Motion Classification and Step Length Estimation for GPS/INS Pedestrian Navigation

ERIC ANDERSSON

KTH Electrical Engineering

Degree Project in
Automatic Control
Master's Thesis
Stockholm, Sweden 2012

XR-EE-RT 2012:011

Abstract

The primary source for pedestrian navigation is the well known Global Positioning System. However, for applications including pedestrians walking in urban or indoor environments the GPS is not always reliable since the signal often is corrupted or completely blocked. A solution to this problem is to make a fusion between the GPS and an Inertial Navigation System (INS) that uses sensors attached to the pedestrian for positioning. The sensor platform consists of a tri-axial accelerometer, gyroscope and magnetometer. In this thesis, a dead reckoning approach is proposed for the INS, which means that the travelled distance is obtained by counting steps and multiplying with step length. Three parts of the dead reckoning system are investigated; step detection, motion classification and step length estimation.

A method for step detection is proposed, which is based on peak/valley detection in the vertical acceleration. Each step is then classified based on the motion performed; forward, backward or sideways walk. The classification is made by extracting relevant features from the sensors, such as correlations between sensor signals. Two different classifiers are investigated; the first makes a decision by looking directly on the extracted features using simple logical operations, while the second uses a statistical approach based on a Hidden Markov Model. The step length is modelled as a function of sensor data, and two different functions are investigated. A method for on-line estimation of the step length function parameters is proposed, enabling the system to learn the pedestrian's step length when the GPS is active.

The proposed algorithms were implemented in Simulink® and evaluated using real data collected from field tests. The results indicated an accuracy of around 2 % of the travelled distance for 8 minutes of walking and running without GPS.

Contents

Acronyms

AHRS Attitude Heading Reference System

DFT Discrete Fourier Transform

DRM Dead Reckoning Module

ECEF Earth-Centered-Earth-Fixed (Coordinate reference frame)

EKF Extended Kalman Filter

EM Expectation-Maximization

FFT Fast Fourier Transform

GMM Gaussian Mixture Model

GPS Global Positioning System

HMM Hidden Markov Model

IMU Inertial Measurement Unit

INS Inertial Navigation System

MEMS Micro-Electro-Mechanical-Systems

pdf Probability Density Function

PNS Pedestrian Navigation System

RMS Root-Mean-Square

Chapter 1

Introduction

"Tall tree, Spy-glass shoulder, bearing a point to the N. of N.N.E.

Skeleton Island E.S.E. and by E.

Ten feet.

The bar silver is in the north cache; you can find it by the trend of the east hummock, ten fathoms south of the black crag with the face on it.

The arms are easy found, in the sand-hill, N. point of north inlet cape, bearing E. and a quarter N."

This is captain J.Flint's direction to the treasure in the novel "Treasure Island" by Robert Louis Stevenson. The way to navigate described above, with bearing and distances, does not differ much from the idea of positioning pedestrians using dead reckoning which is the topic of this thesis.

This is a Master Thesis at Royal Institute of Technology in Stockholm. The work was made at SAAB AB, business unit Security and Defence Systems, in Järfälla.

1.1 Background

The primary source of positioning in Pedestrian Navigation Systems (PNS) is the well known Global Positioning System (GPS). However, for applications including pedestrians walking in harsh environments or indoors, the GPS is not always reliable since the signal is often corrupted or completely blocked. Examples of common GPS error sources are multipath effects, i.e. when the signal is reflected in the surrounding terrain, atmospheric effects and errors due to geometric position of available satellites. The problem can be illustrated with an example in the application of soldier systems.

Today, a group of soldiers could be equipped with a soldier system, enabling them to see the position of their friends plotted on a hand-held device via radio communicated GPS-signals. Imagine a scenario where a soldier walks into a building, blocking the connection to the GPS satellites. His position will now be frozen on his friends devices, leaving them unaware if he went into a vehicle, a building or if something else has happened.

Progress in recent years in the field of Micro-Electro-Mechanical-Systems (MEMS), with development of low-cost miniature inertial sensors, has led to interest for another

positioning approach; Inertial Navigation System (INS). The INS uses an Inertial Measurement Unit (IMU) to perform positioning, typically consisting of tri-axial accelerometer, gyroscope and magnetometer. IMUs are common today, used in almost all smart phones. A quite intuitive way to calculate position from the IMU is double integration of the acceleration. However, this solution suffers from drift since systematic errors present in the IMU signals quickly accumulate in the integration, resulting in positioning errors. A way to overcome this problem is to use a foot-mounted IMU with Zero-Velocity updates [1]. In this method, one takes advantage of the fact that the system is stationary during stance phase, i.e. the system has zero velocity, and uses this information to bound the error growth. However, this requires the IMU to be mounted on the foot which is impractical in some applications.

Another approach is to use a combination of GPS and INS for positioning. When the GPS is active it is responsible for positioning, and the IMU signals are used for step detection. With this information it is possible to estimate the step length. In GPS denied areas the positioning is made by dead reckoning, i.e. counting steps and multiplying with step length. The magnetometer is then used to determine the bearing, which enables calculation of the position. In areas where the GPS signal is weak or corrupted, a fusion between the GPS and dead reckoning can be used to improve the positioning. Clearly this approach offers three big challenges; step detection, step length estimation and bearing estimation, the first two considered in this work.

The main objective of this work is to investigate and develop algorithms for step detection and step length estimation. Furthermore, the detected steps should be classified by the motion performed; forward, backward or sideways walk.

1.1.1 Related work

Many different approaches for step detection have been investigated. A technique using a waist-mounted IMU has been proposed in [2]. This method relies on the fact that the pelvis experiences a vertical displacement during a step. The displacement is estimated by double integration of the vertical accelerometer, using a high-pass filter to remove the drift. Other approaches with IMU mounted on the upper part of the body use the cyclic pattern in the accelerometer signals to detect steps. Common methods include using sliding window, peak detection and zero-crossing [3, 4, 5]. Approaches to differentiate between motion types by analysing patterns in the tri-axis accelerometer signals exist. These includes moving upwards or downwards [6] and right, left, forward or backward [7].

Previous work also includes statistical approaches to the positioning problem. A method to decide the most probable trajectory using Bayesian probabilistic framework has been proposed in [8]. Classification of human motion, including walking, running, falling and going upstairs, using discrete Hidden Markov Models are proposed in [9] and [10].

Investigations and recognition of walking behaviours and the nature of human stride are found in [6] and [5]. Concerning step length estimation, a widely used method is to model the step length as a function of some feature of the sensor data, i.e. variance or step frequency. The GPS and accelerometer signals are then often integrated in an Extended Kalman Filter (EKF) to estimate the step length [3, 4, 2, 5].

Bearing estimation is a crucial part in an INS, and often problematic since the magnetometer suffers from many errors and disturbances. Various methods for this problem have been proposed using a combination of gyroscope and magnetometer signals,

as well as the GPS when it is active [3, 11]. However this work will only focus on step detection and step length estimation.

1.2 System Overview

The system described in this thesis is a part of a larger system referred to as a Dead Reckoning Module (DRM) which is an Inertial Navigation System. The position provided from the INS is ultimately integrated with the GPS position to obtain an improved estimation of the position as compared to GPS alone. This large system, with the GPS/INS integration, is illustrated in Figure 1.1.

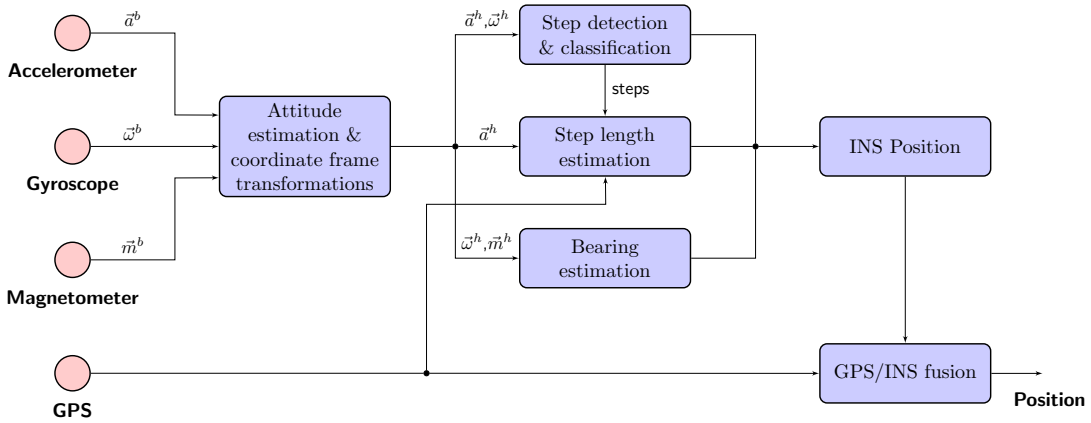


Figure 1.1: Schematic view for the integrated GPS/INS system. Two blocks are considered in the thesis; step detection and classification and step length estimation.

A chest-mounted DRM will be considered in this thesis, the hardware consisting of an IMU-platform and a microprocessor. The coordinate axis of the IMU-platform is aligned to the wearer's body so that the x-axis is pointing forward in the direction of the nose, the y-axis to the right and the z-axis down when the wearer is standing straight. The attitude of the upper-body will change during walk, and the first block in the DRM-system in Figure 1.1 deals with estimation of attitude angles and transformation of sensor data into an appropriate frame of reference. This block will not be considered in this thesis, although its role in the system will be discussed briefly.

The blocks in Figure 1.1 to be investigated in this thesis are "Step detection and classification" and "Step length estimation". These two blocks could be viewed upon as a subsystem that takes the IMU-data transformed to an appropriate coordinate system as input and outputs the distance travelled by the DRM wearer, shown in Figure 1.2. GPS data is of course needed as input for the step length estimator, but it is not used to directly calculate the distance travelled. The predefined requirements on the system shown in Figure 1.2 are:

- The step detector should be able to detect when and in what direction (forward, backward, sideways right or left) a step is made.

- The step length estimator should be adaptive, i.e. perform on-line estimation of the wearer's step length.

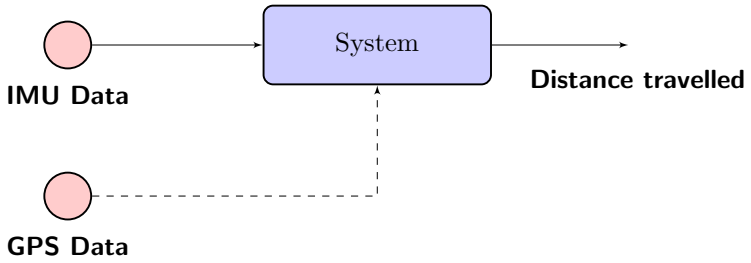


Figure 1.2: Simplified view over the system to be considered in this thesis. The GPS is only used for step length estimation and not to directly produce the output.

The system should be implemented and tested in Simulink® using real data collected from field tests. A hardware prototype of the DRM is available for collection of data, and a Simulink®-block consisting of an attitude observer and coordinate frame transforms for the IMU-data is given in advance.

1.2.1 Prototype and Hardware Specification

A hardware prototype was constructed by SAAB previous to this work for the purpose of data collection. The prototype consists of a CHR-6dm AHRS (Attitude Heading Reference System) chip from CH Robotics [12]. This chip actually produces estimates of the attitude angles, but the open source firmware is modified to log only raw data from the IMU sensors. The chip is put on a circuit with a few other components to enable logging via serial communication to a computer and integration of a GPS. The GPS used is a BR-355 GPS receiver from GlobalSat [13] and the system is powered from a 9 Volt alkaline battery. The circuit is encapsulated in a plastic box that is fastened to the wearers chest using two straps. The GPS receiver is mounted on a cap with Velcro tape.

1.3 Thesis Outline

The thesis consists of 7 chapters:

- Chapter 2 describes the coordinate frame transformation used for IMU data and discusses signal processing tools that is used to extract information from the IMU data. It also introduces a statistical model used for classification of steps.
- Chapter 3 describes the field tests made to collect IMU data and presents features extracted from sensor data that is relevant for detection and classification of steps and step length estimation.
- Chapter 4 introduces algorithms to detect and classify steps.
- Chapter 5 introduces algorithms and methods for step length estimation.
- Chapter 6 presents results from simulations and evaluates the performance of the constructed algorithms.
- Chapter 7 presents conclusions and further work.

Chapter 2

Theory

The task for the DRM subsystems treated in this thesis is to detect steps and estimate step length of the DRM wearer. Furthermore it is of interest to be able to classify the motion of the wearer into different states, such as “walking forward” or “running”. To be able to do this it is necessary to process and extract useful information from the IMU signals.

This chapter starts with a part about coordinate systems used for the IMU. The section discusses how to estimate attitude and make a necessary rotation of sensor data to an appropriate coordinate system. This part is then followed by a section about useful feature extraction methods for IMU data. The last part of the chapter will introduce a statistical model called Hidden Markov Model (HMM), well known in the field of pattern recognition. This model will be used to perform motion classification of the DRM wearer, i.e. to decide if the pedestrian is walking or running, forward or backward etc. Fundamental theory of HMM with continuous observation densities will be discussed, together with a method to perform the classification called the Viterbi algorithm.

2.1 Coordinate System and Attitude Estimation

The IMU consists of a gyroscope measuring the angular velocity of the system and an accelerometer measuring the linear acceleration of the system relative to the moving system itself, both expressed in an inertial reference frame. The measurements are made along the instrumental axes, i.e. the sensitivity axes of the actual sensors. For very precise applications a transformation from the instrumental frame to the IMU platform frame is needed to deal with potential misalignment in the instrumental axes, often referred to as IMU alignment. The IMU used in the prototype for the field tests has pre-made routines for such alignment calibrations. Furthermore, the IMU platform needs to be aligned with the axes of the body it is attached to, in this case the chest of the DRM wearer. However, during the field tests the DRM is mounted in such a way that the platform and body frame is assumed to be perfectly aligned. For a pedestrian standing straight this means that the x-axis is pointing forward in the direction of the nose, the y-axis to the right and the z-axis down.

Traditionally in an Inertial Navigation System, the actual acceleration relative to some earth reference frame is of interest. It is therefore necessary to transform the acceleration from an inertial frame of reference to a non-inertial frame of reference, typically the Earth-Centered-Earth-Fixed reference frame. Since this frame is non-inertial one must consider

fictitious forces such as Coriolis and centripetal forces when making the transformation. However, for the positioning approach presented in this thesis the actual acceleration or angular velocity measured by the sensors are of no interest, but only the signal's pattern. This is one of the benefits of using a dead reckoning approach, rather than double integration of accelerometer for positioning.

