decrease the amount of possibilities significantly.

The choice for a particular activity recognition method is often a trade-off between performance and computational complexity [5]. Considering the scope of this thesis, certain

constraints can be applied, such as that the system should be able to function on a smartphone and must use comparable sensors to those found in such devices, thus MEMS IMUs. Shoaib et al. [43] have performed an extensive survey that focuses on the online activity recognition solely using mobile phone sensors and onboard processing. Other metrics such as resource consumption were also presented. 30 papers were found to meet the requirements of the review. Within the review they indicate that most commonly used classification methods include Decision Tree, support vector machine (SVM), K-nearest neighbor (KNN) and Naive Bayes, in descending order. A third of the papers were found to use decision trees. All but 6 of the papers had the training process occurring offline. The classification process takes the method generated offline, and uses it online to classify new activities. The authors refrain from listing any performance measures, such as accuracy or precision, since no direct comparison between the different methods is made. Ahmad et al. [2] specifically focuses on seeing whether smartphone activity recognition techniques are also applicable to smartwatches, and what parameters, including classifiers, work best. Activities to recognize included walking, up-stairs, down-stairs, running, and jogging. Their results indicated that Decision Trees, SVM and KNN had around 90 percent accuracy a minor difference. Shoaib et al. [44] show that combining information from both a smartwatch and smartphone, complex human activity recognition is improved compared to when only a smart watch is used.

2-3 Proof of Concept System Overview

Summarizing the information and the decision made so far, the overall proof of concept can be found in Figure 2-3. further explanation

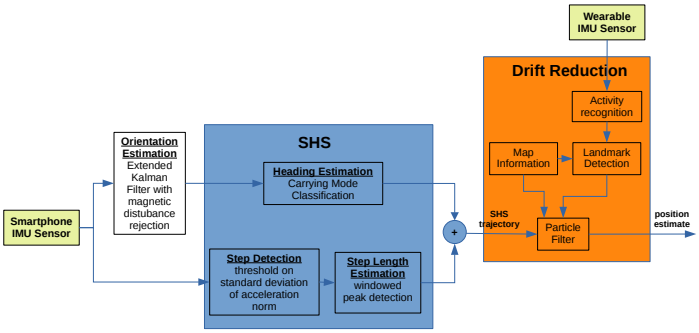


Figure 2-3: Overview of proof of concept SHS-PF with activity recognition from wearable device

Chapter 3

Method

In the previous chapter different indoor localization methods were introduced. From this information an initial system design was destilled. The details of the different components can be found in the following sections.

3-1 Step Detection and Step Length Estimation

introductory text

Step Detection

The method used for step detection is similar to the one presented by [41], introduced in Section 2-1-2. An overview of the different components of this windowed peak detection method is shown in Figure 3-1 . To optimize the algorithm, different combinations of stage components were compared in [41]. A custom made, electronic ground truth device was worn by test subjects for easy comparison with the different parameter settings. 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.

This thesis uses the method and parameters found by Salvi et al. [41] as basis for step detection. The eventually implemented method differs slightly 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 [4]. The values used for the different parameters are based on those found by [41]. 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.

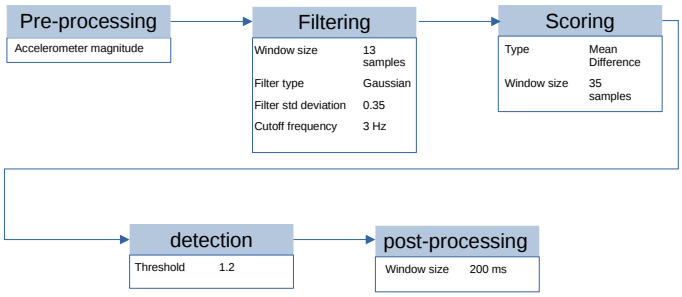


Figure 3-1: Windowed peak detection components and parameters used by [41]

2. **Walk detection**
   Since a person is not walking continuously, the regions in which it does occur need to be indicated. [4] 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 $0.6m/s^2$ 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 3-2.

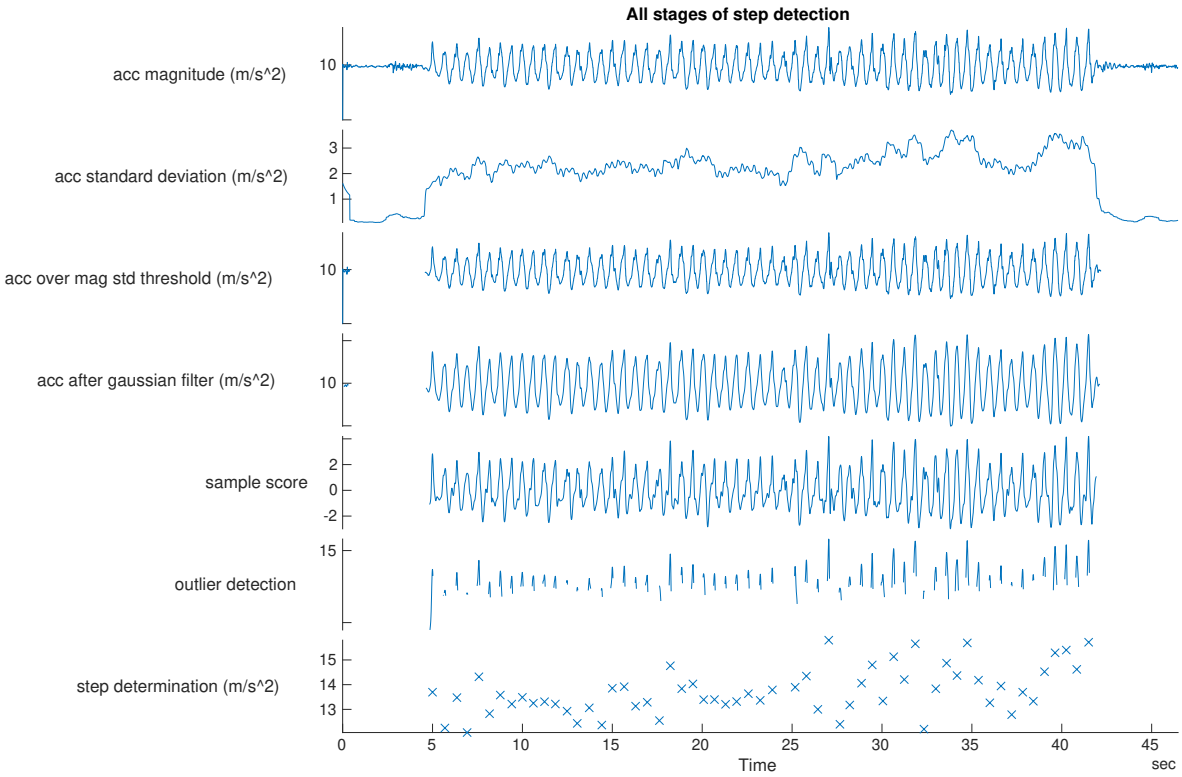


Figure 3-2: All stages of step detection from recorded accelerometer data, with the x's in the last graph indicating the steps taken

Step Length Estimation

For step length estimation the method outlined as working best by [51] will be used. This is (**??**) in Section 2-1-2. Within this method there is a tuneable parameter that needs to be estimate in order for the method to be applied. One way of doing this is by using the data that [51] have made available online. Running this data through the above outlined step detection, the result can be used for parameter estimation.

this section is pretty small at the moment, should it require a heading? What else to add?

3-2 Orientation Estimation

introductory text

3-2-1 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 [29]. 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 [29]. 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.

3-2-2 Orientation Parametrization

While position is often represented by a point in a 3D orthogonal axis frame, orientation has different parametrizations, each able to map to one another. This chapter will introduce rotation matrices (R) and quaternions (q), which will be used in the following section to determine orientation from IMU sensor data.

Rotation Matrix

Rotation matrices $R \in \mathbb{R}^{3\times3}$ 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:

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

Quaternion

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 [23], 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:

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

3-2-3 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 [29]

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

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

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

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

Here (3-10a) is the motion model and (**??**) 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 (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^{nb}_t 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 (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 [29]. 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.

3-2-4 Calibration

For proper orientation estimation, sensor calibration is required.
elaboration need to introduce the subject

Gyroscope Calibration

Gyroscope measurements from MEMS IMUs are generally offset by a bias (o_ω) and influenced by noise ($e_{\omega,t}$) 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 (3-10d). The gyroscope bias is slowly time varying but for relatively short experiments can be assumed to be constant [28]. 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

For magnetometer and accelerometer calibration the method outlined by Kok and Schon [28] 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 [28] 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 [28]

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

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 [15] and YALMIP [31]. 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

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

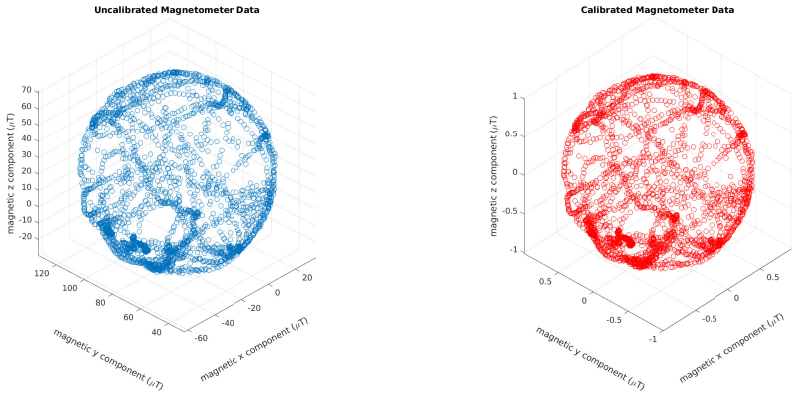
$$\widetilde{D}_{\mu 0}^\top \widetilde{D}_0 = \beta \widehat{A}^{-1} \tag{3-21b}$$

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

An example of this calibration can be found in Figure 3-3 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 3-3: Magnetometer calibration for smart MEMS IMU in one location

3-2-5 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 [50]. 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 (3-10) can be used. As stated in Section 3-2-3, 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

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

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

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 (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

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

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 [29] and [38], 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 3-4, 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.

