Distance estimations by non-linear step length models using accelerometer sensor data from waist-worn smart-phones

Masters Thesis Mathematical Statistics

Lund University
June 16, 2018

Benjamin Drorsén & Daniel Strandberg

Supervisor: Andreas Jakobsson
aj@maths.lth.se

Examinator: Johan Svärd
js@maths.lth.se

Contact:
benjamin.drorsen@gmail.com
JDRStrandberg@gmail.com

Abstract

This thesis examines the possibility of modelling trajectory and estimating distance during a walk using only a smart phone attached to the waist. Due to GPS not functioning indoors, a navigation system without the need of external communication could be the perfect complement for closed spaces. This thesis considers such a system, where the smart phone contains accelerometer and gyroscope sensors for retrieving acceleration and angular rate measurements. Because of measurements errors, there is no straightforward way to estimate the trajectory since double-integration creates a drift. There are different ways to get around this problem, by using zero-velocity updates to reset the system or by using biomechanical models of the human walking patterns. The best approach explored in this thesis is the latter, using non-linear models for estimating step lengths. The step models tend to under- or overestimate and therefore is a scalable constant introduced to handle future distance estimations better. The scalable constants are formed during several tests, using the least-squares method for minimizing the total walked distance errors. The completed models are verified on a validation data set, confirming the method being decent for distance estimation. In conclusion, the distance estimation works properly good for implementation, but since the trajectory can not be reproduced from the non—linear step models, there is a long way left before really contesting the GPS.

Acknowledgements

First, and most importantly, we would like to express our gratitude to our supervisor Prof. Andreas Jakobsson, at the Institution of Mathematical Statistics at LTH, for the continuous support of our work towards this Master's thesis. His motivation and extremely fast email answers 7 days a week has been important to our progress and final result.

Second, the guidance and support from Post doc. Johan Swärd at the Institution of Mathematical Statistics at LTH has been very useful to us. Without his support and open door, this project would have been much harder. No matter if the questions were easy, hard, clever or even stupid, he offered guidance and fine answers, always with smile and sincere interest.

Also, we want to give our gratitude to all personal friends, co-workers from the master thesis-office and LTH personal who have given us support along the way and for letting us use their legs, shoes and bodies to do walking tests. Without them, this would not be possible. A special thanks to Joakim Lübeck for his support in technical issues, in the beginning of our work, when they arose. Knowing that he helped us because he wanted to and not because he had to. For that, we are grateful.

Finally, thanks to friends and family for all the moral support we have received.

Contents

1 Vocabulary

- **ACF** - Auto-Correlation Function
- **COG** - Body Center of Gravity
- **Dead Reckoning** - Position calculated from known starting position and acceleration
- **FC** - Final Contact
- **Gait Cycle** - One step during the walk
- **GPS** - Global Positioning System
- **IC** - Initial Contact
- **IMS** - Inertial Movement Sensors
- **KTH** - Royal Institute of Technology
- **PDR** - Personal Dead Reckoning
- **Stride** - Completing a full cycle for a foot i.e. two steps during a walk
- **ZUPT** - Zero velocity Update

2 Introduction

2.1 Purpose of the project

The system dominating the market and outperforming all others in position tracking today is the Global positioning system (GPS). Although there have been decades of research to complement or substitute GPS, there has been none successful enough to really challenge it [23, 38]. Even though GPS seems hard to beat, there are situations not fit for GPS. Because GPS needs a signal connection to satellites, it struggles when the signal is blocked for example inside buildings. There have been different ways of complementing or solving the indoors position estimation. A solution using WiFi signals have been one of the most prominent, but approaches using Radio Frequency Identification (RFID) and Ultra Wideband (UWB) are also possible candidates. The techniques complementing GPS require external infrastructure development though, making them hard for general distribution. The mentioned methods would work as a complement to GPS, while systems using inertial navigation or bio mechanical step models could substitute GPS completely [38].

The other methods of finding a way of full trajectory estimation in a self-contained (not using any external signal inputs) system have been investigated and analyzed by multiple research groups over the years. A well thought of approach is calculating the distance by directly double-integrate the acceleration measurements from Inertial movement sensors (IMS), which does not work. Due to measurement errors, the double-integration will after some time accumulate a drift, leading to extremely bad position accuracy. There are developed methods to get around this problem, which for example could be resetting the system during stand still [23, 1, 38, 5, 22, 2].

One of the most successful methods of estimating trajectories has been using IMS attached to the foot, both for estimation the foot's movement and for determining when the foot is still. A research group of John-Olof Nilsson, Isaac Skog and Peter Händel at the Royal Institute of Technology (KTH) have presented multiple eminent papers on foot-mounted inertial navigation. The reason for attaching IMS to the foot is to reset the system while the foot is standing still, which gives results of $0.2 - 1\,\%$ position accuracy depending on the trajectory shape [23, 25, 34]. Another approach used by the researches J.C. Alvarez, R.C. González, D. Alvarez and A.M. López are the distance estimation using bio mechanical non-linear step models. By modelling an approximation of a step length, and summing them all calculates the distance, which results in an position accuracy of up to $1 - 2\,\%$. These non-linear step models often need a correction constant, which indicates the simplification in the models themselves. The models also only calculate distance, but there are some ways to estimate the direction as well, by the use of angular rate measurements. Although the acceleration and angular rates measurements combined can estimate the whole trajectory, it is still less successful than a foot-mounted sensor [1, 9, 18, 32].

The purpose of this project is to examine the possibility of reconstructing a trajectory using IMS in smart-phones without the aid of an GPS. There is a clear need of estimating trajectories where it is troublesome for the GPS. For example when firefighters enter a burning building or patients doing walking tests in hospital corridors. To simplify this process, to just use a smart-phone in place of a dedicated IMS, would save time and resources, which can be put to better use.

According to former studies and research, the most successful method uses foot-mounted IMS to recognize when the IMS is stationary [23]. Unfortunately there are no such moments during a walk when the the smart-phone is attached to the front waist. Since the system can not be reset, the better approach is a non-linear step model for estimating the distance [1].

The thesis first approach is to model the trajectory using the foot-mounted device from Oblu, which has its algorithms based on the research of J-O. Nilsson, P. Händel, I. Skoog and A.K. Gupta [23, 26]. Because of the Oblu device not being consistent, the next approach was to estimate the distance using non-linear step models with inspiration from J.C. Alvarez, R.C. González, D. Alvarez and A.M. López

[1]. The thesis ends with an evaluation of how well the algorithm is performing, where the end result are 3.87% position error with a 3.62 percentage point standard deviation at its best.

2.2 Objective

The main objective is to achieve sufficiently good distance estimations. This is done under controlled conditions for healthy persons were the smart phone is attached to a belt pack on the stomach.

- Is it possible to use a smart phone for inertial navigation?
- Can distance be predicted sufficiently good using a smart phone?
- Are there any other relevant applications using mobile phone inertial navigation?

2.3 Limitations

This thesis will only focus on achieving the best distance estimations for simpler and controlled conditions.

2.4 Structure

The structure of the thesis will be as follows.

In the next chapter, the background of the work is outlined, providing the essential information needed for understanding the topic.

Then, in chapter 4, the case of trajectory estimation will be handled. The chapter will present the problem, purpose and a method of execution. Following in the chapter's subsections are the theory, implementation and results. These are discussed regarding the contribution and influence to the end result throughout the subsections. Chapter 4 ends with a general discussion in trajectory estimation.

Next, in chapter 5, is the problem of distance estimation introduced, with purpose and the method of execution. The subsections present the theory, algorithms and results. Throughout the subsections will the algorithms results and contribution to the end result be discussed.

Finally, chapter 6 and 7 contains the main discussion and conclusions of the work.

3 Background

To know your location and where you are going has always been of interest to people. Whether it is a car driving, people walking or animals scattering. There are several methods to predict their locations. This chapter introduces the basic knowledge needed for understanding the way trajectories and distances are estimated in this thesis.

3.1 Methods of measuring positioning

There are several different ways and methods to know a traveled path. It can either be directly measured or constructed from calculations. The most commonly used system today is the GPS, but another way would be to use acceleration measurements from an IMS to calculate the traveled distance. The basics of these systems are presented in this section, with most focus on IMS since it is what the project is based upon.

3.1.1 Global Positioning System - GPS

For the last decades, multiple devices have been developed for measuring distance in different ways. The most common system is the GPS which locates a position with the use of the radio waves sent from satellites. The development of the GPS system started during the mid 1960's with the purpose of tracking satellites and nuclear submarines. This lead to the development of the Navigation System with Timing and Ranging (NAVSTAR) during the 1970's. In 1993 had all of the 24 satellites, used in NAVSTAR, been launched into orbit. Ever since then, the GPS has been improved. This has lead to a high performance at pinpointing positions [19].

Although GPS is the best performing system at locating positions, it has its flaws in some aspects. For the GPS to be well-functioning it has to have a decent connection to the satellites, otherwise it performs poorly. Therefore, closed spaces such as indoor positions are hard to determine accurately, because of the lack of communication with the satellites [22].

3.1.2 Inertial Movement Sensors - IMS

There are other methods of determining distances. For example by using ultrasonic, electromagnetic, optical, infrared or inertial sensors and techniques. These are not as developed as the GPS and may lead to worse performance in comparison. The perks with these methods are that they do not depend on communication with satellites and can therefore work in closed spaces. The most common component for measuring distances by acceleration and angular velocity, is the inertial sensor [22].

IMS are commonly used to measure features such as acceleration and angular velocity. The IMS have been around for some time, but only lately have units decreased in size and cost. As IMS are of relatively small size, light weight, and have low power consumption and high performance, they are easily integrated in technical devices such as smart-watches and smart-phones. There are also IMS components uniquely designed for specific purposes, for example devices connected to ones feet or car [23, 35]. The improvement of IMS have lead them to be easily accessible for all people with smart-phones. Instead of having a special component produced for measuring distance at your feet, a smart phone can be used to do the same thing. This is a huge progress since some IMS components can be troublesome to use, while a smart-phone can be able to solve the same problem, and most people have one available [30].

The perks of IMS are that they only need their own system to measure, for example acceleration and angular velocity. They do not need to communicate with satellites, as the GPS does, and can therefore work in closed spaces, for example inside buildings. The main flaw with IMS is that they do not measure perfectly, there are errors in the measurements which will create problems [22].

3.2 Using IMS for estimating dead reckoning

3.2.1 Dead reckoning

As mentioned in section 3.1.1 and 3.1.2, both GPS and the use of IMS have their advantages and disadvantages. In situations when one may not use GPS, IMS can calculate the location of an object, given an initial fix starting point. This concept is called dead reckoning. Personal dead reckoning (PDR) is dead reckoning on humans. Typically, PDR involves motion sensors attached somewhere to a persons body, tracking accelerations and angular velocities. The accelerations can be double integrated to get the displacements and the position. However, as mentioned, a major drawback of such a system is the errors in the measurements. These errors lead to low accuracy in the estimation of the position, and makes the straightforward approach of double integration pointless.

Generally, PDR systems that uses the IMS can be classified into two categories depending on where the sensors are placed: foot - or waist mounted. [33]. If the sensor is placed on the foot, the algorithm takes advantage of the moments when the velocity is zero when the foot is firmly on the ground. Any non-zero acceleration or angular velocity measurements during this phase are all eliminated due to the foot standing still. This algorithm effectively reduces errors for PDR systems. However, if the sensor is not placed on the foot, but instead in a pocket or mounted on belts around the waist (where smart phones are usually placed) the sensor usually do not experience any predictable zero-velocity phase (ZUPT) during a step. As a result of the device placement, the ZUPT error reduction algorithm is rendered infeasible in the waist-worn case. Instead, step-based PDR tracking methods have been proposed for trajectory estimation without ZUPT [15]. An example is [1], where the IMS is placed on the back and in order to calculate the displacement from the initial point, algorithms taking advantage of empirical observations in the biomechanics of the human gait are used. These kind of methods are adopted as an inspiration to this project, as a smart phone is more easily placed on the waist compared to a foot.

3.2.2 Foot-mounted sensors

The first inspiration for this project was from a Swedish open source project called OpenShoe, which originates from KTH. There they have created a complete foot-mounted inertial navigation system, including both hardware and software design. Clever algorithms help to estimate the steps taken by the person wearing the sensor. The steps can then be summed up to give an accurate relative position of the person. Although the placement of the sensor on the foot obviously has advantages compared to other placements, generally, the foot-mounted sensor has some drawbacks when it comes to user friendliness. Factors contributing are the difficulty of its integration and the lack of suitable hardware platforms, making the technology hard to apply for a wider user group. The sensor placement and lack of a high-level statistical interface is the primary difficulty of integration because of the high rate measurements that has to be transferred from the foot. Wireless links may limit the rate, and they will be power hungry and sensitive due to high strain. The alternative is cabled connections, which have their obvious drawbacks [23].

3.2.3 Body-mounted sensors

A lot of work and research have been conducted in the area of sensors worn on the body and several IMS locations have already been tested, e.g. the waist, trunk, leg and head. The waist or trunk locations are probably the less intrusive IMS placements, and also the most reliable position for heading estimation using gyroscopes or magnetometers [14]. As discussed earlier, the body-mounted sensors placements has a big disadvantage because there is no natural way to use a ZUPT algorithm. To avoid this problem, another approach is taken, namely the step model method mentioned in section 3.2.1, using theory from bio mechanics. The human bipedal gait can be said to consists of two phases; swing and stance [37]. The swing phase extends from the instant the toe leaves the ground (Final Contact, FC), until the heel strikes (Initial Contact, IC). Between FC and IC, the foot is off the ground. The stance phase begins when the heel first contacts the ground, and extends while the foot rolls and it reaches the midstance, producing the forward motion of the body by pivoting of the leg on the ankle. Identifying these points in each step is used in [1] to calculate step lengths.

4 Measuring trajectory with mobile phones

As mentioned, there are sensors (IMS) using acceleration and angular velocity for producing accurate walking trajectories. smart-phones are equipped with similar measurement devices which gives them the same possibility of measuring acceleration and angular velocity. Considering a body-mounted IMS, which have several pros and cons of its placing. Having the IMS connected to different parts of the body can result in both worse and better measurement data. Often, suitable places yielding good measurement data are also inconvenient for the user. For example mounting IMS on a foot or inside the shoe is troublesome but will provide better data [23].

The sole purpose of this project is to find a way to measure traveled distance by using a smart-phone in a belt pack at the lower stomach instead of using any other IMS. There are several methods for solving the proposed problem, one which would be with the help of body-mounted IMS. By using a foot-mounted IMS while also using a smart-phone for measuring, the data collection can be made simultaneously. The collected data can then be analyzed to look for patterns and connections.

The technical specifications for the two smart-phones used in the project are presented later in section 5.1, and as a body-mounted IMS was the device Oblu™ (refered to as Oblu) from the company Oblu co. used [26].

4.1 OpenShoe

A naive approach for calculating a trajectory would be to double-integrate the acceleration measurements from the IMS directly to find out the position. Because of errors in the measurements a drift will occur and the predicted positions will be useless after $\sim 10[s]$ of movement [25]. The open-source project, OpenShoe, found a way around this by using zero-velocity update (ZUPT). By estimating when the foot is stationary, OpenShoe could reset the systems which lead to a better estimation of the trajectory. This is called ZUPT-aided IMS. The method is not perfect, but it is the best way of estimating a pedestrians traveled distance using IMS [28, 23, 24].

4.1.1 Mechanization equations and zero-velocity

The full system for estimating a traveled trajectory consists essentially of the ZUPT-aided IMS combined with mechanization equations. The mechanization equations describes how the position, velocity and quaternion vector are calculated.

$$\begin{bmatrix}\mathbf{p}_k\\ \mathbf{v}_k\\ \mathbf{q}_k\end{bmatrix} = \begin{bmatrix}\mathbf{p}_{k-1} + \mathbf{v}_{k-1}dt\\ \mathbf{v}_{k-1} + (\mathbf{q}_{k-1}\mathbf{a}_{k-1}\mathbf{q}*_{k-1} - \mathbf{g})dt\\ \mathbf{\Omega}(\boldsymbol{\omega}_k dt)\mathbf{q}_{k-1}\end{bmatrix} \tag{4.1}$$

Here, k is the time index, dt the time differential, $\mathbf{p}_k$ the position in all three euclidean dimensions (x, y, z), $\mathbf{v}_k$ the velocity in (x, y, z), $\boldsymbol{\omega}_k$ the angular rate in (x, y, z) and $\mathbf{g} = [0, 0, g]$ the gravity force in z-dimension. The vector $\mathbf{q}_k$ is called a quaternion vector and describes the orientation of the system. It is used to form the rotation matrix which transforms the system back to a reference system setup. The reference system is defined as x-dimension in the walking direction, z-dimension in the vertical direction (gravity direction) and y-dimension orthogonal to both x-, and z-direction (left to right direction). The $\mathbf{\Omega}(\cdot)$ is the quaternion update matrix, which by the use of angular rates, updates the quaternion $\mathbf{q}_{k-1}$ [23, 24].

The mechanization equations in (4.1) work poorly because of the drift, unless zero-velocity updates are used. There are multiple ways OpenShoe has detected zero-velocity, where the most prominent one is called the Stance Hypothesis Optimal dEtector (SHOE). In general, the detector tries the test statistics found in (4.2) against a certain threshold to determine if the foot is stationary under a certain time window. The mathematical formulation of the SHOE detector can be seen in equation 4.3. [34].

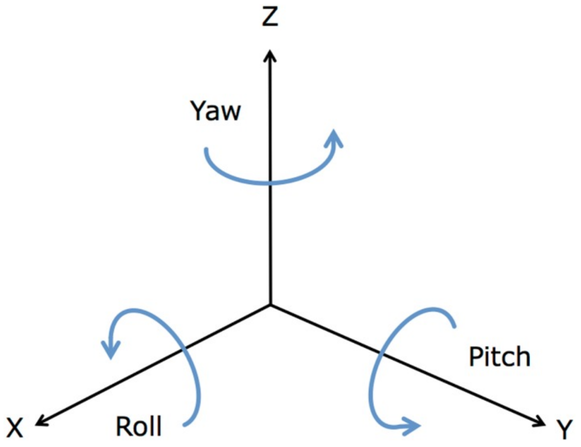

Figure 1: The Euler angles in a Euclidean coordinate system showing the connection between them. The connections are: roll (ϕ)to X, pitch (θ) to Y and yaw (ψ) to Z [6].

$$T(\mathbf{z}_n^a, \mathbf{z}_n^\omega) < \gamma \tag{4.2}$$

$$T(\mathbf{z}_n^a, \mathbf{z}_n^\omega) = \frac{1}{W} \sum_{k=n}^{n+W-1} \frac{1}{\sigma_a^2} ||\mathbf{y}_k^a - g\frac{\overline{\mathbf{y}}_n^a}{||\overline{\mathbf{y}}_n^a||}||^2 + \frac{1}{\sigma_\omega^2}||\mathbf{y}_k^\omega||^2 \tag{4.3}$$

Here are the vectors $\mathbf{y}_n^a$ and $\mathbf{y}_n^\omega$ denoting the measured acceleration and angular rates at time point (sample) n. The matrices $\mathbf{z}_n^a \triangleq \{\mathbf{y}_n^a\}_{k=n}^{n+W-1}$ and $\mathbf{z}_n^\omega \triangleq \{\mathbf{y}_n^\omega\}_{k=n}^{n+W-1}$ are measurement sequences of acceleration and angular rates for a certain sample window and γ is a threshold. The sample window starts at sample n to $n + W - 1$ and has therefore a sample length of W. The controllable parameters, σ_a^2 and σ_ω^2, denote the variance of the acceleration and angular rate measurements. Further on is $|| \cdot ||$ the 2-norm while $\overline{\mathbf{y}}_n^a$ is the sample mean for the considered time window. Worth noticing is the variances σ_a^2 and σ_ω^2 affect different terms in equation 4.3, therefore will the ratio $\sigma_a^2/\sigma_\omega^2$ reflect which disturbances from the accelerometer or gyroscope that will matter most. The tuning parameters for the SHOE detector are then the window size W, detection threshold γ and the ratio $\sigma_a^2/\sigma_\omega^2$. The SHOE detector is a binary operator determining for every sample if the foot should be considered to be moving or to be stationary during a specific moment. The information is then stored as ZUPT to be used as an aid for the IMS. The SHOE detector method is the most prominent one because it takes both acceleration and angular rates measurements into consideration [34, 25]. For further details about the detectors see [34].