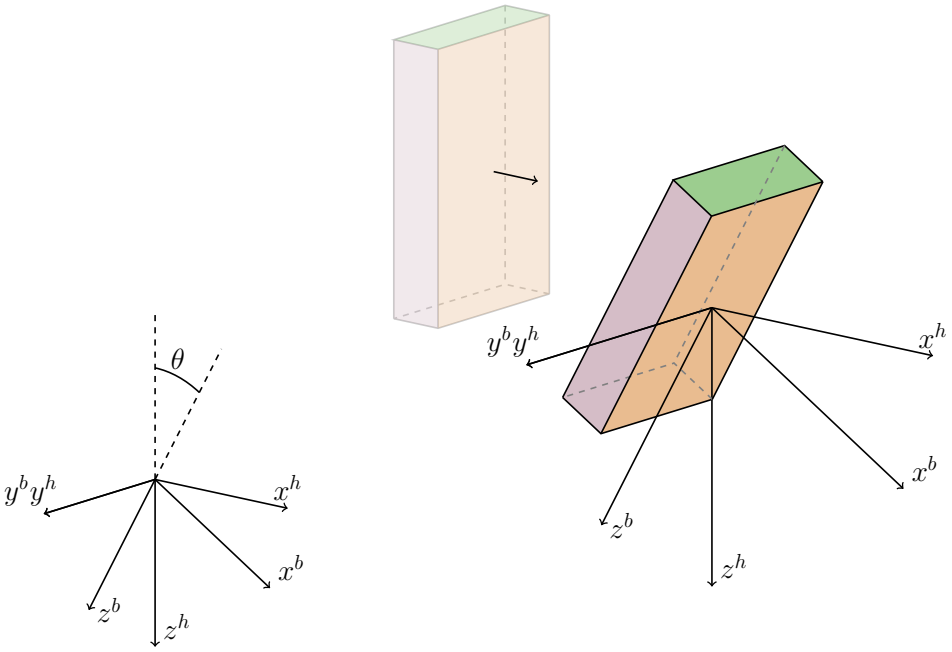


Figure 2.1: Illustration of the IMU platform and the two coordinate systems used; body-fixed and horizontal frame of reference. The platform is rotated around the y-axis, giving the pitch angle (θ) as illustrated to the left in the figure. A rotation around the x-axis would have resulted in a roll angle.

Even though the actual acceleration in the ECEF coordinate frame is not of interest, a transformation from the body fixed frame to an appropriate frame is necessary. The upper body of a human will typically be leaning forward during walk, giving a rotation of the body fixed frame. It seems more useful to know the horizontal and vertical (relative to the gravity force) acceleration. Note that this frame is body fixed as well; the x-axis is pointing in the forward direction of the wearer's anteroposterior axis and the y-axis to the right of the wearer's mediolateral axis. This frame is referred to as the horizontal reference frame, indicated by superscript h. The two coordinate systems are shown in Figure 2.1. The acceleration and angular velocities in the horizontal frame of reference, ignoring all effects caused by the rotation of earth, is written as

$$a^h = g^h + R_b^h a^b \tag{2.1}$$

$$\omega^h = R_b^h \omega^b \tag{2.2}$$

where superscript b indicates body fixed reference frame. Further, g^h is the gravity acceleration, a^b and ω^b are the measurements from the accelerometer and the gyroscope respectively. Moreover, R_b^h is the rotation matrix from the body fixed frame to the horizontal frame, explicitly written as

$$R_b^h = \begin{bmatrix} \cos(\theta) & \sin(\theta)\sin(\phi) & \sin(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.3}$$

where θ is the pitch angle and ϕ the roll angle, i.e. the rotations around the y- and x-axis of the horizontal reference frame, respectively. Knowing the initial attitude, these angles can be obtained through integration of the angular velocities provided by the gyroscope. However, low-cost MEMS gyroscopes are not very accurate which will lead to accumulating errors in the integration. The accelerometer can also provide information of the orientation using the vertical reference of gravitation, but the measurements will be corrupted by noise when moving fast. A fusion between the gyroscope and accelerometer signals could however provide a good estimate of the attitude, for example using an Extended Kalman Filter (EKF). Designing such an attitude observer is not a part of this thesis however, and a Simulink® block for this purpose was therefore provided by SAAB.

When using accelerometer and gyroscope signals throughout the rest of the thesis, they are always expressed in the horizontal frame of reference and denoted a_x, a_y, a_z and g_x, g_y, g_z, respectively.

2.2 Feature Extraction Methods

After rotating the tri-axial gyroscope and accelerometer signals to an appropriate coordinate system, the goal is to extract relevant information from the signals. Initially, the signal is passed through a filter to remove unwanted frequency components. Low-pass filtering can be used to remove high frequency noise, and high pass filtering to remove bias in the sensor signals. Common signal processing methods are often divided into three classes; time-domain features, frequency domain features and the so called time-frequency domain represented by for example wavelets [14] [15]. Only the first two are used in this work.

A sliding window is used when performing the signal processing, which in the case of a rectangular window means buffering of samples. The window is characterized by its length N, and overlap. Choosing an appropriate length of the window ensures that the signal could be considered as stationary within the window, i.e. its statistical properties do not vary in this time period. The overlap affects the update frequency and is chosen based on how often a new output from the signal processing is required.

The time domain features to be considered are tools to estimate statistical properties of the signals. A good measure of the spread of a signal around its mean is the variance. The variance of a signal $\mathbf{x}$ is estimated as

$$s^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \overline{x})^2 \tag{2.4}$$

The square root of the variance is called standard deviation, denoted s, and $\overline{x}$ is the sample mean, defined as

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{2.5}$$

An extension of the variance is the co-variance, giving a measure of the dependence between two signals. Furthermore, a time shift l, is introduced to get the co-variance function

$$\mathrm{Cov}_{xy}(l) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N-|l|}\Big(x_{n+l} - \overline{x}\Big)\Big(y_n - \overline{y}\Big) & \text{if } l \geq 0 \\ \mathrm{Cov}_{yx}(-l) & \text{if } l < 0 \end{cases} \tag{2.6}$$

The case when $l = 0$ and $x = y$ gives the variance according to (2.4). A dimensionless measure of the correlation between two signals are the correlation function, obtained by dividing the co-variance function with the standard deviations

$$\rho_{x,y}(l) = \frac{Cov_{xy}(l)}{s_x s_y} \tag{2.7}$$

The correlation function provides a measure of the correlation in the interval $-1 \leq \rho_{xy}(l) \leq 1$, where 0 corresponds to no correlation at all.

Analysis of signals in the frequency domain is provided by the Discrete Fourier Transform (DFT), defined as

$$X(m) = \sum_{n=0}^{N-1} x_n e^{-j2\pi\frac{m}{N}n} \tag{2.8}$$

Evaluating the transform by a direct use of the definition would give a computational complexity of $O(N^2)$, but fortunately better algorithms called Fast Fourier Transforms (FFT) exists, with complexity of $O(N\log(N))$ [16]. Choosing a window length that is a power of two gives the most efficient calculation of the FFT.

The DFT assumes that the signal is periodic. When using a sliding window this can be problematic since there is a risk of unwanted edge effects. To take care of this issue a window function can be applied and in this case a Hann window is used, defined as

$$w(n) = 0.5\Big(1 - \cos\Big(\frac{2\pi n}{N-1}\Big)\Big) \tag{2.9}$$

2.3 Hidden Markov Model

The HMM is a statistical signal model, widely used in the field of speech recognition [17]. A HMM models a system which is characterized by a set of N "hidden" states, i.e. states which are not directly observable. The observation is instead a probabilistic function of the state. In this case it means that the observable process, which is the features extracted from sensor data, is a function of the hidden state, i.e. the type of motion. At every time instant the system belongs to one of the N states; $S_1, S_2, \ldots, S_N$. In this application the states correspond to different motions such as walking forward or backward. It is called a Markov model since the state sequence is a Markov chain, which means that the state at time t is statistically dependent only on the state at time $t-1$. The HMM can be described with three probability measures as follows. The state transition probability distribution $A = \{a_{ij}\}$, which defines the probability of a transition from state S_i to state S_j. The transitions are illustrated in Figure 2.2. The formal definition of A is

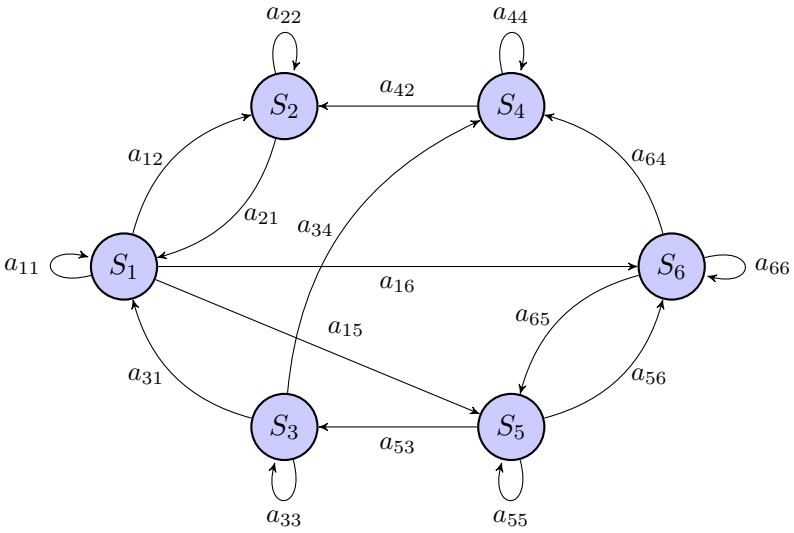


Figure 2.2: Six state Markov chain. Only selected state transitions are shown.

$$\begin{aligned} a_{ij} &= P(q_{t+1} = S_j | q_t = S_i), \qquad 1 \leq i, j \leq N \\ \sum_{j=1}^{N} a_{ij} &= 1 \\ a_{ij} &\geq 0 \end{aligned} \tag{2.10}$$

where q_t denotes the state at time t. The second probability measure is the observation probability distribution in state j, $B = \{b_j(s)\}$. This measure gives the probability of an observation given that the state is S_j. In the case of a discrete HMM, an observation is mapped to a distinct observation symbol, v_s . The set of possible discrete symbols is called the discrete alphabet, which is denoted $V = \{v_1, v_2, \ldots, v_M\}$ (M being the discrete alphabet size). B is defined as

$$b_j(s) = P(v_s \text{ at } t | q_t = S_j), \qquad 1 \leq j \leq N, \qquad 1 \leq s \leq M \tag{2.11}$$

The third component in a HMM is the initial state distribution $\pi = \{\pi_i\}$. This measure gives the probability of being in state S_j at time $t = 1$, formally written as

$$\begin{aligned} \pi_j &= P(q_1 = S_j), \qquad 1 \leq j \leq N \\ \sum_{j=1}^{N} \pi_j &= 1 \\ \pi_j &\geq 0 \end{aligned} \tag{2.12}$$

The three components of a HMM can be used to denote the model in a compact form

$$\lambda = (A, B, \pi) \tag{2.13}$$

Now, since the signals obtained from the IMU are continuous it seems appropriate to use continuous observation densities instead of trying to quantize the observations to discrete symbols. One way to achieve this is to use a probabilistic model for the observation vectors, or more specific a Gaussian Mixture Model (GMM). A GMM is a mixture of M components, each component being a multivariate (or one-dimensional) normal (Gaussian) distribution. The GMM is commonly used since a weighted sum of Gaussian densities can be used to approximate any density function. With this approach the observation probability distribution, $B = \{b_j(\mathbf{O})\}$, can be written on the form

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} \phi_{jm} \mathcal{N}(\mathbf{O}|\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}), \qquad 1 \leq j \leq N \tag{2.14}$$

As can be seen there is N number of GMMs, one for each state. $\mathbf{O}$ is the observation vector with length k and ϕ_{jm} is the mixture weight for component m and state S_j. The sum of the M mixture weights in each GMM equals to one. $\mathcal{N}$ represents the Gaussian probability density function (pdf) with mean vector $\boldsymbol{\mu}_{jm}$ and covariance matrix $\boldsymbol{\Sigma}_{jm}$. The pdf is explicitly written as

$$\mathcal{N}(\mathbf{O}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{O} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{O} - \boldsymbol{\mu}) \right) \tag{2.15}$$

where $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix. When using continuous observation densities we expand the HMM notation from (2.13)

$$\lambda = (A, \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \pi) \tag{2.16}$$

A summary of the HMM parameters and notations follows:

- N – The number of states, where a state is denoted S_j, $1 \leq j \leq N$.
- q_t – The state at time t.
- $\mathbf{O}_t$ – Observation vector at time t, the length of the vector is denoted k.
- $\pi = \{\pi_j\}$ – Initial state distribution.
- $A = \{a_{ij}\}$ – State transition probabilities. a_{ij} is the probability for a transition from state S_i to state S_j.
- $B = \{b_j(\mathbf{O})\}$ – Observation probability density function, represented by a GMM for each state.
- M – The number of components in each GMM.
- ϕ_{jm} – Mixture weight for component m in the GMM for state S_j.
- $\boldsymbol{\mu}_{jm}$ – Mean vector with length k for the Gaussian pdf in component m and state S_j.
- $\boldsymbol{\Sigma}_{jm}$ – Covariance matrix with size $k \times k$ for the Gaussian pdf in component m and state S_j.
- λ – Compact notation for all parameters defining a HMM.

2.3.1 Application of the HMM

The first issue with the HMM is of course how to obtain the model parameters, or to "train" the HMM. It can be solved with the so called Baum-Welch algorithm, which is an Expectation-Maximization[1] (EM) algorithm for finding a good estimate of the HMM parameters [17][18]. However, in this application it seems reasonable to pre-set the state transition probabilities, $A = \{a_{ij}\}$, based on assumptions of how prone humans are to switch between different states. The same reasoning applies for the initial state distribution, $\pi = \{\pi_{ij}\}$. This simplifies the problem to estimation of the GMM-parameters. The estimation is achieved by collecting training data, or more specific observation vectors, from each of the defined states. The GMM-parameters are then estimated individually for each state using an EM-algorithm, which is discussed in Appendix A. Essentially, what this method does is to estimate the probability distribution of the observation vector for each state, and represent this distribution as a GMM.

