Implementation of a Simulator/Demonstrator for the SoftLocation Concept using Bayesian Filters

Master Thesis

by

Mohammed Ahmed Khider Bashir

Information Technology
University of Ulm
November 2005

M/2005/wt/03

Information Technology University of Ulm
University of Ulm · Information Technology · Albert-Einstein-Allee 43 · 89081 Ulm · Germany

Master Thesis

Implementation of a Simulator/Demonstrator for the SoftLocation Concept using Bayesian Filters

Description:

Modern mobile devices such as smartphones or personal digital assistants are not only equipped with network connectivity, significant CPU-, memory- and display resources but increasingly with GPS receivers as location sensors. The knowledge about the user's location is typically employed to facilitate so called "location-based" services that aim to assist the user in tasks such as finding restaurants, shops or other places of interest in personal navigation.
While the accuracy provided by the satellite-based global positioning system is sufficient for many outdoor applications, it strongly suffers in indoor and urban canyon environments due to blockage of the line-of-sight between the satellite and the mobile device. In contrast, other technologies for location determination, such as time-of-arrival, time difference-of-arrival or angle-of-arrival measurements of signals in the mobile networks or inertial measurement units suffer less from these factors but lack global coverage or long-term stability.
A combination of the various techniques to join their strengths is therefore desirable. The concept of "soft location" has been developed in order to achieve an optimum combination of the information provided by the individual sensors. Bayesian filtering algorithms are interesting candidates for realizing the soft location concept
The objective of this thesis is the implementation and integration of Bayesian filtering algorithms into the MOSCITO (Mobile Simulation of Context Information Tool) software platform. Sophisticated visualization (e.g. particles, probability density functions, map and satellite imagery) is required in order to analyze the correctness and performance of the algorithms. Simulations are to be performed in order to quantitatively compare the effect of parameters and algorithmic nuances on the estimation result. The algorithms are to be integrated and tested with hardware sensors (GPS receiver and electronic compass) and real-time date to demonstrate and evaluate their applicability and performance under real world conditions.

Date due:	05.11.2005
Author:	Mohammed Ahmed Khider Bashir
Supervisor:	Dr. Patrick Robertson, Dr. Michael Angermann, Kai Wendlandt, Institute of Communications and Navigation, German Aerospace Center(DLR) Dr. rer. nat. Werner Teich, IT Department, University of Ulm
Examiner:	Prof. Dr.-Ing. J. Lindner
Thesis No.:	M/2005/wt/03

I certify that I have prepared this Master Thesis by my own without any inadmissible outside help.

Ulm, 05.11.2005

(Mohammed Ahmed Khider Bashir)

To
my respected parents
Prof. Ahmed Khider and Mrs. Aziza Ahmed,

and my beloved sisters
Dr. Nada, Shatha, Maha and Allaa,

who inspired me to reach the goal by conquering the mind,
I proudly dedicate this master thesis.

Acknowledgement

First of all, I would like to extend my thanks to the University of Ulm for giving me the opportunity to study Communications Technology at a high standard. Special thanks are sent to the department of Information Technology and for the Communication Systems department at the German Aerospace Center (DLR) for the interesting 6-months period which I spent as a member of their group.

In particular, I would like to express my appreciation with respect and profound gratitude for my supervisors Prof. Dr.Ing. Jürgen Lindner, Dr. rer. nat. Werner Teich, Dr. Uwe-Carsten Fiebig, Dr. Patrick Robertson, Dr. Michael Angermann, Kai Wendlandt for their faith and confidence in me which has pushed me to try to do well. Thanks for guidance, instructions, help and willingness to answer my questions with patience and understanding.

Thanks to anyone who has ever taught me anything and to all those who have contributed to this work in one way or another.

Finally, a special thank is also sent to my aunty Mrs. Fatima Ahmed, who deserves special admiration and love. My thanks are also sent to my most beloved sisters and all my true friends as well as my dear relatives. Above all, this achievement is gratefully dedicated to my parents whose prayers, love, enthusiasm, support and encouragement have always led me to success.

Abstract

The mantra of location based services (LBS) is "to offer the right service, at the right time and in the right place". Accurate knowledge of a user's location is required to achieve this goal. Unfortunately, all location sensors suffer from environmental conditions, such as GPS sensors suffering from signal blocking and multipath in urban canyons. In consequence, the quality of location based services is often poor, due to noisy locations measurements. Using multiple and complementing sensors might be a good solution. However, optimally combining these noisy sensor measurements is a challenge.

This work has started from the existing concept of Soft Location, i.e. representing and combining sensor output by probability density functions instead of "hard" point estimates. Based on this concept the applicability of Bayesian filtering has been investigated. A computationally efficient algorithm for Bayesian filtering, namely Particle Filtering has been selected, discussed in theory and implemented. An advanced movement model for pedestrians, involving several aspects of a user's situation, such as emotional state or age, has been developed for the prediction stage of the Particle Filter. The Particle Filter implementation has been integrated in an experimental testbed for indoor and outdoor positioning with a GPS sensor and an electronic compass. Simulations and real-time experiments have been successfully carried out to qualitatively and quantitatively analyze the performance of the proposed solution. The results are promising and presented and discussed with a focus on position and direction accuracy in the light of the number of particles, a key factor in computational complexity of Particle Filtering.

Contents

1 Introduction **15**
- 1.1 Location Based Services . . . 15
- 1.2 Location estimation methods . . . 15
- 1.3 From location to situation . . . 17
- 1.4 Security and privacy . . . 18
- 1.5 Accuracy problem and solutions . . . 19

2 Background theory **21**
- 2.1 Soft Location Concept . . . 21
- 2.2 Initial Soft Location Method . . . 21
  - 2.2.1 Example 1 - Initial Soft Location method . . . 23
- 2.3 Advanced soft location method - Bayesian Filters . . . 24
  - 2.3.1 Motivation . . . 24
  - 2.3.2 Problem Statement . . . 25
  - 2.3.3 Dynamic System Model . . . 26
  - 2.3.4 Bayesian Filter Algorithm . . . 28
  - 2.3.5 Example 2 - Position estimation using Bayesian Filter . . . 30
- 2.4 Optimal Bayesian Algorithms . . . 32
  - 2.4.1 Kalman Filter . . . 32
  - 2.4.2 Multi-hypothesis tracking (MHT) . . . 34
  - 2.4.3 Grid-based method . . . 34
- 2.5 Sub-Optimal Bayesian Algorithms . . . 36
  - 2.5.1 Extended Kalman Filter . . . 36
  - 2.5.2 Approximate Grid-Based Methods . . . 37
  - 2.5.3 Particle Filtering Methods . . . 38
  - 2.5.4 Example 3 - Position estimation using Particle Filter . . . 47
  - 2.5.5 Comparison between Bayesian Filter and Particle Filter . . . 49

3 Human movement model **51**
- 3.1 Introduction . . . 51
- 3.2 Markovian States . . . 53
- 3.3 System states . . . 53
- 3.4 From states to movement . . . 55
- 3.5 μ - σ Combiner . . . 63
- 3.6 Inferring pedestrian situation from his movement . . . 69

4 System design **73**
4.1 Problem statement . . . 73
4.2 Requirements . . . 73
4.3 System setup . . . 73
4.4 Simulation setup . . . 75
4.5 Software design . . . 76

5 Implementation aspects **79**
5.1 Hardware . . . 79
5.1.1 Hardware components . . . 79
5.1.2 Hardware functionality . . . 80
5.2 Software . . . 81

6 Test, Integration and Experiments **93**
6.1 Simulation setup . . . 93
6.2 Real system integration . . . 93

7 Simulation results **97**
7.1 Qualitative analysis . . . 97
7.2 Quantitative analysis . . . 104

8 Conclusions and Outlook **113**
8.1 Conclusions . . . 113
8.2 Outlook . . . 114

A Statistics tables **117**

B Statistics figures **121**

C States equations **149**

D Equipment photos **153**

E NMEA protocol **155**

Chapter 1

Introduction

1.1 Location Based Services

Location-based services (LBS) provide users of mobile devices with personalized services tailored to their current location. They open a new market for developers, cellular network operators, and service providers to develop and deploy value-added services: advising users of current traffic conditions, supplying routing information, helping them find nearby restaurants, and many more. Providing users with specific services that fit their tastes is one of the hot issues in location-based services.

Location-based services answer questions like: Where am I? What's around me? How do I get there? They determine the location of the user by using one of several technologies for determining position, and then use the location and other information to provide personalized applications and services. As an example, let us consider a wireless 911 emergency service that determines the caller's location automatically. Such a service would be extremely useful, especially to users who are far from home and do not know local landmarks. Traffic advisories, navigation help including maps and directions, and roadside assistance are natural location-based services. Other services can combine present location with information about personal preferences to help users find food, lodging, and entertainment to fit their tastes and budget.

1.2 Location estimation methods

To discover the location of the pedestrian, LBS must use real-time positioning methods. Accuracy depends on the method used and the environment in which the user is. Locations can be expressed in spatial terms or as text descriptions. A spatial location can be expressed in the widely used *latitude-longitude-altitude* coordinate system. Latitude is expressed as 0-90 degrees north or south of the equator and longitude as 0-180 degrees east or west of the prime meridian, which passes through Greenwich, England. Altitude is expressed in meters above sea level. A text description is usually expressed as a street address, including city, postal code, and so on.

There are three competing principles for determining the position of a mobile radio user. The first one operates with directional antennas at the receiver to determine angle

of arrival (AOA) of a radio transmitter. But there are strong limitations in multipath environments (indoor) due to reflections from wrong directions. The second principle needs specialized hardware at the transmitter and the receiver to determine a distance between two stations. This can be achieved by measuring the signal delay points (time of arrival, TOA or time difference of arrival between different stations - TDOA). The use of triangulation algorithms then leads to a position when several such stations are in range. In general this can be the most precise method of location determination depending on the complexity of the setup. The third technique uses signal strength and other parameters measured from a radio link. The data is used either to directly determine a distance (free-space transmission loss proportional to $1/r^2$) or to calibrate a room or an area with typical measurement data.
Significant *accuracy, availability & integrity* improvements might be achieved if two or more of the above mentioned methods could be combined. Several types of positioning methods exist:

- *Using the mobile phone network:* The current cell ID can be used to identify the Base Transceiver Station (BTS) that the device is communicating with and the location of that BTS. Clearly, the accuracy of this method depends on the size of the cell, and can be quite inaccurate. A GSM cell may be anywhere up to 35 kilometers in diameter. TOA, TDOA and AOA along with cell ID can achieve accuracy within 150 meters.

- *Using Global Navigation Satellite Systems (GNSSes):* The Global Positioning System (GPS), controlled by the US Department of Defense, uses a constellation of 24 satellites orbiting the earth. Of course Galelio is the future alternative GNSS. GPS determines the device's position by calculating differences in the times signals from different satellites take to reach the receiver (TDOA). The mobile device has to be equipped with a GPS receiver in order to receive the satellite signals. GPS is potentially the most accurate method (several meters if the GPS receiver has a clear view of the sky), but it has some drawbacks: the extra hardware can be costly, consumes battery while in use, and requires some warm-up after a cold start (no initial position estimates or no almanac satellite data) to get an initial fix on visible satellites. It also suffers from "canyon effects" in cities, where satellite visibility is intermittent, or multipath reception disturbs the range measurements.

- *Using short-range positioning beacons:* In relatively small areas, such as a single building, a wireless local area network can provide locations along with other services. The device's position is determined using signal strength and other parameters measured from the radio link. For example, appropriately equipped devices can use Bluetooth, RFIDs and WLANs for short-range positioning.

- Accelerometers, optical tracking systems (like Lasertracker [1]) and the Cricket Indoor Location System [2] are other sensors which can be used for positioning. Lasertracker and Cricket Indoor systems use TOA principle for location estimation.

In addition, location methods can connect to a mobile position center that provides an interface to query for the position of the mobile subscriber. While applications can be fully self-contained on the device, it is clear that a wider array of services is possible when a server-side application is part of the overall service.

1.3 From location to situation

Three coordinates are needed to determine the exact position of any object. If it is possible to estimate the position fix accurately, a significant improvement in location based services can be achieved.
On the other hand if we extend the three coordinates to have more dimensions, then a point in the space will specify a situation instead of location. Examples of these new dimensions can be the time of the day, the direction of view, the speed, the activity, the hungriness, the weather, the mood, the age, the interest, etc. In such paradigm we will be talking about "Situation Based Services" instead of "Location Based Services" [3]. Services provided to the end user can be personalized and improved according to his situation. As a result, the quality of the service can be improved as illustrated in the following examples:

- If the system knows the location of the pedestrian, it can offer him the next ice cream shop, the next coffee shop, the next restaurant, etc.
- If the system additionally knows that the weather is cold, then offering the next hot drink shop is preferable.
- If the system additionally knows that the end user is tired, then a hot drink shop with a relaxing area is more convenient.

The story does not end on services limits. Location can be used to get information about situation and vice versa. Examples of getting location knowledge from situation are as follows:

- If the end user is clearly shopping, then he is not in an area without a shopping facility.
- If the end user is active, then he is not resting in bed.
- If the end user is sporting - running, then he is most probably in an outdoor recreation area.

Examples of getting situation knowledge from location are as follows:

- If the end user is on the bed, then most probably he is tired.
- If the end user is in a shopping mall, then most probably he is shopping.
- It the end user is in a restaurant, then he is most probably sitting or walking slowly.

Additionally, if the system knows the movement behavior of the pedestrian, then information regarding his activity, activeness, age and other situation states can be approximated. For example if the variation of the angle of movement of the pedestrian is high, then the pedestrian is most probably drunk.
Figure(1.1) shows an example of this coordinate extension. Some possible additional dimensions are listed on the right side of the figure ordered from the slow changing factors to the fastest.

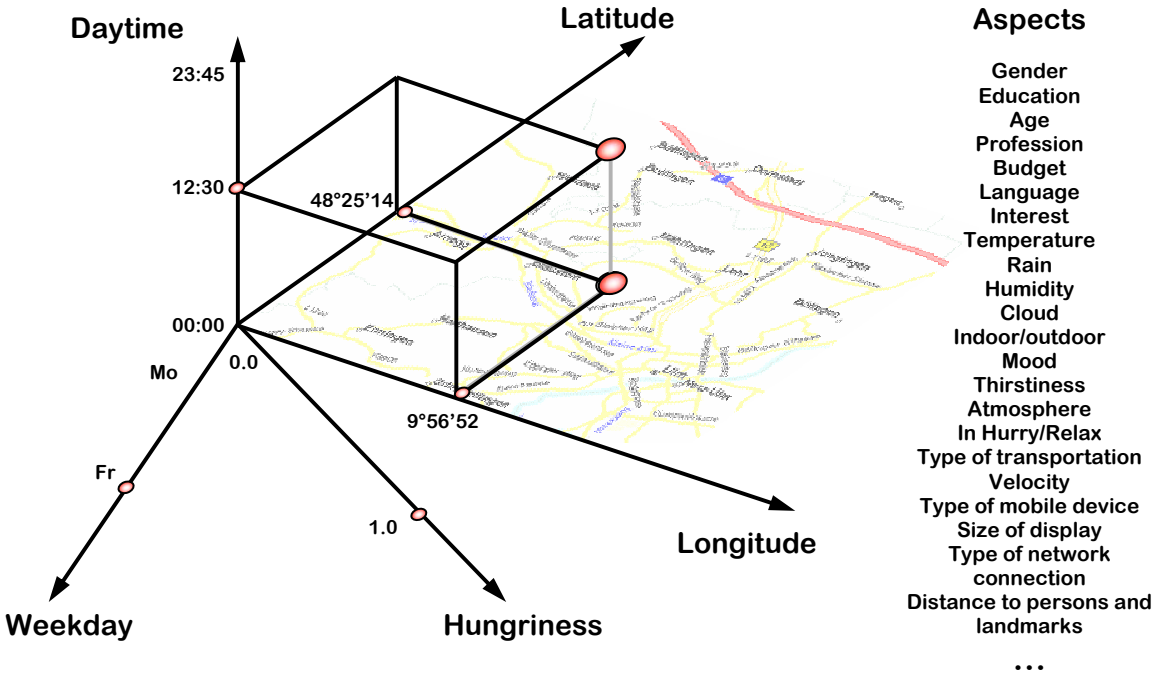

Figure 1.1: Situation Space - General Model, originally found in [3]

1.4 Security and privacy

Many users consider location information to be highly sensitive, and are concerned about a number of privacy issues [4], including:

- *Target marketing:* Mobile users' locations can be used to classify customers for focused marketing efforts.
- *Embarrassment:* One customer's knowledge of another's location may lead to embarrassing situations.
- *Harassment:* Location information can be used to harass or attack a user.
- *Service denial:* A health insurance firm might deny a claim if it learned that a user visited a high-risk area.

- *Legal restrictions:* Some countries regulate the use of personal data.

For these and other reasons, users must know when and how their location is given to any specific application.

1.5 Accuracy problem and solutions

Some applications don't need high accuracy, but others will be useless if the location isn't accurate enough. Its okay for the location of a tourist walking around town to be off by 10 meters, but other applications and services may demand higher accuracy. A car on a highway may cope with perhaps 50m accuracy in order to achieve a certain quality-of-service, where as a system for pedestrian user looking for a shop in the city is only helpful with 5 - 10 meters tolerance. An aircraft on final approach may cope with an accuracy of only few meters. A museum guide will also require an accuracy of few meters.
A big challenge is indoor navigation where positioning data from GPS satellites are inaccurate. That is due to multipath disturbance and shadowing. Other position sensors like Bluetooth, Compasses, accelerometers, RFIDs and WLANs operate at various degrees of accuracy and often suffer from independent errors.

Over the last years, there has been a proliferation of research and papers addressing indoor navigation and suggesting solutions for its challenges.

Soft Location is one of the new paradigms which improve indoor navigation and yields more accurate location estimation.

Chapter 2

Background theory

2.1 Soft Location Concept

Soft Location "SoLo" is an estimate which tells the probability of a user being at a location ("soft decision"), instead of giving a point estimate of the position ("hard decision") [5]. It uses Probability Density Functions (PDFs) for giving soft decisions. An initial soft location method is to represent position sensors output in terms of location more generally as a probability density function (PDF) of the location over two or three dimensional space - typically Cartesian or other coordinates. Combining two or more such PDFs yields a more accurate PDF of the location and improves navigation under difficult circumstances like in indoor scenarios.