With a method to predict when the foot is stationary, a first order deviation (error) model for 4.1 can be setup. Firstly, the quaternion vector $\mathbf{q}_k$ can be expressed in Euler angles $\theta_k = [\phi_k, \theta_k, \psi_k]$ (roll, pitch, yaw). The Euler angles θ contains the same description for the orientation as the quaternion vector $\mathbf{q}_k$, the change is just for analytical convenience. In figure 1, a description of the Euler angels (roll,pitch,yaw) can be seen. The euclidean coordinates (X,Y,Z) follow the same definition as described before.

The deviation model is presented below in 4.4 and contains in total nine states (position-, velocity- and Euler angles-errors).

$$\begin{bmatrix} \delta\mathbf{p}_k \\ \delta\mathbf{v}_k \\ \delta\boldsymbol{\theta}_k \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{I}dt & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & [\mathbf{q}_{k-1}\mathbf{a}_k\mathbf{q}_{k-1}^*]_\times dt \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta\mathbf{p}_{k-1} \\ \delta\mathbf{v}_{k-1} \\ \delta\boldsymbol{\theta}_{k-1} \end{bmatrix} \tag{4.4}$$

Here does $\delta(\cdot)_k$ represents the error states, $\mathbf{I}$ and $\mathbf{0}$ are 3×3 identity and zero matrices, and the notation $[\cdot]_\times$ stands for the cross-product matrix. With the combination of ZUPT, mechanization equations 4.1 and deviation model 4.4, a statistical model of Kalman filtering can be used to propagate a walking trajectory [24, 25]. For more specific details see [7, 8].

4.1.2 Kalman Filter in general

The Kalman filter is a well tested method for reconstruction, interpolation and prediction of state spaces where inputs and outputs of the system are given. The linear state space representation can be seen in equation 4.1.2, where $\mathbf{x}_t$ are the state spaces, $\mathbf{u}_t$ input vector and $\mathbf{y}_t$ the output vector. $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are known matrices and $\mathbf{e}_t$ and $\mathbf{w}_t$ are the process- and measurement noise with covariance $Cov(e_t) = R_e$ and $Cov(w_t) = R_w$. The noises are of different nature and are therefore assumed to be independent of each other. The time index has the notation t in the state space representation.

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{e}_t \tag{4.5}$$
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{w}_t, \tag{4.6}$$

It turns out that the Kalman filter is the optimal linear projector, because it projects the states onto the space spanned by earlier observations. Although it is the optimal filter, the Kalman filter assumes Gaussian processes and linear problems. For other types of problems it might not be optimal, but still performing good. In cases of non-linearity, other version of Kalman filters can be used, for example the Extended Kalman Filter (EKF) [13, 17].

The Kalman filter for a state space system can be calculated recursively as presented below.

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\Big(\mathbf{y}_t - \mathbf{C}\hat{\mathbf{x}}_{t|t-1}\Big) \tag{4.7}$$
$$\hat{\mathbf{x}}_{t+1|t} = \mathbf{A}\hat{\mathbf{x}}_{t|t} + \mathbf{B}\mathbf{u}_t \tag{4.8}$$

where $\hat{\mathbf{x}}_{t+k|t}$ is the optimal prediction of the state space vector. The notation $\mathbf{K}_t$ is known as the Kalman gain and is defined as

$$\mathbf{K}_t = \mathbf{R}^{x,x}_{t|t-1}\mathbf{C}^T\Big[\mathbf{R}^{y,y}_{t|t-1}\Big]^{-1} \tag{4.9}$$

and

$$\mathbf{R}^{x,x}_{t|t-1} = V\{\mathbf{x}_t|\mathbf{Y}_{t-1}\} \tag{4.10}$$
$$\mathbf{R}^{y,y}_{t|t-1} = V\{\mathbf{y}_t|\mathbf{Y}_{t-1}\} \tag{4.11}$$

where $V\{\cdot\}$ is by definition the variance. The recursive update of the variances $\mathbf{R}^{x,x}$ and $\mathbf{R}^{y,y}$ are the following [13]

$$\mathbf{R}^{x,x}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\mathbf{R}^{x,x}_{t|t-1} \tag{4.12}$$
$$\mathbf{R}^{x,x}_{t+1|t} = \mathbf{A}\mathbf{R}^{x,x}_{t|t}\mathbf{A}^T + \mathbf{R}_e \tag{4.13}$$
$$\mathbf{R}^{y,y}_{t+1|t} = \mathbf{C}\mathbf{R}^{x,x}_{t+1|t}\mathbf{C}^T + \mathbf{R}_w \tag{4.14}$$

One need two initial values to start out the recursive filtering. The initial values are not important to be perfect, because the Kalman filter will converge sooner or later. Two decent starting values could be the average and variance formed over many realizations of $\mathbf{x}_1$ which would suggest

$$\hat{\mathbf{x}}_{1|0} = E\{\mathbf{x}_1\} = \mathbf{m}_0 \tag{4.15}$$
$$\mathbf{R}^{x,x}_{1|0} = V\{\mathbf{x}_1\} = \mathbf{V}_0 \tag{4.16}$$

to be good initial values [13].

4.1.3 OpenShoe Kalman filter

In the OpenShoe project a Kalman filter was used for setting up the statistical models for trajectory propagation. They have divided the movement into two cases, one when the foot is moving and one when the foot is stationary. When the foot is moving, the mechanization equation 4.1 is used to propagate the movement and the covariance matrix in equation 4.17 is updated every time-step according to equations 4.12 and 4.13 [24, 25].

$$cov(\mathbf{p}_k, \mathbf{v}_k, \boldsymbol{\theta}_k) = \begin{bmatrix} \mathbf{P}_{\mathbf{p}_k} & \mathbf{P}_{\mathbf{p}_k,\mathbf{v}_k} & \mathbf{P}_{\mathbf{p}_k,\boldsymbol{\theta}_k} \\ \mathbf{P}_{\mathbf{v}_k,\mathbf{p}_k} & \mathbf{P}_{\mathbf{v}_k} & \mathbf{P}_{\mathbf{v}_k,\boldsymbol{\theta}_k} \\ \mathbf{P}_{\boldsymbol{\theta}_k,\mathbf{p}_k} & \mathbf{P}_{\boldsymbol{\theta}_k,\mathbf{v}_k} & \mathbf{P}_{\boldsymbol{\theta}_k} \end{bmatrix} \tag{4.17}$$

where $\mathbf{P}_{x,y} = cov(x,y)$ and $\mathbf{P}_x = cov(x,x)$. As mentioned before, a drift will affect the trajectory propagation and will also make the covariance matrix in equation 4.17 to increase over time. The way around this is using the ZUPT's to tell when the IMS is stationary.

For the case when the IMS is stationary, the OpenShoe project has setup another Kalman type of filter on the deviation model equation 4.4. The deviation model's connection to the state space model would be:

$$\mathbf{x}_t = \begin{bmatrix} \delta\mathbf{p}_k \\ \delta\mathbf{v}_k \\ \delta\boldsymbol{\theta}_k \end{bmatrix} \quad , \quad \mathbf{y}_t = \begin{bmatrix} \mathbf{a}_t \\ \boldsymbol{\omega}_t \end{bmatrix} \tag{4.18}$$

where $\mathbf{y}_t$ are all observations up to time t. The ZUPT's are then considered to be pseudo-measurements implying when the IMS is at stand still. During consecutive periods of ZUPT's, one can say that the mechanization system in equation 4.1 becomes linear and time invariant [23]. The velocity should then be 0 in all directions and the Euler angles roll and pitch are also considered to be 0. The system is in, what is called, a stationary state at these time points [23, 25, 24]. The system's states are then observable during zero-velocity and their covariances decay as one over the number of consecutive ZUPT's. The estimated covariance for the state spaces (using Euler angles instead of quaternion) then converges to

$$cov((\mathbf{p}_k, \mathbf{v}_k, \theta_k)) \approx \begin{bmatrix} \mathbf{P}_{\mathbf{p}_k} & \mathbf{0}_{3\times5} & \mathbf{P}_{\mathbf{p}_k,\psi_k} \\ \mathbf{0}_{3\times5} & \mathbf{0}_{5\times5} & \mathbf{0}_{5\times1} \\ \mathbf{P}^T_{\mathbf{p}_k,\psi_k} & \mathbf{0}_{1\times5} & \mathbf{P}_{\psi_k,\psi_k} \end{bmatrix} \tag{4.19}$$

By the look of the covariance matrix, the errors between the observable states ($\mathbf{v}_k$, ϕ_k and θ_k) and the non-observable states ($\mathbf{p}_k$ and ψ_k) are uncorrelated. Because they are uncorrelated and with the Markovian assumption in equation 4.1 [23], the system can be reset when there are a reasonable number of consecutive ZUPT's. Resetting the system means that the relative position and yaw is set to 0 i.e the system starts over at a new step every time the foot is considered be back at stand still. The reset will handle the drift errors and make them sufficiently small so they do not accumulate over time, but the system should return to a stationary state within intervals of $\sim 2[s]$ or less [25, 23]. With the knowledge

of when to reset the system, the error states can be estimated with the use of a complementary statistical model [24].
With the use of the state space model in section 4.1.2 and the information from ZUPT, a new Kalman type of filter are setup for the deviation model. Using the ZUPT's as pseudo-measurements gives

$$\tilde{\mathbf{y}}_k = \mathbf{H} \begin{bmatrix} \delta\mathbf{p}_k \\ \delta\mathbf{v}_k \\ \delta\boldsymbol{\theta}_k \end{bmatrix} + \mathbf{n}_k \quad , \quad \mathbf{H} = [\mathbf{0} \ \mathbf{I} \ \mathbf{0}] \tag{4.20}$$

where $\mathbf{H}$ is the measurement matrix and $\mathbf{n}_k$ is a measurement noise [24].

OpenShoe uses two Kalman filter approaches to be able to reset the system for a better result than naively propagate the trajectory using only the mechanization equations in 4.1, which makes a drift occur.

4.1.4 OpenShoe - Results

By using the Kalman filter and ZUPT to reset the system, the accuracy of measuring a step are down to a position error in [cm] [25]. The OpenShoe project group have demonstrated its methods by walking in a trajectory symbolizing an eight. In figure 2 is the moving path by the OpenShoe project group presented. The results are extremely accurate and reconstructs the path perfectly. Although one should be aware of a test like this has been done under controlled circumstances and the results may vary between persons, distances and terrain [23, 24, 25].

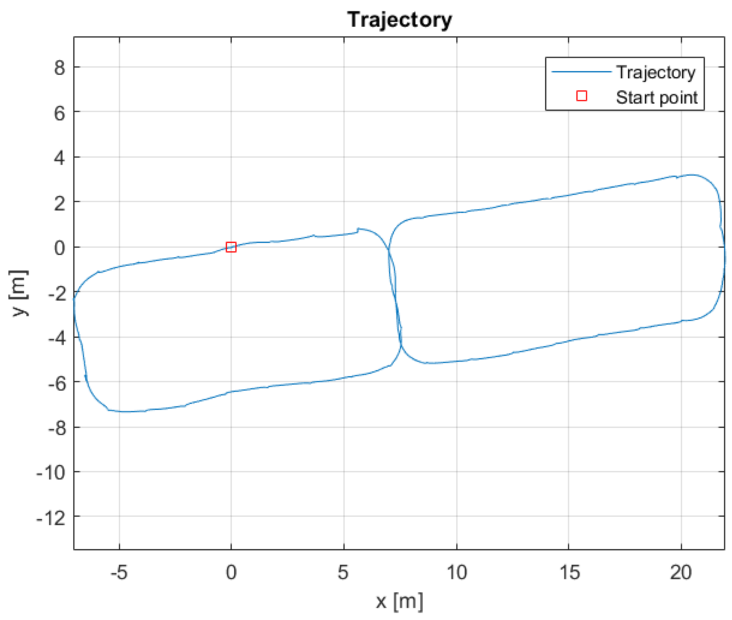

Figure 2: Recorded path done by the OpenShoe project group where an IMS tracks the trajectory by using the developed software and filters for optimal reconstruction of the path [24, 28].

4.2 Oblu

With the knowledge that feet are at some point stationary during walking, Oblu has created the device oblu for tracking the movement of a person. The company has based its algorithms for trajectory estimation on the open-source project OpenShoe originating from KTH, Stockholm [28]. In Figure 3 below, can the oblu device be seen in its case. The usage of oblu is simple, it is strapped on a person's foot and then connected to Oblu's mobile application "Xoblu". Measurements of speed, direction, and trajectory during walking can be received directly in Xoblu application [26]. There is also a way of receiving acceleration and angular rate outputs by connecting a computer to oblu through a micro-USB port. However, this procedure is inconvenient and strenuous as one need a cable connection and a portable computer to follow the oblu around.

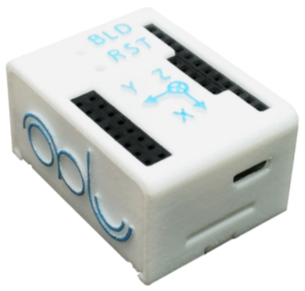

Figure 3: The oblu device connected to a foot for tracking one's movement, speed and direction [27].

4.2.1 oblu - specifications

The oblu device is a circuit board containing four Inertial Movement Sensors (IMS) which measure acceleration (accelerometer) and angular velocity (gyroscope) in all three dimensions X,Y and Z. The accelerometers in the IMS have the max capacity of measuring $\pm 16g$, where g represents the gravity acceleration. In table 1, is the bias stability and velocity for random walk presented. The bias stability represents the minimum bias which can not be estimated and is calculated from the average acceleration output when the oblu is still. The velocity random walk is the average error that will occur when the acceleration output is integrated. The velocity random walk is calculated from the white noise acceleration output when the oblu is still [27, 36].

Table 1: Errors and standard deviation to be expected from the accelerometer axes.

Axis	Bias stability	Velocity Random Walk
X	0.037 mg	3.46 $mg/\sqrt{hr}$
Y	0.034 mg	3.46 $mg/\sqrt{hr}$
Z	0.034 mg	5.99 $mg/\sqrt{hr}$

The gyroscopes have a max capacity of measuring up to $\pm 2000°/s$. In table 2 the bias stability and angle random walk is presented. The gyroscope's bias stability and angle random walk follow the same definition as for the accelerometers.

Table 2: Errors and standard deviation to be expected from the gyroscopes axes.

Axis	Bias stability	Angle Random Walk
X	$0.042°/h$	$0.002°/\sqrt{hr}$
Y	$0.027°/h$	$0.002°/\sqrt{hr}$
Z	$0.026°/h$	$0.002°/\sqrt{hr}$

The oblu device also has a maximum sampling rate of 1 kHz, but can be adjusted after ones preferences. The four IMS are located as a 2x2 array on the circuit board, and together is the array named multiple IMS (MIMS). Further on, the oblu device contains all the necessary systems for collecting and sending the data output from MIMS [27].

4.2.2 oblu - results

Using the oblu connected to the app Xoblu is convenient, while collecting acceleration and angular rates measurement (referred to raw data from now on) from the MIMS is not. The approach was to use the open-source algorithms developed by the OpenShoe project to calculate the trajectory using the oblu device. This worked poorly, which can be seen in figure 4 as it shows a walk through a library and back to the starting point.

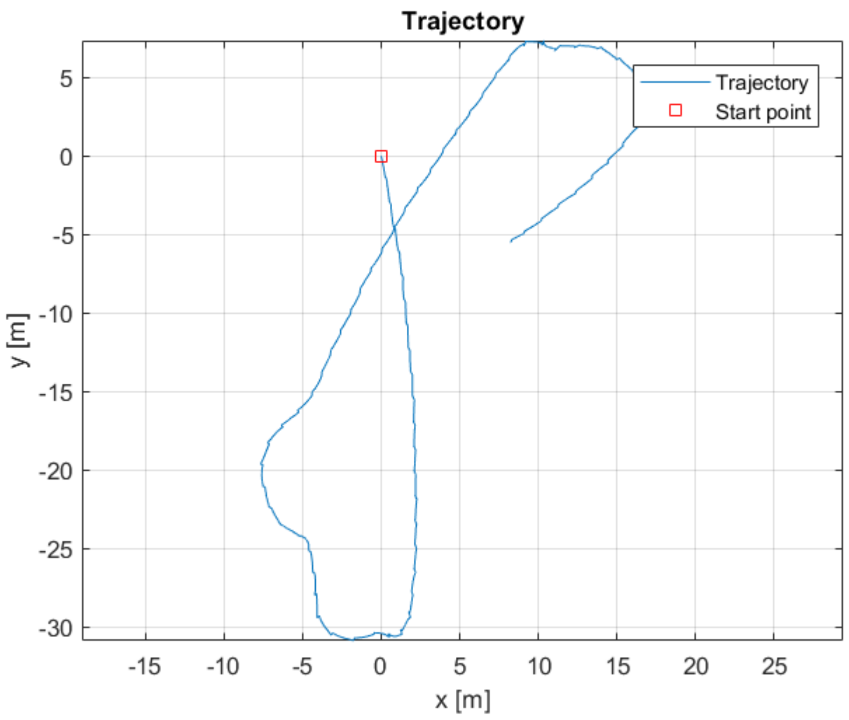

Figure 4: A walk through a library with collecting of raw data (acceleration and angular rates) from an oblu device strapped to one foot. The trajectory path has been reconstructed using OpenShoe's open-source algorithms. The starting point and end point was the same during the trial [28, 26].

As can be seen in the trajectory in figure 4, there is a constant shift in the direction which indicates an underlying drift in the gyroscope measurements. Testing a longer walk outside, using Oblu's own application Xoblu, the reconstruction of the path was much better. In figure 5, the blue path represent the true walking, while the red trajectory is the Xoblu reconstructed path.

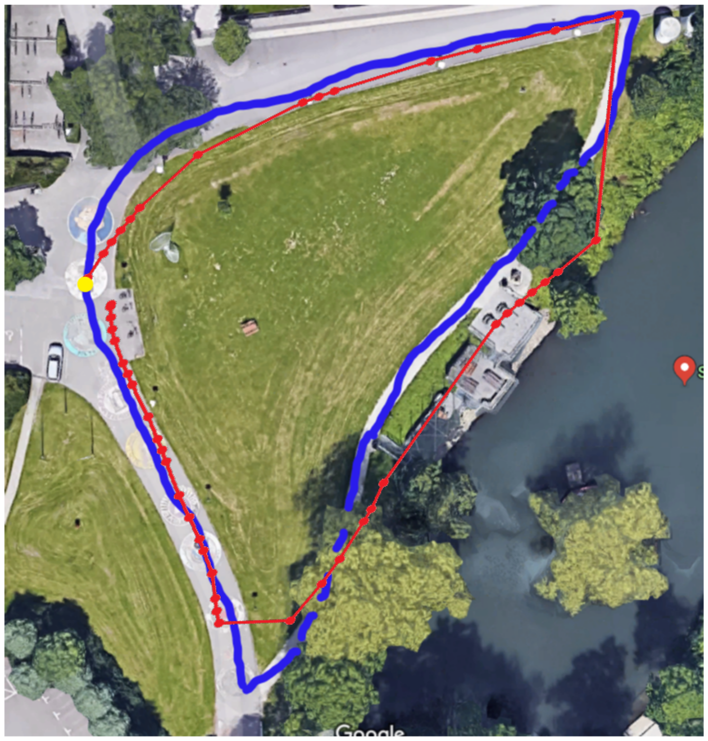

Figure 5: A walk outside the mathematical building, LTH, close to Sjön sjön. The walk start at the yellow point and the true walk is the trajectory in blue colour. The red trajectory was reconstructed with the application Xoblu. The background is provided by Google maps [26, 10].