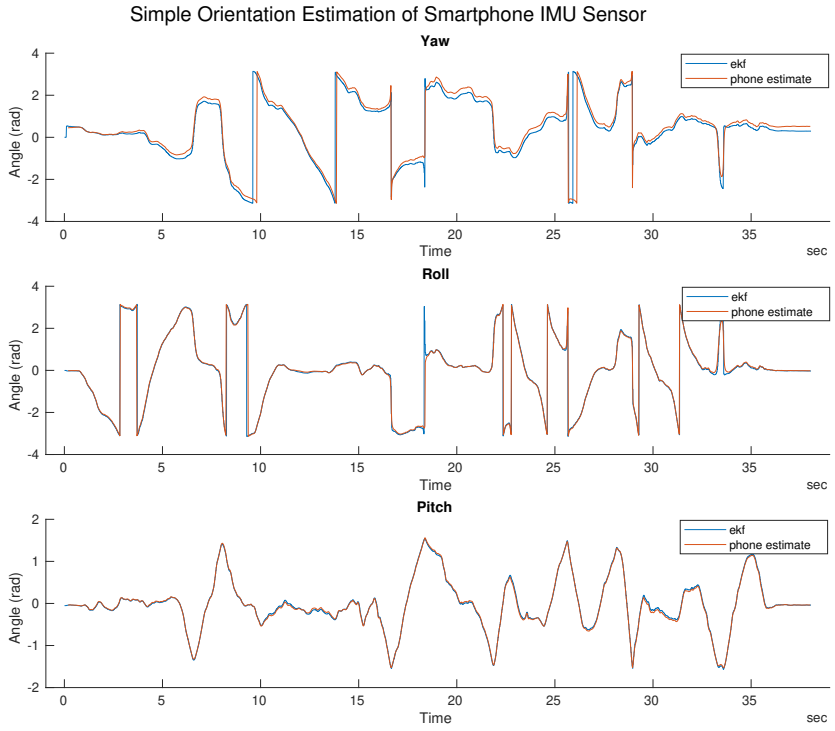


Figure 3-4: Simple orientation estimation comparison between own EKF and android orientation estimation

what other metrics are important to show to proof the performance of the EKF?

3-3 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. 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 [18, 27]. 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 [53].
The particle filter is known to work well in three dimensional state space [18], 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 weights of each particle into account a prior estimate can be made.
Measurement updates are used to calibrate the particles throughout their trajectory. These measurements may come from different sources, including map information and landmark detection. When a measurement occurs the particle cloud is compared with the measurements, adjusting the individual weights accordingly. 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 each particle is $1/N$, where N is the total number of particles. Once initialized the following four steps are performed recursively [53, 54]:

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

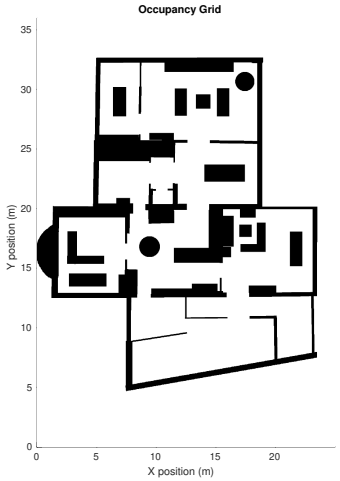
$$c_k = \sum_{i=1}^{N} w_{k-1}^i p\left(y_k|x_k^i\right) \tag{3-30b}$$

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. With a map, the position of physical structures can be compared with the trajectory of particles as a measurement update. 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 3-5a.

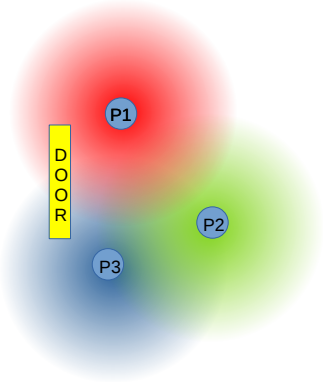
A map can also be used to compare particle position with landmark position of detectable activities. For example, maps indicate where doors are located. If a door opening activity can be detected a conditional probability can be used to determine the weighting of each particle. This conditional probability 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, defined beforehand. This is shown visually by Figure 3-5b. 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. This represent that the closer a particle is to a detected landmark, in this case a door, the likelier the position estimate is around there as well. This is reflected in the weighting that the particle receives. 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.

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 [18]. There are different ways of defining the eventual position estimate. The maximum a posteriori (MAP) estimate picks the particle with the highest weight from the posterior probability function [40], while the minimum mean square error (MMSE) estimate calculates a weighted mean [18, 40] 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



(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 3-5: Map information for measurement updates in the particle filter

filter. There a multiple methods that can be used for resampling including multinomial, stratified resampling, systematic resampling, and residual resampling [24]. Hol et al. [24] 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 [18]. Resampling should therefore occur only when it is needed. Gustafsson [19] proposes the use of *effective number of samples* to trigger a resampling. This can be calculated as

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

where

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

The resampling condition can then be defined as $N_{eff} < N_{th}$ [19], 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 (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, found through the orientation estimation in Section 3-2-5.

3-4 Activity Recognition

section overview:

- Many different type of activity recognition methods, from simple to complex.
- Depends on what needs to be detected. For activities with a clear time based foot print, simple method can be good enough. It is possible to construct own decision tree.
- Figure 3-6 shows ground truth first door interaction and the accelerometer traces from a smartwatch and a standstill detection from SHS.
- standstill detection from SHS is made by determining when the period buildup between steps exceeds over 3 seconds.
- A simple manually constructed decision tree can be used to indicate door interaction. Using both smartwatch data and shs standstill detection the system can distinguish between standing still and interaction with a door.

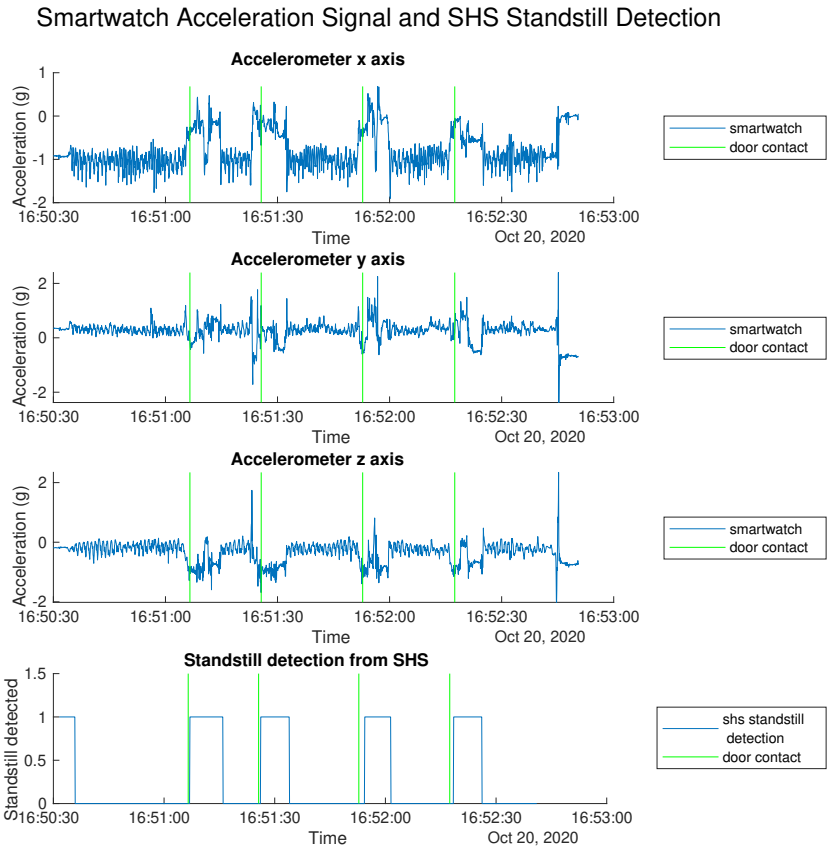


Figure 3-6: Smartwatch accelerometer data and SHS stand still detection

I have more data on smartwatch accelerometer also from different people. I can use that data to determine the threshold on accelerometer data that would work on different people. Do you think that this is a good idea?

Chapter 4

Results

4-1 Step Detection

Salvi et al. [41] 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. The ground truth device consisted of hardware attached to the feet of the test subjects that could exactly measure when the foot came into contact with the floor, hence when a step is taken. The use of this ground truth allows for comparison between the method devised but also with any eventual future techniques.
The raw data of the validation set consists of sampling time, three raw 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 contains data 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. [41] 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 step detection method outlined in Section 3-1 was applied to the validation data of the researchers, allowing for direct comparison between the ground truth and the solution of the researchers. The parameters used for the process coincide with those found by the researchers, an overview of which can be found in Table 4-1.

Figure 4-2 shows how the method, referred to as 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. The results indicate that the method devised performs overall similarly to the method of the researchers. In some cases