A more advanced soft location method is to use Bayesian filter techniques which probabilistically estimate the state of a dynamic system from a sequence of noisy sensor observations. In the most basic form of location estimation, the state of interest is the location of a person or object, and observations are provided by sensors either placed in the environment or carried by the person. Roughly speaking, it provides an answer to the question "What is the probability that the person is at location l if the history of sensor measurement is known?"

Bayesian filters provide a powerful tool to help manage measurement uncertainty and multi-sensor fusion. Their statistical nature makes Bayesian filters applicable to arbitrary sensor types and representations of environments. Bayesian filters provide a sound approach to location estimation using GPS data along with street maps or signal strength information along with topological representations of indoor environments. Furthermore they have been applied with great success to a variety of state estimation problems including speech recognition, target tracking, vision and robotics.
Bayesian filtering algorithms have improved the accuracy and the convergence of state estimation, which is why they are widely used these days.

2.2 Initial Soft Location Method

Representing position sensors output in terms of location as a PDF and combining two or more such PDFs yields a "more accurate" PDF of the location. It provides

significant improvement in indoor position estimation where navigation faces difficult circumstances. A PDF - based representation is selected for this soft decision issue [6] because:

- Only a few parameters might be necessary to define the PDF.
- Super-positioning is an easy mathematical operation
- They can be positioned in an absolute coordinate system.

Now comes the question "how to combine the output of several positioning sources to yield a probability density function (PDF) of the location over space?" First some assumptions should be made:

1. The sources are assumed to be independently distributed, in other words each source of position suffer from statistically independent errors.
2. Each individual positioning function or device i returns a PDF of the location. The location is defined within the coordinate system (x, y, z).
3. The PDF is assumed to be accurate in the sense that it correctly models all errors of the function. These errors must include technical failures, measurement inaccuracies, malevolent attack on the system, and other frequent or infrequent events. Many of these errors will result in the PDF taking on small but nonzero values far from the peak or main area. As a result, the PDF is an accurate indication of the reliability of the correct estimation, without being either too optimistically or pessimistically accurate.

The independence criterion does not imply that positioning sources are uncorrelated; in fact they are expected to be correlated. The PDF of source i is actually a conditional density distribution, $p_l^{(i)}(l|o(i))$ conditioned on a specific set of discrete observations or measurements $o(i)$. For example, a GPS receiver will yield a PDF of the location l conditioned in effect on its antenna input signal and HW/SW characteristics and configurations, as well as the GPS's current status.

Given a number n of these PDFs, $p_l^{(1)}(l|o(1)) \cdots p_l^{(n)}(l|o(n))$, we are able to compute the total PDF of the location given all observations $o(1) \cdots o(n)$,$p_l^{(1)}(l|o(1), o(2), \ldots, o(n))$. The above assumptions can be formulated as follows:

$$P\{o(i) = o_m | l = l_x) = P\{o(i) = o_m | l = l_x, o(j) = o_n\} \; \forall i \neq j; \; \forall o_m, o_n, l_x \tag{2.1}$$

This means that the probability of the estimator with index i receiving its observation $o(i) = o_m$ under the assumption of the location being l_x is independent of the value o_n of the set of observations of another estimator j.

Using Bayes' Rule and the prementioned assumptions an equation for the optimal combiner can be written as follows [6]:

$$p_l^{(t)}(l|o(1), o(2), \ldots, o(n)) = \frac{\prod_{i=1}^{n}(p_l^{(i)}(l|o(i))}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(p_l^{(i)}(l|o(i))dxdydz} \tag{2.2}$$

This means that PDFs are multiplied over location space and then the result is normalized to unity in the integral. If discrete definitions of l - e.g. topological descriptions - are used, then the PDFs can be replaced with probabilities $P^{(i)}\{l|o(i)\}$ and the integrals become sums. This also applied when a discrete grid is considered.

From equation (2.2) we can see that the location PDF is forced to zero for those locations if any single estimator i has a zero in $P^{(i)}\{l|o(i)\}$ at that location. As a consequence, attention should be paid in modelling the PDF even for unlikely locations. For instance, a localized beacon transmitter might carry the ID or the coordinates of a room which is installed in. If this beacon is falsely installed, then it will lead to a failure after any further processing, even if it is "outvoted" by other estimators.

The suggested soft location method has shown noticeable improvement in sensor fusion, and location estimation as a result. On the other hand, it has also shown some weaknesses. Those weaknesses are described next.

i. Error sources are assumed to be independent up to now. In a distributed system it will be difficult to handle correlations between the errors of the different estimators for practical reasons, even if such correlations could be represented mathematically and be computed. Correlations which exist but are not taken into account usually result in over accentuation of the finally calculated PDF of the location. In practice, such correlations will probably pose little problems due to different positioning techniques being used and combined. Errors such as a software bug in a mobile radio system that affects base stations of multiple networks, for example, will produce such correlations, or inaccuracies of commonly used HW elements of a joint mobile radio and satellite navigation receiver.

ii. Another weakness of this initial soft location method is its ignorance of the system dynamics. Using the knowledge of the pedestrian dynamics might improve location estimation significantly.

iii. In addition, the need of a soft location method that estimate situation rather than only location encourages researchers to search for alternative methods.

2.2.1 Example 1 - Initial Soft Location method

Two positioning sources in two dimensional space (i.e. for constant altitude) are used:

1. A "shark" fin PDF that represents the location PDF resulting from the reception of sectorised mobile radio base station signal with a certain signal strength. For a strong signal strength this PDF will be more pronounced, since the likelihood is then high that the receiver is within the antenna sector. A three dimensional view of the PDF is shown in figure (2.1), where x and y are the x- coordinate and the y - coordinate respectively.

2. A round doughnut shaped PDF that depends on the time advance given by a certain mobile radio base station (e.g. the same base station as above). A three dimensional view of the PDF over is shown in figure (2.2), where x and y are the x- coordinate and the y - coordinate respectively.

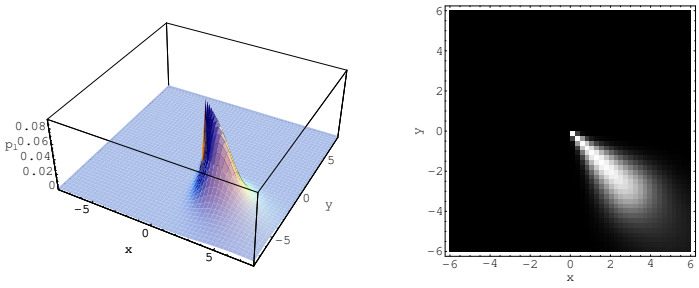

Figure 2.1: PDF of the location using time advance (estimate 1)

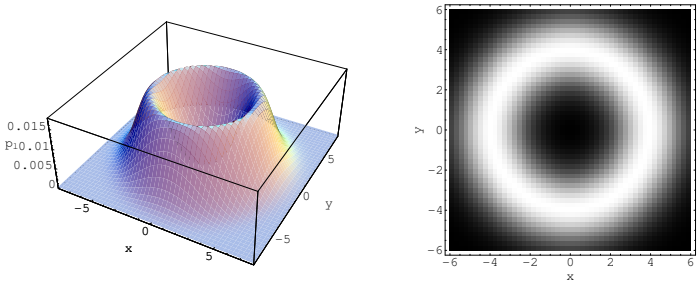

Figure 2.2: PDF of the location using multi-sector antenna (estimate 2)

Combing the two PDFs according to equation (2.2) results in the PDF in figure (2.3). The sharp peak can be used to locate the person.

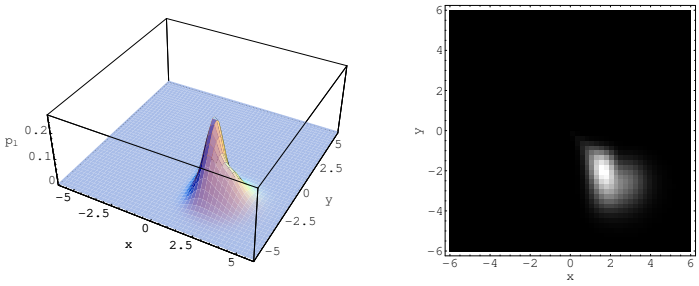

Figure 2.3: PDF of the location after combining estimates 1 and 2

2.3 Advanced soft location method - Bayesian Filters

2.3.1 Motivation

Many real-world data analysis tasks require estimation of the state of a system that changes over time using a sequence of noisy measurements made on the system. In most of these applications, prior knowledge about the phenomenon being modelled is available. This knowledge allows us to formulate a model that can be used to find

a system state of interest at any time step given the previous one. Additionally, in most cases there is one or more sensors that provide some noisy observations regarding the state of interest. Such a system is a dynamic system in which the measurements will be used to calculate likelihood functions relating the noisy observations to the states, and the state transition model will be used to find a prior distribution for the unknown states of interest. A model that is characterized by likelihoods and a state transition model is called a *Bayesian models*. Accordingly, filters that use such models are called Bayesian filters. They represent the state at time t by random variables x_t. At each point in time, the estimation of this random variable is provided as a probability distribution over the state space. The key idea is to generate these posterior distributions at each time step based on all available information, including the set of received measurements. Bayesian filtering appears under different names according to the field of interest, such as optimal(nonlinear) filtering, stochastic filtering and on-line inference and learning. Examples of applications where Bayesian filtering can play a major role include: tracking an aircraft using radar measurements, estimating the position of a walking pedestrian, estimating a digital communications signal using noisy measurements, or estimating the volatility of financial instruments using stock market data.

Arnaud Doucet [7] points out that "If the data are modelled by a linear Gaussian state-space model, it is possible to derive an exact analytical expression to compute the evolving sequence of posterior distributions. This recursion is the well known and widespread Kalman filter. If the data are modelled as a partially observed, finite state-space Markov chain, it is also possible to obtain an analytical solution, which is known as the Hidden Markov model HMM filter". Details on the HMM filter can be found in [8], [9]. The Kalman filter will be discussed in section (2.4.1).

Highly restrictive assumptions should hold in order to have a tractable solution for any of the prementioned filters. Many of the real life applications involves elements of non-linearity, high dimensionality and non-linearity. In such conditions, the assumptions that the analytical solutions are built on become invalid. Many approximations schemes, such as extended Kalman filter, approximate grid-based filters, and particle filters have been proposed to surmount this problem. Different approximations resulted in different computation and design complexity, cost, flexibility and parallelization limits.

2.3.2 Problem Statement

Our objective is to track the state of a system as it evolves over time. Sequentially arriving (noisy or ambiguous) observations are given. The best possible estimate of the state is required. The tracked system is dynamic and its underlined models are nonlinear and non-Gaussian. This dynamic system is modelled using a state space approach. According to Arulampalam [10] "the state space approach is convenient for handling multivariate data and nonlinear/non-Gaussian processes, and it provides a significant advantage over traditional time-series techniques for these problems". For dynamic state estimation, discrete-time formulation is widespread and convenient. Thus, difference equations are used to model the evolution of the system with time, and

measurements are assumed to be available at discrete times. Figure (2.4) summarize what is given and what is required in discrete time.

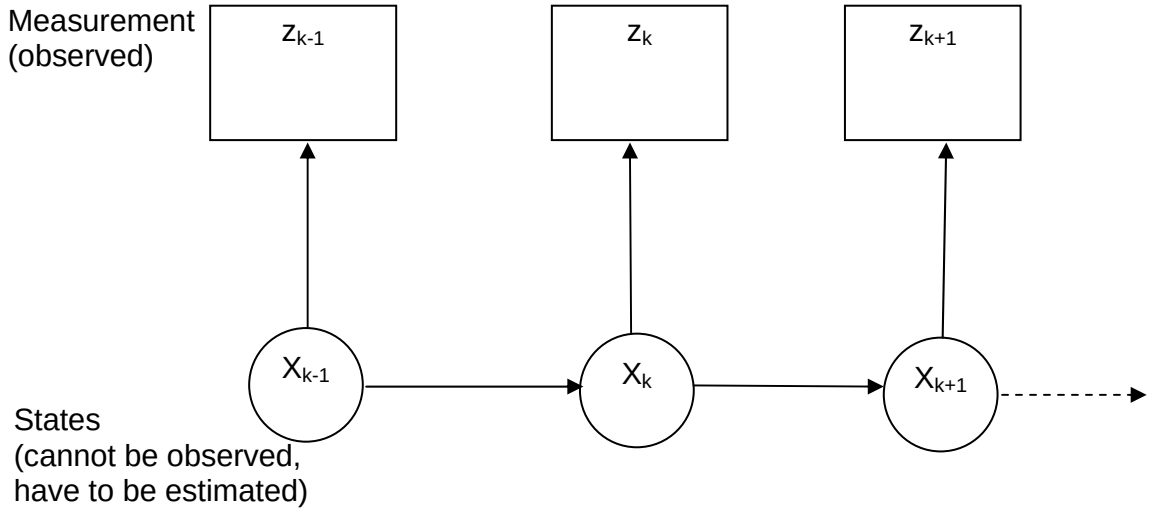

Figure 2.4: Problem statement in discrete time

Often, the observations arrive sequentially in time and one is interested in processing the data as they arrive. This means the that the probability densities should be estimated sequentially conditioned on the noisy observations provided by the sensors. It is therefore necessary to update the posterior distribution as data becomes available. This is called on-line inference [11] and such filtering is called recursive filtering [10]. Computational simplicity is a key advantage of on-line inference, since there is no need to store all the data nor to reprocess existing data if a new measurement becomes available.

In order to analyze and make estimations regarding the state of the pre-described dynamic system, at least two models are required: First, a model describing the evolution of the state with time (the system model). It predicts the state PDF forward from one measurement to the next. The prediction generally translates, deforms and spreads the state PDF since the states are modelled as random variables using the state transition model. Second, a model relating the noisy measurements to the state (the measurement model). It modifies the prediction PDF according to the received measurement. These models are assumed to be available in a probabilistic form. This probabilistic state-space formulation and the need to update the probability densities as new measurements arrive are well suited for the Bayesian approach. This represents a general framework for dynamic state estimation problems.

2.3.3 Dynamic System Model

In the following, the two needed models to analyze any dynamic system will be obtained. The first model is the state transition equation (system model) which is the *movement model* in tracking problems. The state sequence $\{x_k, k \in \mathbb{N}\}$ is given by:

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{2.3}$$

where $f_k : R^{D_x} \times R^{D_v} \rightarrow R^{D_x}$ is a possibly nonlinear function of the state x_{k-1} and $\{v_{k-1}, k \in \mathbb{N}\}$, an i.i.d. process noise sequence;
D_x, D_v: are the dimensions of the state and process noise vectors, respectively;
$\mathbb{N}$: is the set of natural numbers.
From the above equation we can see that a state x_k at k^{th} time is a function of the previous state x_{k-1} and some process noise. The state transition equation can be written in a probabilistic form as an a priori distribution "prior": $p(x_k|x_{k-1})$, which is the conditional probability of any state x at time k given the previous state x_{k-1}.

The second model is the measurement model which is given by:

$$z_k = h_k(x_k, n_k) \tag{2.4}$$

where $h_k : R^{D_x} \times R^{D_v} \rightarrow R^{D_x}$ is a possibly nonlinear function;
$\{n_k, k \in N\}$ is an i.i.d. measurement noise sequence;
D_x, D_v are dimensions of the measurement and measurement noise vectors, respectively.
From the above equation we can see that a measurement z_k at k^{th} time is a function of the k^{th} state and some measurement noise. The measurement model can be written in a probabilistic as a likelihood:$p(z_k|x_k)$, which is the conditional probability of any measurement z at time k given the current state x_k.

The measurements are observed out of some noisy sensors. The states can not be observed and have to be estimated out of those noisy measurements. So they are hidden variables. The System model together with the measurement model form a first order Hidden Markov Model. Diagram (2.5) shows how these two models form a first order Hidden Markov Model.

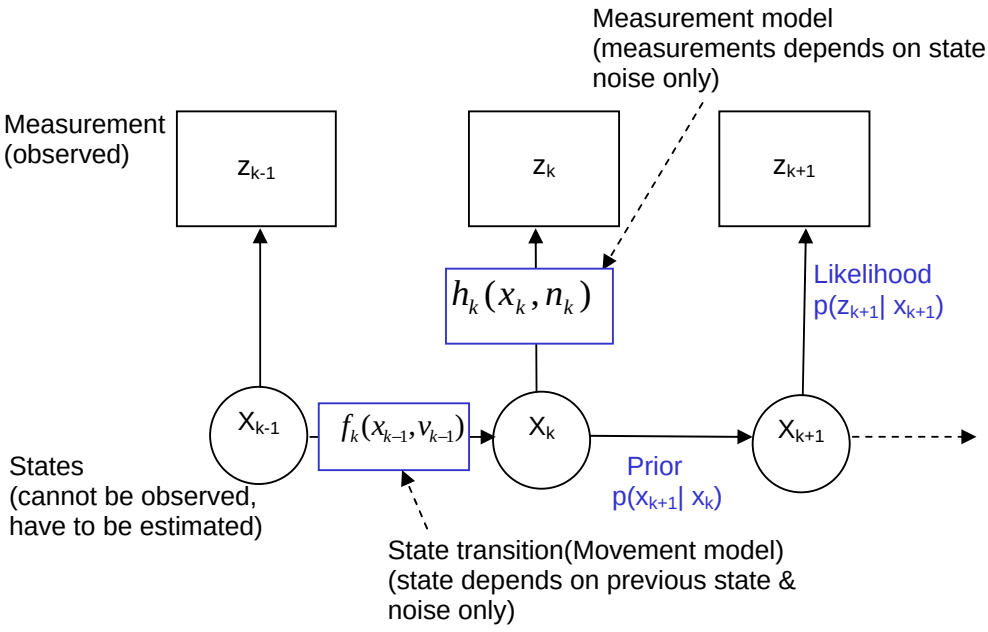

Figure 2.5: First order Hidden Markov Model representing the dynamic system

2.3.4 Bayesian Filter Algorithm

A mathematical formulation of the problem statement is given next. The following prior information is given:

- The initial PDF $p(x_0|z_0) \equiv p(x_0)$ of the state, which is also known as the prior (z_o being the set of no measurements)
- The likelihood $p(z_k|x_k)$ which is obtained from the measurement model.
- The prior $p(x_k|x_{k-1})$ which is obtained from the movement model.
- A set of noisy observations $z_{1:k} = \{z_i, i = 1, \ldots, k\}$.