4.3 Analysis - OpenShoe & Oblu

The OpenShoe project examined the possibility of estimating a path with the use of IMS connected to ones feet. They have produced great results and can reach estimations with errors of only 1 % or lower [25]. The key to their success is the zero-velocity update, ZUPT. The ZUPT implies when the foot should be considered to be still, which creates pseudo-measurements and enables the system to reset. The problem with smart-phones are that they seldomly are strapped to ones feet. In this case, when they are strapped to ones stomach through a belt pack, there is no situation when the system could be considered stationary. Therefore, the ZUPT, in the case when using smart-phones, is not able to render any aid.

It is possible to use the oblu device and OpenShoe open-source algorithms to find correlation and similar features between the oblu measurements and the smart-phone's measurements. One can try many approaches to this method, but the raw data analyzed only renders poor results (see figure 4). The oblu device and OpenShoe open-source algorithms are not robust enough and are therefore not trustworthy to do further analyzes on. Although the Oblu's own application Xoblu performs well, its code is not open-source. There is a difference between OpenShoe's open-source software and Oblu's software since Oblu has only based its product on the open-source software. Because the methods are not retrievable, and analyzing raw data from oblu is not trustworthy, there might be better options to estimate a trajectory using smart-phones.

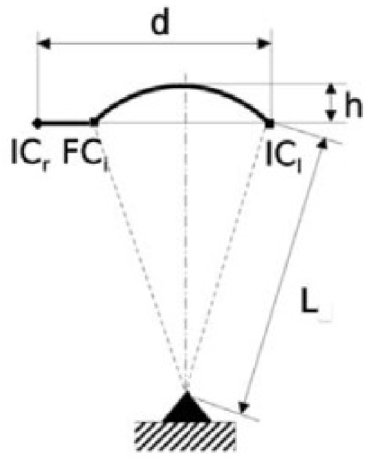

Figure 6: The cyclical nature of a step. Between IC_r and FC_l is the stance phase, where both feet are on the ground, while the swing phase between FC_l and IC_l one taking a step forward. The cycle is then completed and starts all over. The total step length is d, the height COG increases is h, and L is the leg length [1].

5 Estimating distance with smart-phones

As mentioned in the earlier chapters, it is hard to estimate the trajectory using the smart-phone when there is no zero-velocity moments. Although there might exist ways to form the trajectory using a body-mounted smart-phone, but the simpler way is to start by finding a good estimate of just the distance walked. The most common approach for calculating the walked distance, is by setting up non-linear models for estimating step lengths [1].

The idea is to find a mathematical model for human walk and from this estimate the distance traveled. Because of the cyclic nature of the human walk, the properties for each taken step are similar. One obvious property is the swing and stance phase, which has given a scientific clue about how a step may be modeled. As mentioned before, the swing phase represents the foot lifting from the ground while the foot is stationary during the stance phase. Considering the body center of gravity (COG) as the reference point, this may be modeled as an inverted pendulum which is seen in Figure 6. The inverted pendulum is a popular model, but it can only be used to estimate a single step length. If one is interested in forming the whole trajectory, other models have to be used. which takes angular rates into consideration. In this project, the main interest is on distance estimation, why the focus is on forming such an estimator rather than trying to estimate the whole trajectory.

There are several models following the inverted pendulum approach, which uses different methods of estimating the step length. What is common across the models is the interest of when the foot strikes the ground (IC) and when it leaves the ground (FC). In Figure 6 the notations IC and FC can be seen, where the index r and l stands for right- and left foot. The inverted pendulum in Figure 6 represents both feet being stationary in the stance phase (IC_r to FC_l). In the swing phase (from FC_l to IC_l), the right foot stationary is while the left foot swings forward. During the swing phase, the COG becomes higher with the height h. The total distance traveled under both phases is marked as d, and then the period starts over again in a cyclic manner. This is true for all models using the inverted pendulum [1].

The step length is estimated differently by each of the non-linear step models. What is common among all models investigated however, is that they all use the z-acceleration of the COG . Either they use the z-acceleration to estimate the height h during the swing phase or for estimating the step length d. Some models also use the leg and foot length as input parameters, which of course, reduces the convenience as these inputs have to be measured for each individual. [1].

To collect the true acceleration and the angular rates of the COG, one would need IMS inside the body. Most models strap IMS to the lower back (L2/L3 vertebral position) to get a good estimate of the acceleration for the COG [1, 18]. The lower back is chosen because it is the closest position to the COG, but in this project a belt pack positioned on the lower stomach has been used instead. The reason is the convenience, which always is on a trade-off as this position is not as close to the COG. If the more practical solution gives approximately the same results, it is always chosen in first-hand [1].

The following sections in this chapter describes the technical specifications for the used smart-phones, followed by the developed algorithms to estimate the walked distance for a certain path. The last section handles the validation data to see how good the formed models are in reality.

The developed algorithms are assigned specific tasks, where the first one, Cut, makes sure that the only data passing on is during the time the person was walking. The algorithm $Segmentation$ takes care of the cases when one stops during the walk, splitting up the data sets so no steps are falsely detected during a stand still. There are two algorithms, $ICFCyDetector$ and $ICFCzDetector$, which focus on finding the time points when IC's and FC's occur for the feet. The first one, $ICFCyDetector$, estimates the time points of IC's and FC's using the y-acceleration (walking direction). The estimated IC's and FC's will lie in the neighbourhood of the true IC's and FC's [1, 9, 18]. $ICFCzDetector$ then uses that information and the z-acceleration to estimate more accurate time points when IC's and FC's occur. The algorithm $Walk\,Length$ uses distance models to estimate the length walked for a person, but they might need to be scaled. The function also finds the most optimal scaling constant, K, fitted to each distance model.

Table 3: The specifications of the smart-phones OnePlus 5 and Samsung Galaxy S7, which are used for the project as IMS's [12, 20, 21].

Smart Phone	OnePlus 5	Samsung Galaxy S7
max acceleration	$\pm 16\, g$	$\pm 16\, g$
max sensitivity acceleration	$0.730\, mg/LSB$	$0.488\, mg/LSB$
max angular rate	$2000\, dps$	$2000\, dps$
max sensitivity angular rate	$70\, mdps/LSB$	$70\, mdps/LSB$

5.1 Technical specifications

The two different types of smart-phones used in the project are the OnePlus5 and the Samsung Galaxy S7. The technical specifications for the accelerometer and and gyroscope for each phone can be found in table 3 below. The notations are as follows: g is the gravity unit and mg/LSB the sensitivity unit.The unit mg/LSB should be interpreted as the error one can expect per measured value of the acceleration. The angular rate unit dps stands for degrees per second and further on is the angular rate sensitivity unit, $mdps/LSB$, defined likewise for the accelerometer setting. The sampling rate used in this work is 100 $[Hz]$ for both smart phones [12, 20, 21].

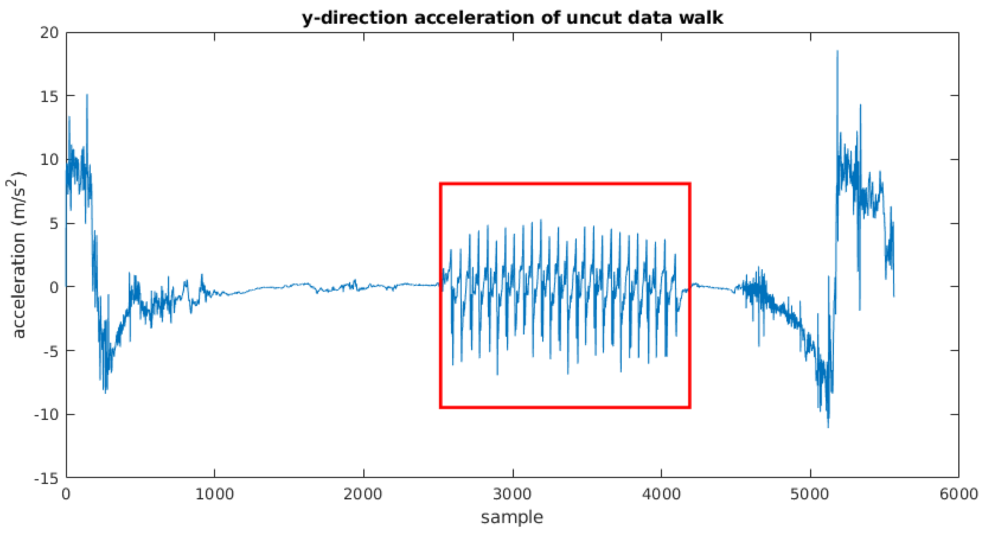

Figure 7: Example of how the data looks before it is cut. In this figure, the y-direction acceleration has been used, but the same behavior is found in all acceleration and gyro directions. The samples outside the red square is cut out of the data measurements.

5.2 Cut

A first step towards getting a good estimate of the distance is to clean the data and optimizing it for further analysis. This includes removing all samples not related to the walk itself, as well as filtering it to reduce the impact of noise from the measurements. From the moment the smart phone is started until it is placed in the pocket, the collected data will not be of any use because of the high fluctuations not corresponding to any walk. The same goes for the samples collected during the time interval from the end of the walk and until the smart phone has been taken out of the pocket and turned off. An example of this can be seen in figure 7, where the red square is the part of the data which consists of walking. As the algorithms for estimating step lengths makes use of periodicity in the gait-cycles for the pedestrian, it is therefore important to remove any obvious noise in the data in order to improve the performance of the algorithm.

To reduce the noise in the data, the data is smoothed by running it through a simple centered moving average filter. Both acceleration and angular velocity data are filtered. A lot of different filtering approaches tested, some more complex and complicated than others, but for this purpose a moving average filter were sufficient. When choosing the filter, it is important that the filtering does not lead to a delay in the data.

5.2.1 Smoothing filter

The smoothing filter that the data is sent through works in the following way. Let Y be an $n \times m$ matrix consisting of all samples, i.e. the acceleration and gyro data. For example, column t in Y will be on the form $\left[\ \mathrm{a}_z^t\ \mathrm{a}_y^t\ \mathrm{a}_z^t\ \mathrm{g}_x^t\ \mathrm{g}_y^t\ \mathrm{g}_z^t \right]^\top$ corresponding to the t sample. Furthermore, let X be an $n \times m$ matrix consisting of the filtered version of Y. The filter smoothes the data by using moving average over the 5 closest neighbours. If there are no neighbours, as will be the case in the beginning and in the end of the data measurements, the neighbouring samples available are used. This means that the filtered signal X will be on the form

$$X = \left[X_{\text{beg}}\, X_{\text{mid}}\, X_{\text{end}}\right], \tag{5.1}$$

with $\left[X_{\text{beg}}\, X_{\text{mid}}\,$ and $X_{\text{end}}\right]$ being on the form

$$X_{\text{beg}} = \begin{bmatrix} \frac{\mathrm{a}_x^1+\mathrm{a}_x^2+\mathrm{a}_x^3}{3} & \frac{\mathrm{a}_x^1+\mathrm{a}_x^2+\mathrm{a}_x^3+\mathrm{a}_x^4}{4} \\ \frac{\mathrm{a}_y^1+\mathrm{a}_y^2+\mathrm{a}_y^3}{3} & \frac{\mathrm{a}_y^1+\mathrm{a}_y^2+\mathrm{a}_y^3+\mathrm{a}_y^4}{4} \\ \frac{\mathrm{a}_z^1+\mathrm{a}_z^2+\mathrm{a}_z^3}{3} & \frac{\mathrm{a}_z^1+\mathrm{a}_y^2+\mathrm{a}_z^3+\mathrm{a}_z^4}{4} \\ \frac{\mathrm{g}_x^1+\mathrm{g}_x^2+\mathrm{g}_x^3}{3} & \frac{\mathrm{g}_x^1+\mathrm{g}_x^2+\mathrm{g}_x^3+\mathrm{g}_x^4}{4} \\ \frac{\mathrm{g}_y^1+\mathrm{g}_y^2+\mathrm{g}_y^3}{3} & \frac{\mathrm{g}_y^1+\mathrm{g}_y^2+\mathrm{g}_y^3+\mathrm{g}_y^4}{4} \\ \frac{\mathrm{g}_z^1+\mathrm{g}_z^2+\mathrm{g}_z^3}{3} & \frac{\mathrm{g}_z^1+\mathrm{g}_y^2+\mathrm{g}_z^3+\mathrm{g}_z^4}{4} \end{bmatrix}, X_{\text{end}} = \begin{bmatrix} \frac{\mathrm{a}_x^m+\mathrm{a}_x^{m-1}+\mathrm{a}_x^{m-2}+\mathrm{a}_x^{m-3}}{4} & \frac{\mathrm{a}_x^m+\mathrm{a}_x^{m-1}+\mathrm{a}_x^{m-2}}{3} \\ \frac{\mathrm{a}_y^m+\mathrm{a}_y^{m-1}+\mathrm{a}_y^{m-2}+\mathrm{a}_y^{m-3}}{4} & \frac{\mathrm{a}_y^m+\mathrm{a}_y^{m-1}+\mathrm{a}_y^{m-2}}{3} \\ \frac{\mathrm{a}_z^m+\mathrm{a}_z^{m-1}+\mathrm{a}_z^{m-2}+\mathrm{a}_z^{m-3}}{4} & \frac{\mathrm{a}_z^m+\mathrm{a}_z^{m-1}+\mathrm{a}_z^{m-2}}{3} \\ \frac{\mathrm{g}_x^m+\mathrm{g}_x^{m-1}+\mathrm{g}_x^{m-2}+\mathrm{g}_x^{m-3}}{4} & \frac{\mathrm{g}_x^m+\mathrm{g}_x^{m-1}+\mathrm{g}_x^{m-2}}{3} \\ \frac{\mathrm{g}_y^m+\mathrm{g}_y^{m-1}+\mathrm{g}_y^{m-2}+\mathrm{g}_y^{m-3}}{4} & \frac{\mathrm{g}_y^m+\mathrm{g}_y^{m-1}+\mathrm{g}_y^{m-2}}{3} \\ \frac{\mathrm{g}_z^m+\mathrm{g}_z^{m-1}+\mathrm{g}_z^{m-2}+\mathrm{g}_z^{m-3}}{4} & \frac{\mathrm{g}_z^m+\mathrm{g}_z^{m-1}+\mathrm{g}_z^{m-2}}{3} \end{bmatrix}, \tag{5.2}$$

and

$$X_{\text{mid}} = \begin{bmatrix} \frac{1}{5}\sum_{j=1}^{5} a_x^j & \frac{1}{5}\sum_{j=2}^{6} a_x^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} a_x^j & \frac{1}{5}\sum_{j=m-5}^{m} a_x^n \\ \frac{1}{5}\sum_{j=1}^{5} a_y^j & \frac{1}{5}\sum_{j=2}^{6} a_y^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} a_y^j & \frac{1}{5}\sum_{j=m-5}^{m} a_y^j \\ \frac{1}{5}\sum_{j=1}^{5} a_z^j & \frac{1}{5}\sum_{j=2}^{6} a_z^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} a_z^j & \frac{1}{5}\sum_{j=m-5}^{m} a_z^j \\ \frac{1}{5}\sum_{j=1}^{5} g_x^j & \frac{1}{5}\sum_{j=2}^{6} g_x^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} g_x^j & \frac{1}{5}\sum_{j=m-5}^{m} g_x^j \\ \frac{1}{5}\sum_{j=1}^{5} g_y^j & \frac{1}{5}\sum_{j=2}^{6} g_y^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} g_y^j & \frac{1}{5}\sum_{j=m-5}^{m} g_y^j \\ \frac{1}{5}\sum_{j=1}^{5} g_z^j & \frac{1}{5}\sum_{j=2}^{6} g_z^j & \dots & \frac{1}{5}\sum_{j=m-6}^{m-1} g_z^j & \frac{1}{5}\sum_{j=m-5}^{m} g_z^j \end{bmatrix}. \tag{5.3}$$

5.2.2 Detect walking

Once the signal is filtered, the part of the data with actual walking is located. For this, a threshold model is chosen. First, during human walking, quite naturally the amplitude of the y-direction accelerations, will at least sometimes be above some threshold. Second, the periodicity of the data in all 6 dimensions will be strong due the the nature of the human gait. Every step is similar to each other and approximately of the same length. These assumptions are assumed likely to be true because of the cyclic behaviour in walks. In Figure 8, the first 1000 samples of walking data can be seen in the left pictures and the first 1000 samples of when the person is not walking in the right plots. Clearly, in all six different acceleration and angular velocity series, the amplitudes are significantly smaller and correlations seems to be smaller when one is not walking. These observations are used to build an algorithm which finds suitable cut off indices and all samples before and after they are cut off. Note that a large amplitude in the y-accelerations is not enough to give a solid method for detecting walking. For an example, see Figure 7, where the accelerations are irregularly large in the beginning and end. As these parts correspond to placing or taking out the smart phone from the belt pack, they most certainly are removed since it is not a person walking.

The algorithm uses overlapping sliding windows to analyze the data, once from the left and once from the right of the measurements. The window has sample length W and overlaps with S samples. The goal is to identify the first window from each side that fulfills certain conditions. To find and check for these conditions, the auto-correlation function (ACF) is introduced, being defined in [13].

$$\rho(t)_{\mathrm{d}_j,w_i} = \frac{r(t)_{\mathrm{d}_j,w_i}}{r(0)_{\mathrm{d}_j,w_i}}, \quad 0 \le t \le T \tag{5.4}$$

where d_j denotes the j'th dimension, w_i denotes the ith window function, and $r(k)$ is the auto-covariance function. As discussed above, the human gait is expected to be cyclic with strong correlation in every step due to the periodicity. The windows sliding once from each side are looking for this periodicity. The process is continued until a window fulfills the following two conditions:

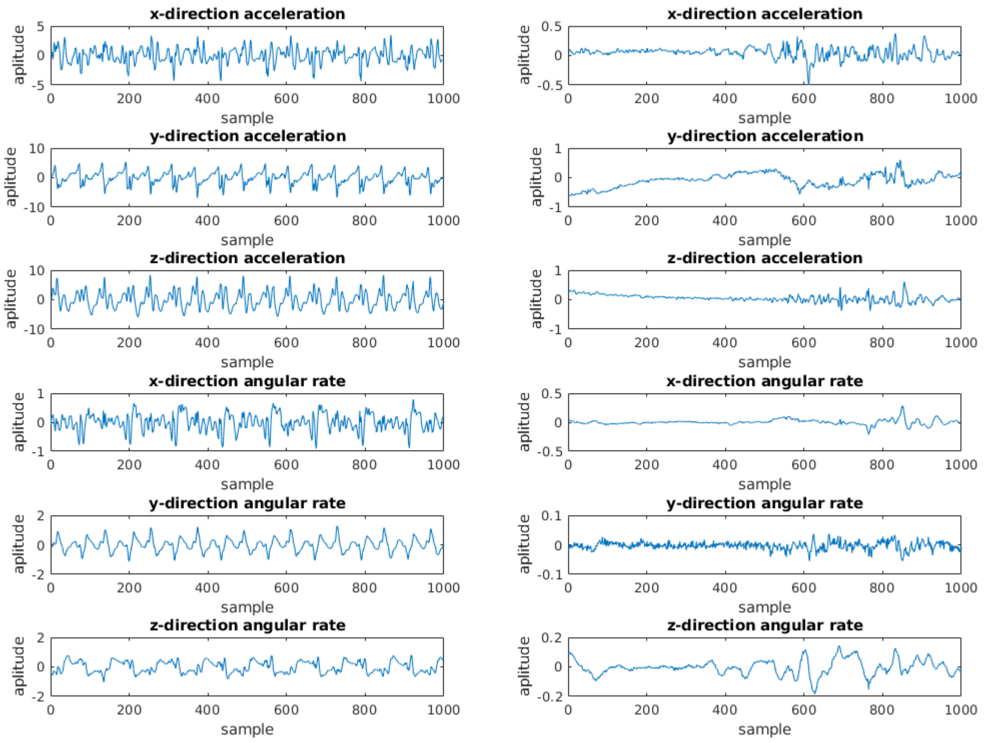

Figure 8: To the left, 1000 samples, 10 seconds of walking, for all 6 different dimensions. To the right, 1000 samples, 10 seconds of not walking, for all 6 different dimensions. Note the scale of the y-axis.

$$A_i^1 = \sum_{j=1}^{6} \min\{\rho(t)_{\mathrm{d}_j,\mathrm{w}_i}\} < C, \quad \text{with } \mathrm{d} = \{\mathrm{a}_x, \mathrm{a}_y, \mathrm{a}_z, \mathrm{g}_x, \mathrm{g}_y, \mathrm{g}_z\}, \tag{5.5}$$

Through empirical tests on people walking, some criteria have been developed by analyzing the periodic human gait cycle and forming thresholds which should be met during walking.
The first criterion is that the sum of the ACF in equation 5.5 is less than C. In this case, $C = -2.4$ and the window length is $W = 400$ samples long. It investigates the correlation for both acceleration and angular rates, and takes advantage of the high negative correlation between the feet taking steps forward and when they are standing still.