The second issue is how to use the HMM to decide which is the most probable state at some time instant according to a well chosen criterion. One possible and commonly used criterion is the most probable state at time t given the model λ and all previous observations. It can be solved using the so called forward algorithm with a scaled forward variable defined as [17]

$$\hat{\alpha}_{j,t} = P(q_t = S_j | \mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_t, \lambda), \qquad 1 \leq j \leq N \tag{2.17}$$

with most probable state at time t

$$q_t = \underset{1 \leq j \leq N}{\operatorname{argmax}}(\hat{\alpha}_{j,t}) \tag{2.18}$$

Note that this criterion determines the most likely state at every time instant, but does not consider if the produced sequence is valid. Consider for example a transition from the state "running forward" to "walking backward". This transition could be regarded as physically impossible, and the corresponding state transition probability would be $a_{ij} = 0$. The criterion (2.18) does not take this "impossible" transition into account and is therefore not suitable for applications containing state transition probabilities equal to zero. A better criterion might be to consider the single most probable state sequence that ends in state S_j at time t

$$\delta_{j,t} = \max_{q_1, q_2, \ldots, q_{t-1}} \Big(P(q_1, q_2, \ldots, q_{t-1}, q_t = S_j, \mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_t | \lambda) \Big), \qquad 1 \leq j \leq N \tag{2.19}$$

This maximization problem has a solution called the Viterbi algorithm [17]. Unfortunately this algorithm needs a "backtracking" step to find the optimal sequence, meaning that the algorithm requires all future observations before a result is produced. In a real time application this is obviously not possible. Nevertheless, a discussion of this algorithm and how it can be used follows.

[1] EM-algorithms are iterative methods to obtain the maximum-likelihood estimate of parameters in statistical models.

2.3.2 The Viterbi Algorithm

The Viterbi algorithm produces the single most probable state sequence for a HMM, by recursively solving the maximization problem defined in (2.19). The recursive structure of the algorithm relies upon the fact that at every time instance t there are only N possible states for the system to be in, and the optimal sequence must go through one of them [19]. At every time instant the parameter $\delta_{j,t}$, defined in (2.19), is stored for each state, together with a pointer to the previous state for this candidate. Only when the sequence ends, at time T, it is possible to find the optimal sequence by backtracking through the stored pointers. The Viterbi concept is shown in Figure 2.3.

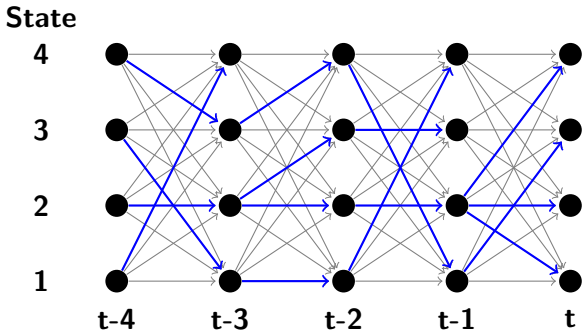


Figure 2.3: Trellis diagram showing the concept of the Viterbi algorithm. Only the most probable partial sequence for each state is considered for each time instance, illustrated with bold lines.

Essentially, what the algorithm does is to maximize separately over partial sequences ending at time t. The Viterbi calculation is quite easy to derive, starting at time $t = 1$ using the initial state distribution

$$\delta_{j,1} = P(q_1 = S_j, \mathbf{O}_1|\lambda) = \pi_j b_j(\mathbf{O}), \qquad 1 \leq j \leq N \tag{2.20}$$

Moving on to time $t = 2$, the next step is to calculate

$$\begin{aligned} &P(q_1 = S_i, q_2 = S_j, \mathbf{O}_1, \mathbf{O}_2|\lambda) \\ &= P(q_1 = S_i, \mathbf{O}_1|\lambda)P(q_2 = S_j, \mathbf{O}_2|q_1 = S_i, \mathbf{O}_1, \lambda) \\ &= P(q_1 = S_i, \mathbf{O}_1|\lambda)P(q_2 = S_j|q_1 = S_i, \lambda)P(\mathbf{O}_2|q_2 = S_j, \lambda) \\ &= \delta_{i,1}a_{ij}b_j(\mathbf{O}_2) \end{aligned} \tag{2.21}$$

where Baye's rule[2] has been used twice in the derivations. A maximization of (2.21) over all possible transitions will give the probability of the best state sequence ending in state S_j at time $t = 2$

$$\delta_{j,2} = \max_{1\leq i\leq N}(\delta_{i,1}a_{ij})b_j(\mathbf{O}_2), \qquad 1 \leq j \leq N \tag{2.22}$$

The general recursion formula is now found by induction

$$\delta_{j,t} = \max_{1\leq i\leq N}(\delta_{i,t-1}a_{ij})b_j(\mathbf{O}_t), \qquad 1 \leq j \leq N \tag{2.23}$$

[2]Baye's rule tells that the joint probability $P(A \cap B) = P(A)P(B|A)$

Now, since the real time application of this system implies infinite time sequences, it is not necessary to store all $\delta_{j,t}$ and pointers to the previous states. Instead, the state at time t is decided as the ending state of the most probable state sequence

$$q_t = \underset{1 \leq j \leq N}{\operatorname{argmax}}(\delta_{j,t}) \tag{2.24}$$

This will not result in the most probable single state sequence, and unfortunately "forbidden" transitions can take place. This is due to the fact that a decision of the state needs to be done at each time instance. Remember that the Viterbi algorithm maximizes separately over several possible partial sequences. The decided state at time t might be part of a different partial state sequence than the state decided at time $t + 1$.

When removing the backtracking step in the Viterbi-algorithm it becomes very similar to the forward algorithm mentioned earlier. The only difference is that the maximization in (2.23) is changed to a summation in the forward algorithm. However, the Viterbi algorithm is chosen since it is easy to implement with log-probabilities, which gives effective computation and numerical stability.

Chapter 3

Analysis of Human Motion through IMU Data

To be able to detect and classify steps, as well as perform step length estimation, it is necessary to extract relevant information from the sensor data. Let us start with a small example. Consider Figure 3.1 showing the z-axis acceleration of a pedestrian during walk. As can be seen the acceleration shows a cyclic pattern, and it is easy to detect when a step is made by looking at the peaks. At time $t = 37.5$ the pedestrian starts running, which is reflected in the amplitude of the acceleration.

This chapter will consider methods to extract features from sensor data that can be used to detect and classify steps. As the example shows, some features are quite easy to detect by the naked eye but, as it turns out, somewhat harder to implement in an algorithm. A fundamental challenge is that human gait differs between individuals, which of course is reflected in the sensor data.

The chapter starts with a part about the experiments made to collect IMU and GPS data, followed by two sections about feature extraction that is relevant for the application of step detection and step length estimation, respectively.

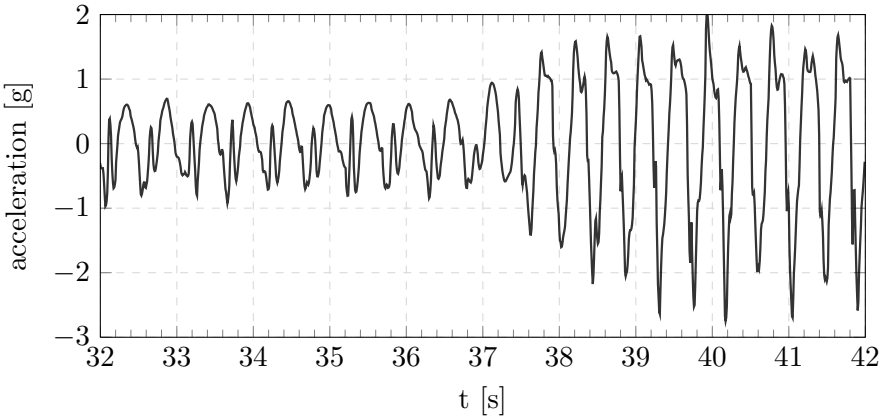


Figure 3.1: z-axis accelerometer output for pedestrian when walking and then running.

3.1 Collection of IMU Data

Two set of field tests were performed for collection of analysis and training data; one indoors collecting only IMU data and one outdoors collecting IMU and GPS data. Both were conducted at SAAB's area in Järfälla. In both cases the IMU was mounted on the chest using two straps around the upper body. The direction of the tri-axial sensors were, when standing up straight the x-axis pointing forward in the direction of the nose, y-axis to the right and z-axis down. For the second test a GPS antenna was mounted on the head using a cap. Logging of data was performed via serial communication to a laptop held by the tester. The sampling rate was 120 Hz.

The first, indoor test, was carried out on seven test subjects. There was one female and six males with heights in the interval $170 - 190$ cm and weight $65 - 95$ kg. Data was recorded when each tester performed six motion types; walking forward, backward, right and left (crossover step), running forward and standing (no lateral or anterior posterior motion, although rotations of body allowed). For each motion type a distance of 230 m were recorded, giving a total 65 minutes of IMU data.

The second test was carried out on four subjects from the first test, all males. Each tester performed two long walks of 390 m, where the tester walked forward at different intensities. Furthermore two series of data were recorded where the tester started off walking forward very slow and then progressively increased the speed until running as fast as possible. The distance for this test was 165 m. Lastly each tester walked 100 m backwards and 100 m sideways (right and left crossover). A total 70 minutes of IMU data was collected in the second field test.

3.2 Feature Extraction for Step Detection and Classification

A rectangular sliding window of 256 samples and 50 % overlap is used for the signal processing, unless otherwise stated. When using FFT a Hann window is applied to reduce the edge effects.

3.2.1 Detection of Steps

During walk, a cyclic pattern is shown in the vertical accelerometer data. Each valley (or peak) corresponds to a step and can be detected using a simple valley-detection algorithm. When looking over all motion types tested, the valleys are more distinct than the peaks and therefore suitable for detection. However, as can be seen in Figure 3.2, the signal is quite noisy and several valleys are present in each step. Particularly, two distinct valleys close in time are seen for each step, which corresponds to the impact of the heel and the sole respectively.

Based on the field tests, the step frequency for normal walk seems to be around 2 Hz, which agrees well with results from previous work in [5]. The maximum step frequency obtained was 3.5 Hz. It seems reasonable to assume maximum step frequencies around 4 Hz, although a step frequency of 4.8 Hz has been measured for 100 metre sprinters [20]. To avoid multiple detections by the valley-detector, a minimum time threshold between two steps could be introduced, based on the assumption of maximum step frequency.

The amplitude of the vertical acceleration varies greatly depending on the walking speed and also between individuals, which implies that having a fixed threshold for the valley detection is inappropriate. A better alternative is a dynamical threshold in form of the standard deviation of the accelerometer signal. The sliding window parameters for this calculation need to be chosen so that the standard deviation is updated every sample and fast enough to capture rapid changes in the accelerometer amplitude.

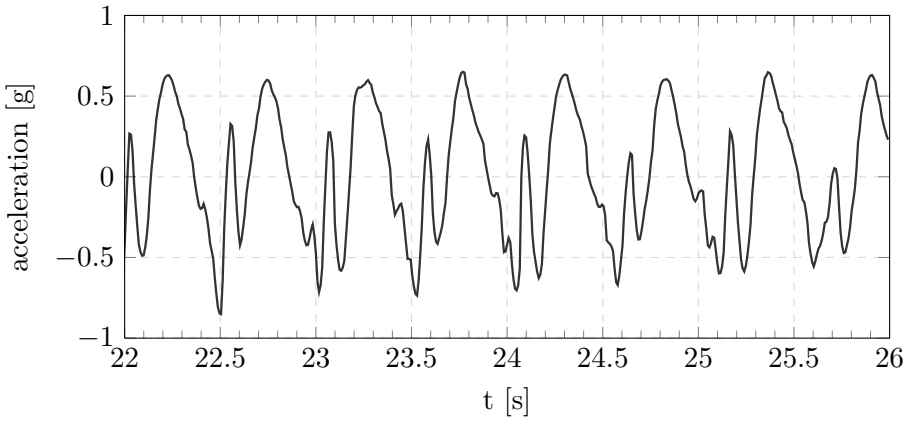


Figure 3.2: Vertical acceleration for a pedestrian walking forward.

3.2.2 Classification of Steps

Classification of steps relies upon the idea of identifying features that enables differentiation between motion types. The motion types, or states, are chosen as; forward, backward, right and left walk, forward running and standing still. The right and left walk is simply sideways crossover steps.

The vertical accelerometer variance for a tester is shown in Figure 3.3. The tester is walking forward and starts running at time $t = 37.5$, which is reflected as a distinct increase of the variance. This indicates that it might be a good candidate to differentiate between standing still, walking and running.

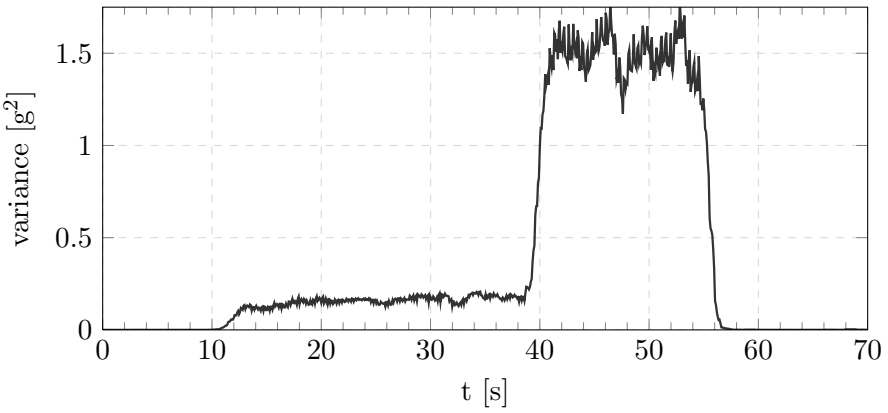


Figure 3.3: Typical variance of vertical acceleration for a pedestrian walking forward and running. The pedestrian starts walking at time $t = 10$ and running at time $t = 37.5$, which is clearly reflected in the variance.