A State vector $x_{0:k} = (x_0, \ldots, x_k)$ should be estimated. In other words, the posterior distribution $p(x_{0:k}|z_{1:k})$ should be calculated. Storing the path $x^i_{0:k-1}$, and the history of observations $z_{1:k-1}$ is inconvenient. Luckily it is more common that a filtered distribution $p(x_k|z_{1:k})$ is required. In these cases Recursive Bayesian Filtering is applicable, which only requires x^i_k to be stored.

A Recursive Bayesian Filter consists of two stages:

- Prediction stage: The movement model is used to predict the state PDF from one measurement to the next
- Update stage: The latest measurement is used to modify the prediction PDF.

A mathematical equation for each stage will be obtained. According to the total law of probability:

$$p(x_k) = \int p(x_k|x_{k-1})p(x_{k-1})dx_{k-1} \tag{2.5}$$

where,
$p(x_k)$: is the probability of any state x_k at time k,
$p(x_k|x_{k-1})$: is the conditional probability of any state x_k at time k given the previous state x_{k-1}.

An equation for the prediction stage could be derived using the pre-mentioned total law of probability:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}, z_{1:k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.6}$$

where,
$p(x_k|z_{1:k-1})$: is the prior PDF of the state x_k at time k given the history of measurements $z_{1:k-1}$ up to time $k-1$,
$p(x_k|x_{k-1}, z_{1:k-1})$: is the joint probability of a state x_k at time k given the previous state x_{k-1} with the history of measurements $z_{1:k-1}$ up to time $k-1$,
$p(x_{k-1}|z_{1:k-1})$: is the PDF of the previous state x_{k-1} given the history of the measurements $z_{1:k-1}$ up to time $k-1$.
The PDF $p(x_{k-1}|z_{1:k-1})$ is assumed to be available at time $k-1$.

Bayes theorem [12], is a mechanism for updating knowledge about the target state in the light of extra information from new data. According to it:

$$p(x_k|z_k) = \frac{p(z_k|x_k)p(x_k)}{p(z_k)} \tag{2.7}$$

where,
$p(x_k|z_k)$: is the conditional probability of any state x_k at time k given the measurement at the same time z_k,
$p(z_k|x_k)$: is the conditional probability of any measurement z_k at time k given the state at the same time x_k. It is also called the likelihood,
$p(x_k)$ and $p(z_k)$: are the probabilities of any state and any measurement at time k consecutively.

An equation for the update stage can be derived using the pre mentioned Bayes Rule:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k, z_{1:k-1})}{p(z_k|z_{1:k-1})} p(x_k|z_{1:k-1}) \tag{2.8}$$

where,
$p(x_k|z_{1:k})$: is the conditional probability of any state x_k at time k given the history of measurements $z_{1:k}$ up to time k. It represents the posterior,
$p(z_k|x_k, z_{1:k-1})$: is the joint probability of the conditional probability of any measurement z_k at time k given the state x_k at the same time with the history of measurements $z_{1:k-1}$ up to time $k-1$,
$p(x_k|z_{1:k-1})$: is the prior PDF of the state at time k given the history of measurements $z_{1:k-1}$ up to time $k-1$, and it is calculated at the prediction stage,
$p(z_k|z_{1:k-1})$: is the probability of any measurement z_k given the history of measurements $z_{1:k-1}$ up to time $k-1$.

From the first order Hidden Markov Model, we notice that if the previous state x_{k-1} is known, then the movement model can be used to find the current state x_k without the need for any previous measurement. Accordingly the prediction equation can be simplified to:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|z_{1:k-1}) dx_{k-1} \tag{2.9}$$

where $p(x_k|z_{k-1}, z_{1:k-1})$ is replaced with $p(x_k|z_{k-1})$.

We also notice that if any state x_k is known at time k then the measurement model can be used to find the current measurement without the need for any previous measurements. Accordingly, the update equation can be simplified to:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)}{p(z_k|z_{1:k-1})} p(x_k|z_{1:k-1}) \tag{2.10}$$

where $p(z_k|x_k, z_{1:k-1})$ is replaced by $p(z_k|x_k)$.

Again using the total law of probability the normalization constant $p(z_k|z_{1:k})$ can be calculated to be:

$$p(z_k|z_{1:k}) = \int p(z_k|x_k, z_{1:k-1}) p(x_k|z_{1:k-1}) dx_k = \text{constant} : \alpha_k \tag{2.11}$$

The final two equations for prediction and update are as follows:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.12}$$

$$p(x_k|z_{1:k}) = \alpha_k p(z_k|x_k)p(x_k|z_{1:k-1}) \tag{2.13}$$

From the above prediction and update equations we can notice the following:

- The prediction equation uses the movement model and the given previous PDF $p(x_{k-1}|z_{1:k-1})$ at time $k-1$ to expect a PDF for the current state.
- At time step k, a measurement z_k becomes available, and it will be used to update our prediction. The arrived measurement gives an indication regarding the correctness of our prediction.
- A closer look to the update equation shows that the update equation is the prediction equation multiplied by the likelihood $p(z_k|x_k)$ and a constant. The likelihood is calculated from the measurement model. So update is nothing more than the prediction weighted with the received measurement at time k.
- In the Bayesian process the prediction and the update are repeated as time evolves. So prediction keeps going till a measurement is received. When a measurement is received an update happens and then prediction continues.

2.3.5 Example 2 - Position estimation using Bayesian Filter

An example of the Bayesian filtering process will be shown. A walking person in one direction should be tracked. A movement model is a person walking in one direction. So the person position at time $z+1$ can be predicted according to this movement model. Using the movement model, the prior $p(x_k|x_{k-1})$ is derived. The prior is needed in the prediction stage as equation (2.9) shows.

This person reports whenever he sees a door. These reports are the only kind of measurements which are available. Using these measurements a measurement model is constructed. Using the measurement model the likelihood $p(z_k|x_k)$ is derived. The update is needed in the update stage to weight our prediction as equation (2.13) shows.

A prediction for the position of the pedestrian keeps going according to the movement model. At the time the guy reports that he sees a door (a measurement is received) the prediction will be weighted or corrected. This means that the prediction distribution where the doors are will get high values and get low values elsewhere. As we see from equation (2.10) the likelihood (from the measurement) is multiplied by the prediction equation for this weighting purpose.
After this update the prediction goes on waiting for another measurement. A graphical illustration is shown in figure (2.6).

The prediction and update equations form a recursive propagation which is only a conceptual solution. In general, it can not be determined analytically. Typically, the normalization constant $p(z_k|z_{1:k})$, the marginal of the posterior $p(x_{0:k}|y_{1:k})$, in particular $p(x_k|y_k)$ can not typically be computed since they require the evaluation of complex

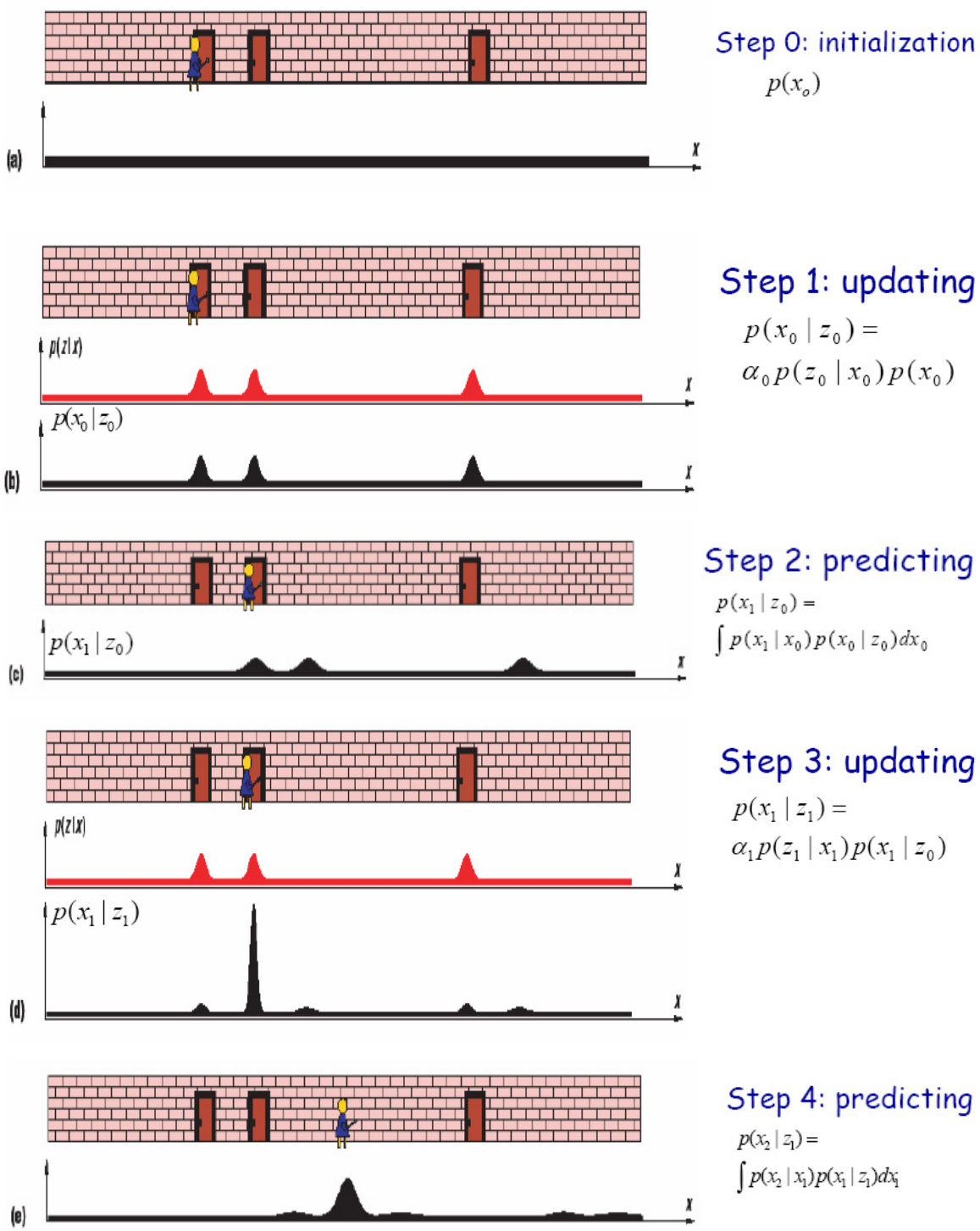

Figure 2.6: Position estimation using Bayesian Filter, originally found in [13]

high-dimensional integrals. Exact solutions exist but under many restrictions and assumptions. On the other hand many approximations also exist for the cases when optimal solution is intractable. The properties of these solutions and their implementations strongly differ in the way they represent the probability density functions (PDFs) over the state x. This results in different types of Bayesian Filters. Accuracy, robustness, efficiency, sensor variety and complexity differ among different types of Bayesian Filters accordingly. Kalman Filter, Multi-hypothesis tracking (MHT) and grid-based filters provide an optimal but restrictive solution. They are optimal because they solve the problem of recursively calculating the exact posterior density. They deduce a solution for this recursion. However, strong assumptions and restrictions have to be made for a valid Kalman, MHT or grid-based solution. On the other hand, Extended Kalman Filter, Approximate Grid Based and Particle Filter provide approximate solutions. They approximate the optimal Bayesian solution. When assumptions for optimal solutions do not hold, then approximate solutions are used. Each of the Bayesian Filter techniques will be briefly discussed. Advantages and disadvantages of each technique will be mentioned. Also the conditions under which each solution or approximation is tractable will be pointed out.

2.4 Optimal Bayesian Algorithms

Optimal finite dimensions algorithms for recursive Bayesian state estimations are formulated into two main categories. In a linear and Gaussian case, the functional recursion of prediction and update results in the Kalman filter. However, if the state space is discrete with a finite number of states, the grid-based methods are optimal. Each of these optimal algorithms will be explained next.

2.4.1 Kalman Filter

The Kalman filter is the most widely used variant of Bayesian Filters. Roughly speaking, the Kalman filter approximates probability distribution over x by unimodal Gaussian distributions, represented by their mean and variance. While the mean gives the expected state the variance represents the uncertainty in the estimate. Even though Kalman filters make strong assumptions about the nature of the sensors and the movement model, they have been applied with great success to various estimations problems.

Following assumptions are made during Kalman filter derivations [14]:

- The movement model is affected by a process noise drawn from a Gaussian distribution and is represented by v_{k-1}.
- The measurement model is affected by a measurement noise drawn from a Gaussian distribution and is represented by n_k.
- Movement model is a known linear function of x_{k-1} and v_{k-1} and is represented by $f_k(x_{k-1}, v_{k-1})$.

- Measurement model is a known linear function of x_k and n_k and is represented by $h_k(x_k, n_k)$

If the above assumptions hold and the prior PDF $p(x_{k-1}|z_{1:k-1})$ is Gaussian, then the posterior PDF $p(x_k|z_{1:k})$ is also Gaussian. As a result, the movement model equation (2.3) and the measurement model equation (2.4) can be re-written as:

$$X_k = F_k x_{k-1} + v_{k-1} \quad \text{(movement model)} \tag{2.14}$$
$$Z_k = H_k x_k + n_k \quad \text{(measurement model)} \tag{2.15}$$

where F_k and H_k are known matrices defining the linear functions of the movement and the measurement models.

In order to derive the prediction and update equation updates of the Kalman filter the following is considered:

- The covariances of v_{k-1} and n_k are named Q_{k-1} and R_k.
- v_{k-1} and n_k are assumed to be statistically independent.
- v_{k-1} and n_k are considered having zero mean.
- The system matrix F_k and the measurement matrix H_k, as well as noise parameters Q_{k-1} and R_k are allowed to be time variant.

Accordingly, the prediction equation (2.9) and the update equation (2.10), can then be viewed as the following recursive relationship:

$$p(x_{k-1}|z_{1:k-1}) = \mathcal{N}(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \tag{2.16}$$
$$p(x_k|z_{1:k-1}) = \mathcal{N}(x_k; m_{k|k-1}, P_{k|k-1}) \quad \text{Prediction Equation} \tag{2.17}$$
$$p(x_k|z_{1:k}) = \mathcal{N}(x_k; m_{k|k}, P_{k|k}) \quad \text{Update Equation} \tag{2.18}$$

where

$$m_{k|k-1} = F_k m_{k-1|k-1} \tag{2.19}$$
$$P_{k|k-1} = Q_{k-1} + F_k P_{k-1|k-1} F_k^T \tag{2.20}$$
$$m_{k|k} = m_{k|k-1} + K_k(z_k - H_k m_{k|k-1}) \tag{2.21}$$
$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} \tag{2.22}$$

where,
$N(x; m, P)$: is a Gaussian density with argument x, mean m, and covariance P,
$z_k - H_k m_{k|k-1}$: is an innovation term having a covariance S_k,
K_k: is the Kalman gain, and the transpose of a matrix M is denoted by M^T.
The covariance S_k and the Kalman gain K_k can be calculated according to:

$$S_k = H_k P_{k|k-1} H_k^T + R_k, \tag{2.23}$$
$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.24}$$

The derived Kalman solution provides an optimal solution for tracking and many Bayesian processes if its highly restrictive assumptions hold. In other words, no algorithm can ever do better than a Kalman filter in this linear Gaussian environment. A description of how to obtain the same result using a least square approach is described in [15].

The Kalman Filter's main advantage is its computational efficiency. Typical sensors used for Kalman filter based estimation are cameras, laser range-finders, and GPS systems. The Kalman filter's weakness is its too restrictive assumptions. Because of such restrictions the Kalman filter is not able to deal with common distributions and models. Example of which are: non-linear models, non-Gaussian noise or posterior, multi-modal distributions and skewed distributions.

2.4.2 Multi-hypothesis tracking (MHT)

Multi-hypothesis tracking extends Kalman filters to multi-modal beliefs[11]. MHT represents the belief by mixture of Gaussians where each hypothesis is tracked using a Kalman filter. The weights of the hypothesis are determined by how well they predict the sensor measurements. Due to their ability to represent multi-modal probability densities, MHT approaches are more widely applicable than the Kalman filter. Details on MHT can be found in [16].

2.4.3 Grid-based method

The Grid-based method is another optimal Bayesian algorithm. It provides an optimal recursion of the filtered density $p(x_k|z_{1:k})$ if the state space is discrete and consists of a finite number of states. It overcomes the restrictions imposed on Kalman filters by relying on discrete, piecewise constant representations of the probability distributions. For indoor estimation, grid-based filters tessellate the environment into small parches, typically of size between 10cm and 1m. Each grid cell contains the probability distribution over the cell. It gives the belief that the person is currently in the cell.

In order to derive the prediction and update equations for the Grid-based filter the following assumptions are made:

- The state space at time $k-1$ consists of discrete states $x^i_{k-1}, i = 1, \ldots, N_s$,
- For each state x^i_{k-1}, the conditional probability of that state, given measurements up to time $k-1$ is denoted by $w^i_{k-1|k-1}$, that is,

$$Pr(x_{k-1} = x^i_{k-1}|z_{1:k-1}) = w^i_{k-1|k-1} \tag{2.25}$$

Accordingly, the posterior PDF at $k-1$ can be written as:

$$p(x_{k-1}|z_{1:k-1}) = \sum_{i=1}^{N_s} w^i_{k-1|k-1}\delta(x_{k-1} - x^i_{k-1}) \tag{2.26}$$

where $\delta(.)$ is the Dirac delta measure.
Substitution of equation (2.26) into the prediction equation (2.9) and the update equation (2.10) yields:

$$\begin{aligned} p(x_z|z_{1:k-1}) &= \int \sum_{i=1}^{N_s} p(x_k^i|x_{k-1}) p(x_{k-1}|z_{1:k-1}) \delta(x_k - x_k^i) dx \\ &= \int \sum_{i=1}^{N_s} p(x_k^i|x_{k-1}^j) p(x_{k-1} = x_{k-1}^j|z_{1:k-1}) \delta(x_k - x_k^i) \delta(x_{k-1} - x_{k-1}^i) dx_{k-1} \\ &= \sum_{i=1}^{N_s} \sum_{j=1}^{N_s} p(x_k^i|x_{k-1}^j) p(x_{k-1} = x_{k-1}^j|z_{1:k-1}) \delta(x_k - x_k^i) \end{aligned} \tag{2.27}$$

and by definition:

$$w_{k|k-1}^i \triangleq \sum_{j=1}^{N_s} p(x_k^i|x_{k-1}^j) p(x_{k-1} = x_{k-1}^j|z_{1:k-1}) \tag{2.28}$$

Substituting 2.28 in 2.27) the prediction and update equations of the Grid-based filer, respectively.

$$p(x_k|z_{1:k-1}) = \sum_{i=1}^{N_s} w_{k|k-1}^i \delta(x_k - x_k^i) \quad \text{Prediction Equation} \tag{2.29}$$

$$p(x_k|z_{1:k}) = \sum_{i=1}^{N_s} w_{k|k}^i \delta(x_k - x_k^i) \qquad \text{Update Equation} \tag{2.30}$$

Substituting equation (2.25) in equation (2.28) yields:

$$w_{k|k-1}^i \triangleq \sum_{j=1}^{N_s} w_{k-1|k-1}^j p(x_k^i|x_{k-1}^j), \tag{2.31}$$

$$w_{k|k}^i \triangleq \frac{w_{k|k-1}^i p(z_k|x_k^i)}{\sum_{j=1}^{N_s} w_{k|k-1}^j p(z_k|x_k^j)} \tag{2.32}$$

The movement model $p(x_k^i|x_{k-1}^j)$ and the measurement model $p(z_k|x_k^i)$ are assumed to be known during Grid-based filter derivation, but the particular form of these discrete densities is not constrained. Again, Grid-based filter is an optimal solution if the assumptions hold.