The other criterion is that the maximum amplitude, in a given window i, needs to be larger than some threshold D, as the first condition (5.5) can be fulfilled even when standing still. Here, the threshold D, is set to 2.25. The implementation of this condition is

$$A_i^2 = \begin{cases} 1, & \text{if any y-direction acceleration sample in the window is larger than } D \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

To decide if the window should be used for determine a cut off, the conditions are merged to one. This is done by checking if (5.7) is satisfied. If it is, the data frame is deemed to contain walking data.

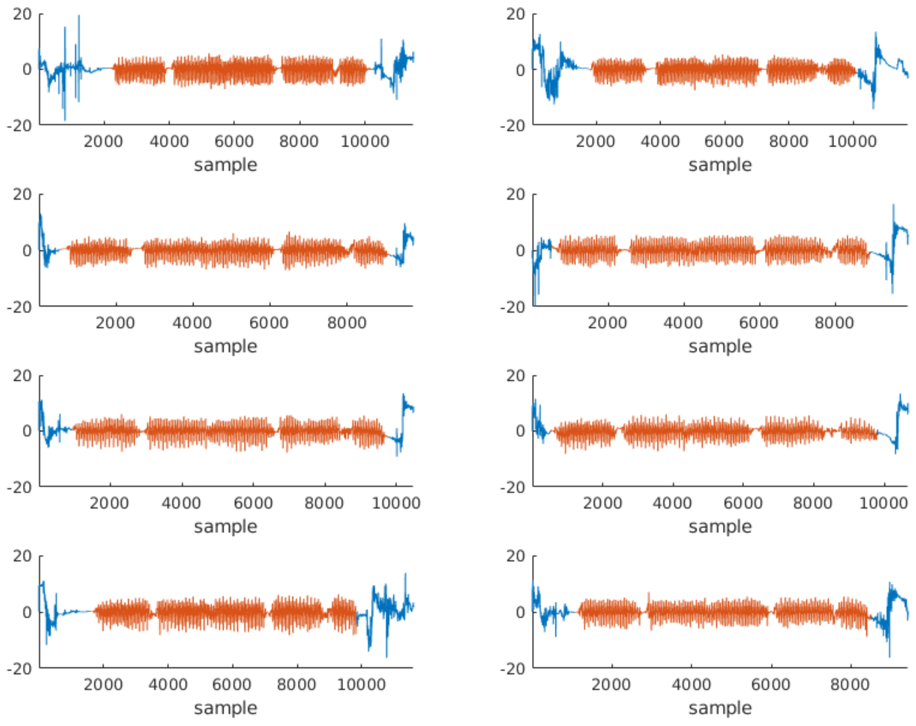

Figure 9: Eight different walks where the beginnings and the ends have been cut off. The blue regions only contains large fluctuations and noise, while it is the red regions where walking is done. The red regions are cut out, which make it easier for further analysis on distance estimation.

$$A_i^1 A_i^2 < K. \tag{5.7}$$

where K is set to -2.7 in this project.

5.2.3 Results

When a frame is deemed to contain walking data, the window i is used to determine the cut off on the side of the series where the window is located. Finally, to actually decide what index to use as the cut off, the first sample in the left window is chosen and likewise the last sample in the right window.

The parameters W, S, T, C, D are tuning parameters, which are used to manage the cut off of the data measurements. This is done successfully in 40 of 40 different walks of different lengths, divided between 17 different people. To illustrate, a selection of these are presented in Figure 9, where the blue part of the series are the parts that have been cut of, and the red is the part which is kept.

5.2.4 Analysis - Cut

As the cut is designed and incorporated in the application to get rid of all noise and non-walking related samples in the walking data, it is seen that it succeeds very well. It manages to cut the data at the appropriate indices. Multiple different ways were tried to choose the cut off samples given the window used, but the final result was not affected as no steps were missed or falsely detected in any of the cases.

The gait cycle was also calculated for the whole walk for different cut offs in the window, these were found to change marginally for different window lengths. The algorithm *Cut* was in a trial excluded from the final application. This however turns out to severely reduce the accuracy of the estimated walk lengths and gives poor results. The main reason is the gait cycle length, which becomes inaccurate if the data is not cut. In some cases the average gait cycle length is changed by more than 100%, from 58 samples [0.58s] to 130 [1.3s]. One important thing is to not cut off too close to the walk, as this potentially could cause some steps to be excluded. However, as the cut offs are set early in the left window and late in the right window, this is never really an issue.

The algorithm uses a lot of tuning parameters. Some were tried to be removed, but in the end, they were set using all 40 test walks. The parameter values are set by empirical tests, and they do not vary a lot from person to person and walk to walk. Tuning parameters in other algorithms for the distance estimation is harder as they deal with the actual attributes of single steps, something which the cut algorithm does not.

5.3 Segmentation

During a walk, there is no obligation that one always needs to continue walking until the end of the path. One can always stop while catching a breath or enjoying a beautiful view, which here will be referred to as a standstill. The problem is that it will affect the accelerations and angular rates cyclic behaviour, which will cause problem when estimating the walked distance. A solution to this problem is to find each of the segments of walking, then separate and analyze them individually.

The function *Segmentation* has this purpose. It locates each walking segment, split them into individual data sets which only contains walking and no stopping. The function also finds when u-turns occur, but u-turns will not contribute with any extra information for distance estimation.

5.3.1 Finding stationary moments

The retrieved raw data of accelerations and angular rates are first handled by the function *Cut*, where the unwanted parts are removed and the data also gets filtered according to section 5.2. The data is now filtered and only contains the parts of walking, and possibly moments of standing still. The function *Segmentation* locates where to split the data set into smaller segments based on the magnitude of the normalized acceleration. The time points where to split the data set will be referred to as splits.

In Figure 10, the magnitude of the normalized acceleration in every direction is presented for an ordinary 90 meter walk on plain terrain. There are three moments of standstill between the start and end of the walk, which are around samples: 2500, 5800 and 7500. There are also two u-turns executed, one around sample 4000 and the second at the last standstill, at sample 7500.

As seen in figure 10, there are similar moments where the acceleration magnitude is low. These moments correspond to standing still, and *Segmentation* is therefore based on locating low acceleration magnitudes. For finding the u-turns, *Segmentation* analyzes the angular rate output in the z-direction. The angular rate in the z-direction will further on be referred to as ω_z throughout this chapter.

The *Segmentation* function makes some assumptions, needed for it to work. First, it assumes that the data sets starts and ends with standstills, i.e., it will force the system to be considered still at start and end. Another assumption is the maximum limit of 15 s of standstill in the beginning and in the end (starting and stopping) of the data set. Furthermore, small windows of less than 2.5 s will be discarded, all for a better robustness.

5.3.2 $\boldsymbol{Acc_{sum}}$ and $\boldsymbol{\omega_z}$

Considering Figure 10, one can see that the acceleration is low in all three directions during a complete standstill. Standstill can be retrieved by analyzing the sum of the acceleration magnitudes and comparing it to a threshold. When below the threshold, the person is considered standing still. Although when a u-turn is executed, there is a higher magnitude in the x- and y-acceleration due to the turning (see around sample $\sim$ 7500 in Figure 10). The data is always split up when a standstill occurs, even though the body is turned on the spot. Therefore the acceleration directions are scaled with different weights, according to their ability to recognize a standstill. The z-acceleration is still low during a u-turn, while y-acceleration fluctuates more and the x-acceleration fluctuates the most. The prioritizing order is then z-, y- and x-acceleration, and the sum is defined as in (5.8). There is no need to construct a sum when analyzing u-turns, since the ω_z contains enough information.

$$acc_{sum} = |acc_x| + 3|acc_y| + 5|acc_z| \tag{5.8}$$

where the weights 1, 3 and 5 have been chosen according to the priority order. The acc_{sum} and the ω_z can be seen in figure 11 below. Although it is clear that the acc_{sum} is low during standstill, the

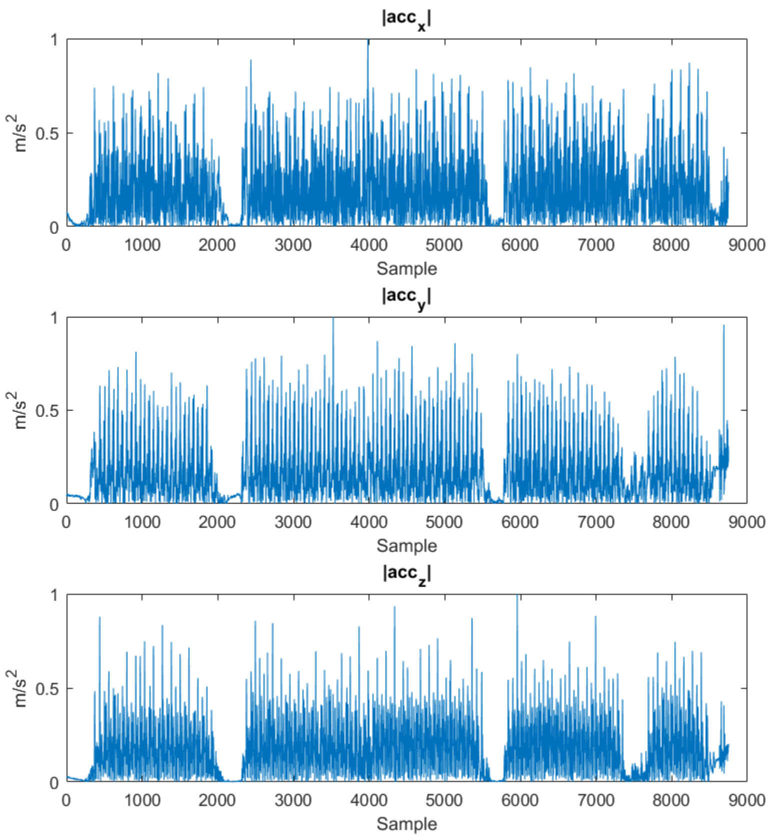

Figure 10: The magnitude of normalized accelerations in all directions of a 90 meter walk. There were three standstills (at sample 2500, 5800 and 7500) and two u-turns (at sample 4000 and 7800) executed during the walk. Notice the fluctuations at sample 7800 when a u-turn is done during standstill.

magnitude can still reach below the threshold for individual samples even when one is moving. A solution is to filter and smooth the data sets for an easier analysis.

5.3.3 Low-pass filter, smoothing and transformation

To reduce the high frequency fluctuations, a low-pass Finite Impulse Response (FIR) filter with a Hamming window is introduced. The FIR filter is defined below in (5.9) with the use of the impulse response sequence $h(n)$ found in (5.10) [29].

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \tag{5.9}$$

$$h(k) = h_d(k)w(k) \tag{5.10}$$

The standard finite filter method presented in (5.9), is an infinite filter approximated for M lags. The notation's $x(n-k)$, $y(n)$ and $h(k)$ correspond to the input signal, the filter response and the impulse response. In (5.10), the impulse for a finite number of discrete time points (M) presented. The $h_d(k)$ stands for the discrete impulse response; where further details can be found in [29]. The restriction in lags correspond to the length M of the window $w(k)$ found in (5.10). The chosen window for this project is the Hamming window, defined as [29].

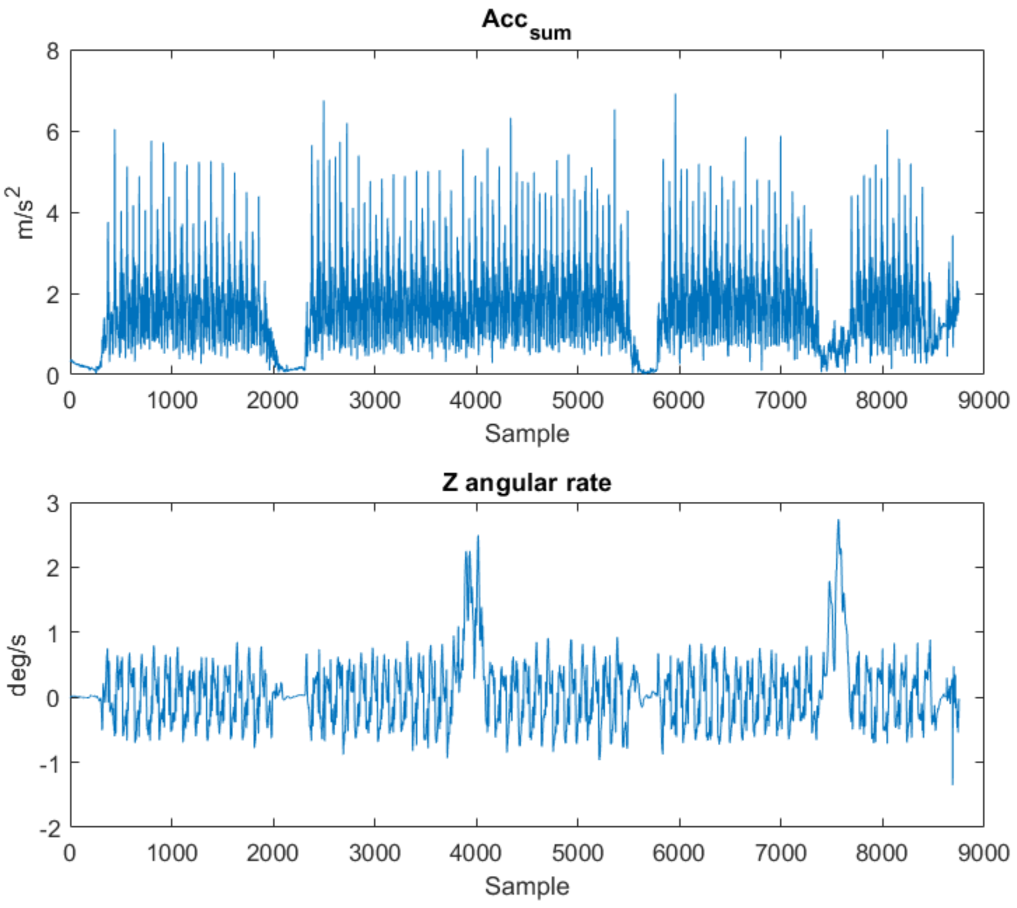

Figure 11: The acc_{sum} and the angular rate in z-direction. As can be seen, there are three standstills and two u-turns where the last standstill and u-turn happen simultaneously.

$$w(k) = \begin{cases} 0.54 - 0.46cos\left(\frac{2\pi k}{M}\right) & 0 \leq k \leq M \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

The cut-off frequency for the Hamming window was set to 2 Hz (low-pass filtering) and the window length M to 30 samples. The higher frequencies in the acc_{sum} data set can then be reduced using the constructed FIR filter. For more details about the FIR filter and Hamming window, see [29].

To smooth the data sets, the same method found in section 5.2.1 is used but with a different number of neighbours. The 150 closest neighbours are taken for the acc_{sum} to form the average, while the same is done for the ω_z but with 100 neighbours instead. When the acc_{sum} both has been filtered and smoothed, it is also transformed by taking the square. There is no transformation done on the ω_z data. These data sets will now be referred to as $acc_{sum,filtered}$ and $\omega_{z,filtered}$.

5.3.4 Thresholds

The acceleration data can differ a lot between persons, while the ω_z is more consistent. Therefore, the threshold for $acc_{sum,filtered}$ will be dynamic and static for $\omega_{z,filtered}$. The thresholds are defined as

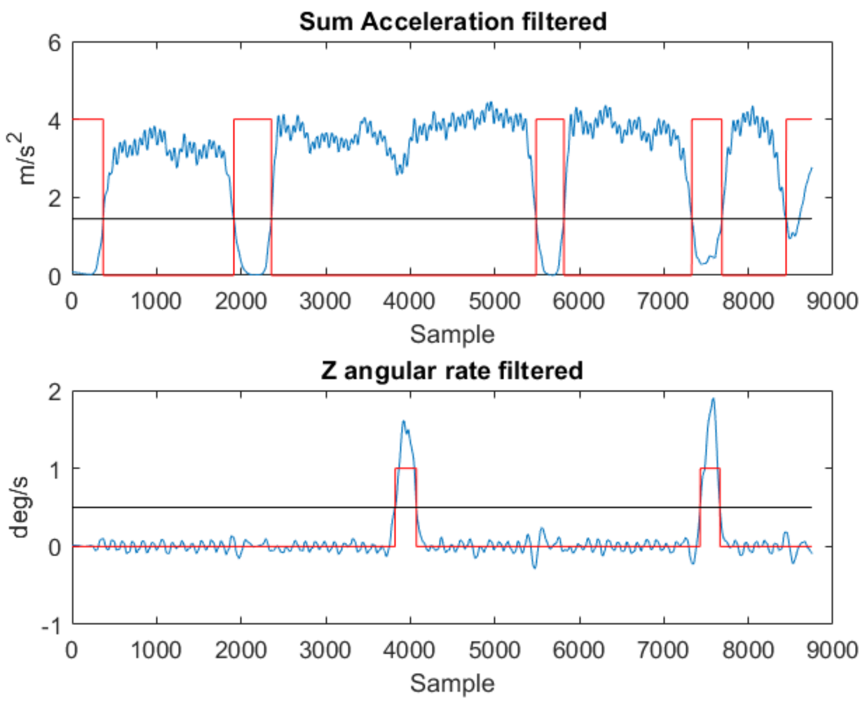

Figure 12: The filtered data sets plotted with their respectively thresholds (black) and logic's (red). The logic in the top image is 1 when a person is considered to be standing still, and 1 in the bottom image when a u-turn is being done.

$$thresh_{sum} = \frac{mean(acc_{sum,filtered})}{2}$$
$$thresh_{\omega_z} = 0.5$$

where they have been retrieved from empirical studies.

5.3.5 Results

Below in Figure 12 are the filtered data sets $acc_{sum,filtered}$ and $\omega_{z,filtered}$ illustrated with the thresholds. The red lines are the logics, describing standstills (when the red line is 1 in the top picture) or u-turns (when the red line is 1 in the bottom picture). Notice in the top picture that the person is considered to be standing still at the end although the $acc_{sum,filtered}$ is above the threshold $thresh_{sum}$ (black).

Although the u-turns are correctly located, they will be of no use when estimating distances. The non-linear walking models use the gait cycles to estimate the distance. Since one can be walking during a u-turn or just turning around on the spot, the main interest is to find when one is at a standstill. The splits are therefore solely based on the acceleration data. *Segmentation* will therefore make the split in the middle of a standing still time-window, which can be seen in Figure 13. In the same figure, the logics for the standstills and u-turns can also be seen with the unfiltered data acc_y and ω_z. In the top image are the splits indicated with a green vertical line, resulting in four smaller data sets.