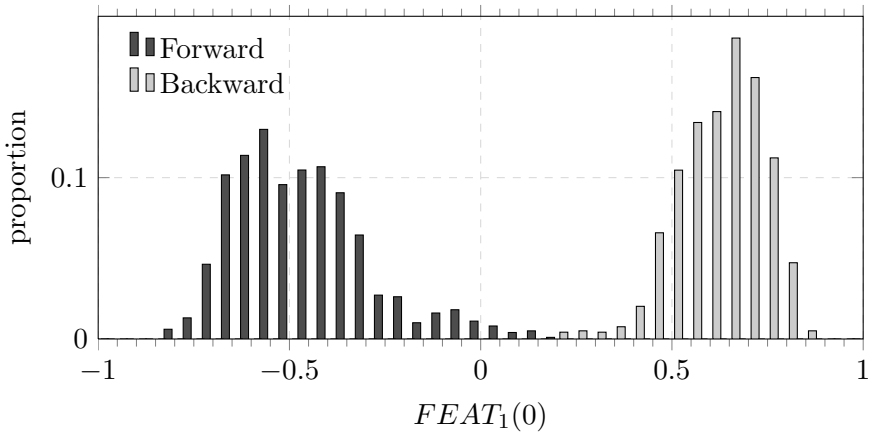


Figure 3.4: Histogram with 50 bins showing the distribution of $FEAT_1(0)$ for backward and forward walk. Data from seven test persons has been considered in the plot. The histogram is normalized with the total number of samples.

One way to avoid using absolute amplitudes and thresholds is to look at the correlation function, $\rho_{xy}(k)$, between signals. The IMU provides six signals, one for each axis for both gyroscope and accelerometer, which gives a total of 36 possible correlations. However, since $\rho_{xy}(k)$ and $\rho_{yx}(k)$ are mirrored versions of each other a total of 21 correlations need to be considered.

All testers showed a significant negative correlation between two of the sensor signals for zero time shift when walking forward, and positive when walking backwards. This correlation is denoted $FEAT_1(k)$, and the result is illustrated by a histogram in Figure 3.4 which shows the distribution of this correlation for zero time shift and all testers. A difference between walking sideways left and sideways right were found in two correlation functions for all testers, denoted $FEAT_2(k)$ and $FEAT_3(k)$. Both correlation functions showed a peak close to zero time shift when walking right and a valley when walking left (they are phase shifted 180 degrees), one of them is illustrated in Figure 3.5.

Looking at correlation functions seems to be a good alternative to differentiate between forward/backward and right/left. However, a feature providing information to differentiate between anterior posterior and lateral movement is needed. A possible

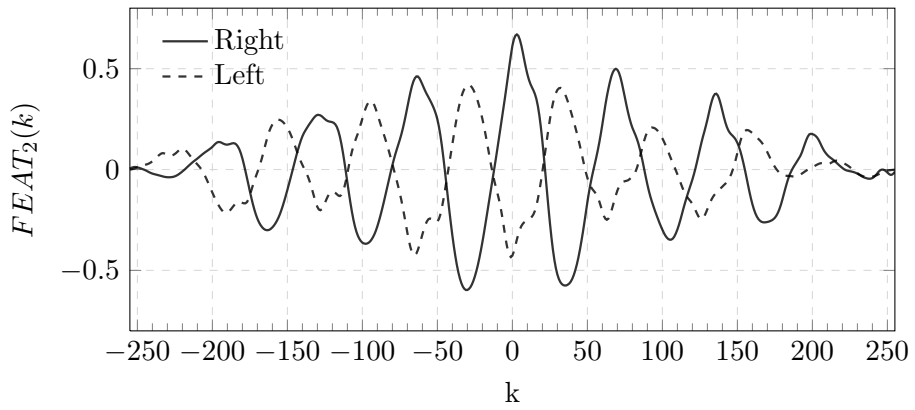
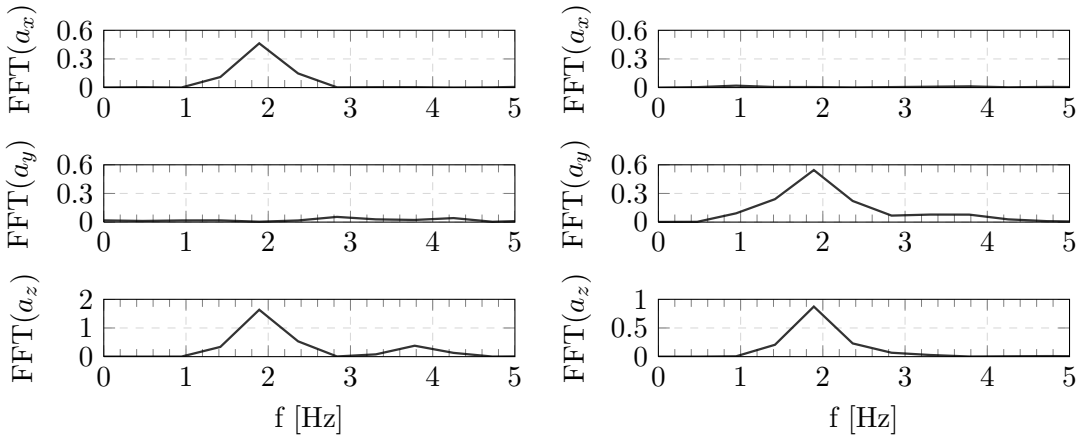


Figure 3.5: Typical correlation function $FEAT_2(k)$ for walking sideways right and left.

feature seems to be the Fourier transform of the accelerometer data, as shown in Figure 3.6. The largest peak in z-axis accelerometer corresponds to the step frequency and when in an anterior posterior motion a peak is found at the same frequency for the x-axis acceleration. Analogously a peak is found at this frequency for the y-axis acceleration when in lateral motion. This is quite intuitive since every step implies a vertical acceleration and an acceleration in the direction of motion. These accelerations will occur at approximately the same frequency, that is the step frequency.



(a) Anterior Posterior **(b)** Lateral

Figure 3.6: Typical FFT of the tri-axis accelerometer data for anterior posterior and lateral walk.

3.3 Feature Extraction for Step Length Estimation

Using GPS and accelerometer data can provide an estimation of the step length for analysis. When analysing the data from the field tests, it is clear that the step length depends upon several factors. Physical parameters, such as the leg-length, will of course affect the step length and makes it very different among individuals. While walking, one could argue that the maximum step length is determined by the length of the leg (and perhaps agility) since one foot always has contact with the ground. While running, however, the increase of step length depends on other physical parameters as well, such as weight and muscle strength. As shown in Figure 3.7 it is not unusual that the step length also differs between left and right steps of a person, sometimes more than 10 cm. Furthermore the step length increases when walking faster, i.e. at higher step frequencies. The field tests also showed that some testers had larger spread of the step length while walking at lower step frequencies and some at higher frequencies.
Due to the irregular nature of the step length, it seems reasonable to use several steps and determine the step length as the mean value of those steps. Preferably an even number of steps is used due to the possible difference between right-leg and left-leg steps. It is desired to be able to express the step length as a function of sensor data, since it changes depending on the walking speed. One suggestion is to express the step length as a function of the difference between the maximum and minimum vertical acceleration during a step, which has been empirically proved in [21]. Another well known candidate

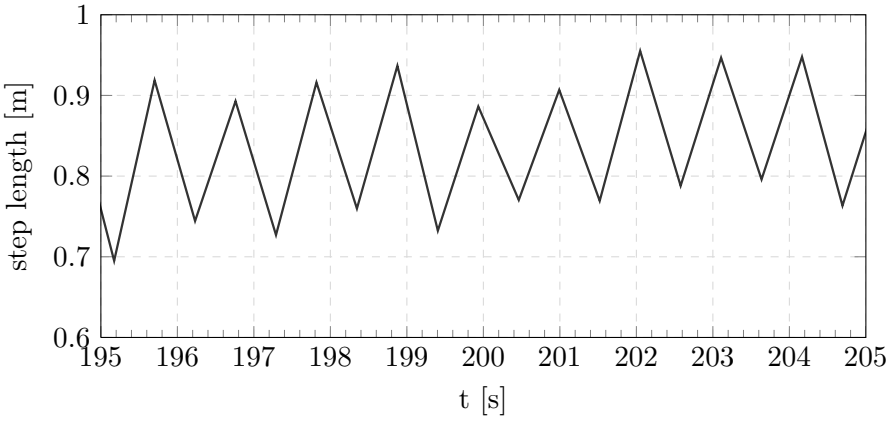


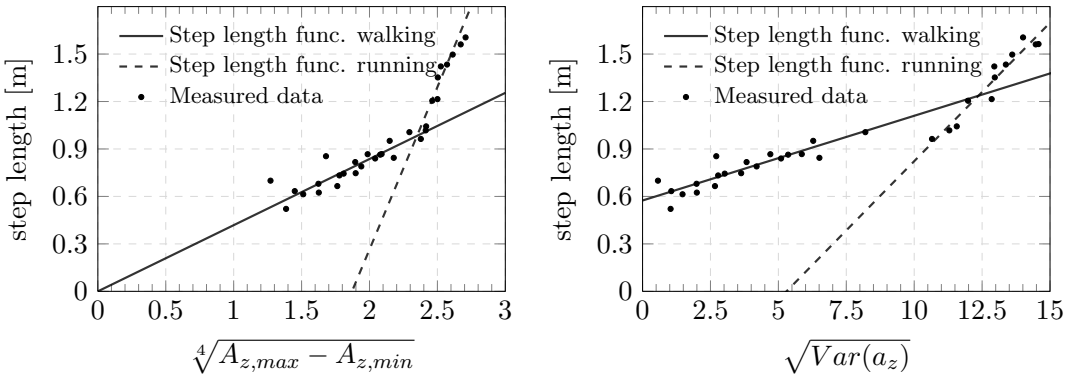
Figure 3.7: Step length for a tester walking forward. Clearly there is a difference in step length between the right and left leg.

is to use the variance of the vertical acceleration. The two suggestions of step length function can be written as

$$SL = b\sqrt[4]{A_{z,max} - A_{z,min}} \tag{3.1}$$

$$SL = c + d\sqrt{\mathrm{Var}(a_z)} \tag{3.2}$$

where b, c and d are regression coefficients to be estimated. Figure 3.8 show the measured step length plotted against the square root of the variance and fourth square root of the amplitude. The figure also contains least square estimates of the relationships. As guessed earlier, the step length dynamics changes significantly when the tester starts running, i.e. when the variance or amplitude is high. It is therefore necessary to use separate step functions for walking and running.



(a) Amplitude difference **(b)** Variance

Figure 3.8: Step length plotted as a function of amplitude difference respectively variance. Step length for each data point is calculated as the mean length of four consecutive steps. The lines are least square fitted to the data, one for walking and one for running in each plot.

When walking backward or sideways, the step length will differ as compared to

walking forward. Also, since humans rarely walk backward or sideways, it might be hard to capture enough data for the parameter estimation to converge. However, these step lengths will also depend on physical parameters such as the leg length, which makes it reasonable to assume a relation between the forward step length and the step length for walking backward or sideways. The step length function for walking backwards or sideways is therefore put on the same form as the forward step length function

$$SL_{bs} = e + f\sqrt[4]{A_{z,max} - A_{z,min}} \tag{3.3}$$

$$SL_{bs} = g + h\sqrt{\mathrm{Var}(a_z)} \tag{3.4}$$

Chapter 4

Step Detection and Classification

Step detection and classification is essentially about pattern recognition; how to extract relevant information from the sensors and use that information to make an output decision or classification that is useful for the system. Put in the context of step detection, the output should answer two questions:

- Did we take a step?
- Which kind of step was it?

The step detection block has two tasks; detection and classification of a step. Whereas the detection part is quite straight forward, the motion classification is a far more complex problem. The step detection system takes IMU data as input and outputs a unit pulse when a step is detected, together with a classification of the step, as illustrated in Figure 4.1.

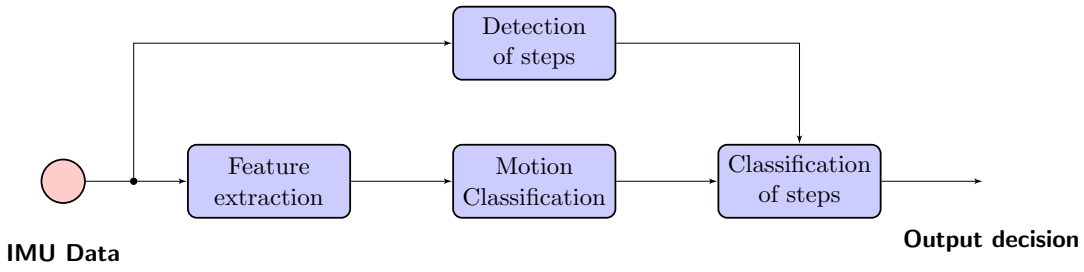


Figure 4.1: The general idea of the step detection system.

The idea of the classification approach is to differentiate between different human motions by using the features, such as correlations, discussed in Chapter 3. To be able to perform classification, the human motion is modelled as to be in one of following six states:

S1. Forward walk

S2. Backward walk

S3. Left walk

S4. Right walk

S5. Forward run

S6. Stand still (Rotations are allowed)

The reason for not making a fusion between the states run forward and forward walk is that, as discussed in Chapter 3, the extracted features changes quite a lot when running as compared to walking. One could of course consider more types of motions such as stair walking or crawling, however the choice of states is made to fulfil predefined system requirements.

This chapter starts with discussing an algorithm for step detection, followed by two sections proposing different approaches for step classification.

4.1 Detection of Steps

Steps are detected by performing a valley detection of the z-axis accelerometer data. If a step is detected, an unit pulse is outputted from the system. A sample is determined as a valley, and hence a step is detected, if it is smaller than the samples surrounding it (a local minimum) and fulfil following conditions:

- The minimum must be smaller than the maximum allowed value $VALLEY\_MAX$
- To assure that the minimum is distinct as compared to its surroundings, the acceleration must go over a threshold defined as the obtained minimum plus $MIN\_INCREASE$ before the minimum is classified as a valley.
- A minimum time, $T\_MINSTEP$ must have passed since the last step before a new step can be detected.
- An obtained minimum is discarded if any of the requirements above is not met before a certain time, $T\_TIMEOUT$
- When a step is detected, the acceleration must cross zero before a new step can be detected.

The pseudo code for the peak detector is found in Algorithm 1. As discussed in Section 3.2, it makes sense to use floating thresholds since the vertical acceleration changes a lot depending on walking intensity. The thresholds $VALLEY\_MAX$ and $MIN\_INCREASE$ is therefore dependent of the standard deviation for the z-axis acceleration. A rectangular sliding window with length $N = 1/T_s$ samples is used when calculating the standard deviation. The overlap is set to $N - 1$, since the step detector runs at the sampling rate.