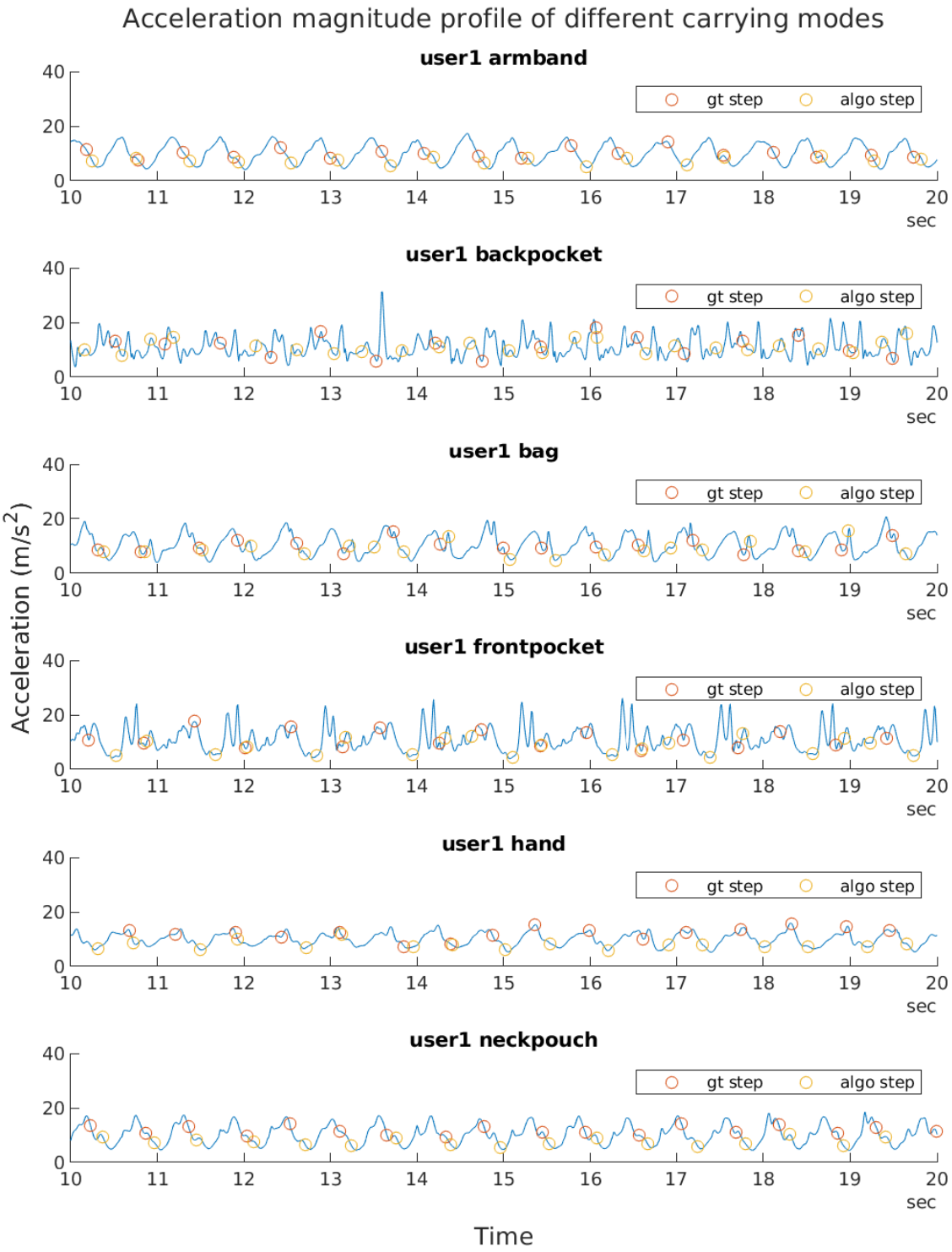
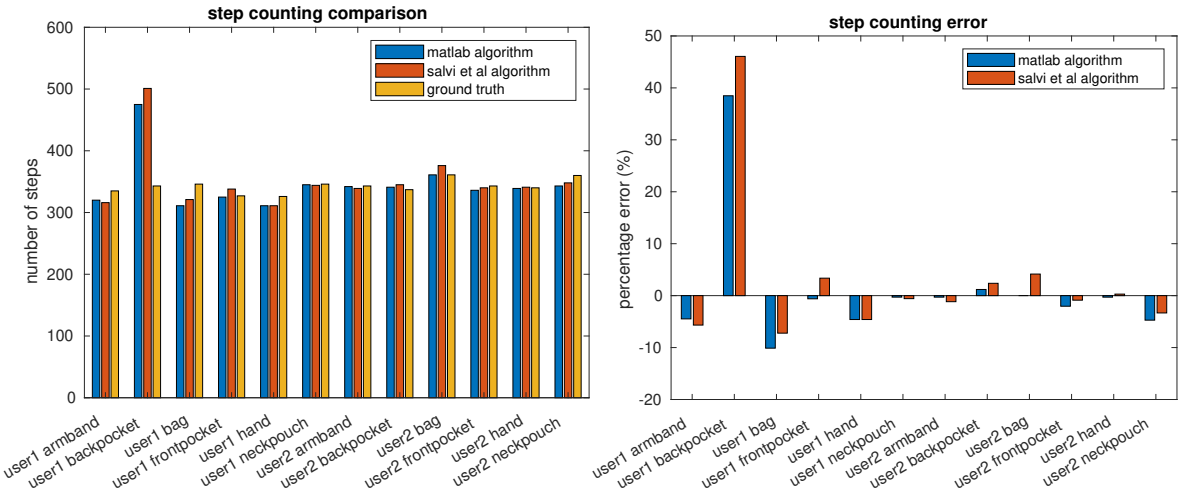


Figure 4-1: Extract of validation dataset from Salvi et al. [41], indicating ground truth steps and step detected by their algorithm.

Stage	Parameters	Value
Filtering	Window size	M = 13
	Filter type	Gaussian
	Filter SD	0.35
Scoring	Type	Mean Difference
	Window size	$N = 35$
Detection	Threshold	1.2
Post-Processing	t_{window}	200ms

Table 4-1: Parameters used

the error is slightly larger, as with user1 bag, while in others it is smaller as with user 2 bag. It is also apparent that with the case of user1 backpocket that there is a large percentage error for both cases. The researchers attribute this to the pocket being loose and allowing the phone to rebound when a step was taken. This would introduce nefarious components in the accelerometer signal, leading to false positives. Brajdic and Harle [4] encountered similar problems with this carrying position, hypothesizing that the relaxing of the gluteus maximus during locomotion could influence the acceleration trace.



(a) Absolute number of steps counted.

(b) Percentage error from ground truth number of steps.

Figure 4-2: Comparison between Salvi et al. [41] 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. Here the backpocket show similar error as with the validation data, further supporting that the method does not work reliably in this carrying position.

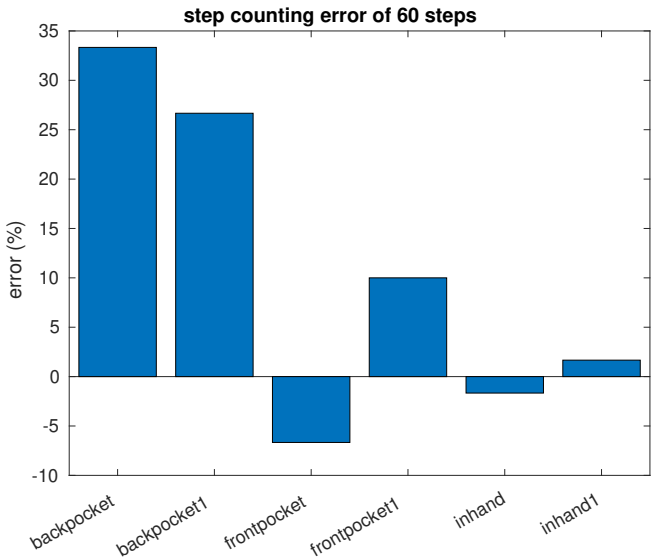


Figure 4-3: Original data step counting

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. [41] algorithm and the algorithm from Section 3-1 perform in this respect, and approach to 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 with the highest amount of true positives for each user and carrying mode can be found in Figure 4-4.

discuss what the results indicate

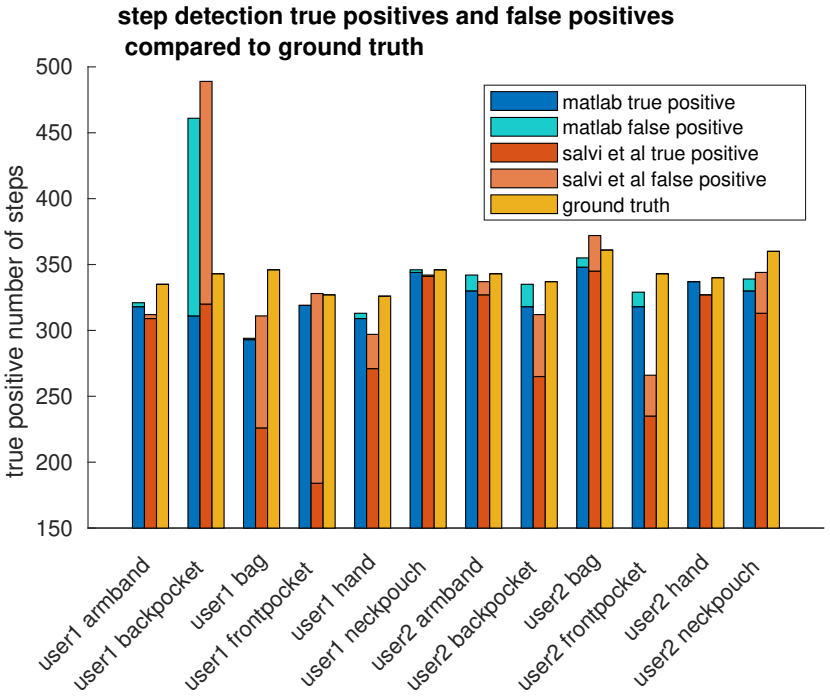


Figure 4-4: Comparison between false positives and true positives for Salvi et al. [41] step detection algorithm, Matlab algorithm and ground truth for different carrying modes.

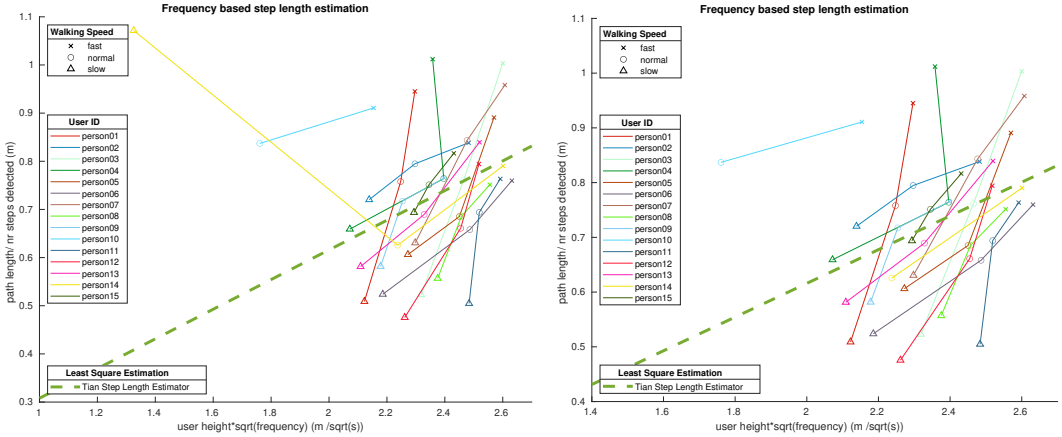
4-2 Step Length Estimation

Vezocnik and Juric [51] 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 [51] 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 (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 [51], and can therefore be used for analysis. The results are shown

in Figure 4-5. Here the parameters for (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.

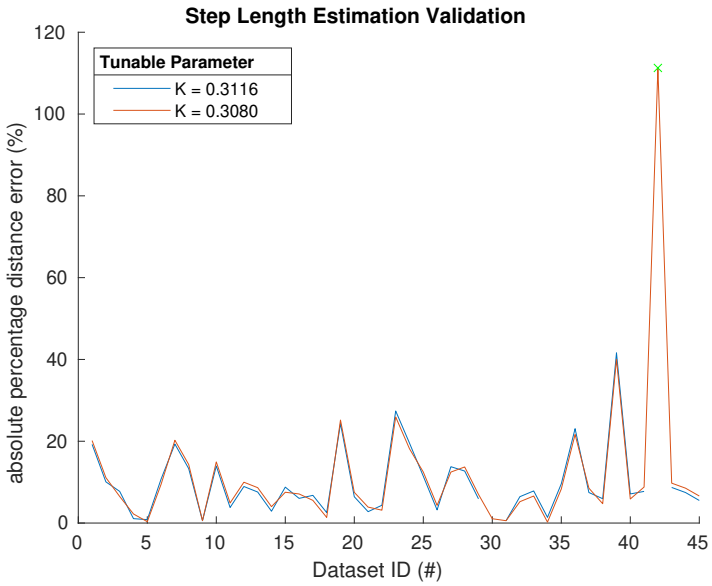


Figure 4-6: Step length estimation using validation dataset

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 [51], 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. The difference in performance is likely to the difference in step detection, which the research do specify exactly.