A key advantage of these approaches is that they can represent arbitrary distributions over the discrete state space. The disadvantage of grid-based approaches is the computational complexity, which makes them applicable to low dimensional estimation problems only, such as estimating the position and the orientation of the person. The computation complexity of grid-based methods can be avoided by non-metric representations of an environment. Details on grid based methods can be found in [17].

2.5 Sub-Optimal Bayesian Algorithms

In many situations of interest, the restrictive assumptions of the optimal solutions do not hold. Optimal solutions are intractable in such cases. The Kalman filter and grid-based methods cannot, therefore, be used as some approximations are necessary. Three approximate nonlinear Bayesian filters are considered next:

(a) Extended Kalman filter (EKF)

(b) Approximate grid-based methods;

(c) Particle filters.

2.5.1 Extended Kalman Filter

Extended Kalman Filter (EKF) generalizes the standard Kalman filter so that it works for non linear systems also. The Gaussianity condition of the system's noise remains as a restriction for the EKF. EKF continually updates a linearization around the previous state estimate, starting with an initial guess. In other words, it only considers a linear Taylor approximation of the system function at the previous state estimate and that of the observation function at the corresponding predicted position. This approach gives a simple and efficient algorithm to handle a nonlinear model. However, convergence to a reasonable estimate may not be obtained if the initial guess is poor or if the disturbances are so large that the linearization is inadequate to describe the system.

If the movement model equation (2.3) and the measurement model equation (2.4) are not linear, then they cannot be rewritten in the form of the Kalman movement model equation (2.14) and the Kalman measurement model equation (2.15). However, a local linearization of the equations may be a sufficient description of the nonlinearity starting from an initial guess. The EKF is based on this approximation.

According to the Gaussian assumptions, the posterior $p(x_k|z_{1:k})$ can be approximated as follows:

$$p(x_{k-1}|z_{1:k-1}) \approx \mathcal{N}(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \tag{2.33}$$
$$p(x_k|z_{1:k-1}) \approx \mathcal{N}(x_k; m_{k|k-1}, P_{k|k-1}) \tag{2.34}$$
$$p(x_k|z_{1:k}) \approx \mathcal{N}(x_k; m_{k|k}, P_{k|k}) \tag{2.35}$$

where

$$m_{k|k-1} = f_k(m_{k-1|k-1}) \tag{2.36}$$
$$P_{k|k-1} = Q_{k-1} + \hat{F}_k P_{k-1|k-1} \hat{F}_k^T \tag{2.37}$$
$$m_{k|k} = m_{k|k-1} + K_k(z_k - h_k(m_{k|k-1})) \tag{2.38}$$
$$P_{k|k} = P_{k|k-1} - K_k \hat{H}_k P_{k|k-1} \tag{2.39}$$

and where now $f_k(.)$ and $h_k(.)$ are non-linear functions and F_k and H_k are local linearizations of these non-linear functions. F_k and H_k are matrices now and can be

formulated as:

$$\hat{F}_k = \left.\frac{df_k(x)}{dx}\right|_{x=m_{k-1|k-1}} \tag{2.40}$$

$$\hat{H}_k = \left.\frac{dh_k(x)}{dx}\right|_{x=m_{k|k-1}} \tag{2.41}$$

$$\tag{2.42}$$

Accordingly, the covariance of the innovation term of the Kalman filter $z_k - H_k m_{k|k-1}$, and the Kalman gain respectively are as follows:

$$S_k = \hat{H}_k P_{k|k-1} \hat{H}_k^T + R_k \tag{2.43}$$

$$K_k = P_{k|k-1} \hat{H}_k^T S_k^{-1} \tag{2.44}$$

As mentioned above the EKF utilizes the first term in a Taylor expansion of the nonlinear system. A higher order EKF that retains further terms in the Taylor expansion exists, but the additional complexity has prohibited its widespread use. Examples of such filter is the "Unscented Kalman filter" [18], [19].

The EKF solution is intractable if the approximation of the $p(x_k|z_{1:k})$ by a Gaussian does not hold. In other words, if the true density is bimodal or heavily skewed, then a Gaussian does not describe it well. In such cases, approximate grid-based filters and particle filters are more suitable approximations and will yield an improvement in performance [20].

2.5.2 Approximate Grid-Based Methods

Approximate Grid-based methods generalize Grid-based methods for cases where the state space is continuous. However, it should be possible to decompose the state space into N_s 'cells', $\{x_k^i, i = 1, \ldots, Ns\}$, in order to be able to approximate the posterior density with a Grid-based approach.

If the above described decomposition is possible, then the equality in the posterior pdf of the Grid-based methods at $k-1$ in equation (2.26) becomes an approximation:

$$p(x_{k-1}|z_{1:k-1}) \approx \sum_{i=1}^{N_s} w_{k-1|k-1}^i \delta(x_{k-1} - x_{k-1}^i) \tag{2.45}$$

Additionally, the equality in the prediction and the update equations of the Grid-based methods in equation (2.29) and equation (2.30) becomes an approximation:

$$p(x_k|z_{1:k-1}) \approx \sum_{i=1}^{N_s} w_{k|k-1}^i \delta(x_k - x_k^i) \quad \text{Prediction Equation} \tag{2.46}$$

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_{k|k}^i \delta(x_k - x_k^i) \qquad \text{Update Equation} \tag{2.47}$$

The grid points $x_k^i, i = 1, \ldots, N_s$, represent regions of continuous state space, and thus, the probabilities must be integrated over these regions. Accordingly, the prior $p(x_k^i|\bar{x}_{k-1}^j)$ can be written as:

$$p(x_k^i|\bar{x}_{k-1}^j) = \int_{x \in x_k^i} p(x|\bar{x}_{k-1}^j)dx \tag{2.48}$$

Substituting equation (2.48) in equation (2.31) and (2.32) yields:

$$w_{k|k-1}^i \triangleq \sum_{j=1}^{N_s} w_{k-1|k-1}^j \int_{x \in x_k^i} p(x|\bar{x}_{k-1}^j)dx, \tag{2.49}$$

$$w_{k|k}^i \triangleq \frac{w_{k|k-1}^i \int_{x \in x_k^i} p(z_k|x)dx}{\sum_{j=1}^{N_s} w_{k|k-1}^j \int_{x \in x_k^j} p(z_k|x)dx} \tag{2.50}$$

where $\bar{x}_{k-1}^j$, denotes the center of the j^{th} cell at time index $k-1$.

For simplicity, a further approximation is made in the evaluation of weights $w_{k|k}^i$ in practice. Specifically, these weights are computed at the center of the "cells" corresponding to x_k^i:

$$w_{k|k-1}^i \triangleq \sum_{j=1}^{N_s} w_{k-1|k-1}^j p(\bar{x}_k^i|\bar{x}_{k-1}^j), \tag{2.51}$$

$$w_{k|k}^i \approx \frac{w_{k|k-1}^i p(z_k|\bar{x}_k^i)}{\sum_{j=1}^{N_s} w_{k|k-1}^j p(z_k|\bar{x}_k^j)} \tag{2.52}$$

Selecting the appropriate number of cells is a challenge in approximate Grid-based methods. The grid must be sufficiently dense in order to get a good approximation to the continuous state space. However, as the dimensionality of the state space increases, the computational cost of the approach therefore increases dramatically. The grid-based approach shows poor results if the state space is not finite in dimensions. This is due to the truncation of the state space caused by the approach. Another disadvantage of grid-based methods is that the state space must be predefined and, therefore, cannot be partitioned unevenly to give greater resolution in high probability density regions, unless prior knowledge is used. More on approximate grid-based methods can be found in [8].

2.5.3 Particle Filtering Methods

Particle Filters represent probability densities over the states x_k by a set of random samples distributed according to the PDF. This random sampling is called Monte Carlo sampling. These samples are called particles. The continuous PDF is represented by N discrete samples (particles). Each of these particles has an associated weight. The points of higher density of the continues PDF will have more particles and those particles will have higher weights. Estimates are computed based on the number of

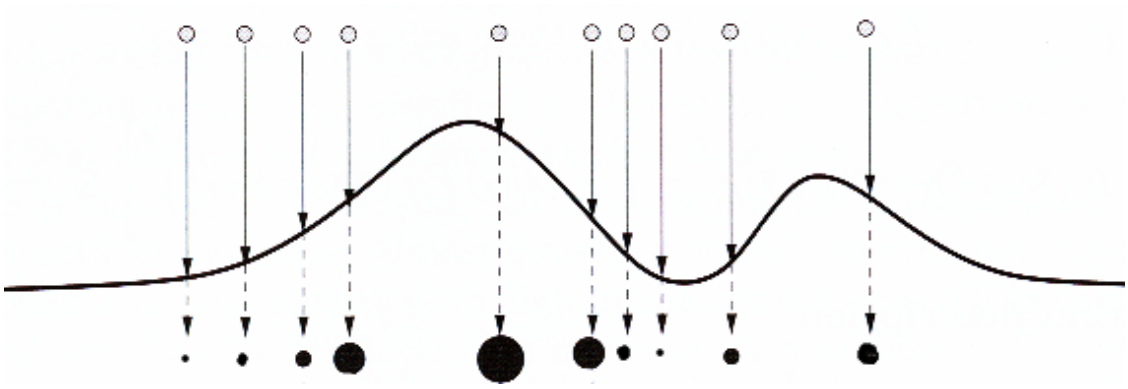

Figure 2.7: Particles representation of continuous PDF

particles and weights. The estimated state is the one at which there are the most number of particles and highest weights. This can be seen in figure (2.7). The discrete equation which is representing the continues PDF is:

$$p(x_k|z_{1:k}) = \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{2.53}$$

where:
N_s: number of samples(particles),
$\{x_k^i, i = 0, \ldots, N_s\}$: set of sampled states,
$\{w_k^i, i = 0, \ldots, N_s\}$: weights of sampled states.

For large numbers of samples the Random (Monte Carlo) Sampling approximation converges to the usual functional description of the posterior PDF, and the particle filter approaches the optimal Bayesian estimate.

Monte Carlo(MC) methods have the great advantage of not being subject to any linearity or Gaussianity constraints in the model, and they also have appealing convergence properties. It is, however seldom possible to obtain samples from these posterior distributions $p(x_k|z_{1:k})$ directly. One therefore has to resort to alternative MC methods, such as importance sampling in which sampling happens from what is called Importance density. By making this general MC technique recursive, one obtains the sequential importance sampling (SIS) method. Unfortunately, it can easily be shown that SIS is guaranteed to fail as time increases. This problem can be surmounted by including an additional selection (resampling) step [21]. According to ARULAMPALAM [22] "Since the introduction of Resampling, research activity in this field has dramatically increased, resulting in many improvements of particle filters and their numerous applications".

I. **The Sequential Importance Sampling (SIS) Algorithm**

Importance sampling [23], [24] is a general Monte Carlo (MC) integration method that is applied to perform nonlinear filtering approximation for the optimal Bayesian

solution. Monte Carlo are numerical methods that can be loosely described as statistical simulation methods, where statistical simulation is defined in quite general terms to be any method that utilizes sequences of random numbers to perform the simulation. Accordingly, the Monte Carlo term refers to random sampling. The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo method that forms the basis for most sequential Monte Carlo filters developed over the past decades [25], [7]. This sequential Monte Carlo (SMC) as the name describe deals with sequentially arriving data. It is known variously as bootstrap filters [21], the condensation algorithm [26], particle filters[27], Monte Carlo filters, interacting particle approximations [28] and survival of the fittest [29].

The key idea of SIS is the same general particle filtering key idea, which is representing the required posterior density $p(x_k|z_{1:k})$ by a set of random samples with associated weights and to compute estimates based on these samples and weights.

In order to develop the details of the algorithm, let $x_j = \{x_j, j = 0, \ldots, k\}$ represents the sequence of all target states up to time k. The joint posterior density at time k is denoted by $p(x_{0:k}|z_{1:k})$. Let $\{x^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ denote random measure that characterizes the posterior PDF $p(x_{0:k}|z_{1:k})$, where $\{x^i_{0:k}, i = 0, \ldots, N_s\}$ is a set of support points with associated weights $\{w^i_k, i = 1, \ldots, N_s\}$. The weights are normalized such that $\sum_i w^i_k = 1$. Accordingly, the posterior density up to time k can be approximated as:

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N_s} w^i_k \delta(x_{0:k} - x^i_{0:k}), \tag{2.54}$$

We therefore have a discrete weighted approximation to the true posterior $p(x_{0:k}|z_{1:k})$. The normalized weights are chosen using the principle of importance sampling. This principle relies on the following. Suppose $p(x) \propto \pi(x)$ is a probability density from which it is difficult to draw samples but for which $\pi(x)$ can be evaluated [as well as $p(x)$ up to proportionality]. In addition, let $x^i \sim q(x), i = 1, \ldots, N_s$ be samples that are easily generated from a proposal $q(.)$ called an importance density. Then, a weighted approximation to the density $p(.)$ is given by:

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i), \tag{2.55}$$

where

$$w^i \propto \frac{\pi(x^i)}{q(x^i)} \tag{2.56}$$

is the normalized weight of the i^{th} particle.

Therefore, if the samples, $x^i_{0:k}$, were drawn from an importance density $q(x_{0:k}|z_{1:k})$ then the weights in (2.54) are defined by (2.56) to be:

$$w^i \propto \frac{p(x^i_{0:k}|z_{1:k})}{q(x^i_{0:k}|z_{1:k})} \tag{2.57}$$

Importance density is defined to be factorized such that:

$$q(x_{0:k}|z_{1:k}) \triangleq q(x_k|x_{0:k-1}, z_{1:k})q(x_{0:k-1}|z_{1:k-1}) \tag{2.58}$$

This means that samples from $x^i_{0:k} \sim q(x_{0:k}|z_{1:k})$ by augmenting each of the existing samples $x^i_{0:k-1} \sim q(x_{0:k-1}|z_{1:k-1})$ with the new state $x_k \sim q(x_k|x_{0:k-1}, z_{1:k})$. However, if $q(x_k|x_{0:k-1}, z_{1:k}) = q(x_k|x_{k-1}, z_k)$, then the importance density becomes only dependent on x_{k-1} and z_k. This assumption holds only if filter estimates of $p(x_k|z_{1:k})$ are required at each time step. Practically, this is the case in most of the Bayesian applications.

Using the previous assumption and as derived in [10] and [22], the weight update is found to be:

$$w^i_k = w^i_{k-1}\frac{p(z_k|x^i_k)p(x^i_k|x^i_{k-1})}{q(x^i_k|x^i_{k-1}, z_k)} \tag{2.59}$$

and the posterior filtered density $p(x_k|z_{1:k})$can be approximated as:

$$p(x_|z_{1:k}) \approx \sum_{i=1}^{N_s} w^i_k \delta(x_k - x^i_k), \tag{2.60}$$

where the weights are defined in (2.59). It is visible that as $N \to \infty$ the approximation (2.60) approaches the true posterior density $p(x_k, z_{1:k})$.

Filtering via Sequential Importance Sampling (SIS) thus consists of recursive propagation of importance weights w^i_k and support points x^i_k as each measurement is received sequentially. A pseudo-code description of this algorithm is given by algorithm 1. SIS is a general algorithm that forms the basis of most particle filters. The choice of importance density functions plays a critical role in particle filtering process.

ALGORITHM 1: SIS PARTICLE FILTER

$[\{x^i_k, w^i_k\}^{N_s}_{i=1} = \text{SIS}\,[\{x^i_{k-1}, w^i_{k-1}\}^{N_s}_{i=1}, z_k]$

- FOR $i = 1 : N_s$
  - Draw $x^i_k \backsim q(x_k|x^i_{k-1}, z_k)$
  - Evaluate the importance weights up to a normalizing constant according to equation (2.59)

$$\tilde{w}^i_k = w^i_{k-1}\frac{p(z_k|x^i_k)p(x^i_k|x^i_{k-1})}{q(x^i_k|x^i_{k-1}, z_k)} \tag{2.61}$$

- END FOR
- Calculate total weight: $t = SUM\,[\{\tilde{w}^i_k\}^N_{i=1}]$
- For $i = 1 : N$
  - Normalize: $w^i_k = t^{-1}\tilde{w}^i_k$
- END FOR

Degeneracy Problem

In the ideal case, the importance density should be the posterior density itself $p(x_k^i|z_{1:k})$. Drawing random samples using the importance density equation (2.58) results in an increase of the variance of the importance weights over time [24]. This means the samples will be spread over the state space. As a result, after a certain number of recursive steps, all but one particle will have negligible normalized weights. This has a harmful effect on the accuracy and leads to a common problem with the SIS particle filter. This phenomena is called the degeneracy problem.

The degeneracy problem is impossible to avoid in the SIS framework and hence it was a major challenge in the development of sequential Monte Carlo methods. Effectively a large computational effort is devoted to updating particles whose contribution to the approximation of $p(x_k|z_{1:k}$ is almost zero. This is a waste of time and memory. A suitable measure of degeneracy of any SIS process is the effective sample size N_{eff} introduced in [23] and [30], and defined as:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \tag{2.62}$$

where w_k^i is the normalized weight obtained using (2.59) and $1 \leqslant N_{\text{eff}} \leqslant N$.