Segmentation works as supposed for walks on plain ground, but problems may occur for walks in rougher terrains. Below in Figure 14 is a long walk for 460 meter on normal ground presented. The

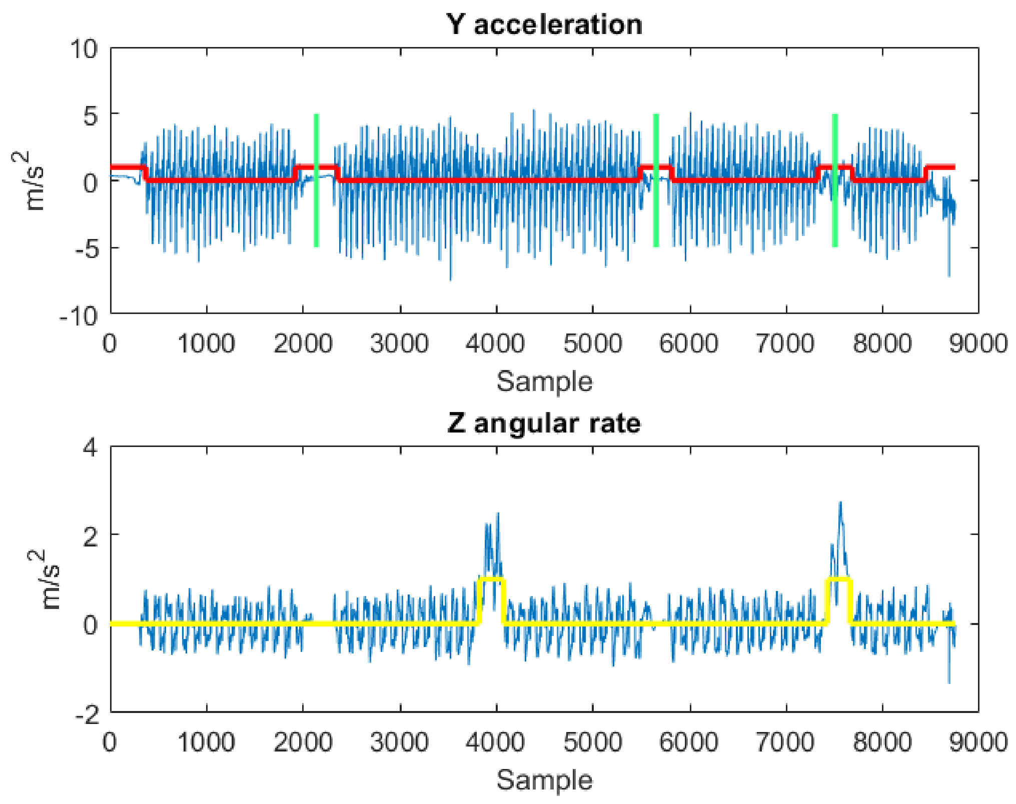

Figure 13: The data sets acc_y and ω_z presented with the logics of standstill (red line) and u-turn (yellow line). The green vertical lines indicates the the splitting occurs and results in four smaller data sets.

walk does not contain any standstill, but the function *Segmentation* falsely detects a split and wants to divide the data set into two smaller segments.

5.3.6 Analysis - Segmentation

The purpose of the function *Segmentation* is to split the data set into smaller pieces for a better analysis. Otherwise, some methods might not work as well, for example when calculating the periodic gait cycle (see section 5.4.1) and finding steps (see chapter 5.4).

The results of *Segmentation* are a bit mixed. It can locate u-turns for a simple walk, but they are of no use when estimating a distance. Because distance is calculated from walking, and a person could be walking or standing still when doing a u-turn. Therefore, u-turns do not provide any extra information, but can be retrieved from *Segmentation* if desired.

Segmentation finds the splits by comparing a sum of accelerations to a dynamic threshold. It works as expected for simpler walks in a plain terrain, but not always for more complex walks in normal terrain (see figure 14). The problem has to do with the measurements in z-acceleration. The walk seen in Figure 14 contains both up- and downhill motions, which certainly will affect the z-acceleration measurements. This leads to a standstill being falsely detected and a degraded estimation quality of the distance walked. The problem with shifts in acceleration measurements in rougher terrain are identified in [16] as well. It can also depend on a lot of other factors, like length of the walk and obstacles changing the behaviour of the person walking.

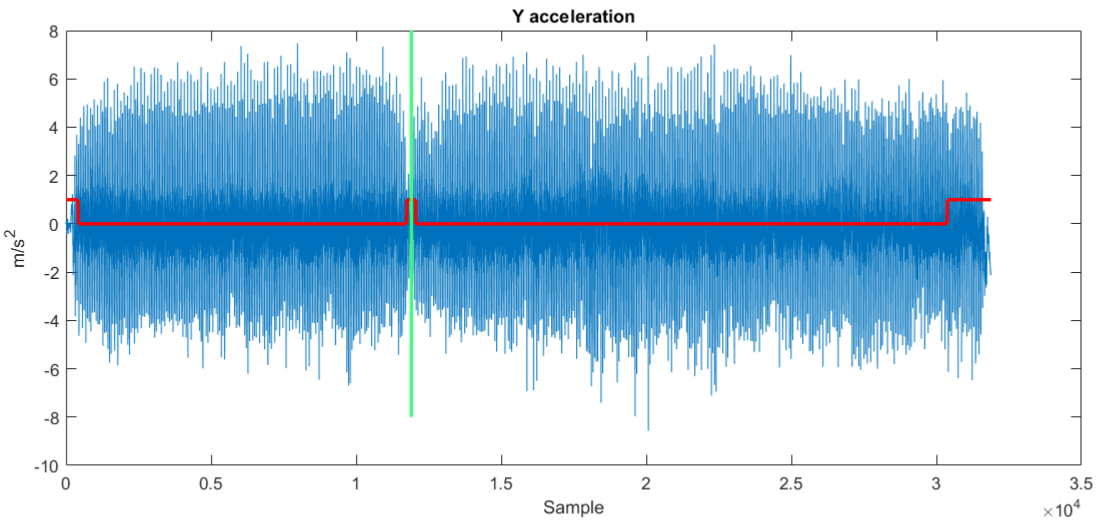

Figure 14: The acceleration in y-direction for a long walk in normal terrain. It is plotted with the logic for standstill (red) and a vertical line for predicted split (green). The split is falsely detected because there was no standstill during the walk.

The function *Segmentation* is used as normal for simple walks on plain ground where standstill is allowed and can be detected. Because the function is not as robust as desired, the investigated walks in normal terrain will in the remainder of this study not contain any standstills. The walks in normal terrain will not have any segmentation, but can be analyzed directly without splitting the data set. To be able to handle all cases of normal walking, the function needs more investigation and development.

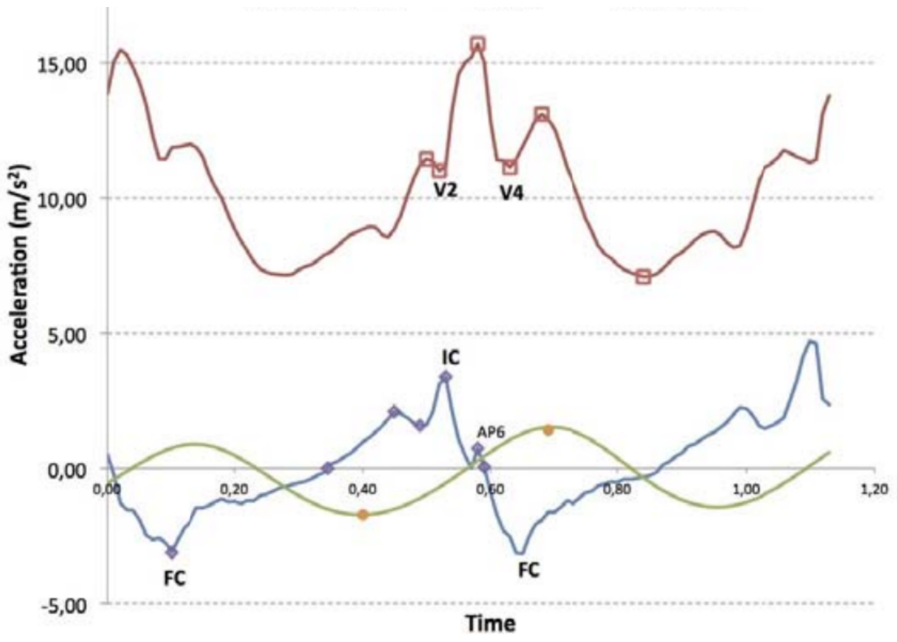

Figure 15: Figure borrowed from [1], to illustrate where to look for IC and FC in the y-directional accelerations. The blue graph in the low part of the figure is y-directional accelerations. It is seen that IC corresponds to the maximum acceleration in each step and likewise, FC corresponds to the minimum of the accelerations in each step. The red graph in the top of the figure shows the z-directional accelerations and V2, V4 corresponds to IC and FC respectively transformed to the z-directional accelerations.

5.4 ICFCyDetector - IC and FC detector in y-direction

After the data is cut and in some cases divided into segments, the next step in the process is to identify the approximate IC and FC with the help of the y-directional accelerations. The reason for working with the y-directional accelerations, as mentioned in the beginning of section 5, is that a lot of inspiration is drawn from the work of Alvarez, Alvarez, López and González [1]. From their work, the theory of where IC and FC are located, is used for identifying IC and FC for this project's walks. The location of IC and FC according to [1] are seen as $V2$ and $V4$ in Figure 15. The markers in the y-directional accelerations in Figure 16 correspond to where IC and FC should be detected in the walking data. The peaks are IC-marks and bottom minimums are FC-marks.

The algorithm is constructed in several different steps. First, the gait-cycle length (GCL) is estimated and will most commonly be around 60 samples long (0.6 s). The algorithm then goes through the y-direction accelerations with sliding windows of length $W = GCL$. An IC and FC should be found for every window length through the data set.

5.4.1 Estimating the length of the gait-cycle - GCL

As seen in figure 16, the gait is highly periodical which is used to find the ICs and FCs. Just as before, in the function *Cut*, is the ACF [13] used for finding the gait cycle. To easily find the patterns, the signal is first filtered through the same low-pass FIR filter described in section 5.3.3.

Secondly, the first peak corresponds to the correlation between the feet, where each step show similarities. The second peak corresponds to the body being in the same position again. The second peak has a much stronger correlation than the first, which is shown in Figure 17. The correlation for each foot

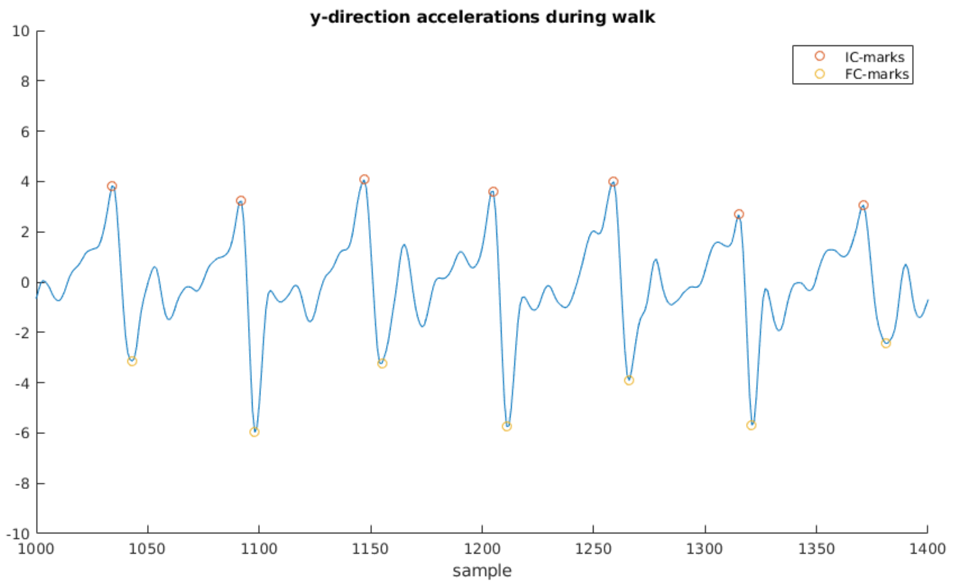

Figure 16: The y-acceleration measurements during a walk where IC (red marks) occurs periodically at every peak and FC (yellow mark) occurs periodically at every minimum.

individually is higher than compared between the feet. As it is seen in the top picture in Figure 17, the ACF has multiple peaks that are of no interest. The highest peak, present in the ACF, does not actually represent the gait cycle period. It represents the period of two gait-cycles. Using the Hamming filtering, a more clear analysis could be made by the function $ICFCyDetector$.

5.4.2 Finding ICs and FCs in y-acceleration

As mentioned, the algorithm works its way through the data set containing the y-direction accelerations by using windows of width W. If it starts at sample point i in the y-directional data, the window covers samples $i, i+1, \cdots, i+W-1, i+W$. In this window, all local maximum and minimum peaks are identified, which are all seen as potential IC and FC detections at this point.

The algorithm identifies the first maximum point in the current window which fulfills two things. First, the maximum point should be larger than the parameter $\max_{lower\_limit}$. Second, the point has a matching minimum point smaller than $\min_{upper\_imit}$. The matching minimum point must be located at most $d_{\text{max(ic-fc)}}$ from the maximum point. Note that both $\min_{upper\_limit}$ and $d_{\text{max(ic-fc)}}$ are parameters that should be tuned in order to minimize false detections and missing detections. In other words, the largest point, which has a small minimum point to the right of it, is identified. This follows the idea illustrated in Figure 16.

After detecting IC and FC points in that particular window, the windows are moved ahead to start from the detected IC point plus half of GCL. The windows are moved forward half a GCL since a new set of IC and FC points will not be found earlier than that. In the case where no suitable pair of IC and FC points are found in a window, the algorithm moves the window ahead by half of GCL as well, as this insures that not a pair of IC and FC points are missed when they are located precisely between two windows. In Algorithm .1 is the pseudo code for the y-directional detector presented.

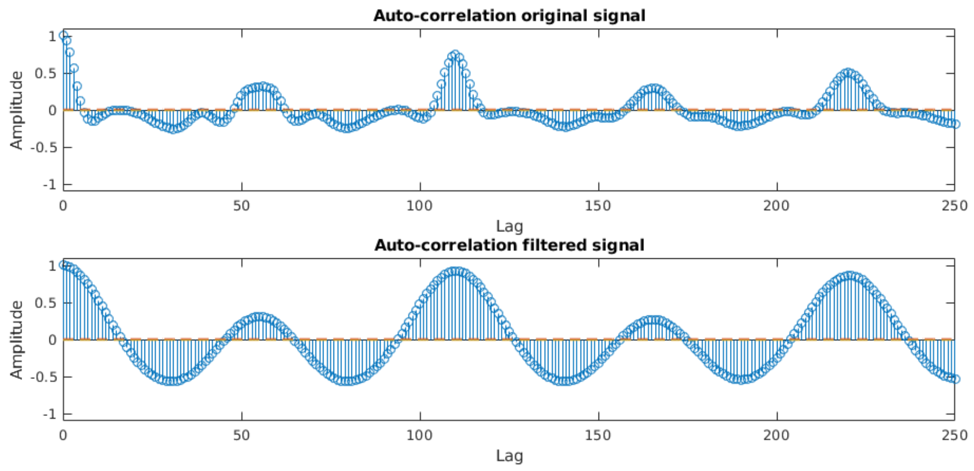

Figure 17: The auto-correlation function for the y-acceleration during walking. The top picture contains the ACF for the original signal while the bottom picture is the ACF for the filtered signal which is more clear. The sample where the first peak occurs corresponds to the number of samples of a gait-cycle.

Algorithm .1 Psuedo code for the IC and FC detector in y-direction.

```
1: Initialize constants, parameters and first window
2: repeat
3:     Search for max and min points in the window
4:     if If max and min point exists in the windows full filling conditions to be IC and FC then
5:         Let these points be IC and FC for the current window
6:         Set window to start at the newly detected IC plus one half window length, for next loop
7:     else
8:         Move window ahead to start 1 window length ahead for the next loop
9:     end if
10: until End of y-directional accelerations, and there is no windows left to test for IC and FC marks
```

5.4.3 Results

As can be seen in Figure 18, the algorithm of marking ICs and FCs is performing well. It will not yield as good results for all cases, but it handles the majority of walks almost flawlessly. The evaluation of the algorithm is done visually, where no false or missing detection of ICs and FCs have been identified during the testing.

5.4.4 Analysis - ICFCyDetector

The *ICFCyDetector* works extremely well, but the y-acceleration measurements are also the data set with the clearest features and patterns. There is an obvious periodical cycle which often can be seen by the naked eye, which makes it easy to determine the gait cycle length, GCL. Filtering away all high frequencies takes away most of the noise and disturbances, which makes a clear improvement in analyzing the period of the data set. One would think of perhaps updating the GCL during walking, but as long as a constant pace is held, it works fine. The next step of improvement would be just that, developing the algorithm so it can handle different types of speed.

In the second part of the algorithm, when forming the windows to locate the IC and FC, a naive approach of window sliding would be to just move them exactly GCL samples forward at every step.

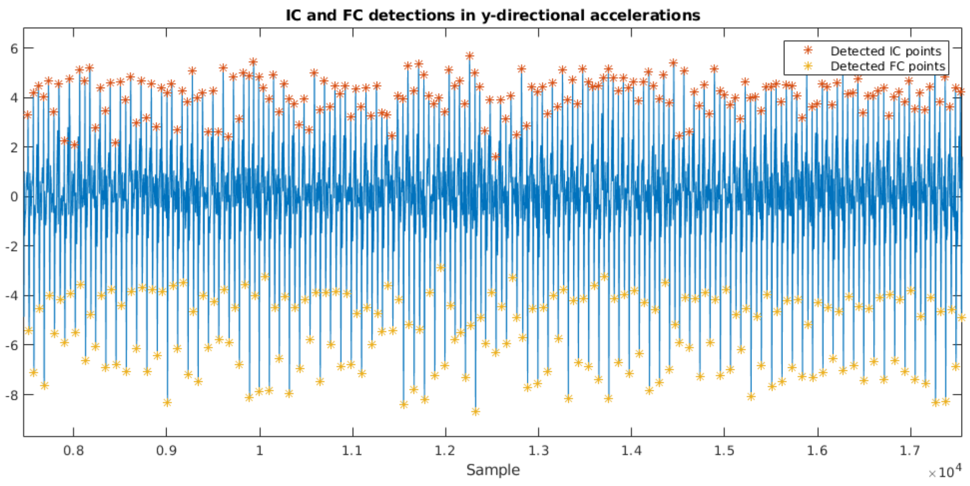

Figure 18: All ICs and FCs detected during a long walk, where red marks are ICs and yellow marks are FCs.

There is a likely possibility that the windows at some point will miss out on an IC and FC or have multiple true ICs and FCs inside the window. The approach now, when the window is moved relative to the last located IC, is much better. Problems of these types are therefore unlikely to occur.

The main point of *ICFCyDetector* is to help the following function *ICFCzDetector* in the estimation of ICs and FCs. Therefore, the function only tries to find the significant maximums and minimums located around the time points where ICs and FCs occur. As long as it is in the region, close to the true ICs and FCs, it helps enough for the function *ICFCzDetector* to properly do its job.

5.5 ICFCzDetector - IC and FC detector in z-direction

To distinguish the detections of ICs and FCs in each acceleration direction, the detection of ICs and FCs in y-acceleration by function *ICFCyDetector* will from now on be referred to as ICys and FCys. In the same way will the detections of the more accurate ICs and FCs in z-acceleration be referred to as ICzs and FCzs. Again, the theory from [1] is used, where the ICys and FCys are transformed to the positions $V2$ and $V4$ presented in Figure 15. There is no guarantee that ICs and FCs between the acceleration y- and z-measurements will occur at the same time point, which can be seen in Figure 19. Because the calculations of the step lengths are based on the occurrence of ICzs and FCzs, it is of great importance that they are located as well as possible.