Algorithm 1 Valley detection (Run for each new sample a_z)
Init at $t = 0$: $min = Inf$, $t_{min} = 0$, $t_{step} = 0$, $searchmin = true$

```
1: if a_z < min then
2:     if a_z < VALLEY_MAX and t − t_step > T_MINSTEP then
3:         min = a_z
4:         t_min = t
5:     end if
6: end if
7: if searchmin then
8:     if a_z > min + MIN_INCREASE then
9:         valley = 1
10:        t_step = t_min
11:        min = Inf
12:        t_min = t
13:        searchmin = false
14:    else if t − t_min > T_TIMEOUT then
15:        valley = 0
16:        min = Inf
17:        t_min = t
18:    else
19:        valley = 0
20:    end if
21: else
22:    if a_z > 0 then
23:        searchmin = true
24:    end if
25: end if
```

4.2 Classification Based on Logic Operations

The logic-based classifier extracts features from sensor data and classifies the motion by making comparisons between well chosen features.

A sliding window of around 2 seconds is used for feature extraction. The sampling rate is 120 Hz, and 256 samples are used as window length. The overlap is 75% which means that new classifications are made at a 64 sample interval, or around every half second. A Hann window is used instead of the rectangular when calculating the FFT.

A flow chart of the classifier is found in Figure 4.2. The first step is to decide if the pedestrian is standing still, walking or running, and this is done by investigating the variance of the z-axis accelerometer. Fixed thresholds are used for this decision, which is not optimal since the levels might differ between individuals. However, the variance difference between standing, walking and running are very distinct for all pedestrians, which allows fixed thresholds in this case.

If a walking motion is determined, the next step is to decide if an anterior posterior or lateral motion is performed. This decision is mainly based on FFT coefficients of the tri-axial accelerometer data, as discussed in Section 3.2. If an anterior posterior motion is decided, the algorithm continues to differentiate between forward and backward motion. The feature used for this decision is a correlation coefficient between two sensor signals, denoted $FEAT_1(k)$ in Section 3.2. A positive value of $FEAT_1(0)$ indicates backward motion and a negative indicates forward motion. If a lateral motion is decided, the next step is to differentiate between sideways right and sideways left. This decision is based on two different correlation functions, denoted $FEAT_2(k)$ and $FEAT_3(k)$ in Section 3.2.

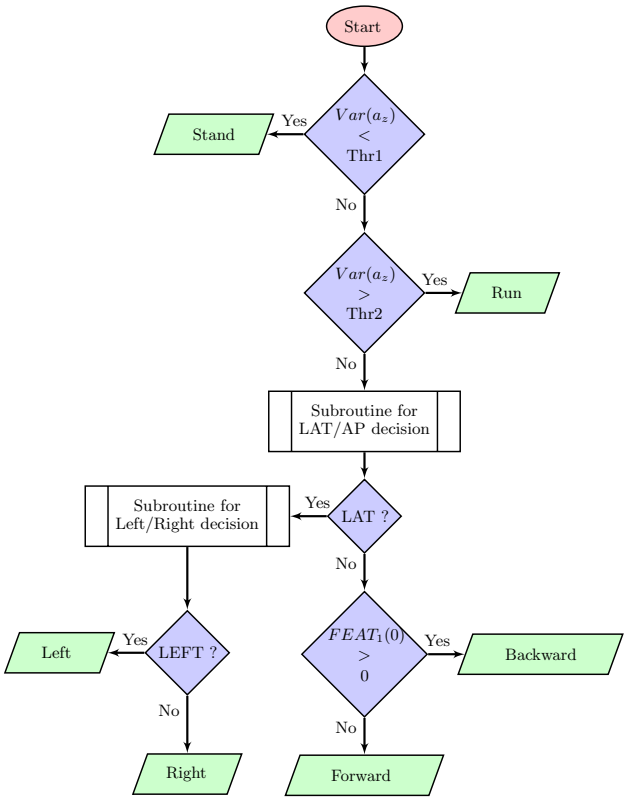


Figure 4.2: Flow chart for the logic classifier.

4.3 Classification Based on Hidden Markov Model

As for the logical classifier, a 256 sample window with 75 % overlap is used when extracting features for the HMM classifier. The sampling rate is 120 Hz. A rectangular window is used except for FFT calculations where a Hann window is used.

A feature vector, or observation vector, is composed of six features with 10 elements from each, giving a total of 60 elements. The features are correlation functions between sensor signals and FFT-coefficients. The 10 values from each feature represents different time shifts in the case of a correlation function and different frequencies in the case of FFT-coefficients. The observation vector is put together as

$$\mathbf{O} = \begin{bmatrix} FEAT_1 & FEAT_2 & FEAT_3 & FEAT_4 & FEAT_5 & FEAT_6 \end{bmatrix} \tag{4.1}$$

The HMM is parametrized by three probability measures, $\lambda = \{A, B, \pi\}$, which all needs to be determined. The initial state distribution, π, is set to favour the state "standing still", since it seems to be the most likely state when turning on the DRM. The rest of the states are set to be equally probable, giving

$$\pi_6 = 0.175 \tag{4.2}$$

$$\pi_j = 0.165, \qquad j \neq 6 \tag{4.3}$$

The state transition probability distribution, A, is also set manually by reasoning. During some tests it became clear that no transition probabilities should be set to zero, even if a transition is physically impossible. This is to reduce the risk of getting stuck in a state. Initially, the transition probabilities were set to

$$a_{ij} = 10^{-10}, \qquad i \neq j \tag{4.4}$$

$$a_{ij} = 1 - 5 \cdot 10^{-10}, \qquad i = j \tag{4.5}$$

which means that the fundamental assumption is that it is most probable to not change state. With this probability distribution as base, some trial and error modifications were made later on for individual transitions.

The observation probability distribution, B, is represented by Gaussian Mixture Models, one for each state. A GMM is represented by three parameters; the mixture coefficient vector, the mean vector and the covariance matrix. These parameters were estimated from collected training data.

4.3.1 GMM Training

Data was collected from seven different testers performing motions in the six different states, as described in Section 3.1. Observation vectors were then obtained by running simulations in Simulink and calculating the chosen features. These observation vector sequences were then used to train a GMM for each state.

The iterative EM-algorithm, with k-means clustering for initialization, was applied for estimation of the GMM parameters, as described in Appendix A. Five components were used for each GMM and the algorithm was set to stop when the likelihood converged according to a threshold. Around 40 iterations were usually required for convergence. For

each state the k-means and EM-procedure was repeated six times, since the algorithm only provides convergence to a local minimum, and the one giving the highest likelihood was kept. Around 1500 seconds of training data was used for training the forward state GMM, and around 500 seconds for each of the other states.

During the GMM-training, some actions were taken to ensure a stable algorithm. A common problem is the occurrence of non positive definite covariance matrices which can be caused by several reasons. To avoid this, and to reduce the complexity of the model, the covariance matrices were set to be diagonal. Furthermore, to ensure numerical stability logarithm probabilities were used as much as possible and the variables of the observation vectors were scaled to be in the same order of magnitude.

4.3.2 The HMM Classifier

With all parameters for the HMM pre-set or pre-trained the motion classification is performed using the Viterbi-algorithm, which is discussed in Section 2.3. When the classification is running on-line, the number of decisions grows as time goes by and so will the number of possible state sequences. This implies that the probability of a single state sequence rapidly will become very low, which in turn will cause numerical problems. To handle this problem the Viterbi-algorithm is implemented using log-probabilities. The negative log-implementation of the algorithm described by (2.20)-(2.24) becomes as follows

1. Initialization:

$$\chi_{i,1} = -\log\big(\pi_i\big) - \log\big(b_i(\mathbf{O}_1)\big), \qquad 1 \leq i \leq N \tag{4.6}$$

2. Recursion:

$$\chi_{j,t} = \min_{1\leq i\leq N}\big(\chi_{i,t-1} - \log(a_{ij})\big) - \log\big(b_j(\mathbf{O}_t)\big), \qquad 1 \leq j \leq N \tag{4.7}$$

3. Decision:

$$q_t = \underset{1\leq j\leq N}{\operatorname{argmin}}(\chi_{j,t}) \tag{4.8}$$

Note that the time t represents frame-time, rather than real time, since the update rate is changed due to the use of a sliding window with 75 % overlap. The log approach offers some minor challenges. The log of $A = \{a_{ij}\}$ and $\pi = \{\pi_j\}$ can be calculated off-line, but the log of $B = \{b_j(\mathbf{O})\}$ is trickier since it is represented by a sum of Gaussian pdf's and needs to be performed on-line. We recall the definition of $b_j(\mathbf{O})$:

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} \phi_{jm}\mathcal{N}(\mathbf{O}|\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})\Big) \tag{4.9}$$

Here M is the number of mixtures, or components, in the GMM. Taking the logarithm of a sum is unwanted in general, but a useful relation is

$$\log(a+b) = \begin{cases} \log(a) + \log\Big(1 + \exp\big(\log(b) - \log(a)\big)\Big) & \text{if } \log(a) \geq \log(b) \\ \log(b) + \log\Big(1 + \exp\big(\log(a) - \log(b)\big)\Big) & \text{if } \log(a) < \log(b) \end{cases} \tag{4.10}$$

The relation above can be derived using common logarithmic identities. Furthermore, the log of a single Gaussian pdf can be written as:

$$\log\Big(\mathcal{N}(\mathbf{O}|\boldsymbol{\mu},\boldsymbol{\Sigma})\Big) = C - \frac{1}{2}D(\mathbf{O}) \tag{4.11}$$

where

$$C = \log\left(\frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2)}}\right) \tag{4.12}$$

$$D(\mathbf{O}) = (\mathbf{O}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{O}-\boldsymbol{\mu}) \tag{4.13}$$

The constant C can be calculated off-line, as well as the log of the mixture coefficients for the GMMs. Algorithm 2 shows the pseudo code for calculating $\log(b(\mathbf{O}))$ using (4.10)-(4.13). The notations used are as follows; **LF** is a vector containing the logarithm of the M mixture coefficients, **C** is a vector containing the constant C for the M mixtures and **D** is a vector containing $D(\mathbf{O})$ evaluated for the M mixtures. The sub-function LOGSUM is simply an implementation of Equation 4.10, calculating $\log(a+b)$ from inputs $\log(a)$ and $\log(b)$. With the function described in Algorithm 2, implementation of the Viterbi-algorithm is straight forward, and it provides an decision of the state q_t at each time instant.

Algorithm 2 Calculation of $\log(b(\mathbf{O}))$

```
1: function LOGGMM(LF,C,D)
2:     logprob=LF_1 + C_1 - 0.5·D_1
3:     for m = 2 → M do
4:         logprob=LOGSUM(logprob, LF_m + C_m - 0.5·D_m)
5:     end for
6:     return logprob
7: end function
```

Chapter 5

Step Length Estimation

The central idea of step length estimation is to model the step length as a function of sensor data. Based on the discussion in Chapter 3, two different suggestions for step length functions will be considered

$$SL = a + b\sqrt[4]{A_{z,max} - A_{z,min}} \tag{5.1}$$

$$SL = c + d\sqrt{\mathrm{Var}(a_z)} \tag{5.2}$$

The coefficients a, b, c and d are the linear regression parameters to be estimated by the system. $A_{z,max}$ and $A_{z,min}$ represents the maximum and minimum z-axis acceleration during a step. The step length dynamics will change depending on if the pedestrian is walking forward, backward, sideways or running. The four situations will be represented by individual functions, each with its own set of parameters. When estimating step length function for forward walk the parameter a is set to zero.

The subsystem of step length estimation inputs both IMU and GPS data as well as the output from the step detection system, and outputs the set of estimated parameters needed for the step length models. The system is illustrated in Figure 5.1. This chapter will discuss estimation of the linear regression parameters necessary for the step length models.

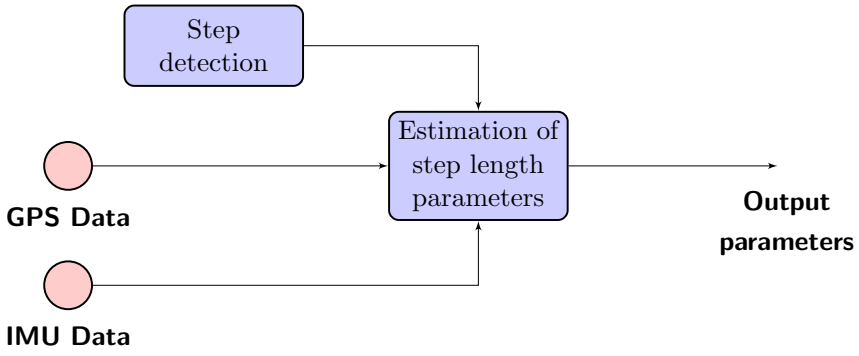


Figure 5.1: System overview for step length estimation

5.1 Forward Walk and Running

The idea of the step length system is to estimate the regression coefficients in (5.1)-(5.2). To clarify, the only difference between the functions proposed is the choice of function variable; either the amplitude difference or the variance. The approach to estimate the function coefficients is the same for both cases, and they will both be referred to as the function variable in this section.

Note that the procedure for estimation of step length coefficients presented next is made separately for the different states. A measurement classified as forward walk will be saved and used for estimation of the forward step length function, a measurement classified as running for the running step length function etc.

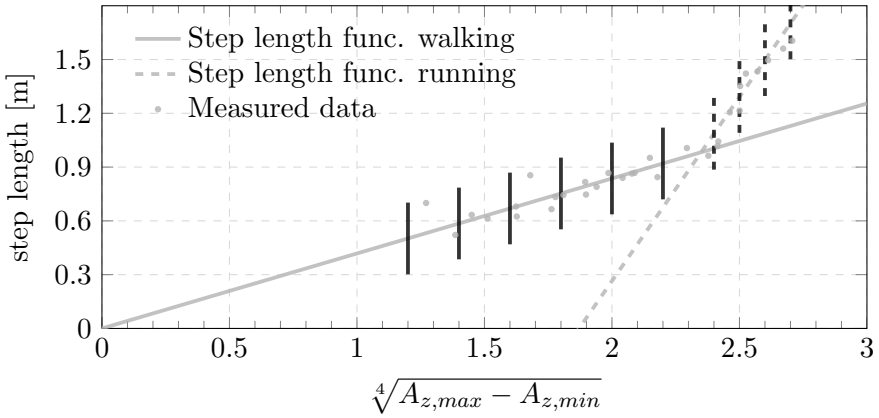


Figure 5.2: Plot showing the idea of using segments for estimation of the step length function. The solid black lines divides the interesting region for forward walk into segments, and the dashed black lines for running. More segments than showed in the plot are used in reality. The amplitude difference is chosen as variable in this figure.