Hence, small N_{eff} indicate a severe degeneracy and vice versa. Using a very large number of random samples N_s is valid solution for the degeneracy problem. However, this is often impractical; because of the extra computation time and effort needed. Accordingly, two other methods are applied in practices to solve the problem of degeneracy. They are the *Good Choice of Importance Density* and the use of *resampling*. A description of each of the approaches is shown next.

a) **Good Choice of Importance Density**
The choice of importance density $q(x_k|x_{k-1}^i, z_k)$ is one of the most critical issues in the design of particle filter. Obtaining and optimal importance density function helps minimizing the variance of the importance weights and as a result reduces the degeneracy problem. The optimal importance density conditioned upon x_{k-1}^i and z_k has found to be [24]:

$$\begin{aligned} q(x_k|x_{k-1}^i, z_k)_{opt} &= p(x_k|x_{k-1}^i, z_k) \\ &= \frac{p(z_k|x_k, x_{k-1}^i)p(x_k|x_{k-1}^i)}{p(z_k|x_{k-1}^i)}. \end{aligned} \tag{2.63}$$

Substitution of (2.63) into (2.59) yields:

$$w_k^i \propto w_{k-1}^i p(z_k|x_{k-1}^i) \tag{2.64}$$

which states that importance weights at time k can be computed before the particles are propagated to the next time step. Additionally, resampling can be carried out if needed before going to the next step.

Using the optimal importance function requires:

i. Sampling from $p(x_k|x_{k-1}^i, z_k)$

ii. Evaluating up to a normalizing constant the probability:

$$p(z_k|x_{k-1}^i) = \int p(z_k|x_k)p(x_k|x_{k-1}^i)dx_k \tag{2.65}$$

The above requirements are major drawbacks of the "good choice of importance density" degeneracy solution technique. That is because generally either of the two requirements may not be straightforward.

However, in some special cases it is possible to use the optimal importance density. The first case is when x_k is a member of a finite set. In such case, sampling from $p(x_k|x_{k-1}^i, z_k)$ is possible since the integral in equation 2.65 becomes a sum. Tracking maneuvering targets is an example where x_k is a member of a finite set [31]. The second special case where it is possible to use the optimal importance density is where the prior $p(x_k|x_{k-1}^i, z_k)$ is Gaussian [24], [32].

Generally, the most popular and convenient choice of importance density is the transitional prior,

$$q(x_k|x_{k-1}^i, z_k) = p(x_k|x_{k-1}^i) \tag{2.66}$$

Substitution of equation (2.66) into equation (2.59) then yields

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{2.67}$$

The above choice is widely spread since it is intuitive and easy to implemented. However, there are many other densities that can be used.

Using the transitional prior as the importance density instead of the optimal one has the following disadvantages:

a) If the optimal importance density is used, then according to equation (2.64) importance weights can be computed before the particles are propagated to time k. Equation (2.67) states that this is not possible with the transitional prior.

b) Using the much broader distribution of the transitional prior $p(x_k|x_{k-1}^i)$ compared to the likelihood $p(z_k|x_k)$ results in having the degeneracy problem again. This is because after some time only few particles will be having high weights.

Methods exist for encouraging the particles to be in the right place. Of course, regions of high likelihood are the right places where the particles should congregate. The auxiliary particle filter [10] is one of these methods. Additionally, bridging densities and progressive correction provide alternative solutions. Details on both algorithms can be found in [7].

b) **Resampling**

The second method by which the effects of degeneracy can be reduced is to use resampling. Resampling is done in the SIS algorithm whenever a significant degeneracy is observed. In other words, whenever N_{eff} falls below some

threshold N_T). Resampling eliminates particles with low importance weights and multiplies samples with high importance weights. Furthermore, resampling allows to apply a particle filter in situations in which the true distribution differs from the proposal.

Resampling involves a mapping of a random measure $\{x_k^i, w_k^i\}$ into a random measure $\{x_k^{i*}, 1/N_s\}$ with uniform weights. Accordingly, the resampled particles are having uniform weights of $w_k^i = 1/N_s$. The mapped random samples x_k^{i*} are generated by resampling(with replacement) N_s times from an appropriate discrete representation of $p(x_k|z_{1:k})$ given by:

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{2.68}$$

so that $P\{x_k^{i*} = x_k^j\} = w_k^j$. Resampling can be implemented by generating N_s independent identically-distributed (i.i.d) random variables from the uniform distribution, sorting them in ascending order and finally picking the corresponding mapped samples [33], [27]. Sorting complexity increase in a logarithmic order of the number of samples N_s. It represents a major limitation in the SIS particle filter.

Systematic resampling [34] is an efficient scheme, simple to implement and above all it has a low computational complexity. Additionally, it reduces the Monte Carlo variations. The pseudocode of the systematic resampling is described in Algorithm 2, where $\mathbb{U}[a, b]$ is the uniform distribution on the interval $[a, b]$. With systematic resampling, the index of the parent i^j for each resampled particle x_k^{j*} is stored if needed.

By now, all the steps needed for a generic particle filter are explained. Its pseudocode is given in algorithm 3.

ALGORITHM 2: RESAMPLING ALGORITHM

$[\{x_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s} = \text{RESAMPLE}\,[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$

- Initialize the CDF: $c_1 = w_k^1$
- FOR $i = 2 : N_s$
  - Construct CDF: $c_i = c_{i-1} + w_k^i$
- END FOR
- Start at the bottom of the CDF: $i = 1$
- Draw a starting point: $u_1 \sim \mathbb{U}[0, N_s^{-1}]$
- FOR $j = 1 : N_s$
  - Move along the CDF: $u_j = u_1 + N_s^{-1}(j - 1)$
  - WHILE $u_j > c_i$
    - $i = i + 1$
  - END WHILE
  - Assign sample: $x_k^{j*} = x_k^i$

  - Assign weight: $w_k^j = N_s^{-1}$
  - Assign parent: $i^j = i$
- END FOR

ALGORITHM 3: GENERIC PARTICLE FILTER

$[\{x_k^i, w_k^i\}_{i=1}^{N_s} = \text{PF}\ [\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- Filtering via SIS according to algorithm 1: $[\{x_k^i, w_k^i\}_{i=1}^{N_s} = \text{SIS}\ [\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$
- Calculate $\widehat{N_{\text{eff}}}$ using equation (2.62)
- IF $\widehat{N_{\text{eff}}} < N_T$
  - Resample using algorithm 2: $[\{x_k^i, w_k^i, -\}_{i=1}^{N_s} = \text{RESAMPLE}[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$
- END IF

Although resampling reduces the effects of the degeneracy problem, it introduces other practical problems:

i. The opportunity of building the implementation using parallel processors is limited since all particles should be combined now.

ii. Resampling results in statistically selecting the particles that have high weights w_k^i many times. This results in a loss of diversity among the particles as the resultant samples will contain many repeated points. This is a known problem in particle filtering and is called *sample impoverishment*. As a result of sample impoverishment, all the particles will collapse to a single point within a few iterations.

iii. Any smooth estimates based on the particles paths degenerate as a result of the particles diversity reduction.

Over the last few years, a lot of research has been done to find solutions for the pre-mentioned disadvantages. The forward-backward filter [35] and the Markov chain Monte Carlo (MCMC) move step [36] are two suggested solution for the degeneration of smooth estimates problem. On the other hand, the resample-move algorithm [37] and the regularization step [7] are two suggested solutions for the sample impoverishment.

As a general rule, the accuracy of any estimate can only decrease as a result of resampling. Therefore if statistical quantities such as mean and variances of the particles are to be reported, these should be calculated prior to resampling.

II. **Other Versions of Particle Filters**

Over the last few years, many versions of particle filters have been developed. However, the sequential importance sampling (SIS) algorithm presented in the previous section forms the basis for most of them. The various versions of particle filters proposed in the literature can be considered as special cases of this general SIS algorithm. They differ in their choice of the importance sampling density and the resampling step. The most famous versions of particle filters are [10]:

(a) Sampling importance resampling (SIR) filter;

(b) Auxiliary sampling importance resampling (ASIR) filter;

(c) Regularized particle filter (RPF).

(d) Local linerization particle filters.

(e) Multiple-model particle filter.

In practise these particle filters can be combined according to the application and to avoid the disadvantages linked to some particle filters.

During this work, SIS particle filter, SIR particle filter and SIR particle filter with N_{eff} are implemented and tested. Their accuracy and performance are also analyzed. The SIS particle filter is explained in the previous section. The SIR particle filter will be explained next.

Sampling Importance Resampling Filter

Sampling importance resampling (SIR) is a very commonly used particle filtering algorithm, which approximates the filtering distribution by a weighted set of particles. It was first proposed in [21] under the name "bootstrap" filter. It is a MC method that can be applied to recursive Bayesian filtering problems.

Very weak assumptions are required to use the SIR filter. These assumptions are:

(a) The state dynamics and measurement functions $f_k(.,.)$ and $h_k(.,.)$ in (2.3) and (2.4), respectively, need to be known.

(b) Sampling from the process noise distribution of v_{k-1} and from the prior $(p(x_k|x_{k-1}^i))$ should be possible.

(c) The likelihood function $p(z_k|x_k)$ needs to be available. Its availability should be at least up to proportionality.

If the above weak assumptions hold, the SIR algorithm is derived from the SIS algorithm by choosing the importance density to be the transitional prior $(p(x_k|x_{k-1}^i))$ and by performing resampling step at every time index k.

According to the above selection of importance density, samples from the transitional prior $(p(x_k|x_{k-1}^i))$ are required. A sample $x_k^i \sim p(x_k|x_{k-1}^i)$ can be generated in two steps. First, a process noise sample $v_{k-1}^i \sim p_v(v_{k-1})$ is generated, where $p_v(.)$ is the PDF of v_{k-1}. Second, setting the support points $x_k^i = f_k(x_{k-1}^i, v_{k-1}^i)$. According to the selected importance density, the importance weights can be calculated using equation (2.67). Using the fact that resampling is applied every time index, importance weights at $k-1$ can be simplified to $w_{k-1}^i = 1/N$ for all $i = 1, \ldots, N$. This has two meanings: first, there is not need to pass on the importance weights from one time step to the next; and second, relation (2.67) simplifies to:

$$w_k^i \propto p(z_k|x_k^i). \tag{2.69}$$

The above simplifications make the computation efforts of the SIR particle filter much simpler compared to other particle filters. It has to be noticed that the weights given by the proportionality in (2.69) have to be normalized before the resampling stage. An iteration of the algorithm is then described in Algorithm 4.

ALGORITHM 4: SIR PARTICLE FILTER

$[\{x_k^i\}_{i=1}^{N_s}] = \text{SIR}\,[\{x_{k-1}^i\}_{i=1}^{N_s}, z_k]$

- FOR $i = 1 : N_s$
  - Draw $x_k^i \sim p(x_k|x_{k-1}^i)$
  - Calculate $\tilde{w}_k^i = p(z_k|x_k^i)$
- END FOR
- Calculate total weight: $t = \text{SUM}[\{\tilde{w}_k^i\}_{i=1}^{N_s}]$
- FOR $i = 1 : N_s$
  - Normalize: $w_k^i = t^{-1}\tilde{w}_k^i$
- END FOR
- Resample using algorithm 2: $[\{x_k^i, -, -\}_{i=1}^{N_s}] = \text{RESAMPLE}\,[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$

The SIR particle filter has two main disadvantages:

a) In the SIR particle filter, the importance density is selected to be the transitional prior $p(x_k|x_{k-1}^i)$ and it is independent of the measurements z_k. As a result, the filter will explore the state space without making use of the observations. Accordingly, this filter can be inefficient and is sensitive to outliers.

b) The filter may suffer from loss of diversity among the particles and the resultant samples will contain many repeated points. This is due to applying resampling at every time step.

On the other hand, the main advantages of the SIR particle filter are as follows:

- It is easy to sample from the importance density since it is selected to be the transitional prior.
- It is also easy to evaluate the importance weights since they are equal to the likelihood.

2.5.4 Example 3 - Position estimation using Particle Filter

The tracking of a walking person in one direction has been demonstrated in example 2 using Bayesian Filters. In this section, the same process will be illustrated but using Particle Filtering. The same movement and measurement models applies. A graphical illustration is shown in figure (2.8)

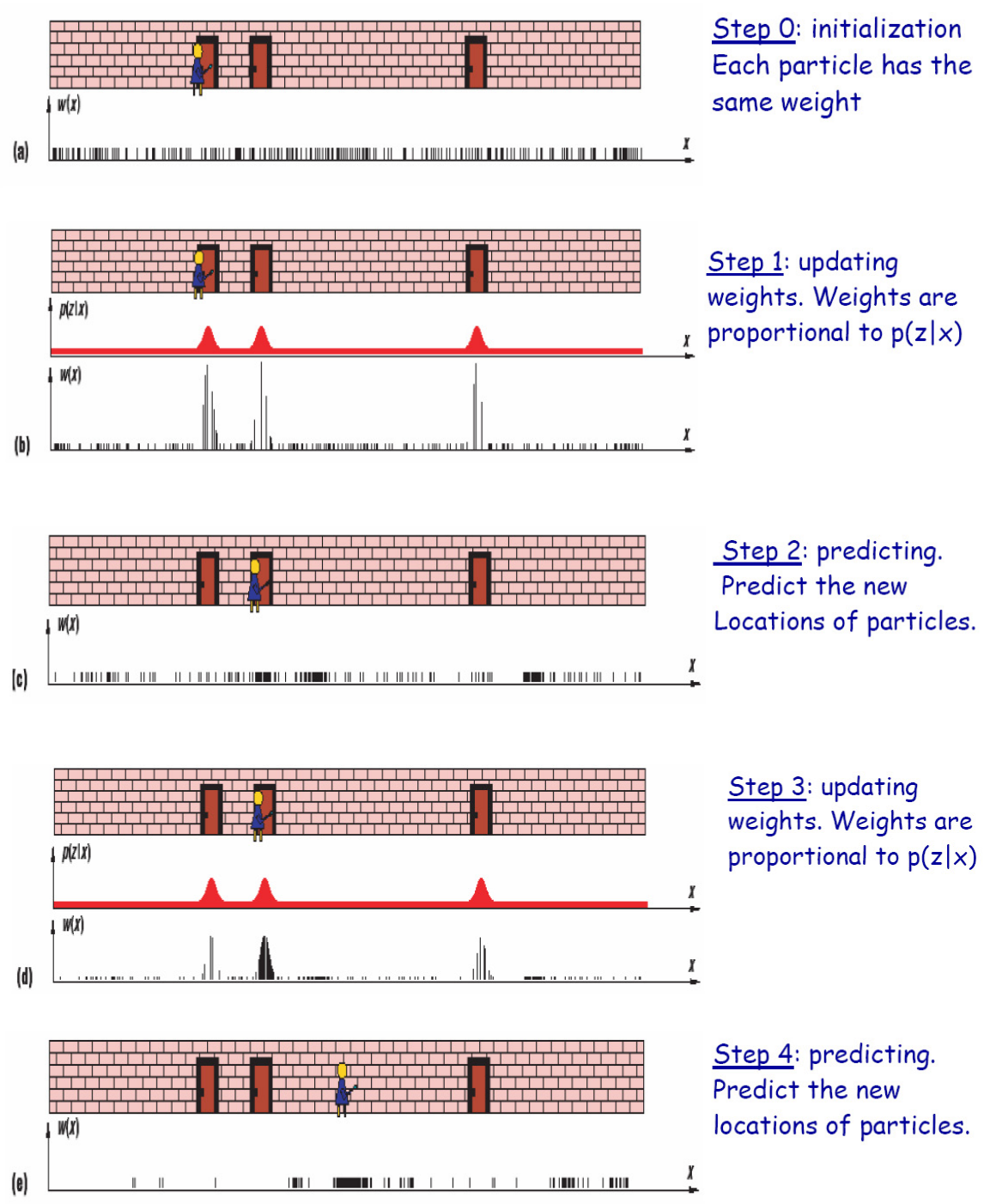

Figure 2.8: Position estimation using Particle Filter, originally found in [13]

2.5.5 Comparison between Bayesian Filter and Particle Filter

Figure (2.9) illustrates a comparison between location estimation using Bayesian Filters (example 2) and the approximation done using Particle Filter (example 3). From the illustrated comparison between Bayesian Filter and Particle Filter in figure (2.9) the following could be noticed:

- Bayesian filtering uses continuous PDFs to represent soft decisions and decisions are made upon states where high PDF densities occur.
- Particle filtering uses discretely sampled PDFs and decisions are made upon states where more particles and more weights occur.
- If the number of samples 'particles' tends to infinity then a Particle filtering will approach Bayesian filtering.
- Particle filtering provides an efficient approximation for Bayesian filtering.
- Particle filtering is easy to formulate and implement.