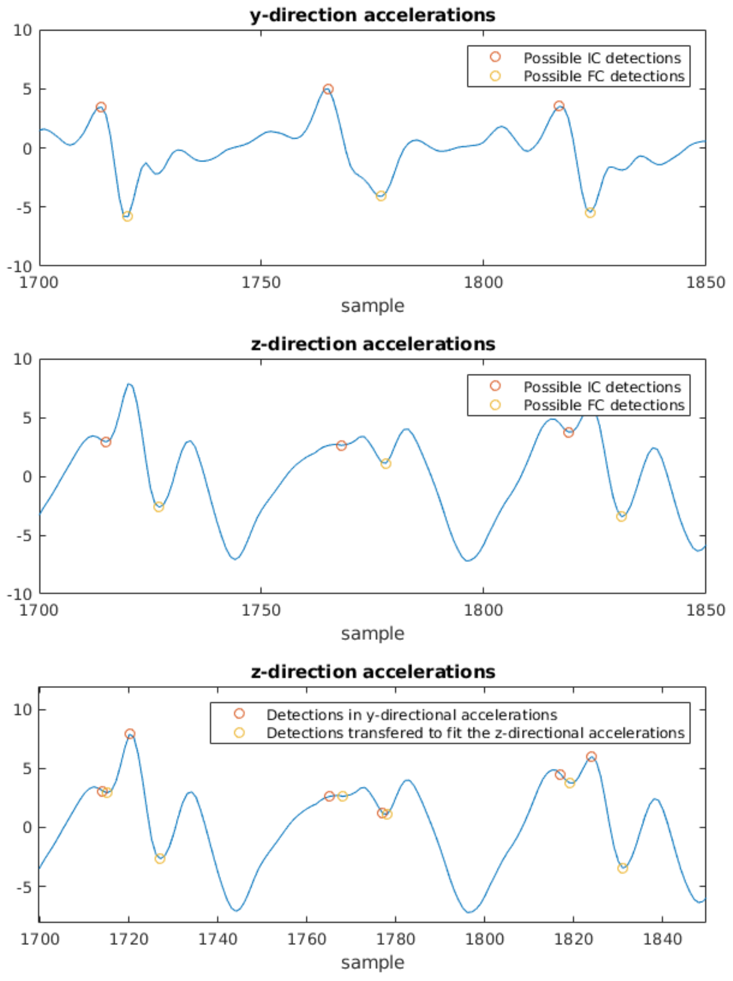

Figure 19: **Top**: The ICys and FCys from the y-directional detector plotted on the y-directional accelerations. **Middle**: The ICys and FCys, transformed to fit the z-directional acceleration in such a way that the ICzs and FCzs end up in the right place. **Bottom**: Both the ICys and FCys from the y-directional accelerations and the ICzs and FCzs shown to visualize how positions are changed.

5.5.1 Finding ICs and FCs in z-acceleration

The algorithm has three steps for finding ICzs and FCzs with the help of ICys and FCys. The first part takes care of all pairs of ICys and FCys points, which are easy to transform. The second part takes the remaining pairs of ICys and FCys points with transformed pairs of ICz and FCz points directly adjacent to the left and right of them. They are transformed with guidance by neighbouring ICzs and FCzs. The third and last part takes all remaining pairs and transforms them, starting with pairs who have ICzs and FCzs to the left or right, which will eventually estimate all ICzs and FCzs.

The first part of the algorithm adds all pairs that are easy to transform. By easy to transform, it means when there is exactly one local minimum in the neighborhoods of both the ICy and FCy. In other words, local minimums in the z-acceleration are searched for in an interval centered around the ICy and FCy points. See Figure 15 for a visual interpretation.

Different interval lengths are tried out, prioritized according to the list presented below, starting out with interval one and moving forward until interval five. For each interval, a check for two local minimums is performed and this would indicate that ICz and FCz are found. In the case when exactly 2 local minimums are found, those are accepted as points for ICz and FCz. ICz being the left local minimum and FCz as the right local minimum. If ICz and FCz have not been located after trying interval 5, the algorithm moves on to the the next pair of ICy and FCy.

a. $I_1 = [IC_z^i - \big(\max(FC_z^i - IC_z^i) + 0\big)\ ,\ FC_z^i + \big(\max(FC_z^i - IC_z^i) + 0\big)]$

b. $I_2 = [IC_z^i - \big(\max(FC_z^i - IC_z^i) + 6\big)\ ,\ FC_z^i + \big(\max(FC_z^i - IC_z^i) + 6\big)]$

c. $I_3 = [IC_z^i - \big(\max(FC_z^i - IC_z^i) - 6\big)\ ,\ FC_z^i + \big(\max(FC_z^i - IC_z^i) - 6\big)]$

d. $I_4 = [IC_z^i - \big(\max(FC_z^i - IC_z^i) + 12\big)\ ,\ FC_z^i + \big(\max(FC_z^i - IC_z^i) + 12\big)]$

e. $I_5 = [IC_z^i - \big(\max(FC_z^i - IC_z^i) - 12\big)\ ,\ FC_z^i + \big(\max(FC_z^i - IC_z^i) - 12\big)]$

When all obvious ICz and FCz points have been transformed, the algorithms moves forward to part two.

The second part of the algorithm adds all pairs that have ICz and FCz points directly adjacent on both sides. For every pair i of ICy and FCy that were not accepted in the first part of the algorithm and has accepted pairs of ICz and FCz, directly to the left and the right, the following is done: the algorithm looks for local minimums in

$$I_{y,IC}^i = \left[\frac{IC_z^{i-1} + IC_z^{i+1}}{2} \pm C\right] \qquad \text{and} \qquad I_{y,FC}^i = \left[\frac{FC_z^{i-1} + FC_z^{i+1}}{2} \pm C\right], \tag{5.12}$$

which are intervals centered between the $(i-1)$'th ICz and the $(i+1)$'th ICz point and $(i-1)$'th FCz and the $(i+1)$'th FCz points respectively detected in the y-directional accelerations. C is a tuning parameter and determines the width of the interval. As the gait cycle is assumed to be constant, the center point should be a good place to look for the i'th ICz and FCz. If at least one local minimum is found in both $I_{y,IC}^i$ and $I_{y,FC}^i$ then the smallest minimum is chosen in each interval as ICz and FCz for that step. In the case when no local minimum was found, the center point of the interval, that is $\frac{IC_z^{i-1}+IC_z^{i+1}}{2}$ and $\frac{FC_z^{i-1}+FC_z^{i+1}}{2}$ respectively, is chosen.

The third and last part adds all pairs that have ICy and FCy points not transformed yet. Starting from the left and alternating in adding points with a pair of ICz and FCz points two steps to the left and a pair of ICz and FCz points two steps to the right. Every time a new step is added, a check is done to determine whether or not part two of the method can be used for the next pair, as this could very well be the case after adding the pair.

The actual position of the ICz and FCz points are done similarly compared to the case of two neighboring pairs. Again if no minimums are found, the center point of the interval is chosen. The difference is that the used intervals are the ones below

$$I^i_{y,IC} = \begin{cases} \left[\frac{(IC^{i+2}_y - 2GCL)+(IC^{i+1}_y - GCL)}{2} \pm C\right] & \text{If pairs to the right are used and and both} \\ & IC^{i+2}_y \text{ and } IC^{i+1}_y \text{ has been transformed.} \\ \left[\frac{IC^{i+2}_y - 2GCL}{2} \pm C\right] & \text{If pairs to the right are used and} \\ & IC^{i+1}_y \text{ has not been transformed.} \\ \left[\frac{(IC^{i-2}_y + 2GCL)+(IC^{i-1}_y + GCL)}{2} \pm C\right] & \text{If pairs to the left are used and both} \\ & IC^{i-2}_y \text{ and } IC^{i-1}_y \text{ has been transformed.} \\ \left[\frac{IC^{i-2}_y + 2GCL}{2} \pm C\right] & \text{If pairs to the right are used and} \\ & IC^{i-1}_y \text{ has not been transformed.,} \end{cases} \tag{5.13}$$

and in the same manner,

$$I^i_{y,FC} = \begin{cases} \left[\frac{(FC^{i+2}_y - 2GCL)+(FC^{i+1}_y - GCL)}{2} \pm C\right] & \text{If pairs to the right are used and both} \\ & FC^{i+2}_y \text{ and } FC^{i+1}_y \text{ has been transformed.} \\ \left[\frac{FC^{i+2}_y - 2GCL}{2} \pm C\right] & \text{If pairs to the right are used and} \\ & FC^{i+1}_y \text{ has not been transformed.} \\ \left[\frac{(FC^{i-2}_y + 2GCL)+(FC^{i-1}_y + GCL)}{2} \pm C\right] & \text{If pairs to the left are used and both} \\ & FC^{i-2}_y \text{ and } FC^{i-1}_y \text{ has been transformed.} \\ \left[\frac{FC^{i-2}_y + 2GCL}{2} \pm C\right] & \text{If pairs to the right are used and} \\ & FC^{i-1}_y \text{ has not been transformed.,} \end{cases} \tag{5.14}$$

The pseudo code for this algorithm is presented in Algorithm .2 below.

Algorithm .2 Psuedo code for the IC and FC detector in z-direction.

```
1: Initialize constants, parameters
2: Import IC and FC detections from y-directional detector
3: Initialize list $L$ of IC and FC points to be detected in z-directional accelerations
4: for Each pair of IC and FC points in $L$ do
5:     Form interval surrounding the current pair of IC and FC from y-directional accelerations
6:     Search for min points in interval
7:     if exactly two min points exists fulfilling conditions of being IC and FC in z-directional accelerations then
8:         Let these points be IC and FC for the current current interval
9:         Remove current IC and FC point from $L$
10:    end if
11: end for
12: for Each pair of IC and FC points in $L$ with neighbouring IC and FC pairs already removed from $L$ do
13:    Form IC and FC intervals around both current IC and current FC from $L$, centered at current IC and FC
14:    Search for min points in both intervals
15:    if Min points exists in IC interval then
16:        Let the smallest minimum point be IC for the current interval
17:    else
18:        Let the center point of the IC interval be IC for the current interval
19:    end if
20:    if Min points exists in FC interval then
21:        Let the smallest minimum point be FC for the current interval
22:    else
23:        Let the center point of the FC interval be FC for the current interval
24:    end if
25:    Remove current IC and FC point from $L$
26: end for
27: for Each pair of IC and FC points in $L$, alternating from left to right in $L$ do
28:    Form interval based on already detected IC and FC points in around current pair of IC and FC points in $L$
29:    Search for min points in both intervals
30:    if Min points exists in IC interval then
31:        Let the smallest minimum point be IC for the current interval
32:    else
33:        Let the center point of the IC interval be IC for the current interval
34:    end if
35:    if Min points exists in FC interval then
36:        Let the smallest minimum point be FC for the current interval
37:    else
38:        Let the center point of the FC interval be FC for the current interval
39:    end if
40:    Remove current IC and FC point from $L$, and let them be detected
41: end for
```

5.5.2 Results

The resulting algorithm is tested on all sets used to detect ICy and FCy points in the y-directional accelerations. These detections are used as input to the ICFCzDetector function. The transformation of the function proved harder than expected for some cases. As mentioned in section 5.4, the z-directional

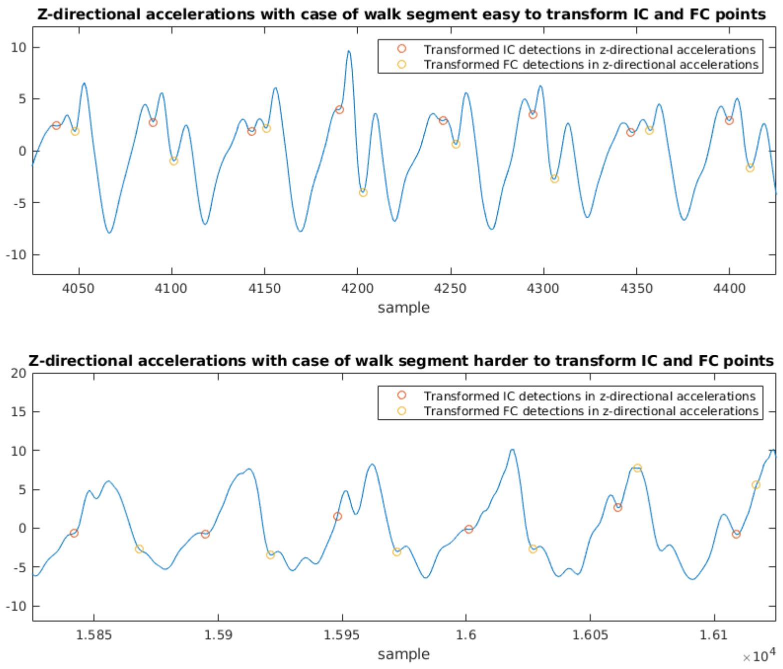

Figure 20: **Top**: A case of a walk segment where the algorithm has no big problems in transforming the ICy and FCy points with good precision. The accelerations looks as expected and the transformed IC and FC points ends up in the right valleys. **Bottom**: A case of a walk segment where the algorithm has problems in transforming the ICy and FCy points with good precision. The accelerations in this case do not look as expected, which causes problems.

detector detected ICy and FCy to almost 100% of the cases when studying the result visually. The top part of figure 20 shows a segment where the detector succeeds in transforming the IC and FC points correctly. Note that in a lot of cases, there is more ambiguity and the z-directional accelerations in each step vary from step to step as compared to the y-directional accelerations. The bottom of Figure 20 illustrates a good example of a walking segment that is not very easy to decide or detect where the correct transformed ICz and FCz should end up. The identification of the local minimums are not very clear, and in some cases, non-existent.

5.5.3 Analysis - ICFCzDetector

The *ICFCzDetector* performs decently, but compared to *ICFCyDetector*, it is not as easy to tell whether it finds the true ICzs and FCzs. This is due, mostly, to the z-acceleration measurements being hard to analyze, lacking the wanted features such as local minimums. Although, the function *ICDCzDetector* becomes robust, because the expected time points of ICz and FCz can be calculated from close laying and known ICzs and FCzs.

The three parts of the algorithm try to catch every case of walking, but might still miss special difficult cases which have not occurred during this project. For example, it is always expected that some ICzs and FCzs will be found in part 1 of the algorithm. One could prepare for a case where this does not happen, but such a walk, which do not contain any significant features, will most likely not be good for this type of distance estimation anyway. The form and definition of each part has been fit to the empirical tests during this project.

In summary, *ICFCzDetector* is working perfectly for z-accelerations with clear features and patterns, while it can struggle and almost have to guess where to mark an ICz or FCz. It leads to a more insecure estimation of ICzs and FCzs which affect the step length estimation and in the end, the calculated distance. Therefore it is of greatest importance to make *ICFCzDetector* as good as possible, but it is still limited by z-acceleration lacking the features it should have according to the theory found in [1].

5.6 Walk Length

As mentioned in the beginning of this chapter, it is not an easy task to estimate the walked distance without the use of a GPS. The difficulty when only using accelerometers and gyroscopes is the drift, which occurs because of measurement errors. There are different ways around this problem, where the chosen one for this project is by estimating step lengths with non-linear step models. In the following, the non-linear step length models will be referred to as step models.

The algorithms, *ICFCyDetector* and *ICFCzDetector*, recognizes IC and FC for each step taken. The recognition is essential for individual analysis of each step, where the step models either need the measurements between the time points FCz to ICz for the same foot or ICz to ICz from one foot to the other. The step models estimate the step lengths by analyzing the acceleration in z-direction, which is why finding the best estimation of ICz and FCz is of great importance.

The models are typically applied when using IMS attached to the lower back, which is as close as possible to the body's COG. Having the smart phones in a belt pack on the stomach will most likely affect the IMS measurements and the step models are therefore expected to be scaled with a constant K.

5.6.1 Non-linear models for estimating step lengths

There were eight step models found relevant for this project, which are all presented below:

Step model 1 [38, 18]

$$l_1 = K \cdot \sqrt[4]{max(acc_z) - min(acc_z)} \tag{5.15}$$

Step model 2 [38]

$$l_2 = K \cdot \sqrt[3]{\frac{\sum_{i=1}^{N} |acc_{z,i}|}{N}} \tag{5.16}$$

Step model 3 [38]

$$l_3 = K \cdot \frac{\frac{\sum_{i=1}^{N} |acc_{z,i}|}{N} - min(acc_z)}{max(acc_z) - min(acc_z)} \tag{5.17}$$

Step model 4 [9]

$$l_4 = K \cdot 2\sqrt{2L_4 h - h^2} + 0.75F \tag{5.18}$$

Step model 5 [1]

$$l_5 = K \cdot 2\sqrt{2L_2 h - h^2} + 0.75F \tag{5.19}$$

Step model 6 [9]

$$l_6 = K \cdot 2\sqrt{2L_3 h - h^2} \tag{5.20}$$

Step model 7 [18]

$$l_7 = K \cdot 2\sqrt{2L_1 h - h^2} \tag{5.21}$$

Step model 8 [5]

$$l_8 = K \cdot \left(\sqrt{2L_1 h_1 - h_1^2} + \sqrt{2L_1 h_2 - h_2^2}\right) \tag{5.22}$$

Here, step models 1-5 estimate the step length during the time period from FCz to ICz, and the step models 6-8 estimate from ICz to ICz. The notation acc_z is the acceleration sequence in z-direction during that time period for each step model. The models 4-8 contains the notation L_k $(k = 1, 2, 3, 4)$, which corresponds to specific leg lengths for each person. The parameter F found in step model 4 and 5 is the feet length.

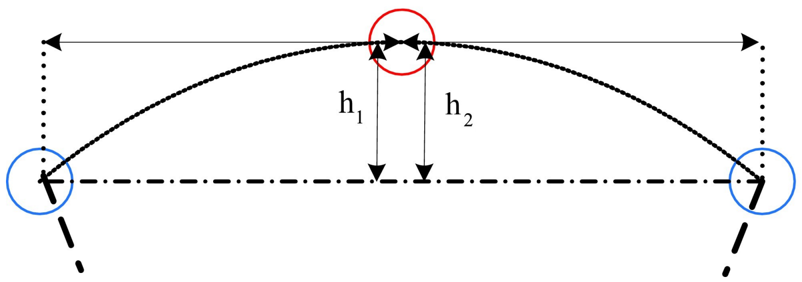

Figure 21: A zoomed in image of the swing phase during walking. The heights h_1 and h_2 represents the maximum height differences during the first half of the swing phase respectively the second half [5].

The variable h is the maximum height the COG is translated in the z-direction during a swing phase. The variables h_1 and h_2 corresponds to the maximum height difference during the first half of the swing phase respectively the second half. An illustration of h is shown in Figure 6 above in the beginning of this chapter, where h_1 and h_2 can be seen in Figure 21.

The calculations of h, h_1 and h_2 is done by forming the recursive mechanization equations in (5.23) [23], where a_z is the acceleration measurements in z-direction. The equations in (5.23) uses double-integration, which gives a drift if done over long time periods. Since a step is considered to take a short amount of time, the double-integration is no problem here.

$$\begin{aligned} p_{z,k+1} &= p_{z,k} + v_{z,k}dt + \frac{a_{z,k}^2}{2}dt \\ v_{z,k+1} &= v_{z,k} + a_{z,k}dt \end{aligned} \tag{5.23}$$

Here, $p_{z,k}$ is the position in z-direction with starting value $p_{z,0} = 0$. The velocity $v_{z,k}$ is defined in the same way, also with starting value $v_{z,0} = 0$. The variable h is calculated as the maximum difference in the $p_{z,k}$ sequence. The variables h_1 and h_2 are in the same way calculated as the maximum for the first respectively second half of the same sequence $p_{z,k}$ [5, 1, 23].
By estimating each step length and then summing up, the total walked distance can be calculated for each foot. Since each foot walk the total distance, the average between them forms the estimated distance length.

5.6.2 Replacing outliers

The steps are estimated by non-linear models which are not perfect and each step may differ a bit from the true one. When the walking pace is around a constant level, the estimated steps could empirically be seen as normal distributed. In Figure 22 below, the estimated steps from person 2 walking 460 m around Sjönsjön at LTH (see section 5.6.4) are presented as a histogram with a normal distribution fit to the steps. A normal distribution seems to fit decently when analyzed empirically. Therefore, the steps outside of a 95% confidence interval will be considered as outliers and replaced by the mean of a step during the walking period.