A measurement consists of the obtained step length together with its corresponding obtained function variable and a classification of the state. To get as good estimations as possible the main interest is to obtain measurements at several different values of the function variable. In other words, the quality of the estimation is dependent on the spread of the obtained values rather than the quantity. To take this into account, the relevant region of the function variable is divided into several smaller segments, see Figure 5.2. When a measurement is obtained, it is sorted into a segment based on the value of the function variable. Furthermore, instead of saving several measurements in each segment, the obtained values are averaged using a low-pass filter. This procedure will give a single averaged value for each segment that is used for estimation of the regression coefficients.

The step length measurements are collected in a vector y and the function variable in a vector x, each element in the vectors correspond to a segment. A new measurement belonging to segment i is saved as

$$x_i^{new} = (1-p) \cdot x_i^{old} + p \cdot x_{measured}, \qquad 0 < p < 1 \tag{5.3}$$

$$y_i^{new} = (1-p) \cdot y_i^{old} + p \cdot y_{measured}, \qquad 0 < p < 1 \tag{5.4}$$

Here, 24 segments are used for estimation of a step length function, which gives 24 elements in x and y. A good value of the parameter p seems to be between 0.05 and 0.1, making the averaged value in each segment resistant to outlier measurements. However, such a low value of p implies quite a long settling time, i.e. many measurements are required for each segment before it approaches the true value. A way to overcome this is to have good initial guesses for each segment, which is hard since the step length differs between individuals. Therefore, the first five measurements obtained in each segment is averaged by the sample mean. This provides a starting value for the segment, and from the sixth obtained measurement the new values are incorporated using (5.3)-(5.4).

The step length is measured using a combination of GPS and output from the step detection system. Due to the irregular nature of the step length, as discussed in Chapter 3, the step length is decided as the mean length of four steps. When a step is detected by the step detector, an integration of the GPS velocity is started. After four steps, the integration stops and the obtained distance is divided by four to get the step length. The four steps must be consecutive and classified as the same motion by the step classifier. At each detected step the obtained function variable is stored, and the average of the four steps is used as a measurement.

Whenever a new measurement is obtained, the regression coefficients for the corresponding state are re-estimated using least square fit

$$\beta = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{j=1}^{n} y_j}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\big(\sum_{i=1}^{n} x_i\big)^2} \tag{5.5}$$

$$\alpha = \overline{y} - \beta\overline{x} \tag{5.6}$$

where a horizontal bar over a variable represent the sample mean. The coefficient β is equivalent to the parameters b or d in (5.1)-(5.2), depending on which function variable is used, and the coefficient α is equivalent to a or c. Note that the coefficient a is set to zero for forward walk, forcing the step length function to go through the origin. This simplifies the estimation of the remaining parameter to

$$b_{forward} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} \tag{5.7}$$

During estimation some segments might be empty of measurements, with the corresponding elements in x and y being zero. These elements are removed before applying (5.5)-(5.7). Bad or outlying measurements will occasionally be obtained during estimation. If one segment has received ten measurements and another just one measurement, it therefore seems reasonable to trust the value in the former more. This is achieved by weighting each segment with the number of measurements received before estimation of regression coefficients, although an upper limit of the weight is set to avoid segments being to heavy weighted.

A problem with the backward and sideways step length function estimation is that humans rarely perform these motions. This implies that very few measurements for these states will be obtained, and it might be difficult for the parameter estimation to converge. An alternative is to use the forward step length function and make some simple manipulations of the coefficients to obtain reasonable estimations of the sideways and backward step length functions.

When the GPS is inactive, the length of a detected step is calculated using the estimated coefficients and the function variable obtained from IMU data according to (5.2)-(5.2). Different sets of estimated coefficients are used depending on the motion type decided by the step classification.

Chapter 6

Simulation Results

The solutions for step detection, classification and step length estimation discussed in the previous chapters were implemented as a model in Simulink®. A new set of data was collected to test and evaluate the performance of the implemented system.

The collection of this data set was carried out on five test persons, none of whom participated in the collection of training data. The testers were all males with heights in the interval 162-191 cm and weights between 60-90 kg. Each tester performed four tests, all conducted at SAAB's area in Järfälla.

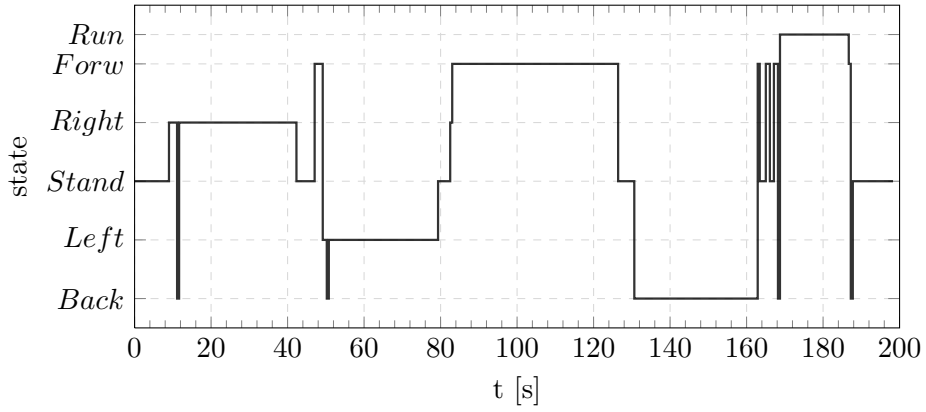
During the first test each person performed a mixed walk where all motion types were represented; sideways right and left, forward, backward and running. This test was recorded mainly to evaluate the classifiers. The second test was a long walk, around 720 metres, where the tester walked forward with two short intervals of running and the third test was a short walk backwards. Throughout the chapter, these three tests will be referred to as mixed walk, long walk and backward walk. The last tests was recorded to be used as learning data for the system, i.e. to simulate a scenario where the DRM wearer is walking outside with the GPS active so the system can estimate the step length parameters. Both IMU and GPS data was recorded for all tests, although the GPS data is only used by the system when learning the step length.

The chapter starts with an evaluation of the classifiers, mainly based on the mixed walk. This section is followed by an investigation of the parameter estimation for the step length functions. The last section presents an evaluation of the overall system, i.e. how well the travelled distance is estimated by the system. When talking about amplitude and variance throughout the chapter, it is the two alternatives for step length function that is referred to.

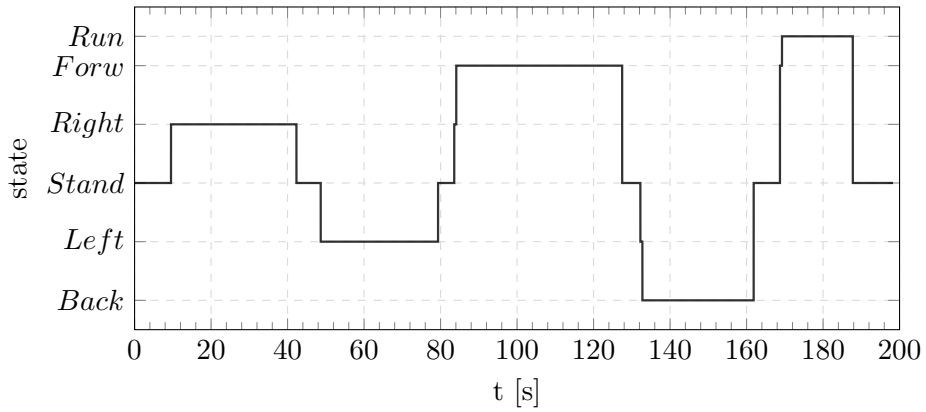
6.1 Evaluation of Step Detection and Classification

Two classifiers were proposed in the previous chapters; one naive approach using logic and one statistical approach using a Hidden Markov Model. To evaluate the classifiers each tester did a test where they performed motion in all states; sideways right and left, forward, backward and running forward. The resulting classifications for one tester, as simulated in Simulink®, are shown in Figure 6.1. As can be seen, the logic classifier are more prone to switch between states.

Table 6.1 presents detailed results from the steps detected and classified for the five testers. The table shows the true number of steps made in each state and the steps



(a) Logic classifier



(b) HMM-classifier

Figure 6.1: Simulation of the classification for one tester. Each state is represented as a level, indicated on the y-axis. The HMM classification shown in **(b)** is almost 100 % correct, and can be seen as the reference. Clearly, the logic classifier are more prone to switch between states.

detected by the two classifiers. It is clear that the HMM-classifier outperforms the logic classifier for all states except walking backward.

Note that it is the performance of the step detection algorithm and classification together that is evaluated. A correct decision is made if the step is detected and correctly classified. The step detection algorithm will occasionally detect false steps, mainly when the wearer is standing still at a place making rotations and irregular movements. The idea is that the classifier should recognise that the wearer is standing still and disregard the false detections. The last column in Table 6.1 shows the number of false detections for the two classifiers. Obviously the HMM-classifier is better at disregarding the false detections.

It is hard to grade the classifiers using for example a percentage of correct classifications. Almost all the failed detections for the HMM-classifier occur in the transitions between states, more precisely it often fails to detect the first step when changing state. This implies that making a test where the tester constantly switches between states would yield a lower accuracy, and a test where the tester performs a long walk in the same state would yield a high accuracy.

	Right	Left	Forward	Backward	Run	False
Tester 1, Steps	**48**	**45**	**35**	**81**	**40**	-
Detected Logic	47	45	32	81	40	2
Detected HMM	48	45	34	81	39	0
Tester 2, Steps	**51**	**51**	**74**	**46**	**63**	-
Detected Logic	49	49	72	46	62	9
Detected HMM	51	51	74	44	63	4
Tester 3, Steps	**44**	**54**	**74**	**44**	**50**	-
Detected Logic	42	46	34	44	50	7
Detected HMM	44	47	74	44	49	1
Tester 4, Steps	**51**	**51**	**79**	**42**	**50**	-
Detected Logic	45	44	79	42	48	17
Detected HMM	51	47	79	42	49	1
Tester 5, Steps	**59**	**59**	**85**	**53**	**54**	-
Detected Logic	59	56	85	53	52	8
Detected HMM	58	57	85	51	54	1
Total, Steps	**253**	**260**	**347**	**266**	**257**	-
Detected Logic	242	240	302	266	252	43
Detected HMM	252	247	346	262	254	7

Table 6.1: Results from the step detection and classification for the five testers. For each tester, the number of correct classified steps for each motion type is shown, as well as the number of false detections during the test.

One more field tests were evaluated for the HMM-classifier, namely the long walk as discussed in the beginning of this chapter. For all testers a total of 4272 steps were recorded during the long walk, and the HMM-classifier managed to detect and correctly classify 4252 of them. In addition, a total of 12 false detections were made.

Looking at the step detector alone, it only failed to detect 11 steps out of 5655 steps in total for the mixed and long walk. The steps that were not detected occurred in two situations; during sideways walk where the accelerometer pattern was very distorted and some peaks became too small, and in the transition from running to walking where the floating threshold did not change fast enough. As regards the false detections, they can be divided into two categories; false detections that occur when the pedestrian is not walking but making irregular movements and false detections that occurs when the pedestrian is walking. The former is left to the classifier to disregard, but the latter is a problem assigned to the step detector. For the two field tests discussed above only 5 false detections of the second category was found and they all occurred during sideways walk.

6.2 Learning the Step Length

The data used for step length estimation was an around 400 m long walk where the testers were instructed to walk normally and perform a short distance of running during the test. Both IMU and GPS data were recorded for this test, and during simulation this

data was used by the system to learn the testers step length, i.e. estimate the necessary parameters for the forward and running step length functions. To achieve as good step length estimations as possible it would have been better to instruct the testers to perform a walk with progressively increasing speed from very slow to very fast, which would have given the step length estimator as much information as possible. However, that kind of scenario seems unlikely to happen in reality and therefore it seemed more interesting to record an ordinary walk with a short distance of running.

Three or four parameters need to be estimated, depending on the choice of step length function variable. However, during the simulation both the variance and the amplitude difference were considered to be able to make a comparison. The step length functions to be estimated are

$$SL_{f1} = b\sqrt[4]{A_{z,max} - A_{z,min}} \tag{6.1}$$

$$SL_{f2} = c + d\sqrt{Var(a_z)} \tag{6.2}$$

$$SL_{r1} = e + f\sqrt[4]{A_{z,max} - A_{z,min}} \tag{6.3}$$

$$SL_{r2} = g + h\sqrt{Var(a_z)} \tag{6.4}$$

where subscript f and r indicates forward or running and b-h are the parameters to be estimated.

Figure 6.2 and 6.3 shows the estimation of forward step length parameters b, c and d for one tester. Note that parameter c and d remains constant at some intervals, which is caused by limitations of the parameters. The slope d is restricted to be positive, but the least square estimate renders a negative value which therefore is discarded. This reveals a weakness with using two parameters in the step length function; when the obtained measurements are concentrated to a small interval of the function variable the least square fit can produce parameters far from the true values. The one parameter alternative, as defined in Equation 6.1, is more robust giving a smoother estimation of the parameter b as shown in Figure 6.2.

Note that it is difficult to talk about settling times for the parameter estimations; convergence towards the true value requires obtained measurements at several interval of the function variable and is therefore ultimately dependent of the wearer's movement pattern.

Estimation of the running step length functions leaves no choice; two parameters must be estimated regardless of the function variable used. Figure 6.4 shows the estimation of parameters e to h. As can be seen, less than 30 seconds of training data is acquired for the estimation, and the parameter values e and f are still changing quite a lot when the estimation stops.