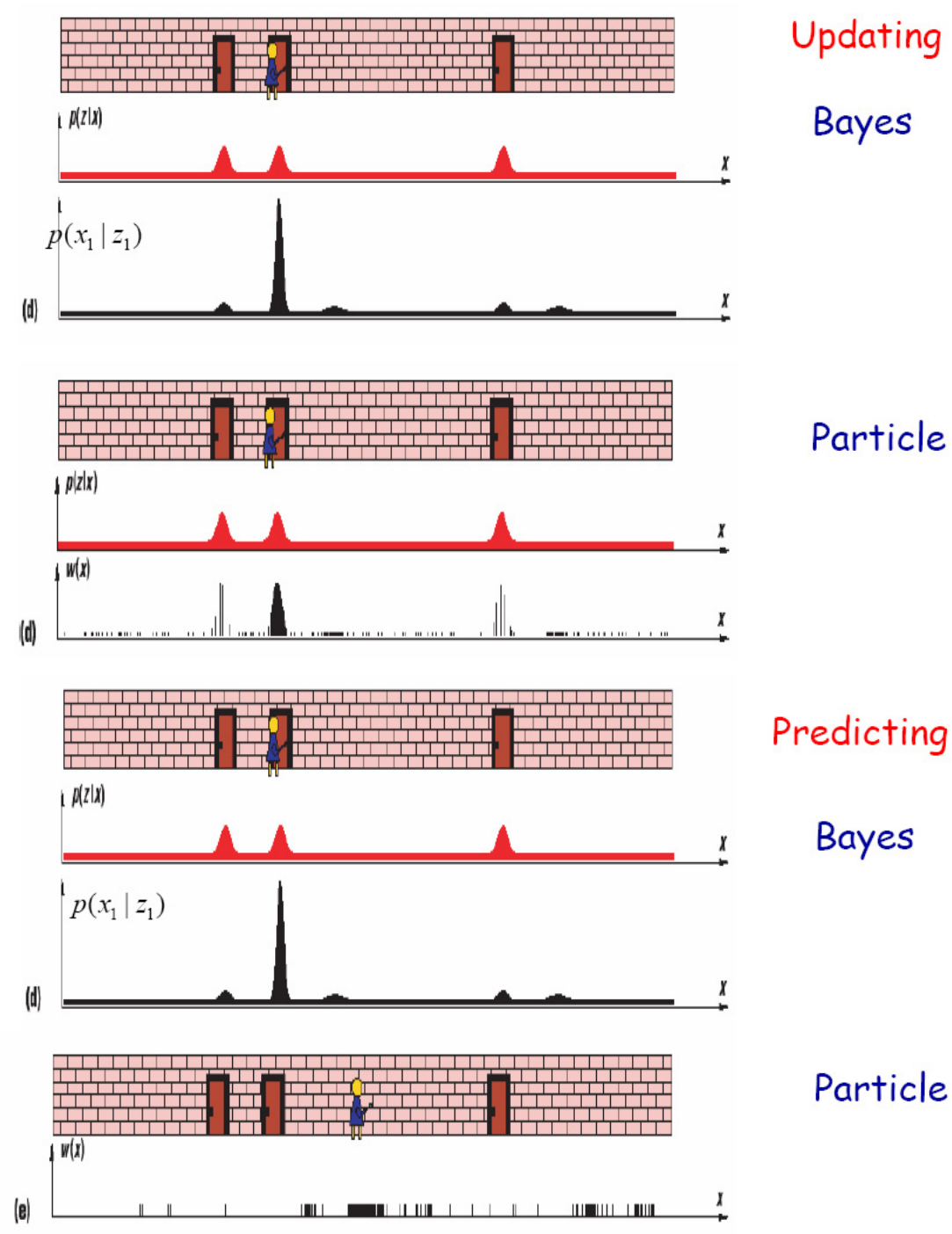

Figure 2.9: Comparison between Bayesian Filter and Particle Filter, originally found in [13]

Chapter 3

Human movement model

3.1 Introduction

Having a movement model that reproduces the behavior of the real dynamic system is a prerequisite in Bayesian algorithms. The Prediction stage depends entirely on the movement model to predict where most probably the pedestrian will be at the next time step. With a good movement model, the system can predict accurately the position of the pedestrian at time $k + 1$. This will decrease the dependency on the measurements for estimation. It will also allow the system to take correct decisions regarding the position of the pedestrian at the prediction stage, since we will have less contradiction between the prediction and the measurements later on. Developing a good movement model was one of the main tasks during this work. A detailed explanation of the developed movement model and its elements will be given in this chapter.

Several parameters affect the movement of the human being. Speed and direction and as a result the position vary according to these parameters. Some of these parameters affect the movement more than the others. Activity, activeness, disorientation and emotions are examples of these parameters.

Human movement is parameterized by physical parameters like speed, direction and position. Movement constraints that control these physical parameters are categorized into two groups. The first category includes parameters that the system can determine accurately such as age, weather, time of day and weekday and parameters that can be derived from external data such as ground steepness or obstacles at the pedestrian's position. The other category includes parameters that are varying according to the human behavior. The system is not capable of answering questions regarding this category. Examples of the members of this group are activity, disorientation, activeness, arousal and emotions.

Eleven parameters that affect the human movement are considered during this work. Figure (3.1) shows an illustration of these parameters. Previous speed and direction are used to calculate the new position which is needed in order to specify the condition of some of the states. For example this new position is needed to specify the geographical situation of the area where the pedestrian is navigating.

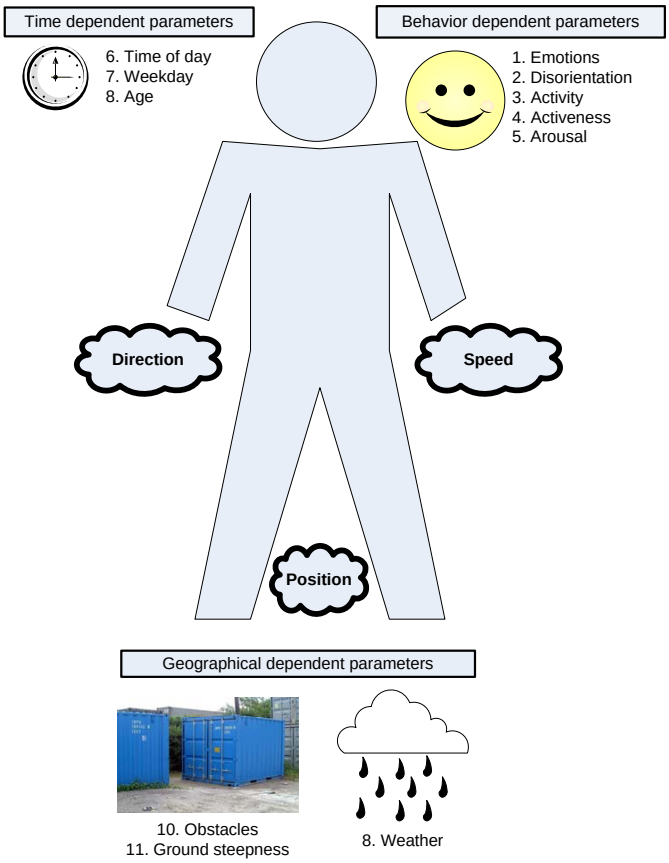

Figure 3.1: States that affect the human movement

The system is capable of answering question like telling the weather condition, the day of the week, the time of the day and the age of the pedestrian. The system is also capable of telling the obstacles and the steepness conditions at the location of the pedestrian.

The system is not capable of measuring the emotions, the arousal, the activeness, the activity and the disorientation states of the pedestrian. These variables are modelled via Markov models. Transition probabilities are set according to some statistics. For example, if at $t = 0$ the pedestrian is half drunk then at time $t+1$ his disorientation state will change according to some transition probability following a Markov model.

By knowing the specific condition of our eleven states, the situation of the pedestrian could be specified accurately. From this situation we can speculate how his movement will be at the next time step using some biological statistical data.

We used some biological statistics to parameterize our model assuming Gaussian distributions for its random variables. For example, for a totally drunk walking pedestrian we assume a mean speed of about 43 m/minute, standard deviation of 22m/sec, mean direction which is same as the old direction, standard deviation of direction of 90^0. Eleven constraints will result in eleven mean and variance for speed and direction. Single mean and variance are calculated by performing a weighted average over the eleven states. The two means and variances are used to build a two dimensional Gaussian distribution. New speed and new direction are sampled out of this two di-

mensional Gaussian distribution. The new position is calculated out of the speed and the direction using the equations of motion. This new position will be the input for the next time step.

As time evolves changes in the states of the first category items are received from the system, while changes in the states of the second category items are calculated using their Markov models. Speed, direction and the resulting position are then calculated at every time step.

The prediction stage became more accurate as a result of this improved movement model. Accordingly, the system is capable of approximating the position of pedestrian more accurately before the measurement comes. This results in less contradiction between the prediction and the measurement. The developed movement model has shown promising results in location estimation. Location based services may improve considerably with accurate movement models. A flow diagram of how the developed movement model works is shown in figure (3.2).

3.2 Markovian States

As can been seen from figure (3.2), activity, disorientation, emotions, arousal and activeness are modelled using Markov models. The idea of using Markov chains for describing human behaviors could be found also in [38], [39] and [40]. Transition probabilities of the Markov models are set according to some statistics [41] and common sense. The measurements arrive every second, so transition probabilities are set on a one second time interval base. The input for each of the Markov models is the previous state and the output will be the new state. For example, if the pedestrian's activeness old state was "active", then his new activeness state might remain "active" with a probability of 90%, might change into "just ok" with a probability of 5% and might change into "tired" with a probability of 5%. The state and transition probabilities of the Markov models of the five Markov processes are demonstrated in tabular forms in tables (3.1) to (3.5). Vertical table elements are old states, while horizontal table elements are new states.

3.3 System states

From figure (3.2) we can see that there are two types of system states. The first type includes states that the system is able to reply with their condition without the need of knowing their previous states. Examples of those states are weather, weekday, time of day and age. Currently those are configured using an XML file.

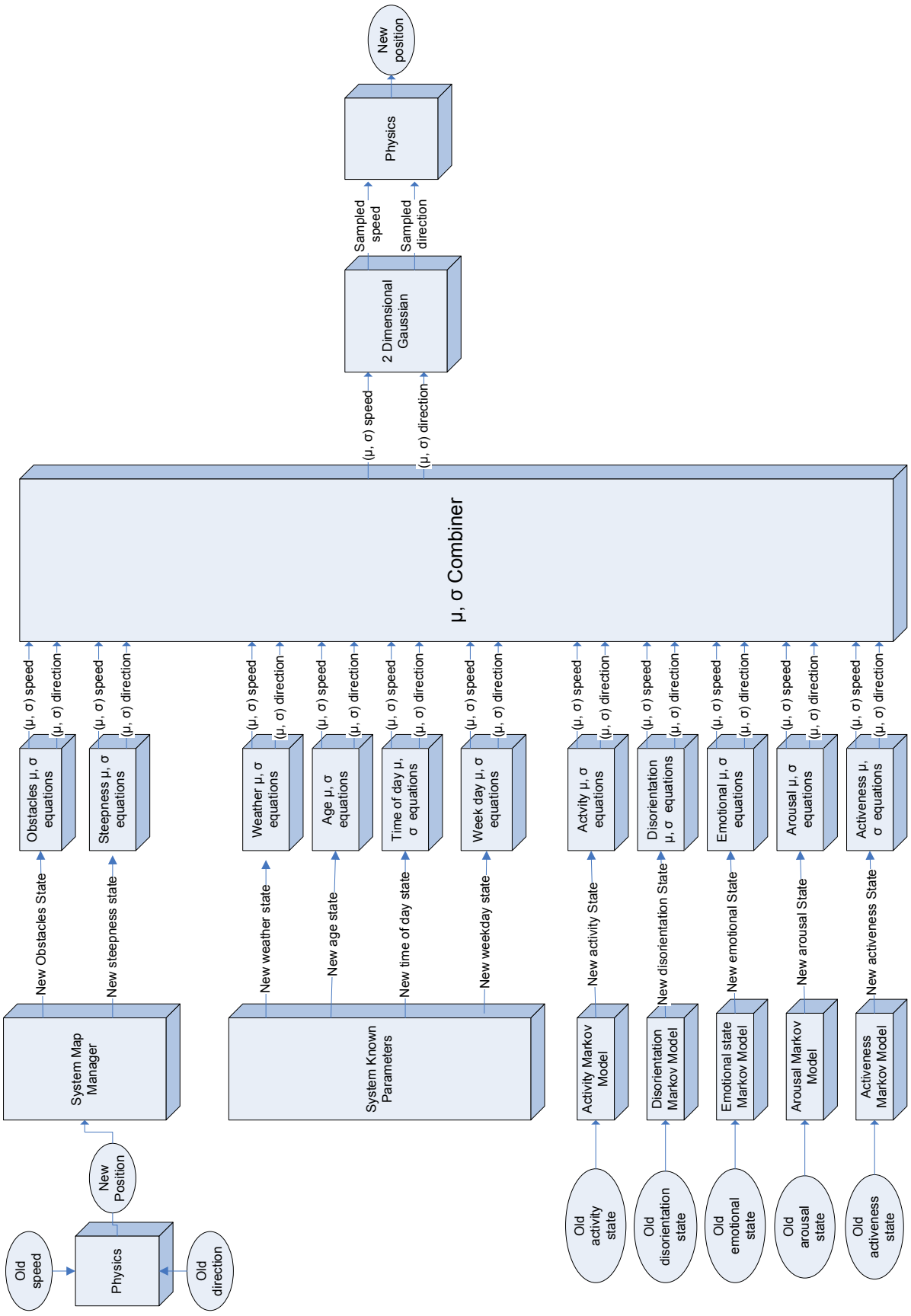

Figure 3.2: Developed movement model

Activity	Escaping	Sporting (running)	Harry walk	City walk	Hiking	Shopping	Stopping
Escaping	0.9	0.005	0.03	0.02	0.02	0.02	0.005
Sporting running	0.01	0.9	0.03	0.02	0.03	0.005	0.005
Hurry walk	0.01	0.025	0.9	0.025	0.02	0.01	0.01
City walk	0.01	0.015	0.02	0.9	0.015	0.02	0.02
Hiking	0.01	0.02	0.025	0.02	0.9	0	0.025
Shopping	0.01	0.01	0.02	0.03	0	0.9	0.03
Stopping	0.01	0.01	0.015	0.025	0.015	0.025	0.9

Table 3.1: Activity Markov model

Disorientation	Sober	Quarter drunk	Half drunk	3Q drunk	Totally drunk
Sober	0.95	0.03	0.02	0	0
Quarter drunk	0.1	0.85	0.04	0.01	0
Half drunk	0.04	0.11	0.82	0.02	0.01
3Q drunk	0	0	0.25	0.7	0.05
Totally drunk	0	0	0	0.3	0.7

Table 3.2: Disorientation Markov model

Emotions	Sad	Normal	Happy
Sad	0.9	0.07	0.03
Normal	0.05	0.9	0.05
Happy	0.03	0.07	0.9

Table 3.3: Emotions Markov model

For the second type of states, the system needs to know the new position to give any information regarding their conditions. Old speed and old direction are used to calculate the new position of the pedestrian. Ground steepness and obstacles are the members of this type. To model these two states, a map of the simulated/demonstrated area should be used. The map is colored using a gray scale in which each gray level represents some steepness condition. Another gray scale map could be generated in which each gray level represents some obstacles condition. An 8 level gray scale map is used in the current implementation. The same map is used for steepness and obstacles. A Java class that takes the new position as input and produce a steepness and obstacles conditions using the gray scale map is implemented. A satellite image for DLR is shown in figure (3.3). A gray scale map for DLR premises is shown in figure (3.4).

3.4 From states to movement

A specific condition for each of the states is determined as explained in the two previous sections. Each specific condition is used to calculate a mean and standard deviation for speed and direction as shown in figure (3.2). For example an escaping pedestrian

Figure 3.3: Satellite image of DLR premises

Figure 3.4: Gray scale map for DLR premises using 8 colors depth

Madness	Calm	Neutral	Angry
Calm	0.9	0.07	0.03
Neutral	0.05	0.9	0.05
Angry	0.03	0.07	0.9

Table 3.4: Arousal Markov model

Activeness	Tired	Just Ok	Active
Tired	0.9	0.08	0.02
Just Ok	0.05	0.9	0.05
Active	0.02	0.08	0.9

Table 3.5: Activeness Markov model

will have a mean speed of 250 m/min, standard deviation of speed of 5 m/min, mean direction which is the same as the old direction and standard deviation of direction of 15°.
To get such values statistical biological information [41], [42] is used. Some of these statistics are shown in table (3.6).

Speed (m/minute)	Status
40	average slow walk
82	average normal walk
110	average fast walk
above 130	start running speed
200	average running speed
300	maximum runnig speed (normal pedestrian)
618	maximum running speed (athlete)

Table 3.6: Some statistics used to build the movement model

The maximum running speed ever was achieved by Michael Johnson. It takes an athlete like Michael Johnson or Maurice Greene 5 to 6 seconds to reach their maximum speed [43]. However, the designed movement model was meant for a normal pedestrian, so average normal human limits were used.

Equations that relate mean and standard deviation of speed and direction to the human states were generated under consideration of these biological statistics. This resulted in four equations for each of the eleven states. For simplicity mean direction is always assumed to be the same as the old direction. This reduce the number of equations needed for each of the states to three.

To generate such equations a table was created for each of the states. This table contains various conditions of the state on one side, and the resultant mean and the standard deviation of speed and direction on the other side. As mentioned previously, biological statistical data and some common sense were used to create these tables. For simplicity, the correlation between speed and direction was assumed to be zero while creating these tables. Curves for mean speed, standard deviation of speed and

direction were created out of the tables. No curve was needed for mean direction since it was assumed to be the same as the old direction. Curve fitting was used to create three equations for each of the states out of the curves. The generated table for age state is shown in table (3.7). The rest of the tables for the other ten states are shown in appendix (A).

Age (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Kid	45	old direction	15	45
Youth	82	old direction	13	15
Old	60	old direction	11	30

Age (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Kid	120	old direction	25	20
Youth	200	old direction	34	15
Old	160	old direction	20	17

Table 3.7: Age statistics

Using the resultant tables, three charts for each of the states are generated. These charts relate mean speed, sigma speed and sigma direction as functions of the specific condition of each of the states. Curve fitting is used to generate an equation out of each of those charts. The generated charts for age state are visualized in figure (3.5) up to figure (3.10). The rest of the charts for the other ten states are visualized in appendix (B). The curve fitted equation is also visualized in each chart.

The conditions of each of the states are represented by numbers in the x-axis of the charts. For example kid age state is represented by zero in the age charts.

Curve fitting is applied to generate a mathematical equation for each of the charts. This was done using Microsoft Excel. Trendlines are used for this curve fitting purpose. Trendlines are used to graphically display trends in data and to analyze problems of prediction. Such analysis is also called regression analysis. Regression analysis are forms of statistical analysis used for forecasting. Regression analysis estimates the relationship between variables so that a given variable can be predicted from one or more other variables. By using regression analysis, trendlines can be extended in a chart beyond the actual data to predict future values. Trendline is most reliable when its R-squared value is at or near 1. R-squared value is a number from 0 to 1 that reveals how closely the estimated values for the trendline correspond to the actual data. R-squared is also known as the coefficient of determination. When a trendline is fitted to the data, Excel automatically calculates its R-squared value.

The generated equation and the R-squared values are displayed on each of the chart. A summary of the resultant equations that relate age state to the means and standard deviations (σ) of speed and direction are shown in equation (3.1) up to equation (3.8). The rest of the equations that relate the other ten states to the means and standard deviations of speed and direction are shown in appendix (C).