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.

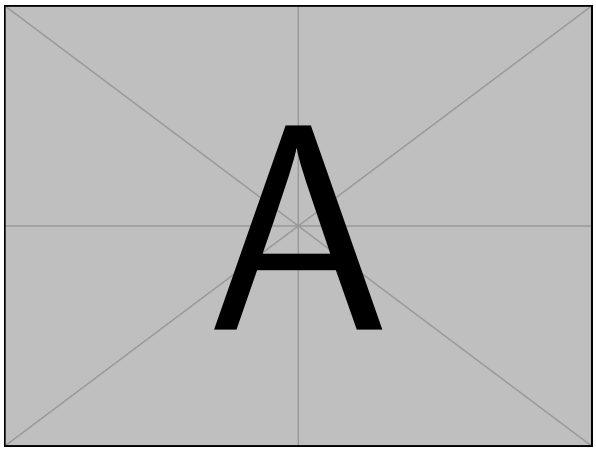


Figure 4-7: Original data step length estimation

still need to add comment on results of test with original data and see how it compares with the data from [51]... The results so far show that general parameters are not good. In the experiments indoor

I think that step length estimation is one of the largest error sources ATM. Is it work while switching to a different technique? Vezocnik and Juric [51] also indicate on that is good for personalized parameters ....? I think that this could be fixed quickly

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

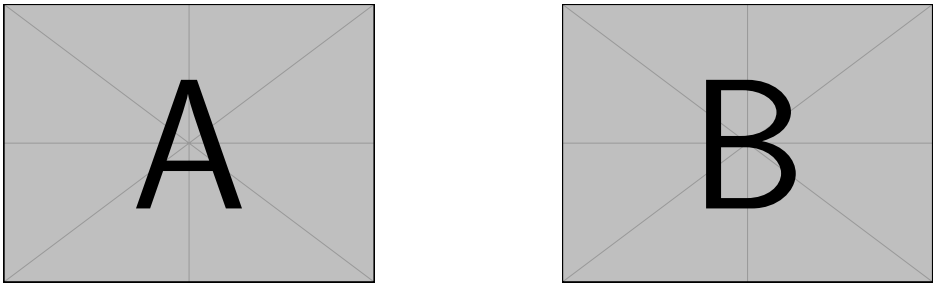
(c) Particle Filter map result

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 experiment 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 3-2-5 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

These two examples indicate that the two estimation techniques have a similar performance. There is no reason for a direct comparison between the orientation estimation outline in Section 3-2-5 and the attitude estimation of the, since this will be relative to each other and not to the ground truth which is not available.

A rudimentary comparison could be made using the ground truth of the experiment, where the assumption is made that the angle of the displacement vector can be considered the orientation of the phone. The different orientations can be compared by showing the difference in angle for the initial orientation. An example of this can be found in Figure 4-10, also showing the orientation estimations of the phone and the EKF method.

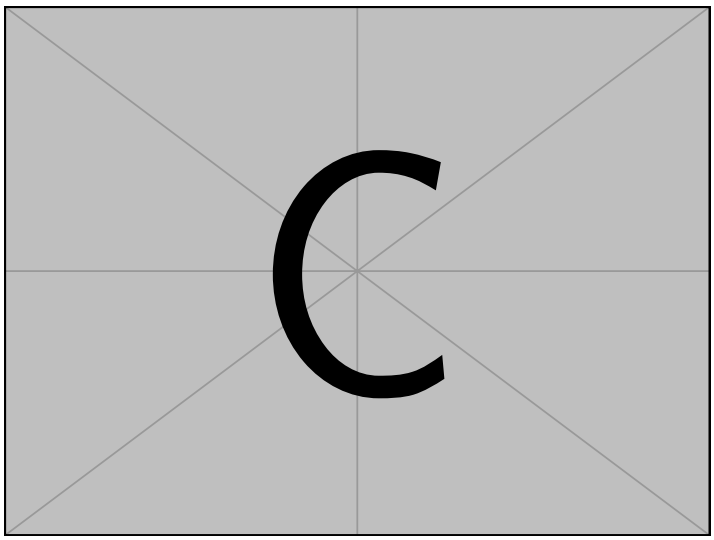


Figure 4-10: Orientation distilled from ground truth compared to orientation estimation of android system and EKF

The error between each trial and the orientation distilled from the ground truth trajectory can be found in Figure 4-11.

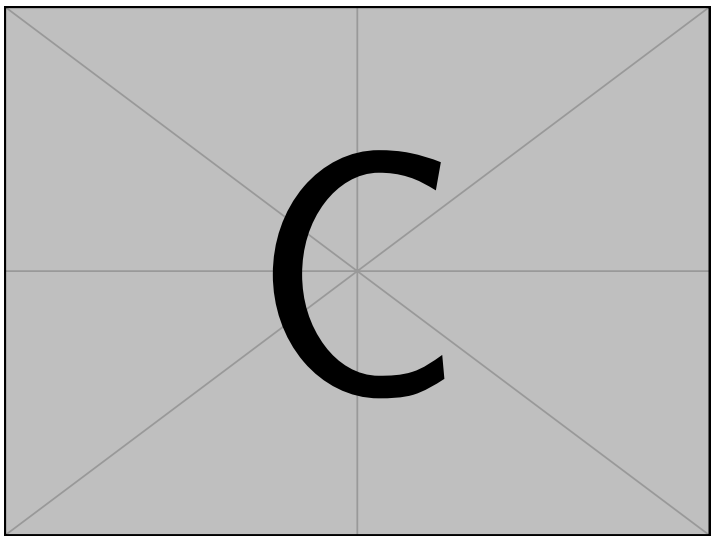


Figure 4-11: Error between ground truth orientation and android system and EKF

As can be seen this plot has not been generated yet. Depending on the outcome a comparison can be made over all trials to determine performance of the different methods. What do you

think, is this a valid way of comparing orientation estimate?

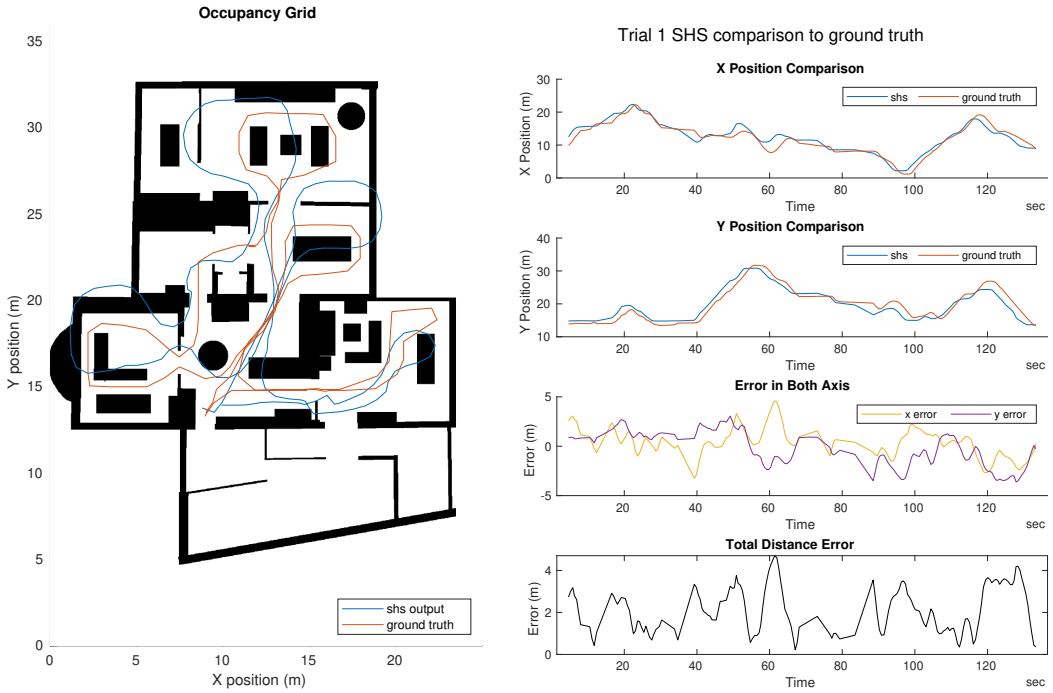
4-3-2 Step Length Estimation

is it worthwhile to compare step length estimation with the video ground truth generated?

4-3-3 Step and Heading System

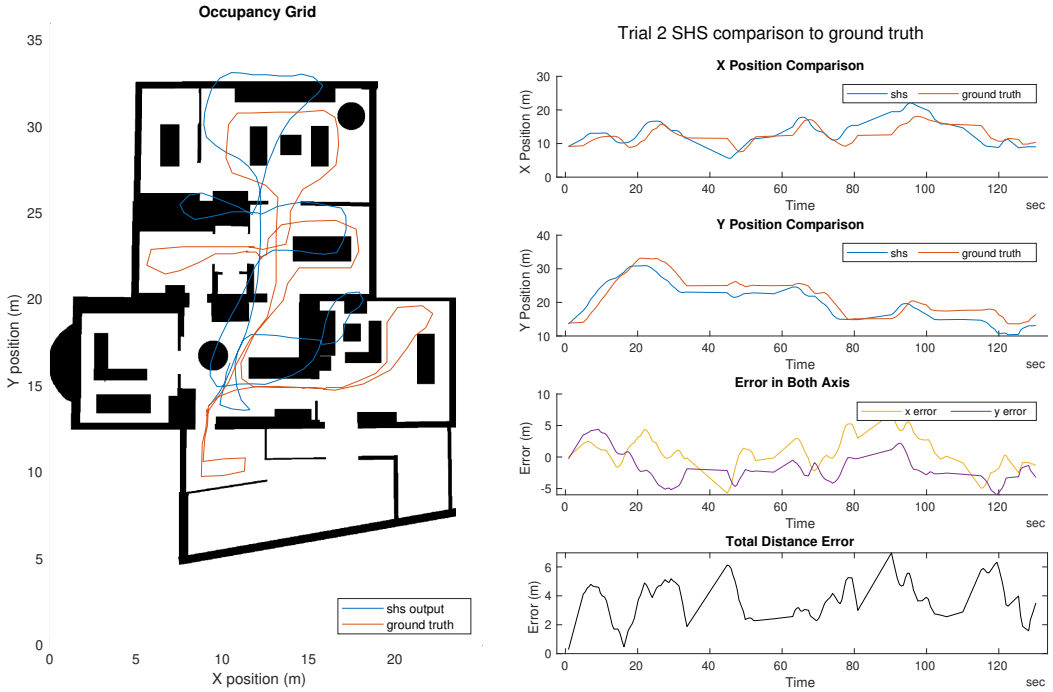
trial naming at this point is not consistent, also not in the plots ... needs to be changed

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 complete system performance. 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 section. Two trials with their trajectory projected onto the map and side by side comparison with the ground truth can be found in Figure 4-12 and Figure 4-13. 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. Since this heuristic rotating is only used to give an indication of the shs output compared to the ground truth, it is not possible to compare it quantitatively with the ground truth, since it is not a standalone solution for indoor localization.