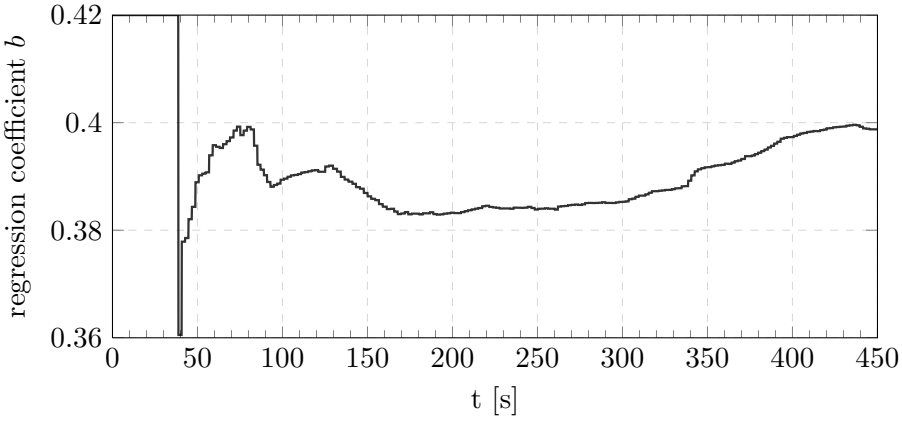
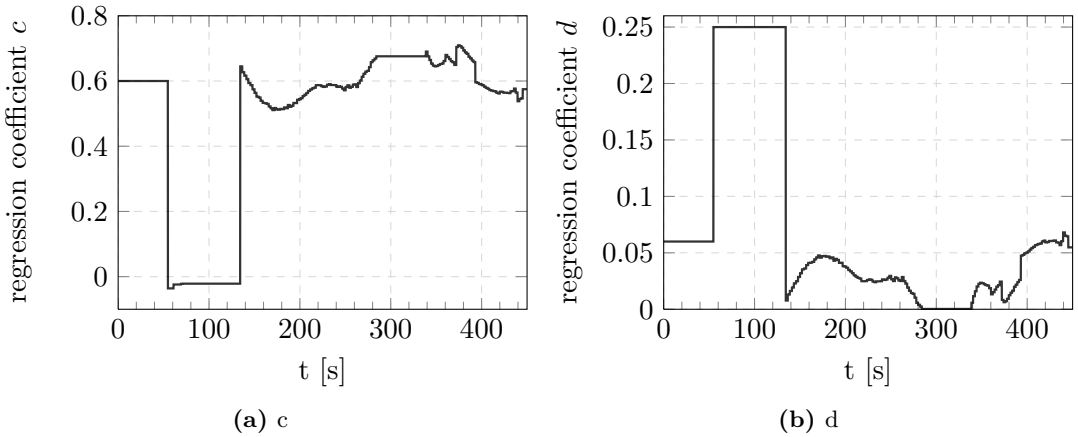
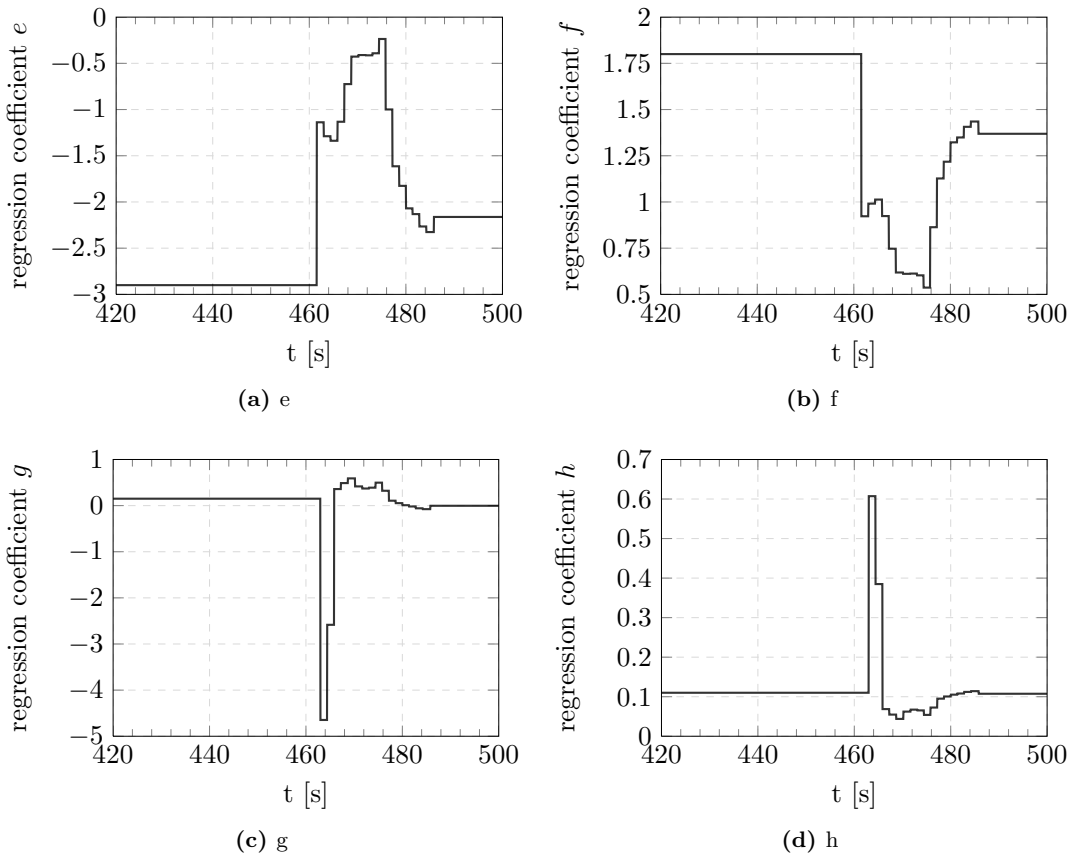


Figure 6.2: Estimation of the regression coefficient b, used in forward walk step length function.



(a) c **(b)** d

Figure 6.3: Estimation of the linear regression coefficients c and d, used in the forward walk step length function.



(a) e **(b)** f **(c)** g **(d)** h

Figure 6.4: Estimation of the linear regression coefficients e-h used in the step length function for running.

6.3 Evaluation of the Overall System Performance

The overall system is evaluated based on a approximately 720 m long walk performed by the five testers, where they walked forward all the time except for two short stops and two short intervals of running. The walk included walking on both asphalt and dirt road, on plain and hilly terrain and making both left and right turns. Walking only forward seems like a realistic scenario since humans in a normal situation almost exclusively walk forward. Nevertheless walking backward and sideways are also evaluated in separate tests. Note that the step length functions for these states are not trained from GPS data, but estimated from the forward step length function. The results from these tests are mainly an evaluation of this estimation.

Three different measures are used as reference distance when evaluating the performance of the system; measured distance, distance calculated from GPS speed and distance calculated from GPS coordinates.

The measured distance was obtained using a simple measuring wheel, measured only at one occasion. The field tests followed distinct land marks in the surroundings, such as the roadside, to make sure the testers walked the same distance. However, it is likely that the the true distance for each tester differs slightly from the measured distance.

Travelled distance from GPS speed is obtained by integration and the accuracy is vaguely specified to 0.1 m/s by the manufacturer. The provided value is speed over ground, and since the test distance included walking slightly uphill it is likely that the distance obtained is a little shorter than the true distance. Also the relatively low GPS update frequency of 1 Hz could contribute to errors when calculating the distance through integration.

The distance from GPS-coordinates is obtained using Bowring's formulae. A potential problem is that the GPS coordinates does not remain constant when standing still. The small changes in coordinates quickly adds up to a significant distance, although it could be reduced by introducing a speed threshold for the distance calculation. Distance calculated from GPS coordinates does not account for hilly terrain but the former issue will probably make the obtained distance slightly longer than the true distance.

The true distance for each of the tests could probably be considered to be somewhere in between the results obtained from the three sources discussed above. All three are considered in the evaluation to capture some sort of worst case performance.

6.3.1 Long Walk

The results from the simulations of the long walk is shown in Table 6.2. Two distances for the dead reckoning are presented; one using amplitude difference and one using variance as function variable. The maximum error obtained, considering all testers and all references, was 3.21 % when using amplitude difference and 9.05 % when using the variance. Table 6.3 presents the RMS-error of the five tests for each of the reference distances. Using amplitude difference for step length estimation gave the best result, outperforming the variance-alternative no matter which distance used as reference. However, as can be seen in Table 6.2, it is mainly the results from Tester 3 that contributes to the bad results of the variance-function.

Figure 6.5 shows the travelled distance for Tester 3 at a selected time-interval. As can be seen, the distance calculated from GPS coordinates is a bit longer than the distance calculated from GPS speed, as expected. At time $t = 137$ the tester starts running. It

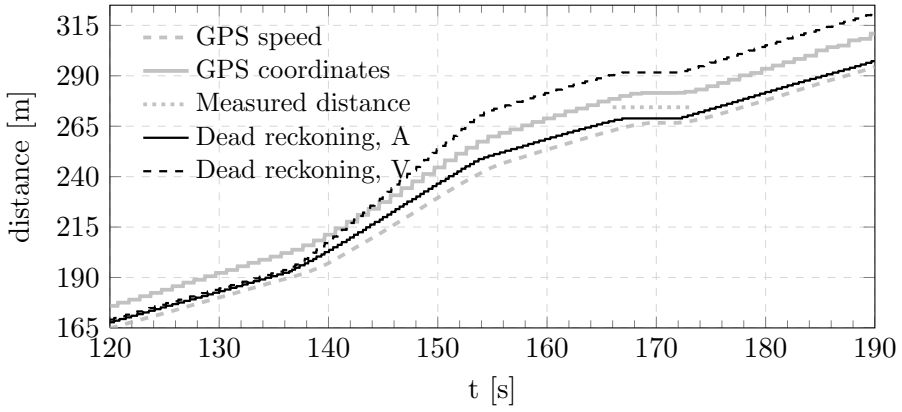


Figure 6.5: Travelled distance for Tester 3 in a selected time interval. Black lines shows distance from dead reckoning, solid for amplitude difference (A) and dashed for variance (V). The grey lines shows the three references. The reference distance obtained from measuring wheel is only plotted at the interval where the tester stands still, that is around $t = 169$.

is clear from the figure that the variance-function overestimates the step length when running, which explains the bad performance as compared with the amplitude-function. The tendency of the variance-function to overestimate the running step length is seen for all testers. Since the same data is used for training of the step length parameters, this indicates that using amplitude difference is better. However, during forward walk both alternatives seems to estimate the step length equally well.

	Dead reckoning		
	Amplitude	Variance	Reference [m]
Tester 1, distance [m]	**711.1**	**702.3**	
Error, GPS Speed	1.79 %	0.53 %	698.6
Error, GPS Coord	0.73 %	1.95 %	716.3
Error, Measured	1.35 %	2.57 %	720.8
Tester 2, distance [m]	**698.6**	**708.4**	
Error, GPS Speed	1.84 %	0.46 %	711.7
Error, GPS Coord	3.21 %	1.86 %	721.8
Error, Measured	3.08 %	1.72 %	720.8
Tester 3, distance [m]	**713.1**	**768.8**	
Error, GPS Speed	1.15 %	9.05 %	705.0
Error, GPS Coord	2.25 %	5.39 %	729.5
Error, Measured	1.07 %	6.66 %	720.8
Tester 4, distance [m]	**706.9**	**734.5**	
Error, GPS Speed	0.18 %	4.10 %	705.6
Error, GPS Coord	2.72 %	1.07 %	726.7
Error, Measured	1.93 %	1.9 %	720.8
Tester 5, distance [m]	**704.5**	**734.6**	
Error, GPS Speed	1.21 %	5.53 %	696.1
Error, GPS Coord	1.91 %	2.28 %	718.2
Error, Measured	2.26 %	1.91 %	720.8

Table 6.2: Results from the five testers. The table shows the results for the two different step length functions (amplitude difference and variance) compared to the three available references expressed as a percentage error.

	GPS Speed	**GPS Coord**	**Measured**
Amplitude	1.37 %	2.32 %	2.06 %
Variance	5.09 %	2.92 %	3.50 %

Table 6.3: RMS-errors taking all five testers into account, calculated for each of the reference distances.

6.3.2 Backward and Sideways Walk

The step length functions for sideways and backward walk are estimated from the forward step length function, i.e. no GPS-measurements of the testers actual step lengths for these motions are used. The RMS distance errors for sideways walk are shown in Table 6.4. The variance-function is the best alternative based on the results. The travelled distance during test was around 60 m (30 m right respectively left) for each of the tester, which is quite long for a sideways walk. It seems reasonable to assume that a typical distance performed sideways is around 10 m at maximum, which means that an error of 8 % would give around 80 cm in absolute error.

	GPS Speed	**GPS Coord**	**Measured**
Amplitude	8.21 %	9.88 %	6.68 %
Variance	6.75 %	8.20 %	5.88 %

Table 6.4: RMS-errors for sideways walk taking all five testers into account, calculated for each of the reference distances.

The results of the backward test is shown in Table 6.5. As for the sideways test, the variance-function performed better than the amplitude-function. The travelled distance for this test was not measured using a measuring wheel, and only two reference distances are therefore provided. The distance travelled by the testers was approximately 60 m. From the tests it seems like a typical error of around 6 % could be expected.

	GPS Speed	**GPS Coord**
Amplitude	5.81 %	7.37 %
Variance	4.52 %	6.65 %

Table 6.5: RMS-errors for backward walk taking all five testers into account, calculated for two reference distances.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

The proposed algorithm for step detection worked well and simulation showed that it only failed to detect 11 out of over 5500 steps. However, it also made false detections due to peaks in the accelerometer from irregular movements when the pedestrian was not walking. These false detections were expected, and the idea is that the classifier should discard them.

Two classifiers were implemented, the statistical approach using a Hidden Markov Model clearly outperformed the naive approach using logical operations. For a mixed walk, where five testers each performed motion in all states (forward, backward, sideways walk and running), the HMM-classifier showed 1361 correctly detected and classified steps out of 1383 steps made, whereas the naive approach showed 1302 correct classifications. Another major difference was that the HMM managed to discard most of the false detections made by the step detector, as opposed to the logical classifier. It is however a trade-off; most of the wrongly classified steps by the HMM occurred in the transition between standing still and starting to move where the classifier sometimes discarded the first steps. Changing some parameters of the HMM will enable these steps to be detected but also reduce its ability to discard false detections. However, occasionally failing to detect a step or two seems less devastating for the overall performance than making lots of false detections.

The overall system with step detection, classification and step length estimation was simulated using data from five testers performing a 720 meter distance with mixed walking and running. The simulation was run for two different step length functions, and the function using amplitude difference of the vertical acceleration as variable performed better than the one using the variance of the vertical acceleration. The former gave a RMS distance error of around 2 % whereas the latter had a RMS-error of around 3.5 %. The two alternatives performed equally well for walking, but the variance-function overestimated the running step length which explains the poor result. Since the very same data was used for estimation, this indicates that using the amplitude difference is more robust. Another major advantage with using the amplitude difference is that only one parameter needs to be estimated for the forward step length function which gives a much more robust estimation, especially when the number of measurements are few. The RMS-error per step for the best step length function was around 1.6 cm, and with knowledge of the highly various nature of human stride this seems to be a fairly

satisfying result.