5.6.3 Scaling constant K

Most models from equations (5.15)-(5.22) are adapted to IMS measuring at the back (to get as close as possible to COG). The models might not function exactly the same for IMS measuring at the stomach, therefore is the scaling constant K applied to handle the deviation better. Some of the models already use scaling constant by definition, for example does step model 4 (equation 5.18) tend to underestimate

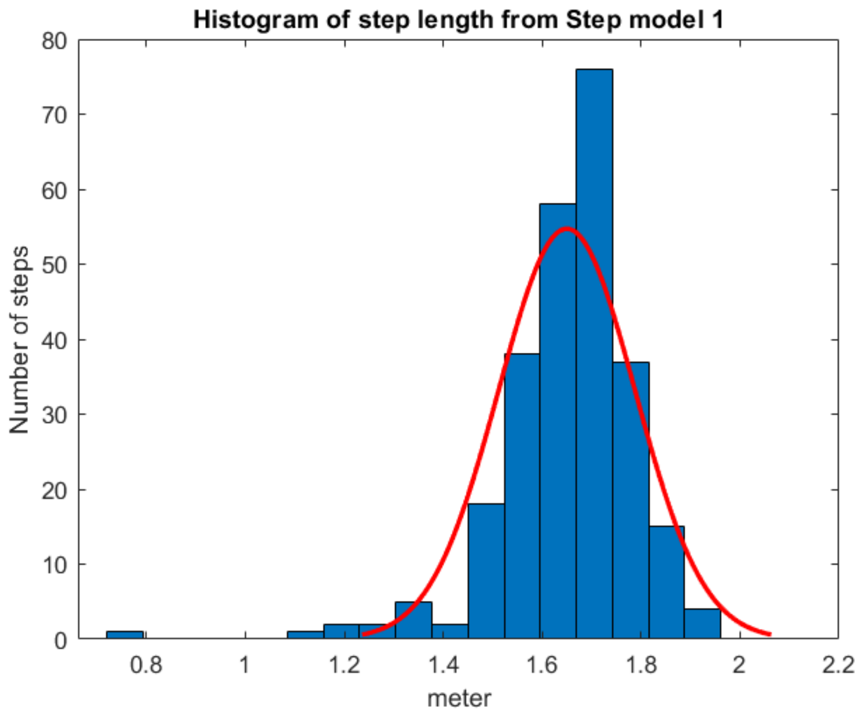

Figure 22: A histogram presenting the empirical distribution of steps taken in step model 1. The red line corresponds to a Gaussian distribution fitted to the the empirical distribution of the steps.

and is usually scaled with $K = 1.25$ during IMS measuring at the back [9].

When the true distance of a walk is known, the scaling constant K can be estimated. Different persons walking the same distance will have a deviation among their results. The most optimal individual K would for each person scale their measured distance to its true one. The most optimal K for a step model would minimize the error regarding all walks for a known distance. This is done by estimating K using the LS-method [13].

5.6.4 Results

Four tests have been setup to decide which step models are of relevance for estimating the scaling constant K. The aim is to find the best performing step model which also is the most convenient. Therefore, the performance forms a trade-off against inconvenience, where convenience is prioritized as long as the model is performing well.

The first test selects the step models of interest for further investigation, while the next three tests are of varying types to get a good estimation of K. When the chosen step models have been estimated, they are tried out on validation data to see their performance.

The first test determines which step models that are relevant. Four persons walked two walks each on a straight forward distance of 20 m. These results are presented in Table 4 below. The optimal K for each model is estimated by the LS-method, which results can be seen in Table 5. Further, the scaled distances (scaled with their respectively scaling constant K) are presented in table 6. The scaled distances mean percentage error and standard deviation are seen in Table 7.

Table 4: Eight walks by four different persons on a 20 m straight forward path. The models are consistent, but the majority is in need of scaling to correctly estimate 20 m.

	Step model 1	**Step model 2**	**Step model 3**	**Step model 4**	**Step model 5**	**Step model 6**	**Step model 7**	**Step model 8**
Person 1	19,87 m	15,57 m	9,61 m	13,08 m	13,32 m	9,25 m	10,12 m	8,24 m
Person 1	20,24 m	15,87 m	9,48 m	12,74 m	12,98 m	10,06 m	11,01 m	8,32 m
Person 2	19,32 m	14,75 m	12,33 m	13,25 m	13,50 m	10,69 m	11,69 m	8,18 m
Person 2	19,92 m	15,08 m	11,29 m	13,75 m	14,01 m	9,35 m	10,21 m	8,08 m
Person 3	20,58 m	15,31 m	11,06 m	13,33 m	13,64 m	9,87 m	10,68 m	8,10 m
Person 3	21,12 m	15,92 m	10,72 m	13,16 m	13,47 m	10,16 m	11,01 m	8,35 m
Person 4	20,37 m	15,24 m	10,12 m	12,32 m	12,55 m	8,74 m	9,48 m	8,04 m
Person 4	20,57 m	15,63 m	10,15 m	13,07 m	13,32 m	8,61 m	9,35 m	7,64 m

Table 5: The scaling constant K corresponding to each step model to make each distance scale as best as possible to 20 m.

	Step model 1	**Step model 2**	**Step model 3**	**Step model 4**	**Step model 5**	**Step model 6**	**Step model 7**	**Step model 8**
K	0,99	1,30	1,87	1,53	1,50	2,07	1,91	2,46

As can be seen in Table 7, the distance mean percentage error and its standard deviation are the lowest for step models 1, 2, 4, 5 and 8. The step models 3, 6 and 7 all have big errors and are therefore of less relevance for further analysis. Since the step models 4-8 uses input variables such as foot length and leg length, they are considered to be inconvenient and less relevant. The step models 1, 2 and 3 do not have any extra input variables, where 1 and 2 perform as well as any other model. Because of the simplification of no extra input variables, the step models 1-3 are chosen for further analysis. Even though step model 3 performs badly it is easily implemented without any extra input variables, and can function as a benchmark for further analysis.

In the second test, the models performances were tried out on a longer range of path. Two test persons walked four times each. Of the four walks, two were 160 m with three turnarounds just on the spot and two were 168 m walks with three wide turnarounds. The walks with a turnaround on the spot are divided into segments according to the function *Segmentation* in section 5.3. The person walking is not truly standing still while doing a wider turnaround, so the measured data is not divided into segments for these cases. The scaling constants K for each distance with their mean percentage error and its standard deviation are presented in table 8. Further-more, the scaled distances for each test are shown in Table 9.
The third test was about estimating the distance while there were stops during the walk. The total distance was 90 m with two stops at 20 and 60 m. Two turnarounds on the spot were done at the distances 40 and 80 m. Due to the stops and turnarounds, the data is segmented in accordance with the function *Segmentation* described in section 5.3.

Table 6: The distances for each step model scaled with their respectively K. The distances lie close to 20 m now, but some deviates more than others.

	Step model 1	**Step model 2**	**Step model 3**	**Step model 4**	**Step model 5**	**Step model 6**	**Step model 7**	**Step model 8**
Person 1	19,61 m	20,19 m	18,01 m	19,96 m	19,94 m	19,20 m	19,28 m	20,29 m
Person 1	19,98 m	20,57 m	17,77 m	19,45 m	19,43 m	20,87 m	20,97 m	20,48 m
Person 2	19,07 m	19,12 m	23,11 m	20,23 m	20,21 m	22,18 m	22,27 m	20,13 m
Person 2	19,66 m	19,54 m	21,17 m	20,99 m	20,97 m	19,39 m	19,46 m	19,89 m
Person 3	20,32 m	19,84 m	20,73 m	20,35 m	20,42 m	20,48 m	20,36 m	19,94 m
Person 3	20,84 m	20,63 m	20,09 m	20,10 m	20,16 m	21,09 m	20,97 m	20,55 m
Person 4	20,11 m	19,76 m	18,96 m	18,81 m	18,78 m	18,13 m	18,06 m	19,80 m
Person 4	20,31 m	20,26 m	19,02 m	19,96 m	19,94 m	17,87 m	17,81 m	18,82 m

Table 7: The distance mean percentage error and its standard deviation for each step model after scaling with their respectively constant K. Low mean percentage error and low standard deviation correspond to a more consistent and robust model between persons.

	Step model 1	Step model 2	Step model 3	Step model 4	Step model 5	Step model 6	Step model 7	Step model 8
Distance mean perc. err.	2,03	2,12	7,09	2,17	2,29	6,26	6,22	1,87
Distance mean perc. err. std.	2,70	2,60	8,95	3,20	3,27	7,53	7,67	2,73

Table 8: The estimated scaling constants K estimated for each model and for the two walks on 160 respectively 168 m.

	Step model 1	Step model 2	Step model 3
160 m			
K	0,97	1,26	1,94
Mean perc, error	1,41	1,34	3,69
Mean perc, Error std,	0,30	0,44	2,39
168 m			
K	0,96	1,25	1,98
Mean perc, error	1,75	2,10	3,48
Mean perc, Error std,	0,76	0,90	1,46

The test was performed with nine new test persons and two previous ones. Just as before, the scaling constants with the distance mean percentage errors and its standard deviation can be seen in Table 10. The estimated distances scaled with their respectively constant can be seen in table 11.
The fourth and last test was another walk outdoor around the lake sjonsjon at LTH. The walking distance was 460 m, which was measured both by GPS tracking system [31] and by Google maps measuring system [11]. Notably is that the path has changes in altitude, which makes it troublesome for *Segmentation* to split up data correctly if standing still. Therefore, all walks were performed without any stand still. Seven persons did the test, out of which three were new test subjects. The walking path can be seen in Figure 23 where the starting point is marked in yellow.
In the same way as before, the scaling constant, K, was estimated using the LS-method. The estimations of K with the distance mean percentage error and its standard deviation are presented in table 12. The resulting distances can be seen in table 13.

Table 12: The scaling constants and the distance mean percentage error and its standard deviation presented for the walk of 460 m around lake Sjonsjon at LTH.

	Step model 1	Step model 2	Step model 3
K	0,98	1,23	1,97
Distance mean perc. error	4,83	4,17	4,55
Distance mean perc. error std.	2,39	2,94	4,68

The four tests have all estimated a scaling constant K, fitted specifically to each test. They can all be found in the tables 5, 8, 10 and 12. A summarize of the former estimated Ks can be seen in table 14 below. The scaling constants for future distance estimations are defined as the mean between all former estimated Ks.

Table 9: The estimated distances for the three step models on 160 m and 168 m.

	Step model 1	**Step model 2**	**Step model 3**
160 m			
Person 1 (160 m)	161,94 m	161,31 m	166,12 m
Person 1 (160 m)	162,50 m	162,93 m	165,14 m
Person 2 (160 m)	157,19 m	157,53 m	158,44 m
Person 2 (160 m)	158,23 m	158,11 m	149,17 m
168 m			
Person 1 (168 m)	161,50 m	161,57 m	166,82 m
Person 1 (168 m)	163,98 m	165,00 m	163,88 m
Person 2 (168 m)	156,32 m	156,16 m	151,77 m
Person 2 (168 m)	157,98 m	156,94 m	156,65 m

Table 10: The scaling constants K for eleven test persons on a 90 m walk with stops and turnarounds. Notice that the mean percentage error and its standard deviation have become higher compared to before.

	Step model 1	**Step model 2**	**Step model 3**
K	0,92	1,19	1,58
Mean perc. error	3,54	4,41	10,60
Mean perc. Error std.	3,62	3,17	5,76

Table 14: The K_{mean} for future distance predictions presented, which has been estimated by the mean of K_is from previous tests.

	Step model 1	**Step model 2**	**Step model 3**
K_1	0,99	1,30	1,87
K_2	0,97	1,26	1,94
K_3	0,96	1,25	1,98
K_4	0,92	1,19	1,58
K_5	0,98	1,23	1,97
K_{mean}	0,96	1,25	1,87

5.6.5 Predicting distance

As a final test, the estimation of the most optimal scaling constants, K, have been defined as the K_{mean}, which are found in Table 14. These are now used to estimate the distance for seven completely new test subjects, each walking 400 m outdoor on a training track. The final results are presented in Table 16 below, where its scaling constants K_{mean} and the mean distance percentage error and its standard deviation can be seen in Table 15.

Table 11: The estimated distances for the 90 m walk with two stops and two turnarounds.

	Step model 1	**Step model 2**	**Step model 3**
Person 1	92,45 m	91,53 m	83,00 m
Person 2	86,05 m	82,36 m	75,21 m
Person 5	92,85v	95,93 m	97,69 m
Person 6	90,63 m	92,06 m	102,90 m
Person 7	90,30 m	88,38 m	78,86 m
Person 8	79,87 m	81,25 m	110,95 m
Person 9	89,97 m	88,85 m	79,73 m
Person 10	92,02 m	95,24 m	86,68 m
Person 11	87,78 m	89,76 m	82,52 m
Person 12	90,67 m	91,90 m	92,03 m
Person 13	85,89 m	84,04 m	95,98 m
Person 14	98,91 m	95,57 m	79,05 m

Table 13: The estimated distances for a walk of 460 m outdoor on asphalt around the lake Sjonsjon at LTH, shown in Figure 23.

	Step model 1	**Step model 2**	**Step model 3**
Person 1	481,86 m	478,58 m	478,91 m
Person 2	443,96 m	452,48 m	474,11 m
Person 7	465,19 m	457,20 m	469,94 m
Person 9	484,08 m	483,21 m	452,82 m
Person 15	435,80 m	444,23 m	455,00 m
Person 16	482,05 m	481,29 m	483,88 m
Person 17	418,01 m	415,06 m	392,62 m

Table 15: The scaling constant K_{mean} is defined as the most optimal scaling constant for future predictions of the distance walked. Notice the difference between the distance mean percentage error and its standard deviation between each step model.

	Step model 1	**Step model 2**	**Step model 3**
$\boldsymbol{K_{mean}}$	0.96	1.25	1.87
Distance mean perc. error	3,87	7,38	6,92
Distance mean perc. Error std.	3,62	5,47	3,88

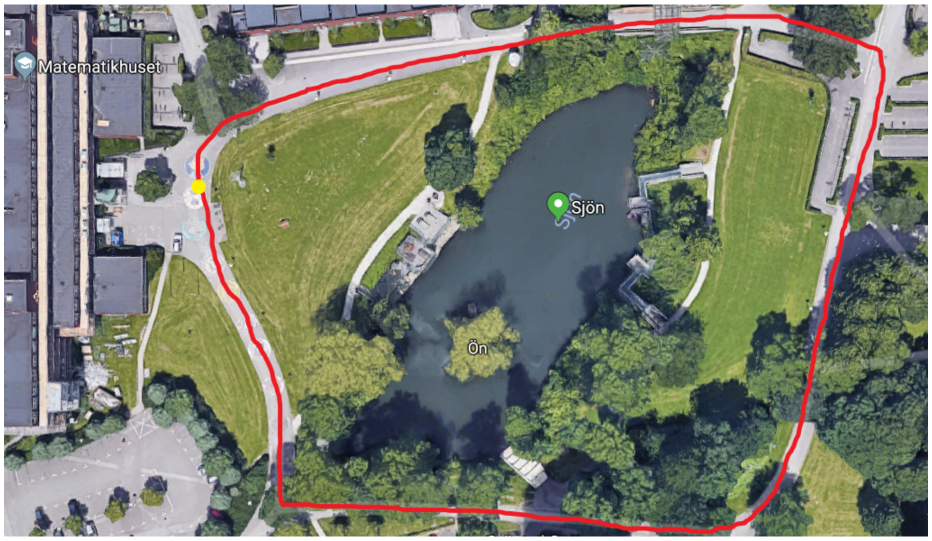

Figure 23: The walking path for seven test subjects where the walking distance is 460 m [11, 31].

Table 16: The final results of a 400 m walking test performed by seven test subjects with no stops during the walk.

	Step model 1	**Step model 2**	**Step model 3**
Person 18	393,71 m	413,40 m	376,24 m
Person 19	393,29 m	393,35 m	448,42 m
Person 20	415,62 m	455,10 m	419,09 m
Person 21	446,86 m	465,14 m	410,42 m
Person 22	406,27 m	421,19 m	411,31 m
Person 23	410,63 m	423,32 m	436,02 m
Person 24	416,06 m	421,70 m	444,75 m

As can be seen in the tables above. the results when predicting distances are a bit worse than before, but still decently good. In this case, the optimal scaling constant K is not used. Instead the estimated K_{mean} is used. The distance mean percentage error and its standard deviation is a bit higher, which is expected.

5.6.6 Analysis - Walk Length

The aim of this section was to find the best non-linear step models for estimating distance. In five different tests were the best step models selected, their scaling constant K for each test calculated, and finally, a K_{mean} was formed.

The choice of which step models to use is not an easy one. The main priority of this project is to simplify the tests and make them more user-friendly. Therefore, no-input models are strongly preferred compared to models needing input. When the the first three step models perform as good as one of the latter, they are prioritized.

The first three step models (5.15)-(5.17) were implemented for further studies, due to their simplicity.

The calculated scaling constants K_i were similar for each test, but always lower than the mean during the fourth test of the 90 m walk. This might be due to the many stops and splitting of the data, but it is also the only test with twelve unique persons. Otherwise it was only a maximum of six different persons doing a specific test. In the second test, only two persons are walking and the distance mean percentage error and its standard deviation is really low. This tells us that the scaling constant K seems to be specific to each person. The errors will then be low if there are only two persons and their Ks are by chance similar to each other. K is here considered to be a constant and not dependent on anything, but it might have dependence with related variables such as leg length and feet length. Due to lack of time has this not been investigated in this project, but could be done as future work.

Expecting each step length to be similar to each other, it was shown empirically that they follow a normal distribution. By replacing outliers with the mean step length of the walk, there was some decrease in the distance mean percentage error and its standard deviation.

5.7 Correlation K

It is possible that there might be ways to increase the performance of the step models, e.g. one could investigate dependencies of the scaling constant K. Since its introduction, K has only been considered as just a scaling constant with no dependency to any other factors. In this section, Ks is investigated to see for any linear dependencies with the factors: distance, leg length and foot length. Because step model 1 showed the best performance, it is used throughout this whole section and therefore, K should be interpreted as the scaling constant for step model 1.

5.7.1 Linear dependency

For the comparison between K and an external factor, the true value for K is needed for each walked distance. Therefore, every walk considered here, is scaled to its true distance, meaning that K becomes the best scaling constant K for the data at hand. Thereafter, the considered factors of distance, leg length and foot length are compared to their respectively K, to see whether K follows a linear dependency or not. To determine the linear relationship, a linear regression model is used and defined as the following:

$$y_i = \alpha + \beta x_i + \epsilon_i, \quad \epsilon_i \in N(0, \sigma) \tag{5.24}$$

Where x_i are the measurements of the different external factors and y_i is the calculated scaling constant K for each specific walk. The parameters α, β are estimated as

$$\begin{aligned} \beta^* &= \frac{S_{xy}}{S_{xx}} \\ \alpha^* &= \overline{y} - \beta^* \overline{y} \\ S_{xx} &= \sum_{i=1}^{n} (x_i - \overline{x})^2 \\ S_{xy} &= \sum_{i=1}^{n} \Big((x_i - \overline{x})(y_i - \overline{y}) \Big) \end{aligned}$$

Here, n denotes the number of measurements, σ the noise variance, and $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, where $\overline{y}$ is defined likewise. The parameters $\alpha*$, β^* are normally distributed

$$\alpha* \in N\left(\alpha, \sigma\sqrt{\frac{1}{n} + \frac{\overline{x}^2}{S_{xx}}}\right) \tag{5.25}$$

$$\beta^* \in N\left(\beta, \frac{\sigma}{\sqrt{S_{xx}}}\right) \tag{5.26}$$

The 95%-confidence interval for the parameters

$$\begin{aligned} I_{\alpha^*} &= \alpha^* \pm \lambda_{0.025} \sigma \sqrt{\frac{1}{n} + \frac{\overline{x}^2}{S_{xx}}} \\ I_{\beta^*} &= \beta^* \pm \lambda_{0.025} \frac{\sigma}{\sqrt{S_{xx}}} \end{aligned}$$

The interesting parameter here is β^*, which corresponds to the slope of the linear trend. If its confidence interval is strictly separated from zero, then there is a 95% confidence that there is a linear trend between K and the investigated factor [3].

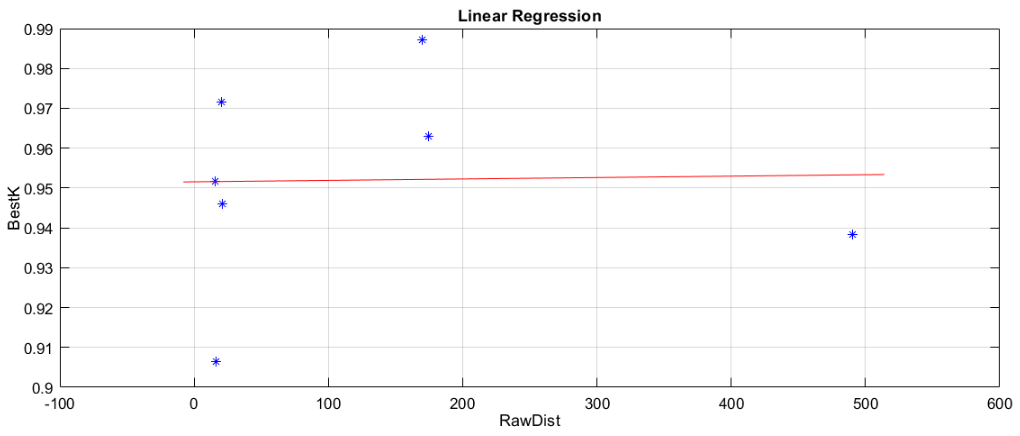

Figure 24: The linear regression model (red line) estimated for investigating a correlation between K and the total walked distance.