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

Figure 4-12: Qualitative SHS comparison of trial 1 with ground truth



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

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

4-3-4 Activity Recognition

- calculate the difference between the ground truth start of door interaction and the begin of the simple decision tree method outlined in the thesis

What do you think is also important to show about activity recognition

4-3-5 Particle Filter

Using the output of the SHS, the method outlined in Section 3-3 for the particle filter, and the map constructed for the indoor experiments, the indoor localization particle filter could be constructed. For initialization a known location on the map was available, around which the particles were distributed with a Bayesian profile. The output of the SHS was used to propagate the particles, with the map information removing particles that collided with the walls.

Determining the estimate

Within Section 3-3 there are two approaches to determining the estimate. These are the maximum a posteriori (MAP) and the minimum mean square error (MMSE). While running

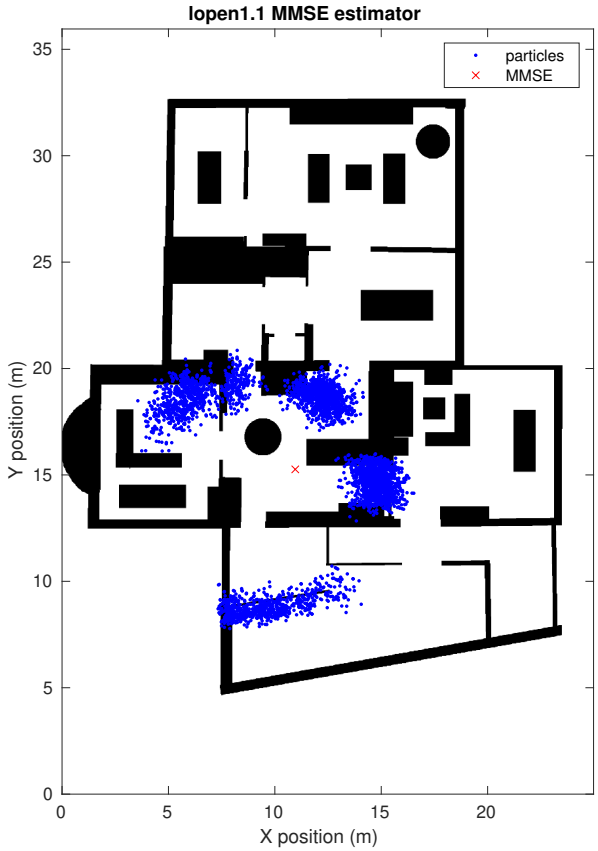


Figure 4-14

the SHS particle filter method defined in this report, both forms of estimation proofed inadequate. First, MMSE does not work well in situations in which there are multiple equal likely hypotheses. In such a case it will take the position in between the different hypothesis as the position estimate. A good example of this can be found in **??**

Another approach was used in which the particles that survived the whole SHS trajectory made during the experiment were used. By recording the ancestors of these particles, the whole trajectory could be calculated by backtrack [quote needed?].

- For this section I want to show MAP, RMSE and Ancestor Tracking (AT) on an example particle filter output to show why AT is better than th other two.

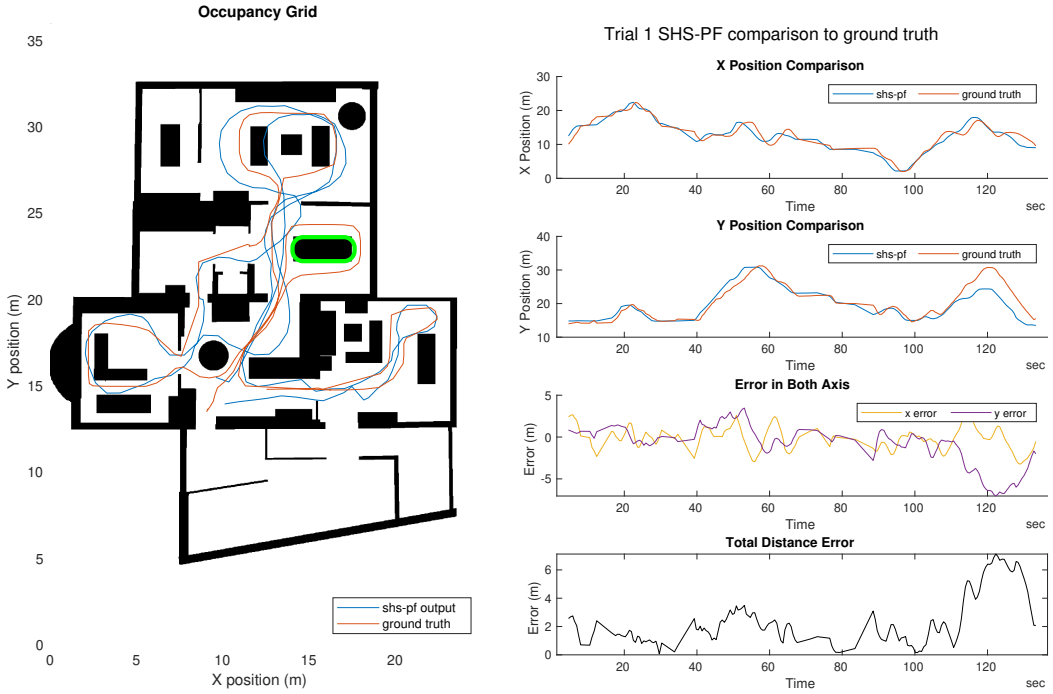
Do you think that I need a quantitative comparison between the different position estimate methods?

With activity

During the experiment the possibility existed of recording at timestamp to indicate the beginning of interaction with a door within the indoor environment. This could be used as a ground

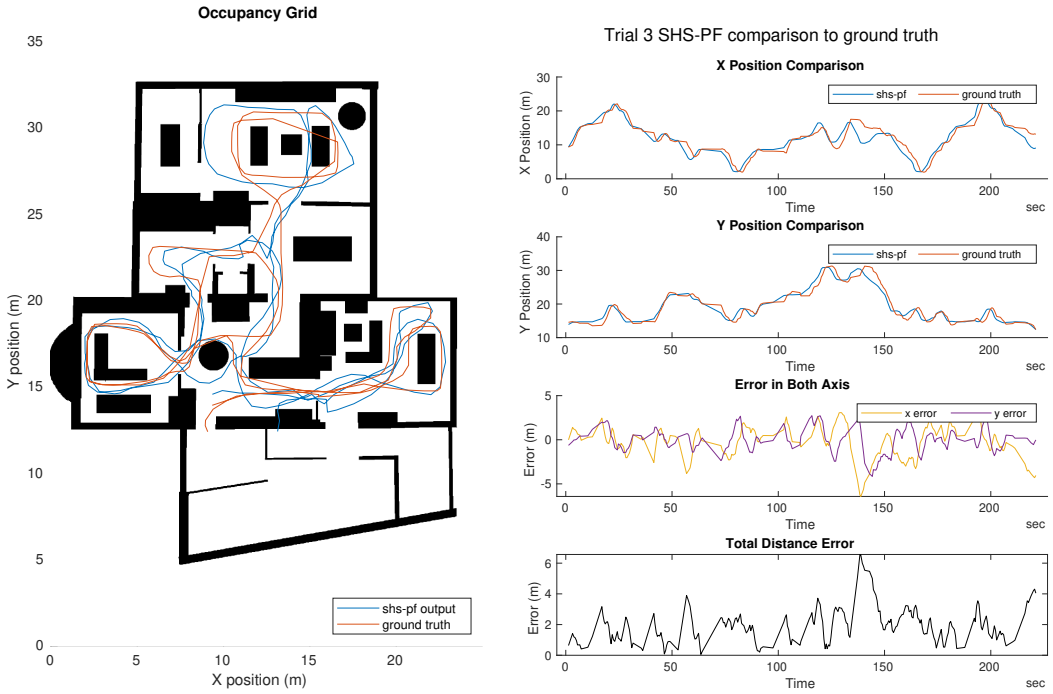
truth comparison for eventual activity recognition, but also as a way of bypassing the activity recognition. At the different timestamps recorded, a particle filter measurement update can be initiated. Using this method the potential affect of activity recognition can be evaluated, by comparing the use of the timestamps with when they are not used. Two examples can be found in Figure 4-17 and Figure 4-16 which show the ground truth generated from video and the output of the particle filter using their respective SHS trajectories as input.

the results here are only using the "ground truth" door detection method, but can be replaced by the actual activity recognition algorithm. I suspect that the results will not be very different since from AR testing, door activity is detected well using the method in Section 3-4