The performance of the system for backward and sideways walk was evaluated in separate tests, giving a RMS distance error of around 6.5 % and 8 % respectively. The results are not very impressive, but two things need to be kept in mind. Firstly, the step length functions for these motions have not been trained but only estimated from the forward step length function. Secondly, it seems reasonable to assume that pedestrians rarely walk longer distances than, say, 10 m backwards or sideways. With this assumption we talk about absolute errors of less than 1 meter. The main benefit from being able to detect backward and sideways walk is that the motion tracked by the INS is in the right direction. Consider a 10 m backward walk. If all motion is assumed to be forward walk this would result in a positioning error of around 20 m since the movement actually is in a direction 180 degrees from the heading. A distance error of a couple of meters but with a movement in the right direction is significantly better.

The results from the simulations are promising and maybe even sufficient depending on the application. The most problematic part in this kind of INS is often the bearing estimation, since magnetometers are prone to suffer from disturbances. It is likely that the errors assigned to the bearing estimation will be the major part of the positioning error.

7.2 Further Work

Although the simulations showed promising results there are several aspects that need to be investigated more thoroughly and further work are needed in the following areas:

- Investigation of how the step detection and classifier works in different scenarios, depending on application. Examples could be walking up- or downhill with a large slope, stair walking, walking with a heavy backpack, walking in a crowed urban environment and at different surfaces such as mud.
- Investigations of which states that are most important to use. For example, sideways walk might not be critical to detect since humans rarely walk sideways. It might be more interesting to be able to detect stair walking or crawling depending on application.
- Investigations of how sensitive the step detector and classifier is to potential misplacement of the IMU-platform, for example if it is not tightly fastened towards the wearer's chest.
- Development of algorithms to decide when training of the step length can take place. Training in areas where the GPS suffers from multipath effects or other errors could corrupt the step length estimation.

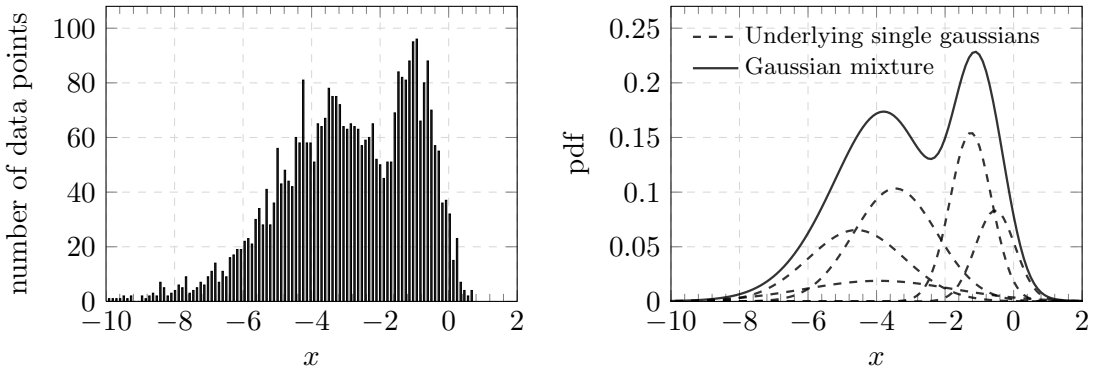
Before the actual GPS/INS system can be taken into use further work is needed in the following areas which were not addressed in this thesis:

- Development of a method for bearing estimation.
- Development of an algorithm for GPS and INS fusion.

Appendix A

Gaussian Mixture Models

The Gaussian Mixture Model is powerful in the sense that it can be used to model any probability distribution. Consider Figure A.1a, showing the histogram of a one-dimensional feature recorded during forward walk. From the histogram its clear that a single Gaussian is not sufficient to approximate the distribution, since it is not symmetric and has several peaks. However, using a mixture of five Gaussian distributions gives a good approximation of the feature distribution, as can be seen in Figure A.1b. The tool for solving the estimation problem is called Expectation Maximization.



(a) Histogram for variable x.

(b) EM-estimated GMM for x.

Figure A.1: The variable x is one of the Fourier coefficients from the FFT of the z-axis accelerometer during forward walk. A total of 3555 data points from seven different pedestrians were collected. The left figure is a 100-bin histogram showing how the variable x is distributed. The right figure shows the EM-estimated GMM and its five underlying components.

A.1 Training of GMM using Expectation Maximization

As mentioned in Section 2.3, the HMM observation probability distribution is modelled using GMMs. Since each HMM state has its own distribution, this implies a total N number of GMMs for the HMM. Each GMM are trained separately based on acquired data for respectively state. As an example, consider the case of training a GMM for the state “walking forward”. A subset of training data is collected from a pedestrian

walking forward. The data consists of a set of observation vectors $(\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_T)$, and each observation vector consists of k well chosen features extracted from the IMU signals. Several subsets of training data can be collected from different pedestrians walking forward and put together to the final set of training data

$$D = \big((\mathbf{O}_{1,1}, \ldots, \mathbf{O}_{1,T_1}), (\mathbf{O}_{2,1}, \ldots, \mathbf{O}_{2,T_2}), \ldots, (\mathbf{O}_{P,1}, \ldots, \mathbf{O}_{P,T_P})\big) \tag{A.1}$$

The idea is to assume that an observed feature vector $\mathbf{O}$ is the outcome of a stochastic process defined by a GMM, and use the training data to estimate the GMM parameters. Recall the pdf for a GMM

$$f(\mathbf{O}) = \sum_{m=1}^{M} \phi_m \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}_m|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{O} - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{O} - \boldsymbol{\mu}_m) \right) \tag{A.2}$$

$$\sum_{m=1}^{M} \phi_m = 1$$

The parameters to be estimated are the mean vectors $\boldsymbol{\mu}_m$, the covariance matrices $\boldsymbol{\Sigma}_m$ and the mixture weights ϕ_m. One parameter has to be pre-set, namely the number of components, M.

The EM algorithm is an iterative algorithm that gives increased likelihood each iteration, although convergence to a global maximum is not guaranteed [18]. The measure to be maximized is the probability that the observed data set D is the outcome from a stochastic process defined by a GMM with parameter set $\theta = \{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, expressed as a log likelihood

$$\log\big(P(D|\theta)\big) = \sum_{t=1}^{T} \log\left(\sum_{m=1}^{M} \phi_m \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \right) \tag{A.3}$$

Obviously it is desired to obtain as big likelihood as possible for this measure. Maximization of (A.3) with respect to θ seems a bit tricky, since the second sum appears inside of a logarithm. To solve this problem EM proposes to introduce a latent, unobservable, variable z_{tm} that provides information about from which of the M single Gaussian components the observation $\mathbf{O}_t$ was drawn [18][19]. At every time instance t this variable is equal to 1 for the component that produced the observation and 0 for the $(M-1)$ remaining components. It is now possible to rewrite the measure to be maximized as

$$\log\big(p(D, \mathbf{z}|\theta)\big) = \sum_{t=1}^{T} \sum_{m=1}^{M} z_{tm}\big(\log\phi_m + \log\mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big) \tag{A.4}$$

Now, the hitch is that there is no way to know the latent discrete variable z_{tm}. It is however possible to calculate the expected value of the latent variable. The probability of z_{tm} being equal to 1 is decided by the mixture weights, $P(z_{tm} = 1) = \phi_m$, and the probability of an observation $\mathbf{O}_t$ is found in (A.2). The probability of an observation $\mathbf{O}_t$ given the latent variable z_{tm} is of course decided by the Gaussian distribution, $P(\mathbf{O}_t|z_{tm} = 1, \theta) = \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. The desired conditional probability of z_{tm} being equal to 1 given the observation $\mathbf{O}_t$ is found using Baye's rule

$$\gamma_{tm} = P(z_{tm} = 1|\mathbf{O}_t, \theta) = \frac{\phi_m \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{j=1}^{M} \phi_j \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{A.5}$$

The EM-function to be maximized can now be written as the expectation with respect to z_{tm} of (A.4)

$$q(\theta', \theta) = \sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_{tm} \big(\log \phi'_m + \log \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}'_m, \boldsymbol{\Sigma}'_m) \big) \tag{A.6}$$

where the prim notation is used to denote the new parameters to be estimated from the old one. The interested reader can find the derivations to obtain the parameter set θ' that maximizes (A.6) in [18], the resulting algorithm becomes

1. Expectation:

$$\gamma_{tm} = \frac{\phi_m \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{j=1}^{M} \phi_j \mathcal{N}(\mathbf{O}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{A.7}$$

2. Maximization:

$$\phi'_m = \frac{1}{T} \sum_{t=1}^{T} \gamma_{tm} \tag{A.8}$$

$$\boldsymbol{\mu}'_m = \frac{\sum_{t=1}^{T} \gamma_{tm} \mathbf{O}_t}{\sum_{t=1}^{T} \gamma_{tm}} \tag{A.9}$$

$$\boldsymbol{\Sigma}'_m = \frac{\sum_{t=1}^{T} \gamma_{tm} (\mathbf{O}_t - \boldsymbol{\mu}'_m)(\mathbf{O}_t - \boldsymbol{\mu}'_m)^T}{\sum_{t=1}^{T} \gamma_{tm}} \tag{A.10}$$

After each EM-iteration the new model is evaluated by calculating the log-likelihood $\log \big(P(D|\theta') \big)$ according to (A.3) and compare it to the log-likelihood for the previous model. If the difference is below some fixed threshold, the algorithm stops.

The observant reader realizes that an initial guess of the model is necessary in order to start the EM-algorithm. Since EM only guarantees convergence to a local maxima this initial guess will strongly affect the outcome of the algorithm. One way to obtain an initial model is to use a clustering algorithm, e.g. the well known k-means clustering. Starting off with M random centroids, each observation vector $\mathbf{O}_t$ in the training data is assigned to the closest centroid in a euclidean sense. After assigning all observation vectors to a cluster, new centroids are calculated as the sample mean of the data in each cluster. The procedure with assigning each observation vector to the closest centroid starts over, and the algorithm is repeated until the centroids do not change any more according to some threshold. Mean-vectors and covariance matrices are then calculated for each cluster, and the mixture weights is determined by the proportion of data in each cluster.

Since the EM-algorithm only provides a local maxima, the procedure with K-means clustering followed by EM-estimation could be repeated a few times. The GMM with largest resulting likelihood is then kept.

Bibliography

[1] P. Strömbäck, J. Rantakokko, S.-L. Wirkander, , M. Alexandersson, K. Fors, I. Skog, and P. Händel, "Foot-Mounted Inertial Navigation and Cooperative Sensor Fusion for Indoor Positioning," in *Proceedings of the International Technical Meeting of The Institute of Navigation (ION 2010)*, (San Diego, CA, USA), Jan. 2010.

[2] P. Goyal, V. Ribeiro, H. Saran, and A. Kumar, "Strap-Down Pedestrian Dead-Reckoning System," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN 2011)*, (Guimaraes, Portugal), pp. 1–7, Sept. 2011.

[3] W. Chen, R. Chen, Y. Chen, H. Kuusniemi, and J. Wang, "An Effective Pedestrian Dead Reckoning Algorithm Using a Unified Heading Error Model," in *Position Location and Navigation Symposium (PLANS)(IEEE/ION 2010)*, (Indian Wells, CA, USA), pp. 340–347, May 2010.

[4] W. Chen, Z. Fu, R. Chen, Y. Chen, O. Andrei, T. Kröger, and J. Wang, "An Integrated GPS and Multi-Sensor Pedestrian Positioning System for 3D Urban Navigation," in *Urban Remote Sensing Event, 2009 Joint*, (Shanghai, China), pp. 1–6, May 2009.

[5] Q. Ladetto, "On Foot Navigation: Continuous Step Calibration Using Both Complementary Recursive Prediction and Adaptive Kalman Filtering," in *Proceedings of the 13th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2000)*, (Salt Lake City, UT, USA), pp. 1735–1740, Sept. 2000.

[6] S. Lee and K. Mase, "Recognition of Walking Behaviours for Pedestrian Navigation," in *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA '01)*, (Mexico City, Mexico), pp. 1152–1155, Sept. 2001.

[7] Q. Ladetto, J. van Seeters, S. Sokolowski, Z. Sagan, and B. Merminod, "Digital Magnetic Compass and Gyroscope for Dismounted Soldier Position & Navigation," in *Military Capabilities enables by Advances in Navigation Sensors, Sensors & Electronics Technology Panel, NATO-RTO meetings*, (Istanbul, Turkey), Oct. 2002.

[8] T. Nguyen, Y. Zhang, and M. Griss, "ProbIN: Probabilistic Inertial Navigation," in *Proceedings of the 2010 IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, (San Francisco, CA, USA), pp. 650–657, Nov. 2010.

[9] G. Shi, Y. Zou, Y. Jin, X. Cui, and W. Li, "Towards HMM Based Human Motion Recognition Using MEMS Inertial Sensors," in *Proceedings of the 2009 IEEE Interna-*

tional Conference on Robotics and Biomimetics, (Bangkok, Thailand), pp. 1762–1766, Feb. 2009.

[10] W. Wan, H. Liu, L. Wang, G. Shi, and W. Li, "A Hybrid HMM/SVM Classifier for Motion Recognition Using μIMU Data," in *Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics*, (Sanya, China), pp. 115–120, Dec. 2007.

[11] V. Gabaglio, Q. Ladetto, and B. Merminod, "Kalman Filter Mechanization for GPS-INS Pedestrian Navigation," in *Proceedings of the Global Navigation Satellite System Meeting (GNSS 2001)*, (Sevilla, Spain), May 2001.

[12] CH Robotics, *CHR-6dm Attitude and Heading Reference System, Product Datasheet*, rev. 1.1, preliminary ed.

[13] GlobalSat Technology Corporation, *Product User Manual, RS232 GPS Receiver BR-355*.

[14] S. Preece, J. Goulermas, L. Kenney, and D. Howard, "A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data," *IEEE Transactions on Biomedical Engineering*, vol. 56, pp. 871–879, March 2009.

[15] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso, "Preprocessing Techniques for Context Recognition from Accelerometer Data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[16] "Fast Fourier Transform, R2011b MATLAB documentation." http://www.mathworks.se/help/techdoc/, Feb. 2012.

[17] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb. 1989.

[18] J. A. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," Tech. Rep. TR-97-021, International Computer Science Institute and Computer Science Division U.C. Berkeley, Berkeley CA, April 1998.

[19] A. Leijon, "Pattern Recognition; Fundamental Theory and Exercise Problems." Lecture Notes, Royal Institute of Technology, 2007.

[20] N. Romanov, "Distinctive Characteristics of Usain Bolt's Running Technique." http://www.posetech.com/training/archives/000791.html, Nov. 2009.

[21] H. Wienberg, "Using the ADXL202 in Pedometer and Personal Navigation Applications." Analog Devices AN-602 Application Note, 2002.