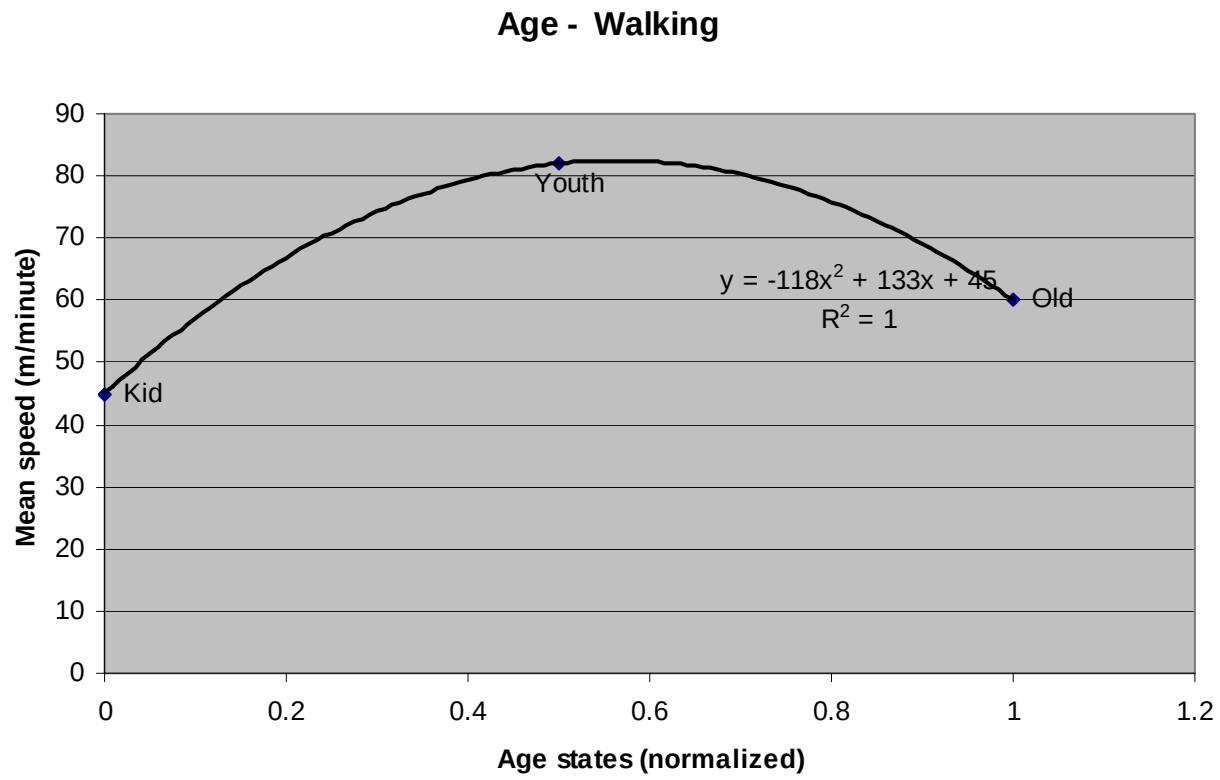

Figure 3.5: Mean speed as a function of the age during walking

Figure 3.6: Sigma speed as a function of the age during walking

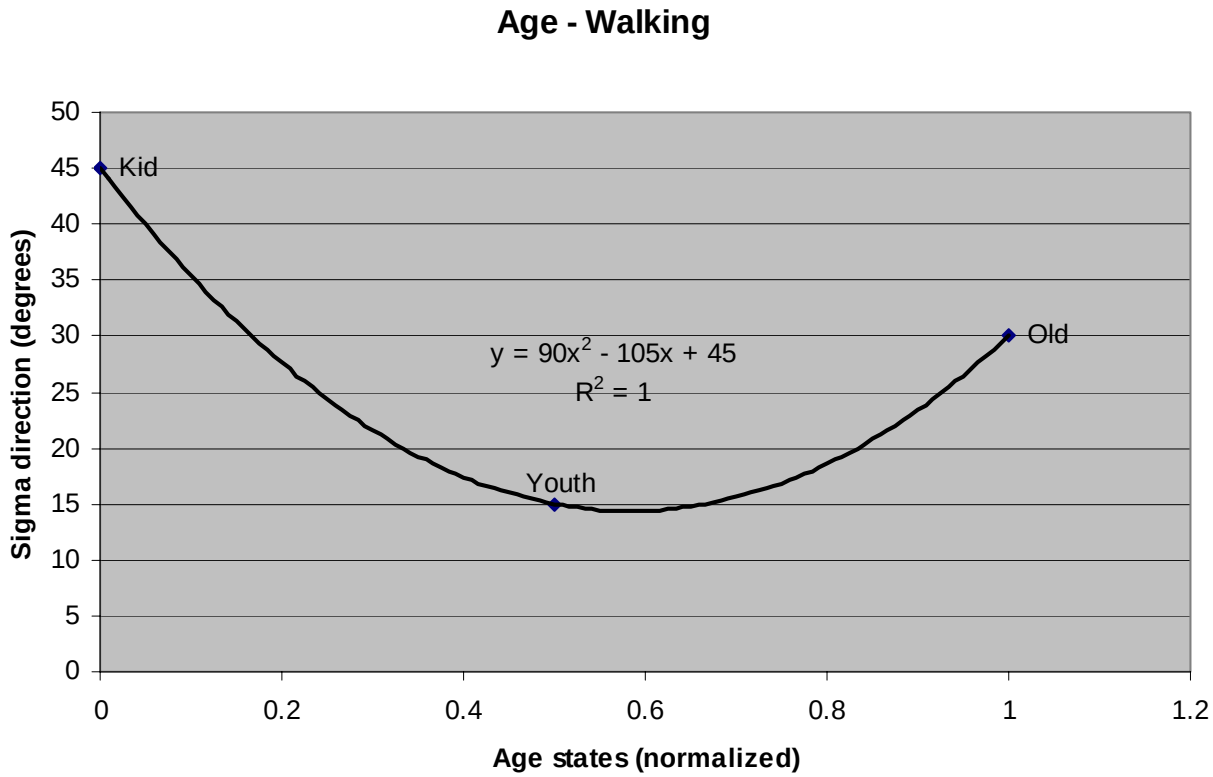

Figure 3.7: Sigma direction as a function of the age during walking

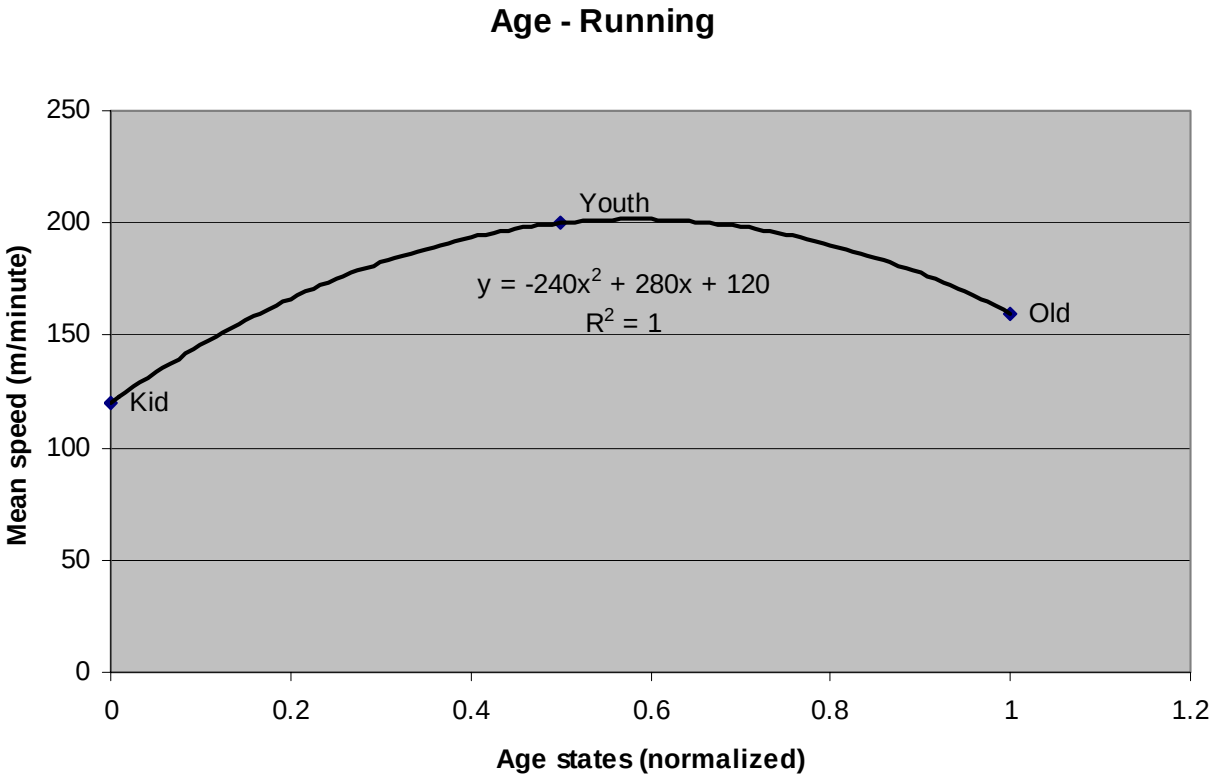

Figure 3.8: Mean speed as a function of the age during running

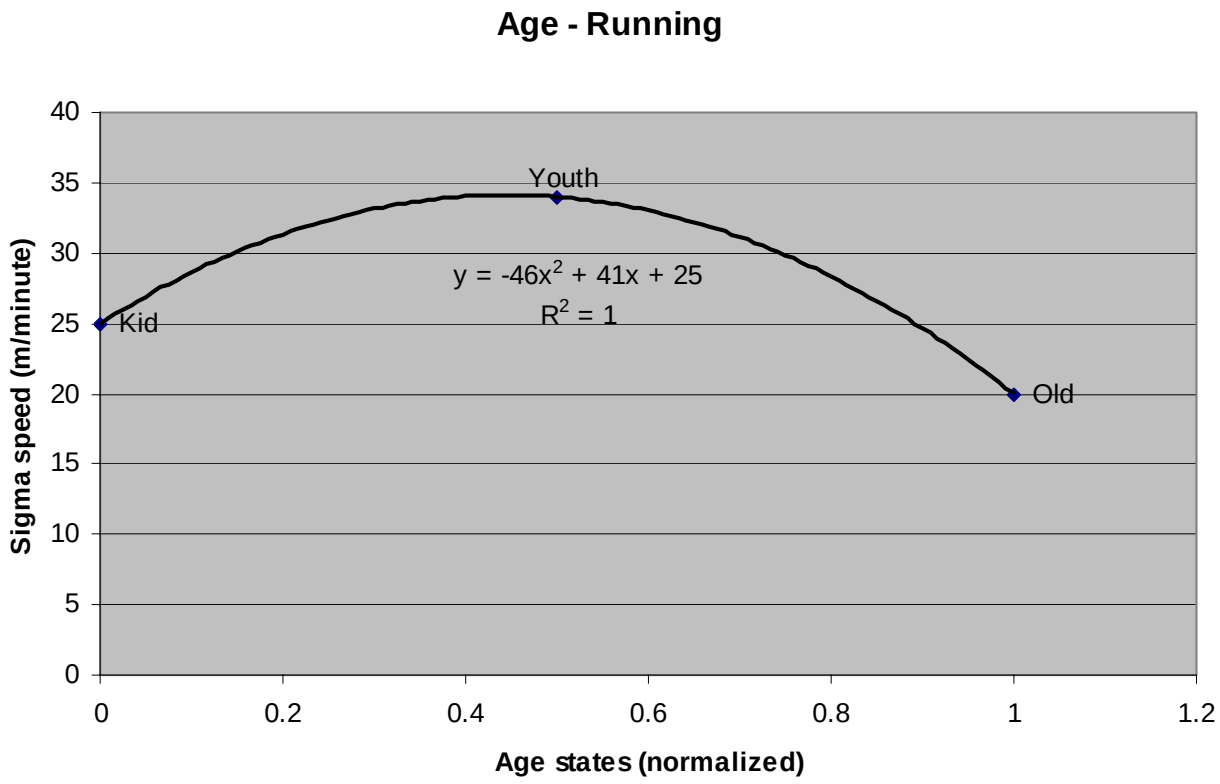

Figure 3.9: Sigma speed as a function of the age during running

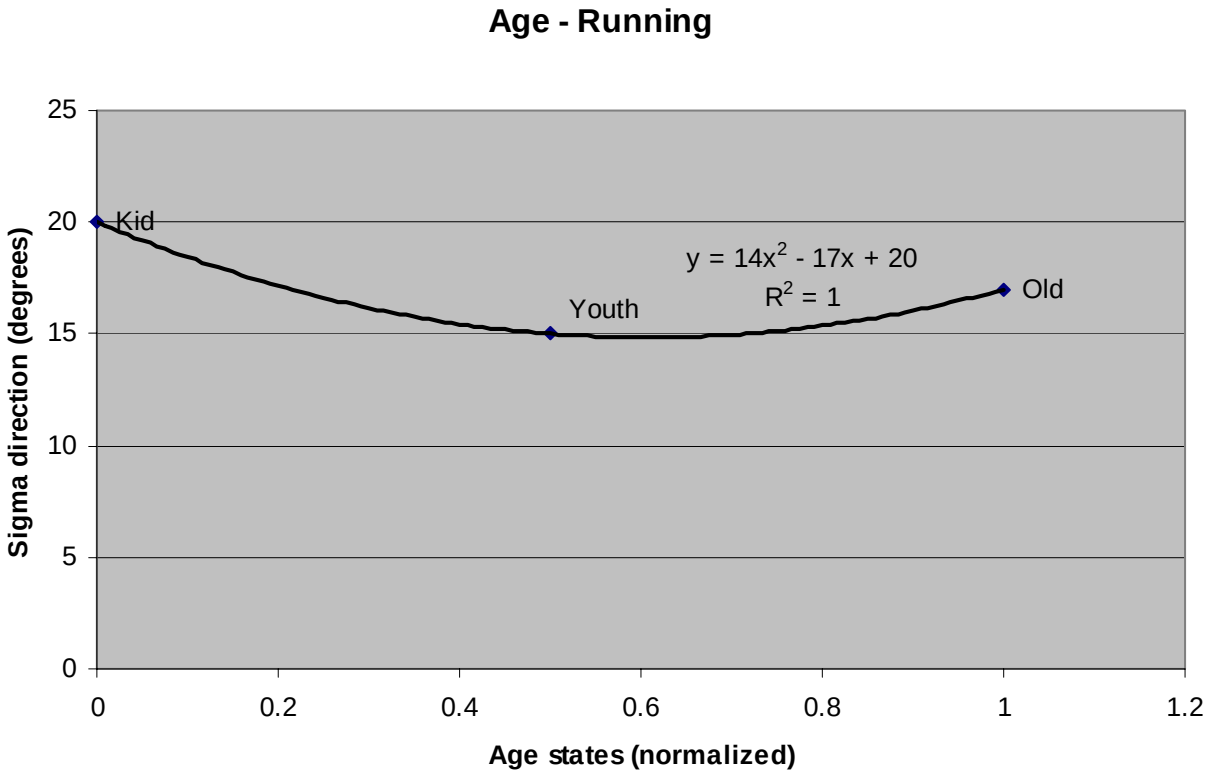

Figure 3.10: Sigma direction as a function of the age during running

Age state

- Walking

$$\mu_v = -118(\text{age state})^2 + 133(\text{age state}) + 45 \tag{3.1}$$
$$\sigma_v = -4(\text{age state}) + 15 \tag{3.2}$$
$$\mu_\alpha = \alpha(k-1) \tag{3.3}$$
$$\sigma_\alpha = 90(\text{age state})^2 - 105(\text{age state}) + 45 \tag{3.4}$$

- Running

$$\mu_v = -240(\text{age state})^2 + 280(\text{age state}) + 120 \tag{3.5}$$
$$\sigma_v = -46(\text{age state})^2 + 41(\text{age state}) + 25 \tag{3.6}$$
$$\mu_\alpha = \alpha(k-1) \tag{3.7}$$
$$\sigma_\alpha = 14(\text{age state})^2 - 17(\text{age state}) + 20 \tag{3.8}$$

where,
μ_v: is the mean speed,
σ_v: is the standard deviation of speed,
μ_α: is the mean direction,
σ_α: is the standard deviation of direction.

By now eleven means and standard deviations for speed and direction can be calculated. Next step as can be seen from figure (3.2) is to combine these means and standard deviations into a single mean and a single standard deviation for both speed and direction. This is done using the Mean - Standard Deviation combiner. An explanation of how this combiner works is shown below.

3.5 μ - σ Combiner

Specific weights are assigned for each of the states according to its importance for the human movements. The assigned weights are shown in table (3.8).

Using the weights specified in table (3.8) weighted sums for the change in μ_v , μ_α, σ_v and σ_α for each of the states are calculated. Equations used to calculate these weighted sums of changes are shown from equation (3.9) up to equation (3.12).