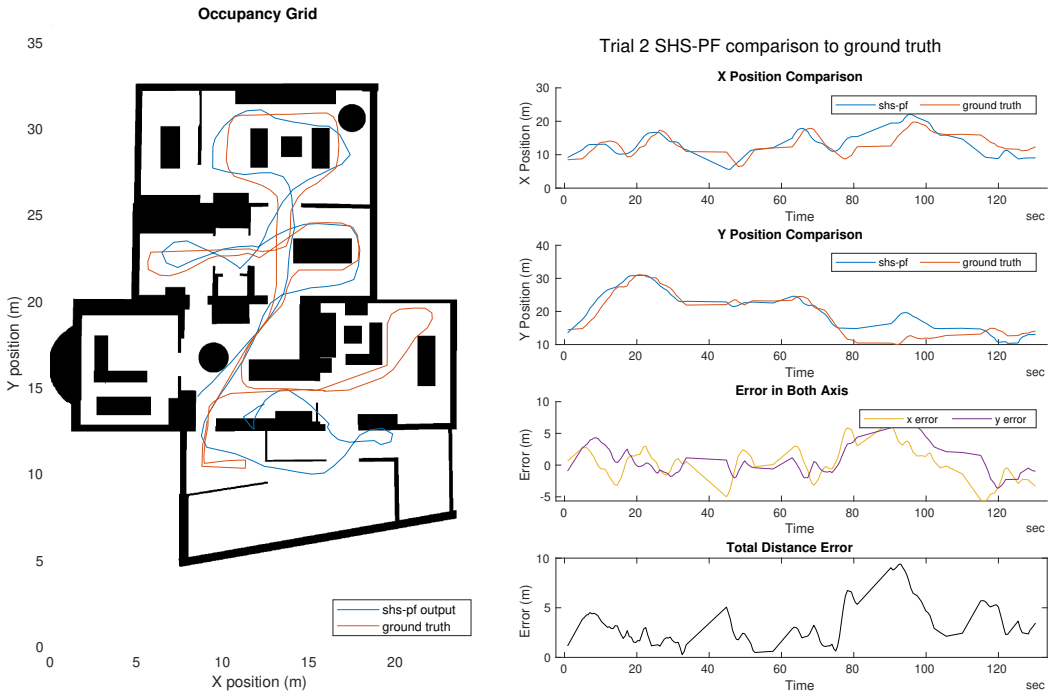
(a) trajectory comparison **(b)** axis comparison

Figure 4-15: SHS-PF comparison of trial 1 with ground truth



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

Figure 4-16: SHS-PF comparison of trial 3 with ground truth



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

Figure 4-17: SHS-PF comparison of trial 3 with ground truth

Running the particle filters with the same settings for lopen1.1-1.3, with 5 iterations each, generated the following results. The individual trajectories can be found in Appendix B-3.

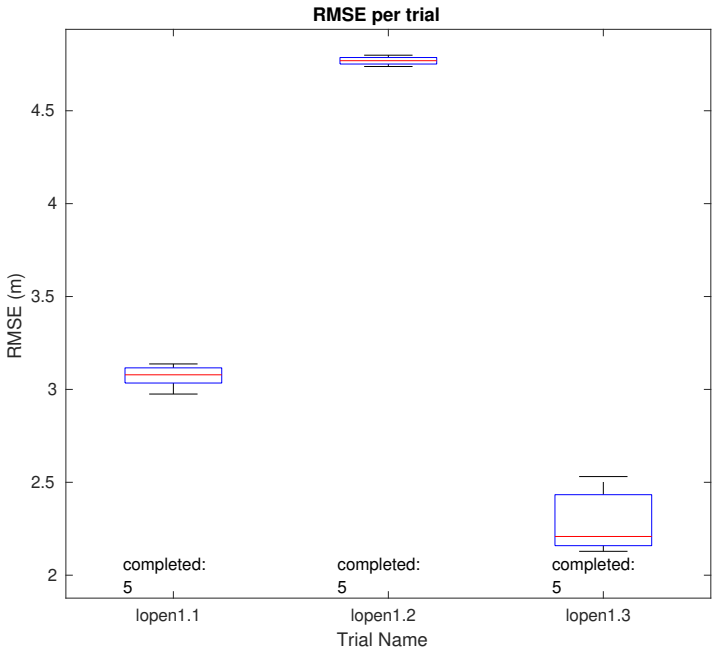


Figure 4-18: Particle Filter position estimation performance with door interaction measurment update, 5 itterations per trial. Number under "completed" indicates how many trials had particles surviving for the whole SHS trajectory.

Things to notice:

- All particle filters generate similar results and complete the whole trajectory
- the RMSE of lopen1.2 is quite high. The reason for this can be found in figure Figure 4-17, with the table that is circled in green. For all particle filter cases this object could not be circle, drastically adding to the RMSE of this trial.
- The particle did not perform consistently for lopen1.5 where it either worked well an example of which can be found in Figure 4-16 , or diverged completely. I think this is due to the SHS trajectory input, with a focus on wrong step length estimation. This can be shown in a plot. What do you think?

there are about 4 more trials that can be processed, do you think that this is worthwhile?
I do not yet know how the particle filter will handle false positives. Can I discuss this with you during the meeting?

Without activity

The same SHS trajectory were used, however now door interaction landmarks were not used as measurement update. The results can be found in Figure 4-19.

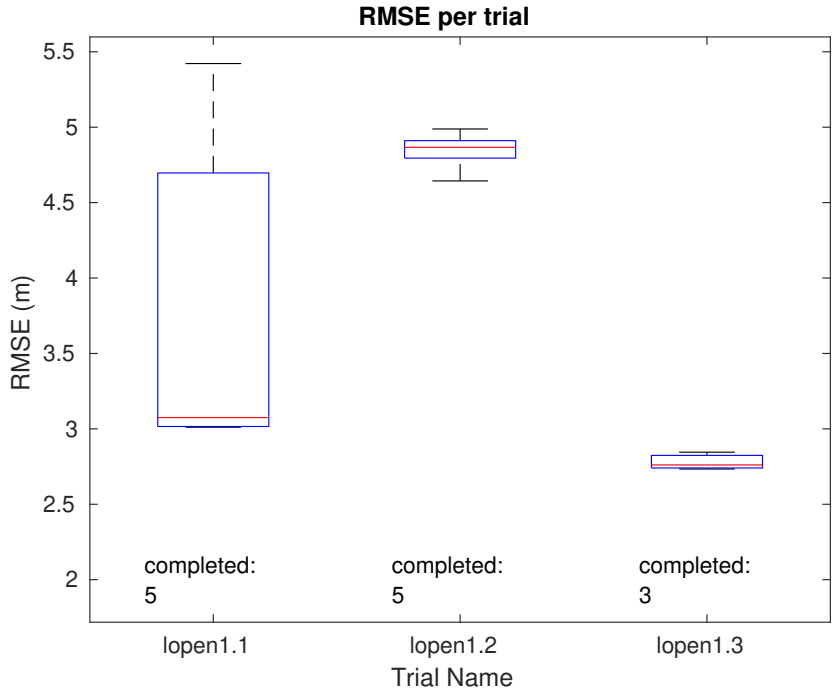


Figure 4-19: Particle Filter position estimation performance without door interaction measurment update, 5 itterations per trial. Number under "completed" indicates how many trials had particles surviving for the whole SHS trajectory.

things to notice

- Trial one has very different RMSE value per run
- Strangely it did not have much effect on lopen1.2
- Although the RMSE of lopen1.3 is similar to the one with door activity two of the 5 trials did not complete the SHS track. This means that all particles.
- So far this does indicate that having activity recognition does help with position estimation within a particle filter

Chapter 5

Discussion

I thought it best to first discuss the discussion during the meeting, since I am not completely sure what I should write here. Furthermore I also think that this is very dependent on the results that I will be showing throughout this thesis, which is also not yet set at the moment

5-1 Answering Research Questions

this will depend on the the eventual research questions that are found

5-2 Future Work

Step detection

- using data from [41] determine if parameters found are also the best parameters for the implementation used in this thesis.
- Step detection is a form of activity recognition, investigate if other machine learning techniques can perform better
- testing was performed walking on a hard floor and it will be necessary to investigate softer surfaces like carpet or grass.
- Determine robustness against more complex movements

Step length estimation

- generate more first hand data, maybe in a motion capture lab
- use [3], a data set that has recently been made to develop smartphone based pedestrian navigation

- make the assumption that step length is constant and determine effect on performance
- when making a turn both feet do not travel the same distance. The inside foot will make a smaller step, while the outer a large one. Determine how turning affects the step length estimation.

Orientation estimation

- use the method of [32] to handle magnetic field disturbances

Step heading estimation

- The testing was limited to one carrying mode, so that the yaw of the device would represent the direction in which the test subject is moving. Step heading estimation will need to be applied in order for the phone to be in other carrying modes.

Particle filter

- Test without furniture or with a different probability density for furniture
- decrease probability close to walls since people do not generally walk close to walls
- different representation of map, instead of grid based more node and edge approach.
- be able to detect when moving to a different floor
- combine other drift reduction methods
- initialize particles over whole map instead of defining a starting point.
- find a way to use less particles in order to be implementable on portable device. Above solutions could already help with this.

Activity Recognition

- Detect more useful activities that can be used as additional particle filter measurement update. For example, climbing stairs for only smartphone sensors, or more intricate activities such as cooking
- implement activity recognition locally on device so little inter device communication is required.

Testing

- different building testing. The current contained may different ways of walking around, while other buildings with more corridor structures are more restrictive. Potentially the SHS-PF technique could have better performance in these settings.
- better ground truth determination for both positioning and orientation. ORB-SLAM2 was tried but could not get it to work with footage from phone.
- online calculation of position on device
- test with different people to determine how user sensitive this is.

Chapter 6

Conclusion

still needs to be written

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:

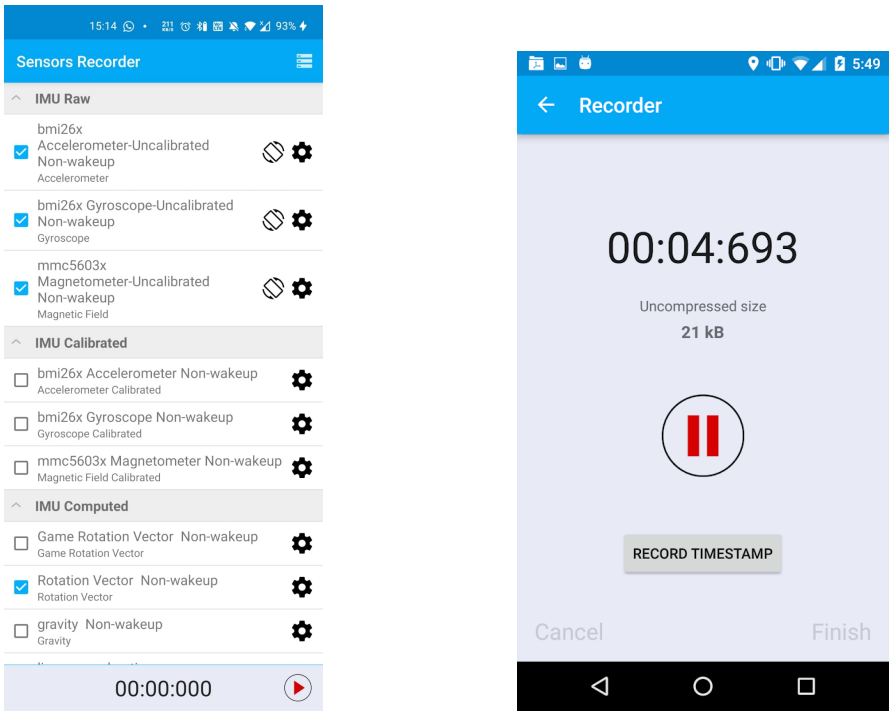
1. Calibrate One Plus Nord with the calibration method outlined above
2. Determine with which hand you will hold the phone, strap the smartwatch to the other hand
3. Ensure that all doors in test location are closed
4. Define start location within test location and record it

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: Indoor experiment android app configuration and button

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.