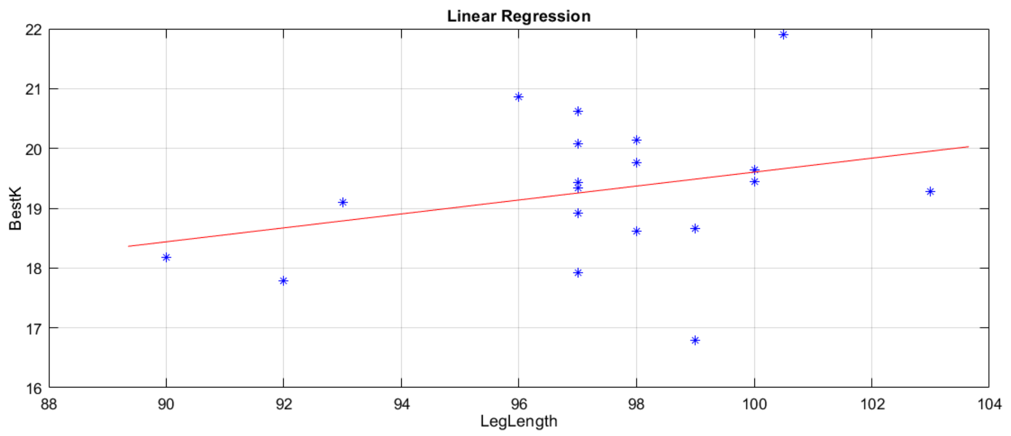

Figure 25: The linear regression model (red line) estimated for investigating a correlation between K and leg length of the person walking.

5.7.2 Results

The first case investigated the correlation between K and the distance of the walks. In this test, one person walked the distance 15, 20, and 168 m twice and 460 m once, resulting in a total of seven measurements. The linear regression model defined in (5.24) for this case can be seen as the red line in Figure 24. The parameters were estimated to $\alpha^* = 0.9516$ and $\beta^* = 3.563 \cdot 10^{-6}$, where the confidence interval for β^* is $I_{\beta^*} = [-0.0001674 \quad 0.0001745]$.

In the second case, was the correlation between K and the leg length on a straightforward distance of 20 m tested. The test was performed by 13 persons walking 19 walks in total. Just as before, the linear regression model was estimated in the same way, resulting in $\alpha^* = 7.956$ and $\beta^* = 0.1165$ with β^*'s confidence interval being $I_{\beta^*} = [-0.03787 \quad 0.3068]$. The linear regression model for this case can be seen in Figure 25.

In the third and last case, the correlation between K and feet length. This was tested in the same way as the second one, with 13 persons walking 19 walks on a 20 m straightforward path. The linear regression model was estimated as before, resulting in $\alpha^* = 12.15$ and $\beta^* = 0.2422$ with β^*'s confidence

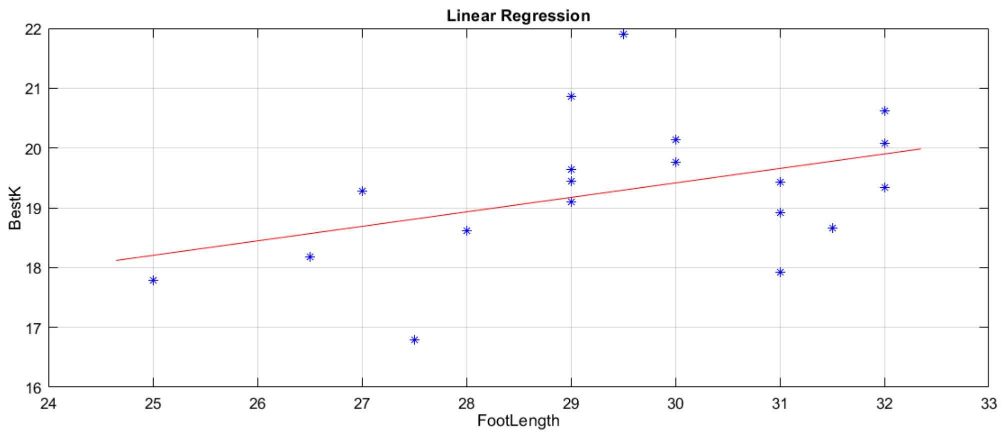

Figure 26: The linear regression model (red line) estimated for investigating a correlation between K and foot length of the person walking.

interval being $I_{\beta^*} = [-0.0331 \quad 0.5175]$. This linear regression model can be seen in Figure 26.

5.7.3 Analysis

What can be concluded, for all three cases, is that the parameter β^* is non-significant. This means there are no correlation between K and the external factors considering a 95% confidence level. The main problem with these tests is that the number of measurements might be too low. Looking at Figures 24-25, it is apparent that the variances are big, making the inference even harder. A good idea could be to further conduct clinical tests to increase the quality of the inference. There are several other factors which also could be investigated, such as K's correlation with the gait cycle or the circumference around the waist.

6 Discussion

In this chapter, the main objectives of this project is discussed. In the following,there is a conclusion of this work and a summarize of the achievements of this project.

Why not using the foot-mounted sensor?

There has been a lot of research conducted on retrieving the walked trajectory by using a foot-mounted sensor. The performance of having IMS attached to the feet is really well, and it is mainly because of one big advantage. A foot has a stationary moment during a walk, which enables the system to be reset and all accumulated drift errors disappears at that moment. This is what makes a foot-mounted sensor work really well, the only downside is its inconvenience. Therefore, a smart-phone is the more convenient choice, but with the disadvantage that it has no stationary moment. Therefore, the system can not be reset, which will make the errors accumulate and the trajectory estimation bad.

There might be a way to map the IMS measurements between the foot to the stomach, but the best similarities between these data sets are visible only in the frequency domain and not in the time domain, which is the domain used for estimating IC and FC. Until there is a smart way of using the foot-mounted sensor for improving smart-phone inertial navigation, the best approach is probably non-linear step models for estimating the step length.

What is needed for estimating the distance with smart-phone attached to the stomach?

When it was clear that the same approach that OpenShoe and Oblu had used for estimating the trajectory could not be applied in our work, the focus changed to non-linear step models. These have also been examined thoroughly (see section 5.6.1), but mostly for the cases where the IMS is attached to the back near COG. When the smart-phone is attached to the stomach, it is also reasonable close to COG for the step models to be applied to this case as well.

The method of estimating the distance is found in chapter 5.6, where all used algorithms for extracting the results are thoroughly explained. The first problem when using a smart-phone for examining the walk is that one needs to remove the non-walking segments. There are extremely high fluctuations in the acceleration and angular rates measurements when the smart-phone is put down into the belt pack. These do not follow any kind of cyclic behaviour, which a walk segment does. In this way, the walking data can be extracted by the introduced function *Cut*.

During a walk, people might be stopping or turning around. The function *Segmentation* handles these cases by splitting the walking regions individually. This is mainly done because the first FC and last IC are hard to find. They will therefore be manually added to the start and end since a FC should start the whole walk and an IC should end it. Unfortunately, the function is not robust and needs further development to make sure it applies to any types of walk styles.

Data segments containing walking, follow some specific features such as gait cycle, IC and FC, which are the ones retrieved in this case. There is a distinct pattern in the y-direction (forward) acceleration when the foot leaves the ground (FC) and when it makes contact again (IC). These are detected by the function *ICFCyDetector* in chapter 5.4, which works as a help function for locating IC and FC in the vertical acceleration (z-direction). The function *ICFCzDetector* does just that, which is explained in detail in chapter 5.5. The said functions works perfectly for data measurements where the features IC and FC can clearly be located. There are though a lot of irregularities in the data measurements, where it does not help how many criterions you have for finding IC and FC. The good part is that IC and FC do not always have to be perfect for the models to perform well, just close to the time region when they happen. Unlike the function *Segmentation*, the two functions *ICFCyDetector* and *ICFCzDetector* are robust and find all IC and FC reasonable well. There are cases where the functions perform worse, for example in rougher terrain outdoors. They could be improved with endlessly criterions for IC and FC, but it is not certain it would increase the results since there is a built-in uncertainty in the non-linear step models already.

The part of estimating the distance walked is found in chapter 5.6, which is done by non-linear step models. The simplest and most user-friendly models were chosen because they did not perform worse than models with extra input variables. There could be more tests using other models on different paths and terrain as well, but the chosen step models did not perform badly compared to previous research. Therefore, only the simpler models were implemented.

The scaling constant ,K, and the details about its implementation is seen in chapter 5.6.3. The scaling constant is estimated using the LS-method for each step model for a known distance, which gives the most optimal scaling constant for that type of test. The problem is that K is not assumed to have any dependency, just being constant. It is something which it not known at the moment, but a reasonable hypothesis would be that K changes with different factors which for example could be different persons or path length. This should further be examined, which would have been done if there was more time.

The results should be considered satisfactory. Compared to other projects with IMS attached to the back, which have errors between $1 - 3.5\,\%$ [5, 4, 9], the presented errors differ between $2 - 4.5\,\%$ when using the most optimal K for each test. A bit higher errors are to be expected when using the estimated K_{mean} for future distance predictions. The distance estimation could be further improved by investigating the scaling constant K's correlation to other factors, as has been done in section (5.7). There might be other approaches than using non-linear models to estimate the distance as well. Since the models only are approximations, they are never expected to be truly correct about every step taken during a walk.

The predicted distances seen in section 5.6.5 should also be considered reasonable. The distance mean percentage error is a bit higher for step models 2 and 3, but stays decently low for step model 1. A 4% error rate for the best model is not good enough to challenge the GPS. Even if GPS may have the same error rate when used outdoors, it still tracks the trajectory which the non-linear step models can not do. There are lots of improvements which can be done to make a robust and stable inertial movement system for estimating the trajectory. One can add trajectory estimation from the gyroscope measurements, use machine or deep learning to model human walking or find a way of resetting the errors and remove the drift. There are many ways to try new methods, which may be the best approach further on.

What could be done to enable trajectory estimation?

The aim of the project starting out was to model the walking trajectory using only smart-phones. To do this the angular rates have to be considered to model the direction. The only time angular rates have been used in this project is to detect u-turns. It appeared more and more challenging to model the trajectory using the acceleration and angular rates measurements from a smart-phone. It is a tough problem because today it seems like the only way to get rid of the drift is by resetting the system. Although it might be possible in an approach not thought of yet. Or if it is made possible to model the true acceleration of the body and diminish the errors, it could be a decent estimation for trajectory estimation.

7 Conclusion

In section 2.2, the objectives for this project presented, which will be answered during this chapter. The objectives are:

Is it possible to use a smart-phone for inertial navigation? The short answer is yes, distance estimation is decently good but all research done throughout the last two decades have trouble model trajectories. It is hard to get around the drift problem without any kind of resetting of the system, which appears hard when IMS is attached somewhere else than on the foot.

Can dead reckoning be predicted sufficiently good using a smart-phone? Sufficiently good for some applications yes, but it can not measure itself with GPS or a foot-mounted sensor. Smart phones are a bit from performing as good as GPS and foot-mounted sensors when estimating distance, but there is no good way of an accurate trajectory estimation using a smart phone attached to the waist region.

Are there any other relevant applications using smart-phone inertial navigation? Distance estimation is the most straightforward application when using smart-phones accelerometer and gyroscope. Another interesting application would be to measure how much a person stumble sideways (x-direction) when walking. Due to lack of time, this application was not implemented.

8 Future Work

Shoe sensor - OpenShoe and Oblu

The idea in this project was to use the open source algorithms from OpenShoe and the oblu walking device from Oblu to know the ground truth and data measurements for feet walking. Unfortunately, it was not that easy, since the algortihms for the oblu device was just based on the OpenShoe algorithms and not open source at Oblu.

What one can do in the future is to improve the open source algorithms from OpenShoe to work as supposed for data collected by the oblu device on the foot. Then, analyzing the connection between the IMS measurements from the stomach and the oblu measurements from the foot can lead to a good estimation of trajectories [23, 25].

Correlation for scaling constant K

By finding the dependency of the scaling constant, K, can the distance estimation of the non-linear step models be improved. The improvement might be good enough to take the step models to the next level and get close to the GPS performance.

Machine Learning

Another approach which may be useful is to use machine learning, where perhaps methods as neural networks and deep learning could be implemented for modelling the IMS measurements between stomach and feet. Perhaps these methods can be directly applied to modelling trajectories with the IMS at the stomach, which could be a smarter way than using non-linear step models.

Smart phone in pocket

It is more convenient to have the smart phone in a belt pack at the stomach than a foot-mounted sensor at the foot. Although it can still be made easier, for example getting rid of the belt pack. Next step to investigate could be to examine the distance or trajectory estimation when having the smart phone in the pocket. It is a more general and harder problem because the smart phone might not always be fixed and pockets can vary between users.

References

[1] J. C. Alvarez et al. "Pedestrian navigation based on a waist-worn inertial sensor". In: *Sensors* 12.8 (2012), pp. 10536–10549.

[2] T. Bennett, R. Jafari, and N. Gans. "An extended kalman filter to estimate human gait parameters and walking distance". In: *American Control Conference (ACC), 2013*. IEEE. 2013, pp. 752–757.

[3] G. Blom et al. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, 2005. ISBN: 91-44-02442-8.

[4] E. M. Diaz and A. L. M. Gonzalez. "Step detector and step length estimator for an inertial pocket navigation system". In: *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*. IEEE. 2014, pp. 105–110.

[5] T. N. Do et al. "Personal dead reckoning using IMU mounted on upper torso and inverted pendulum model". In: *IEEE Sensors Journal* 16.21 (2016), pp. 7600–7608.

[6] K. Ellis et al. *Identifying Active Travel Behaviors in Challenging Environments Using GPS, Accelerometers, and Machine Learning Algorithms.* 2014. DOI: 10.3389/fpubh.2014.00036..

[7] J. A. Farrell. *Aided navigation*. Mc Graw Hill, 2008.

[8] E. Foxlin. "Pedestrian tracking with shoe-mounted inertial sensors". In: *IEEE Computer Graphics and Applications* 25.6 (Nov. 2005), pp. 38–46. ISSN: 0272-1716. DOI: 10.1109/MCG.2005.140.

[9] R. C. Gonzalez et al. "Ambulatory estimation of mean step length during unconstrained walking by means of COG accelerometry". In: *Computer methods in biomechanics and biomedical engineering* 12.6 (2009), pp. 721–726.

[10] *Google Maps Sjon Sjon, Lund*. 2018. URL: https://www.google.com/maps/@55.7105287,13.2076427,114m/data=!3m1!1e3.

[11] *Google Maps Sjon Sjon, Lund*. 2018. URL: https://www.google.com/maps/@55.7103502,13.2085737,187m/data=!3m1!1e3.

[12] *iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope*. LSM6DS3. ST life.augmented. May 16, 2018. URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/a3/f5/4f/ae/8e/44/41/d7/DM00133076.pdf/files/DM00133076.pdf/jcr:content/translations/en.DM00133076.pdf.

[13] A. Jakobsson. *An Introduction to Time Series Modeling*. 2nd ed. Studentlitteratur, 2015. ISBN: 978-91-44-10836-0.

[14] A. R. Jimenez et al. "A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU". In: *2009 IEEE International Symposium on Intelligent Signal Processing*. Aug. 2009, pp. 37–42. DOI: 10.1109/WISP.2009.5286542.

[15] Y. Jin et al. "A robust dead-reckoning pedestrian tracking system with low cost sensors". In: *2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Mar. 2011, pp. 222–230. DOI: 10.1109/PERCOM.2011.5767590.

[16] J. W. Kim et al. "A step, stride and heading determination for the pedestrian navigation system". In: *Positioning* 1.08 (2004), p. 0.

[17] E. Lindström, H. Madsen, and J-N. Nielsen. *Statistics for finance*. 1st. ed. CRC Press, Taylor Francis Group, 2015. ISBN: 978-1-4822-2899-1.

[18] A. M. López et al. "Validity of four gait models to estimate walked distance from vertical COG acceleration". In: *Journal of applied biomechanics* 24.4 (2008), pp. 360–367.

[19] T. Mai. *Global Positioning System History.* Apr. 11, 2018. URL: https://www.nasa.gov.

[20] *MEMS digital output motion sensor: ultra-low-power high-performance three-axis "nano" accelerometer.* LIS3DSH. ST life.augmented. May 5, 2018. URL: http://www.st.com/resource/en/datasheet/lis3dsh.pdf.

[21] *MEMS motion sensor: three-axis digital output gyroscope.* L3GD20. ST life.augmented. May 5, 2018. URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/43/37/e3/06/b0/bf/48/bd/DM00036465.pdf/files/DM00036465.pdf/jcr:content/translations/en.DM00036465.pdf.

[22] B. Muset and S. Emerich. "Distance Measuring using Accelerometer and Gyroscope Sensors". In: *Carpathian Journal of Electronic and Computer Engineering* 5.1 (2012), pp. 83–86.

[23] J. O. Nilsson, A. K. Gupta, and P. Händel. "Foot-mounted inertial navigation made easy". In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN).* Oct. 2014, pp. 24–29. DOI: 10.1109/IPIN.2014.7275464.

[24] J. O. Nilsson and P. Händel. "Standing still with inertial navigation". In: (Oct. 2013). ISSN: 2153-358X. DOI: 10.1109/PLANS.2012.6236875.

[25] J. O. Nilsson et al. "Foot-mounted INS for everybody - an open-source embedded implementation". In: (Apr. 2012), pp. 140–145. ISSN: 2153-358X. DOI: 10.1109/PLANS.2012.6236875.

[26] *Oblu.* https://www.oblu.io/. 2018.

[27] Oblu. *oblu, A shoe-mounted indoor GPS.* 2017. URL: https://www.inertialelements.com/oblu/resources/oblu-datasheet.pdf.

[28] *OpenShoe.* 2018. URL: http://www.openshoe.org/.

[29] A. V. Oppenheim. *Discrete-time signal processing.* Pearson Education India, 1999.

[30] M. S. Pan and H. W. Lin. "A Step Counting Algorithm for Smartphone Users: Design and Implementation". In: *IEEE Sensors Journal* 15.4 (Apr. 2015), pp. 2296–2305. ISSN: 1530-437X. DOI: 10.1109/JSEN.2014.2377193.

[31] *Runkeeper Sjön Sjön, Lund.* 2018. URL: https://runkeeper.com/.

[32] S. Shan, Z. Hou, and J. Wu. "Linear Kalman Filter for Attitude Estimation from Angular Rate and a Single Vector Measurement". In: *Journal of Sensors* 2017 (2017).

[33] W. Y. Shih, L. Y. Chen, and K. C. Lan. "Estimating Walking Distance with a Smart Phone". In: *2012 Fifth International Symposium on Parallel Architectures, Algorithms and Programming.* Dec. 2012, pp. 166–171. DOI: 10.1109/PAAP.2012.33.

[34] I. Skog, J. O. Nilsson, and P. Händel. "Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems". In: (Sept. 2010), pp. 1–6. DOI: 10.1109/IPIN.2010.5646936.

[35] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte. "A high integrity IMU/GPS navigation loop for autonomous land vehicle applications". In: *IEEE Transactions on Robotics and Automation* 15.3 (June 1999), pp. 572–578. ISSN: 1042-296X. DOI: 10.1109/70.768189.

[36] *Xsens.* 2018. URL: https://www.xsens.com/tags/accelerometers.

[37] F. Zajac, R. Neptune, and S. Kautz. “Biomechanics and muscle coordination of human walking Part I: Introduction to concepts, power transfer, dynamics and simulations”. In: *Gait Posture* 16 (2002), pp. 215–232.

[38] R. Zhou. “Pedestrian dead reckoning on smartphones with varying walking speed”. In: *Communications (ICC), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–6.