Appendix B

Indoor Experiment Results

B-1 SHS-PF parameter search

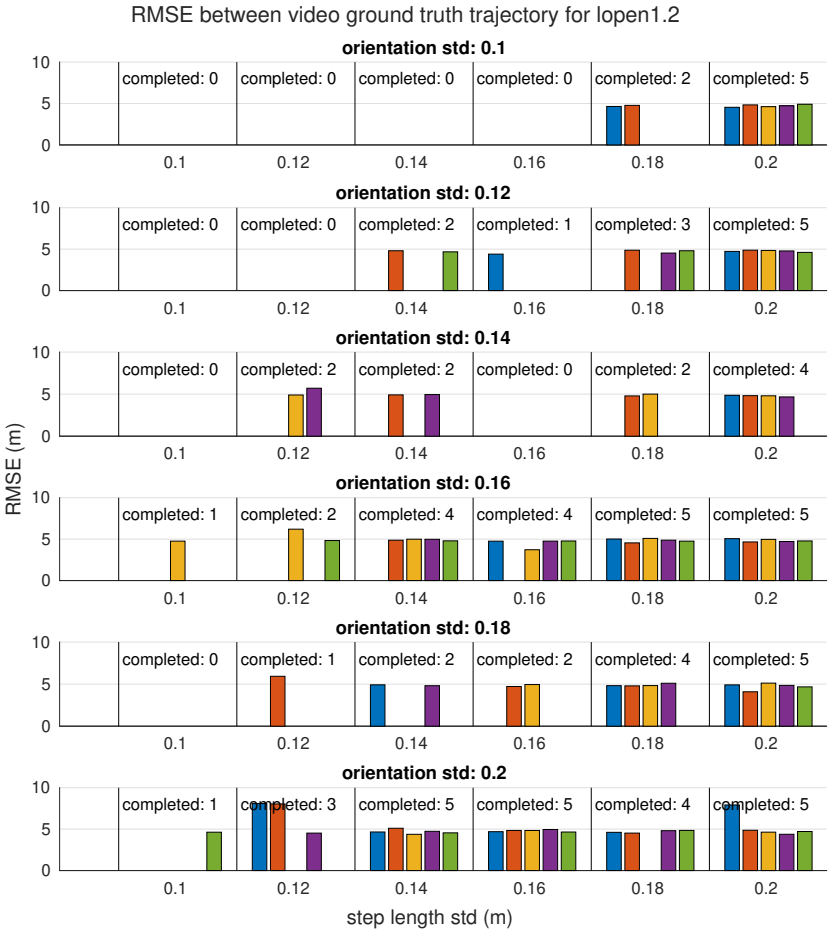


Figure B-1

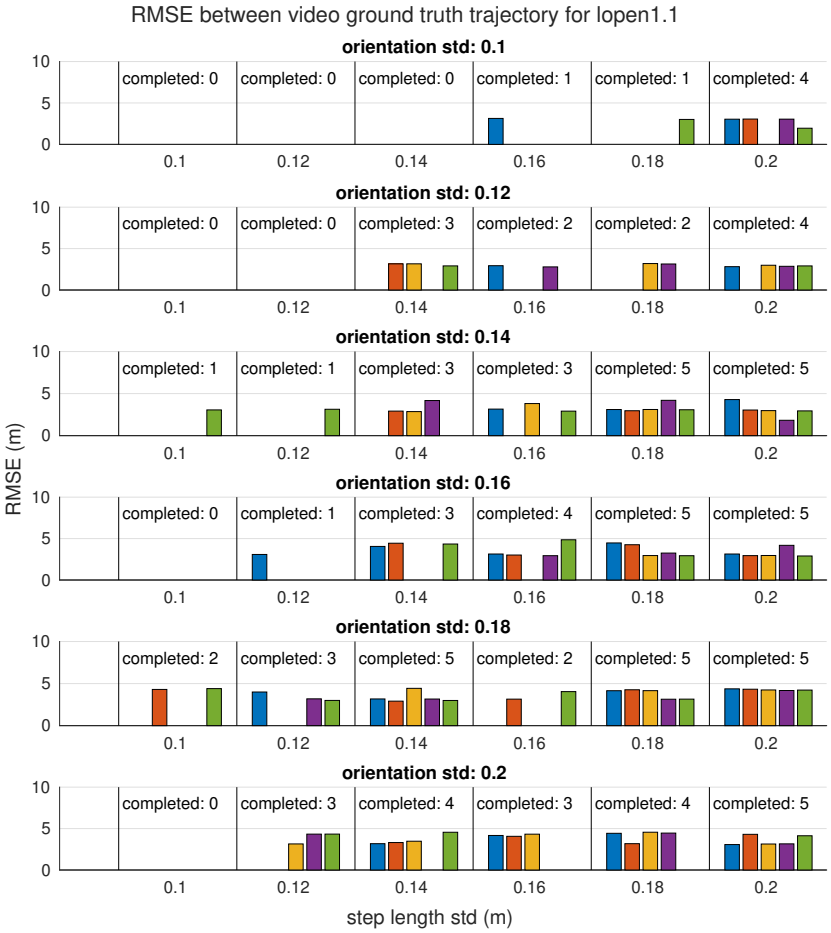

RMSE between video ground truth trajectory for lopen1.1
orientation std: 0.1
completed: 0
completed: 0
completed: 0
completed: 1
completed: 1
completed: 4
orientation std: 0.12
completed: 0
completed: 0
completed: 3
completed: 2
completed: 2
completed: 4
orientation std: 0.14
completed: 1
completed: 1
completed: 3
completed: 3
completed: 5
completed: 5
orientation std: 0.16
completed: 0
completed: 1
completed: 3
completed: 4
completed: 5
completed: 5
orientation std: 0.18
completed: 2
completed: 3
completed: 5
completed: 2
completed: 5
completed: 5
orientation std: 0.2
completed: 0
completed: 3
completed: 4
completed: 3
completed: 4
completed: 5
RMSE (m)
0
5
10
0.1
0.12
0.14
0.16
0.18
0.2
step length std (m)

Figure B-2

B-2 SHS-PF estimator comparison

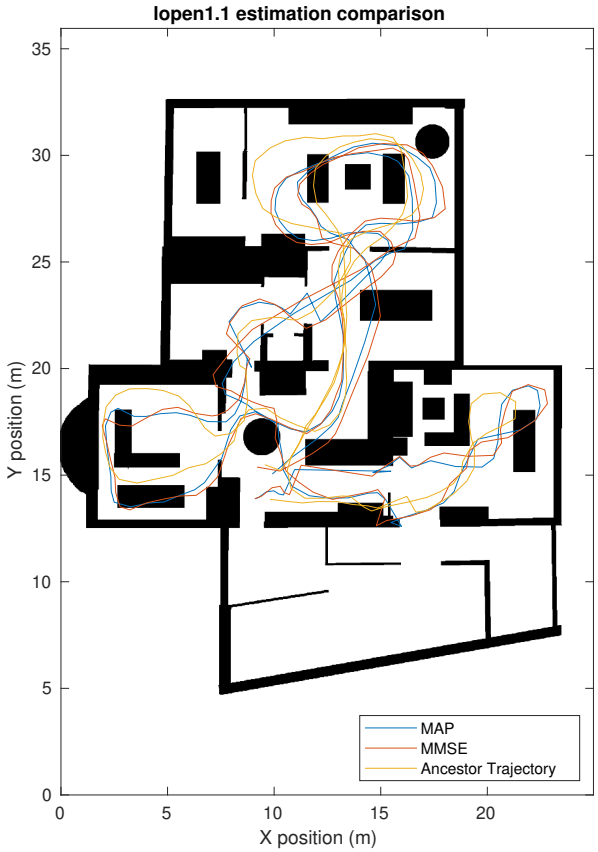


Figure B-3

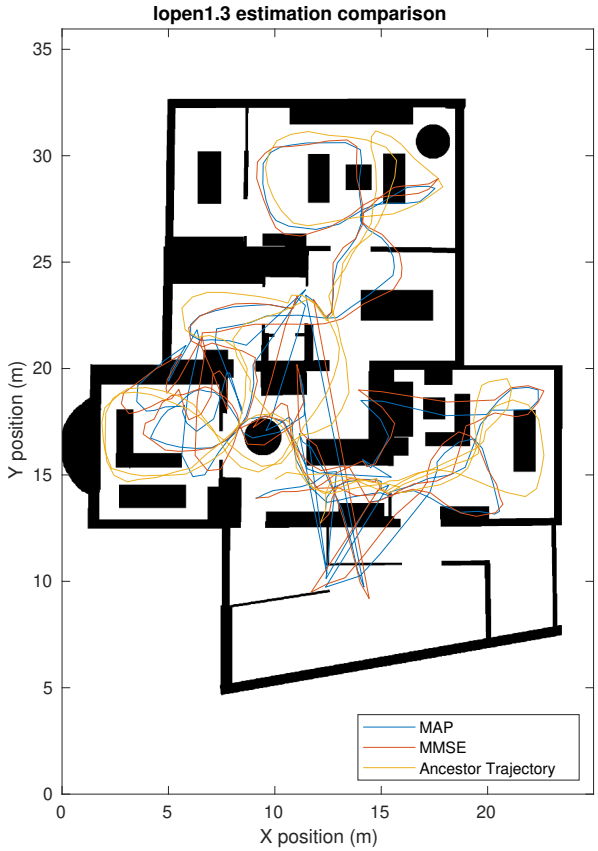
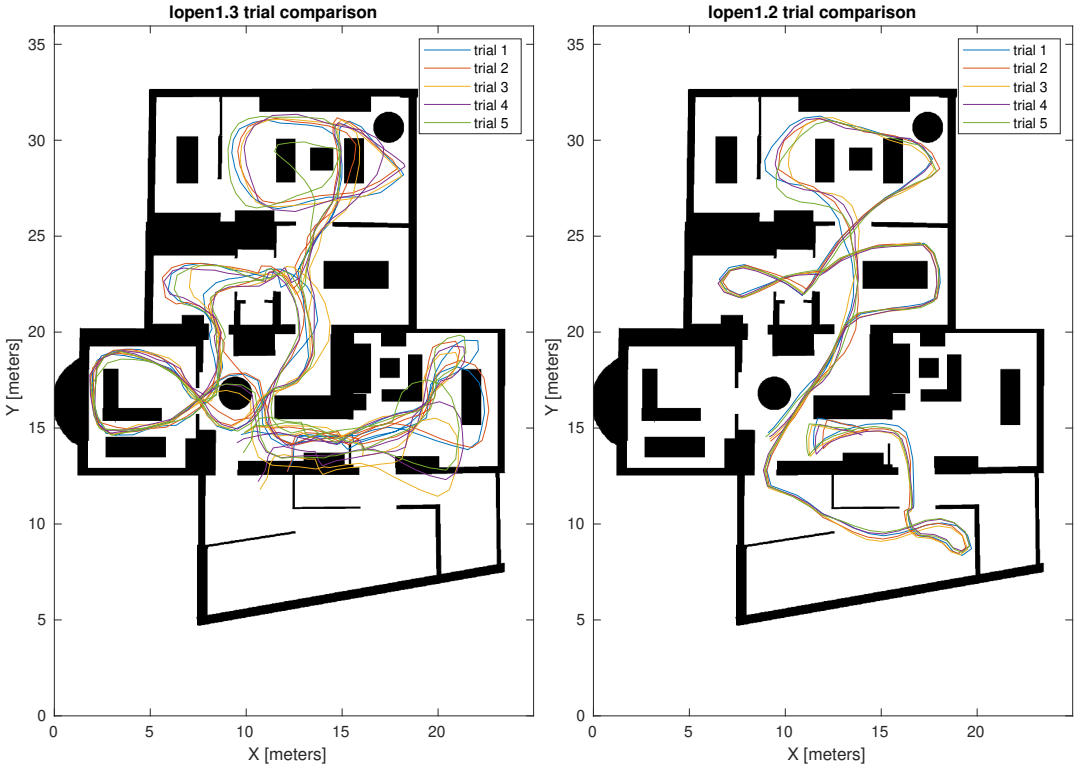

Iopen1.3 estimation comparison
Y position (m)
X position (m)
0
5
10
15
20
25
30
35
MAP
MMSE
Ancestor Trajectory

Figure B-4

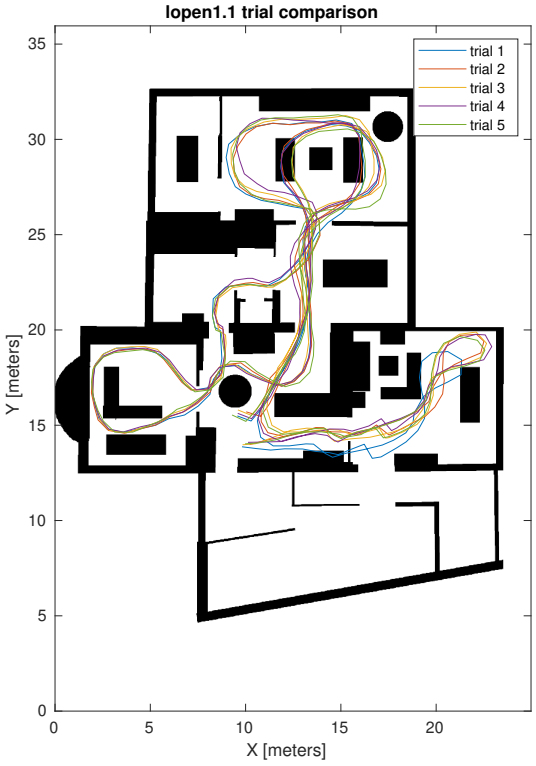

lopen1.2 estimation comparison
Y position (m)
X position (m)
0
5
10
15
20
25
30
35
MAP
MMSE
Ancestor Trajectory

Figure B-5

B-3 SHS-PF performance



(a) **(b)**


lopen1.1 trial comparison
trial 1
trial 2
trial 3
trial 4
trial 5
Y [meters]
X [meters]
0
5
10
15
20
25
30
35

(a)

Bibliography

[1] Khairi Abdulrahim, Chris Hide, Terry Moore, and Chris Hill. Aiding low cost inertial navigation with building heading for pedestrian navigation. *Journal of Navigation*, 64 (2):219–233, 2011. ISSN 03734633. doi: 10.1017/S0373463310000573.

[2] Muhammad Ahmad, Adil Khan, Manuel Mazzara, and Salvatore Distefano. Seeking Optimum System Settings for Physical Activity Recognition on Smartwatches. *Advances in Intelligent Systems and Computing*, 944:220–233, 2020. ISSN 21945365. doi: 10.1007/978-3-030-17798-0_19.

[3] Andrey Bayev, Ivan Chistyakov, Alexey Derevyankin, Ilya Gartseev, Alexey Nikulin, and Mikhail Pikhletsky. RuDaCoP: The dataset for smartphone-based intellectual pedestrian navigation. *2019 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2019*, 2019. doi: 10.1109/IPIN.2019.8911823.

[4] 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.

[5] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014. ISSN 03600300. doi: 10.1145/2499621.

[6] 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.

[7] 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.

[8] Alejandro Correa, Marc Barcelo, Antoni Morell, and Jose Lopez Vicario. A review of pedestrian indoor positioning systems for mass market applications. *Sensors (Switzerland)*, 17(8), 2017. ISSN 14248220. doi: 10.3390/s17081927.

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

[10] Estefania Munoz Diaz and Maria Caamano. Landmark-based online drift compensation algorithm for inertial pedestrian navigation. *2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*, 2017-Janua:1–7, 2017. ISSN 1424-8220. doi: 10.1109/IPIN.2017.8115966.

[11] 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.

[12] 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.

[13] 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.

[14] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6):38–46, 2005.

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

[16] Slawomir Grzonka, Frederic Dijoux, Andreas Karwath, and Wolfram Burgard. Mapping indoor environments based on human activity. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 476–481, 2010. ISSN 10504729. doi: 10.1109/ROBOT.2010.5509976.

[17] Fuqiang Gu, Xuke Hu, Milad Ramezani, Debaditya Acharya, Kourosh Khoshelham, Shahrokh Valaee, and Jianga Shang. Indoor Localization Improved by Spatial Context—A Survey. *ACM Computing Surveys*, 52(3):1–35, 2019. ISSN 03600300. doi: 10.1145/3322241.

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

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

[20] 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.

[21] Michael Hardegger, Daniel Roggen, Alberto Calatroni, and Gerhard Tröster. S-smart: A unified Bayesian framework for simultaneous semantic mapping, activity recognition, and tracking. *ACM Transactions on Intelligent Systems and Technology*, 7(3), 2016. ISSN 21576912. doi: 10.1145/2824286.

[22] 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.

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

[24] 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.

[25] Robert Jackermeier and Bernd Ludwig. Exploring the limits of PDR-based indoor localisation systems under realistic conditions. *Journal of Location Based Services*, 12(3-4): 231–272, 2018. ISSN 17489733. doi: 10.1080/17489725.2018.1541330.

[26] Yoonhyuk Jung, Seongcheol Kim, and Boreum Choi. Consumer valuation of the wearables: The case of smartwatches. *Computers in Human Behavior*, 63:899–905, 2016.

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

[28] 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.

[29] 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.

[30] Weslllen Sousa Lima, Eduardo Souto, Khalil El-Khatib, Roozbeh Jalali, and Joao Gama. Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors (Switzerland)*, 19(14):14–16, 2019. ISSN 14248220. doi: 10.3390/s19143213.

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

[32] Thibaud Michel, Pierre Genevès, Hassen Fourati, and Nabil Layaïda. Attitude estimation for indoor navigation and augmented reality with smartphones. *Pervasive and Mobile Computing*, 46:96–121, 2018. ISSN 15741192. doi: 10.1016/j.pmcj.2018.03.004.

[33] 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.

[34] 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.

[35] Fredrik Olsson, Manon Kok, Kjartan Halvorsen, and Thomas B. Schon. Accelerometer calibration using sensor fusion with a gyroscope. *IEEE Workshop on Statistical Signal Processing Proceedings*, 2016-Augus:660–664, 2016. doi: 10.1109/SSP.2016.7551836.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] Jianga Shang, Fuqiang Gu, Xuke Hu, and Allison Kealy. APFiLoc: An infrastructure-free indoor localization method fusing smartphone inertial sensors, landmarks and map information. *Sensors (Switzerland)*, 15(10):27251–27272, 2015. ISSN 14248220. doi: 10.3390/s151027251.

[43] Muhammad Shoaib, Stephan Bosch, Hans Scholten, Paul J.M. Havinga, and Ozlem Durmaz Incel. Towards detection of bad habits by fusing smartphone and smartwatch sensors. *2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015*, pages 591–596, 2015. doi: 10.1109/PERCOMW.2015.7134104.

[44] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J.M. Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors (Switzerland)*, 16(4):1–24, 2016. ISSN 14248220. doi: 10.3390/s16040426.

[45] 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.

[46] 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.

[47] Zain Bin Tariq, Dost Muhammad Cheema, Muhammad Zahir Kamran, and Ijaz Haider Naqvi. Non-GPS positioning systems: A survey. *ACM Computing Surveys*, 50(4), 2017. ISSN 15577341. doi: 10.1145/3098207.

[48] 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.

[49] Attila Torok, Andras Nagy, Laszlo Kovats, and Peter Pach. DREAR - Towards infrastructure-free indoor localization via dead-reckoning enhanced with activity recognition. *Proceedings - 2014 8th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST 2014*, pages 106–111, 2014. doi: 10.1109/NGMAST.2014.47.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] Zheng Yang, Chenshu Wu, Zimu Zhou, Hong Kong, X U Wang, Yunhao Liu, Xinglin Zhang, and Xu Wang. Mobility Increases Localizability: A Survey on Wireless Indoor Localization using Inertial Sensors. *ACM Comput*, 33, 2014. doi: 10.1145/0000000.0000000. URL http://doi.acm.org/10.1145/0000000.0000000.

[56] Ning Yu, Xiaohong Zhan, Shengnan Zhao, Yinfeng Wu, and Renjian Feng. A Precise Dead Reckoning Algorithm Based on Bluetooth and Multiple Sensors. *IEEE Internet of Things Journal*, 5(1):336–351, 2018. ISSN 23274662. doi: 10.1109/JIOT.2017.2784386.

[57] 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
MEMS	micro-electromechanical systems
SLAM	Simultaneous Localization and Mapping