State	Weight
Activity	11
Disorientation	12
Age	6
Time of day	4
Weather	5
Activeness	6
Obstacles	8
Steepness	8
Weekdays	4
Emotions	2
Arousal	3

Table 3.8: State weights according to their importance for human movement

$$\begin{aligned}\sum_{i=1}^{11} \triangle\mu_{v_i} \times \gamma_i = \quad & \gamma_{activity} \times \triangle\mu_{v,activity} \\ & + \gamma_{disorientation} \times \triangle\mu_{v,disorientation} \\ & + \gamma_{age} \times \triangle\mu_{v,age} \\ & + \gamma_{time\ of\ day} \times \triangle\mu_{v,timeofday} \\ & + \gamma_{weather} \times \triangle\mu_{v,weather} \\ & + \gamma_{activeness} \times \triangle\mu_{v,activeness} \\ & + \gamma_{obstacles} \times \triangle\mu_{v,obstacles} \\ & + \gamma_{steepness} \times \triangle\mu_{v,steepness} \\ & + \gamma_{weekdays} \times \triangle\mu_{v,weekdays} \\ & + \gamma_{emotions} \times \triangle\mu_{v,emotions} \\ & + \gamma_{arousal} \times \triangle\mu_{v,arousal} \end{aligned} \tag{3.9}$$

where, γ is the weight of the state and $\triangle$ is how much change has happened to the quantity of interest since last time step.

$$\begin{aligned}\sum_{i=1}^{11} \triangle\mu_{\alpha,i} \times \gamma_i = \quad & \gamma_{activity} \times \triangle\mu_{\alpha,activity} \\ & + \gamma_{disorientation} \times \triangle\mu_{\alpha,disorientation} \\ & + \gamma_{age} \times \triangle\mu_{\alpha,age} \\ & + \gamma_{time\,of\,day} \times \triangle\mu_{\alpha,timeofday} \\ & + \gamma_{weather} \times \triangle\mu_{\alpha,weather} \\ & + \gamma_{activeness} \times \triangle\mu_{\alpha,activeness} \\ & + \gamma_{obstacles} \times \triangle\mu_{\alpha,obstacles} \\ & + \gamma_{steepness} \times \triangle\mu_{\alpha,steepness} \\ & + \gamma_{weekdays} \times \triangle\mu_{\alpha,weekdays} \\ & + \gamma_{emotions} \times \triangle\mu_{\alpha,emotions} \\ & + \gamma_{arousal} \times \triangle\mu_{\alpha,arousal} \end{aligned} \tag{3.10}$$

$$\begin{aligned}\sum_{i=1}^{11} \triangle\sigma_{v,i} \times \gamma_i = \quad & \gamma_{activity} \times \triangle\sigma_{v,activity} \\ & + \gamma_{disorientation} \times \triangle\sigma_{v,disorientation} \\ & + \gamma_{age} \times \triangle\sigma_{v,age} \\ & + \gamma_{time\,of\,day} \times \triangle\sigma_{v,time\,of\,day} \\ & + \gamma_{weather} \times \triangle\sigma_{v,weathe} \\ & + \gamma_{activeness} \times \triangle\sigma_{v,activeness} \\ & + \gamma_{obstacles} \times \triangle\sigma_{v,obstacles} \\ & + \gamma_{steepness} \times \triangle\sigma_{v,steepness} \\ & + \gamma_{weekdays} \times \triangle\sigma_{v,weekdays} \\ & + \gamma_{emotions} \times \triangle\sigma_{v,emotions} \\ & + \gamma_{arousal} \times \triangle\sigma_{v,arousal} \end{aligned} \tag{3.11}$$

$$\begin{aligned}\sum_{i=1}^{11} \triangle\sigma_{\alpha,i} \times \gamma_i = \quad & \gamma_{activity} \times \triangle\sigma_{\alpha,activity} \\ & + \gamma_{disorientation} \times \triangle\sigma_{\alpha,disorientation} \\ & + \gamma_{age} \times \triangle\sigma_{\alpha,age} \\ & + \gamma_{time\ of\ day} \times \triangle\sigma_{alpha,time\ of\ day} \\ & + \gamma_{weather} \times \triangle\sigma_{\alpha,weather} \\ & + \gamma_{activeness} \times \triangle\sigma_{\alpha,activeness} \\ & + \gamma_{obstacles} \times \triangle\sigma_{direction,obstacles} \\ & + \gamma_{steepness} \times \triangle\sigma_{\alpha,steepness} \\ & + \gamma_{weekdays} \times \triangle\sigma_{\alpha,weekdays} \\ & + \gamma_{emotions} \times \triangle\sigma_{\alpha,emotions} \\ & + \gamma_{arousal} \times \triangle\sigma_{\alpha,arousal} \end{aligned} \tag{3.12}$$

where,

$$\triangle\ \mu_v = \mu_v(k) - \mu_v(k-1) \tag{3.13}$$
$$\triangle\ \mu_\alpha = \mu_\alpha(k) - \mu_\alpha(k-1) \tag{3.14}$$
$$\triangle\ \sigma_v = \sigma_v(k) - \sigma_v(k-1) \tag{3.15}$$
$$\triangle\ \sigma_\alpha = \sigma_\alpha(k) - \sigma_\alpha(k-1) \tag{3.16}$$

Weighted averages for $\triangle\mu_v$, μ_α, σ_v and σ_α for each of the states are calculated out of the weighted sums according to equations (3.17) up to (3.20).

$$\text{Weighted average} \triangle\ \mu_v = \frac{\sum_{i=1}^{11} \triangle\mu_{v,i} \times \gamma_i}{\sum_{i=1}^{11} \gamma_i} \tag{3.17}$$
$$\text{Weighted average} \triangle\ \mu_\alpha = \frac{\sum_{i=1}^{11} \triangle\mu_{\alpha,i} \times \gamma_i}{\sum_{i=1}^{11} \gamma_i} \tag{3.18}$$
$$\text{Weighted average} \triangle\ \sigma_v = \frac{\sum_{i=1}^{11} \triangle\sigma_{v,i} \times \gamma_i}{\sum_{i=1}^{11} \gamma_i} \tag{3.19}$$
$$\text{Weighted average} \triangle\ \sigma_\alpha = \frac{\sum_{i=1}^{11} \triangle\sigma_{\alpha,i} \times \gamma_i}{\sum_{i=1}^{11} \gamma_i} \tag{3.20}$$

μ_v, μ_α, σ_v and σ_α are generated out the weighted sums according to equations (3.21) up to (3.24).

$$\begin{aligned}
\mu_v(k) = &\ \mu_v(k-1) \\
&+max(min(\text{weighted average } \triangle\mu_v, \triangle v_{max}), \triangle v_{min}) \qquad (3.21)\\
\mu_\alpha(k) = &\ \mu_\alpha(k-1) \\
&+max(min(\text{weighted average } \triangle\mu_\alpha, \triangle \alpha_{max}), \triangle \alpha_{min}) \qquad (3.22)\\
\sigma_v(k) = &\ \sigma_v(k-1) \\
&+max(min(\text{weighted average } \triangle\sigma_v, \triangle \sigma_{v_{max}}), \triangle \sigma_{v_{min}}) \qquad (3.23)\\
\sigma_\alpha(k) = &\ \sigma_\alpha(k-1) \\
&+max(min(\text{weighted average } \triangle\sigma_\alpha, \triangle \sigma_{\alpha_{max}}), \triangle \sigma_{\alpha_{min}}) \qquad (3.24)
\end{aligned}$$

where,
$\triangle v$: is how much change in speed might happen in $\triangle t$ if old speed is given.
$\triangle_\alpha$: is how much change in direction might happen in $\triangle t$ if old direction is given.
$\triangle\sigma_v$: is how much change in standard deviation of speed might happen in $\triangle t$ if old standard deviation of speed is given.
$\triangle\sigma_\alpha$: is how much change in standard deviation of direction might happen in $\triangle t$ if old standard deviation of direction is given.

From the above equations we can observe that the changes in the means and the standard deviations are limited between a maximum and a minimum values. These two bounds represent the maximum and the minimum changes that a normal human being can make during the time interval of interest. For example, the change in the mean speed is not allowed to be more than $\triangle v_{max}$ or less than $\triangle v_{min}$.

An acceleration profile is created to set limits for the change of $\triangle\mu_v$. This is done so that the change in speed will not be more or less than what the human can accelerate or de-accelerate in one second.

Again statistics are used to create this acceleration profile. Some of the facts that are used to create this profile are as follows:

- At speed zero human can only accelerate, so the de-acceleration is zero.
- Maximum speed that a human can reach is about 8 m/s or 480 m/min, which gives an acceleration of 1.35 m/s^2. This is the maximum acceleration a human can have.
- Maximum human acceleration happens when the human is at zero speed.
- Human de-accelerate faster than his acceleration. So maximum de-acceleration is set to be -2.7 m/s^2, which is double the maximum acceleration.
- At maximum speed the human can have his maximum de-acceleration.
- At maximum speed the human being can only de-accelerate. So his acceleration will be zero.
- A linear profile is assumed between these maximas and minimas.

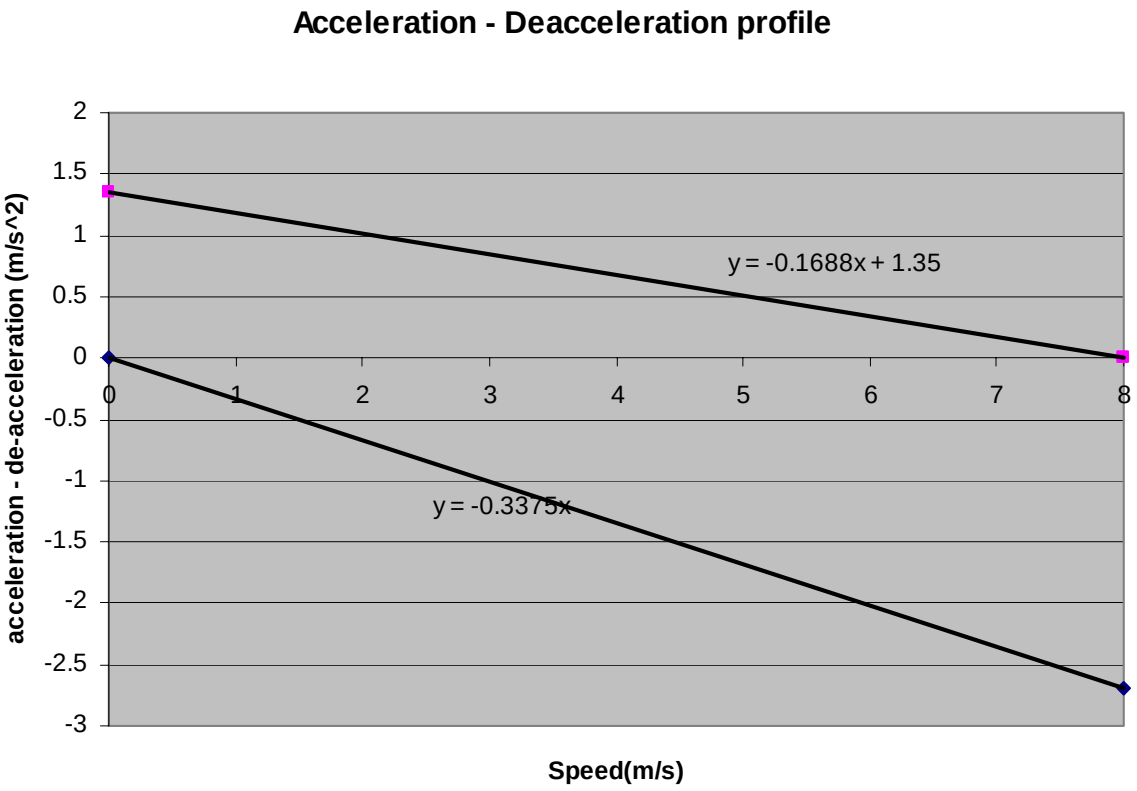

Figure 3.11: Acceleration -deacceleraation profile

A graphical illustration of the resulting profile is shown in figure (3.11).

The movement model normally suggests that the pedestrian will make a change in speed of some amount during an interval of one second. This change will be passed to the acceleration profile and according to the old speed of the pedestrian an allowable change will be decided. If the suggested change by the model was more than the profile allows, the change will be limited to the maximum at that old speed point. For each old speed point there are two limiting points, acceleration limit and de-acceleration limit. If the suggested speed change was positive acceleration limit will be used, while de-acceleration limit will be used if suggested speed change was negative. To calculate these two limiting points for each old speed point, curve fitting is used. Using curve fitting, an equation for the acceleration part and another equation for the de-acceleration part of the profile are created. The two generated equations are also shown in figure (3.11).

For simplicity, no limiting profiles are implemented for the mean direction, the standard deviation of the direction and the standard deviation of the speed. This is means whatever change is suggested by the movement model regarding these quantities will be accepted. This might be improved by including some human factors that affect how much the human being can change his direction, σ_v and σ_α in one second. These factors might be used to create the limiting profiles.

At this point μ_v, μ_α, σ_v and μ_α are available. According to figure (3.2) the four values will parameterize a two dimensional Gaussian block. Speed and direction are

assumed to be independent. So instead of sampling from a two dimensional Gaussian, sampling is done from two separate one dimensional Gaussian [12]. Equation (3.25) shows the Gaussian distribution equation which is used for the sampling purpose.

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{(-(x-\mu)^2)/(2\sigma^2)} \tag{3.25}$$

where,
$f_X(x)$: is the probability function (also called the probability density function (PDF) or density function) of a continuous distribution.

The sampling process results in a sampled speed and a sampled direction. According to figure (3.2) the sampled speed and direction will be used to calculate the new position for the next time step. The distance l walked by the pedestrian within a time duration of *vartrianglet* and with a speed v can be calculated according to:

$$l = v \,.\, \triangle\, t \tag{3.26}$$

Using Pythagorean Theorem, the new position can be calculated as follows:

$$x(k+1) = l.cos[\alpha(k+1)] \tag{3.27}$$
$$y(k+1) = l.sin[\alpha(k+1)] \tag{3.28}$$

The new position is a prediction for the pedestrian position. When a measurement arrives, it gives a weight for this prediction according to prediction and update equations (2.9) and (2.10). The new position will be the input to the movement model at the next time step and prediction continues. The movement model will use this new position to calculate obstacles and steepness states at the next time step. While it will use the old activity, orientation, arousal, emotions, activeness conditions to calculate the new conditions at the next time step. Figure (3.12) demonstrate how the movement model evolves with time and its causal relationships.

3.6 Inferring pedestrian situation from his movement

Till now the developed movement model uses the pedestrian situation to give information regarding his movement and as a result his position. But, isn't it possible to use the pedestrian position to get information regarding his situation? In this case we will be able to infer the pedestrian's situation from his movement.

Figure (3.12) represents a Bayesian network that can be used to go back from the effect to the cause. Noisy position and direction observations are used to estimate the position of the pedestrian at any time step. For example at time $k+1$, the particle filter will use these noisy observations to give a better estimation for the position. On the other hand, the estimated position at that time step might be used to get information regarding his speed and direction at the previous time step k. It is also noticeable

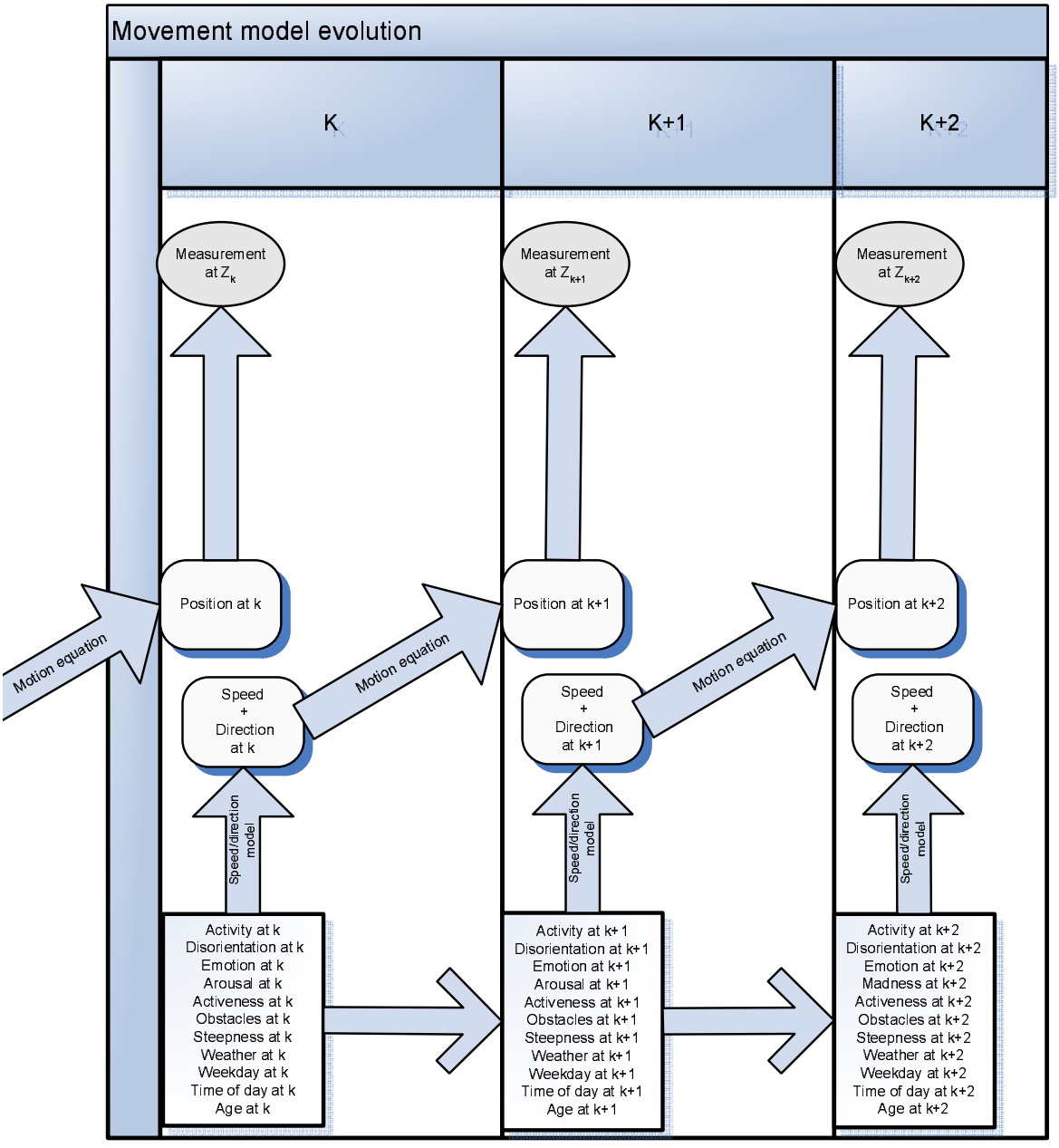

Figure 3.12: Movement model evolution with time and its causal relationships

from the figure that the pedestrian speed and direction at time k might be used to get an indication about his situation. As a result states like: activity, disorientation, emotions, arousal, activeness, age, weather, time of the day, weekday, obstacles and ground steepness might be approximated (inferred) out of the speed and the direction. Another inference application is described in [40].

To include the inference in the particle filtering paradigm several issues are considered. First, particles states are extended to have not only position and direction, but additionally all the eleven states. As a result, each particle can have different activity, disorientation, emotions, arousal, activeness, age, weather, time of the day, weekday, obstacles and ground steepness conditions. Second, particles are initialized with random conditions of each of the eleven states. As time evolves particles will start moving according to their specific conditions of each of the eleven states. In other words, they will start moving according to their situations. This means that particles with different situation compared to the pedestrian will not move as he does. So, they will move "away" from him. When the next measurement arrives, it will be used to weight the prediction according to equation(2.10). This means that particles will get weights accordingly. As a result, particle that moved far from the pedestrian will get low weights, while particles that are close to the pedestrian will get high weights. The resampling step explained in section (2.5.3) will cause the distant particles to vanish, while particles that are close to the pedestrian will be propagated. Finally, the situation of the surviving particles is used to give an indication for the situation of the pedestrian. This might be done by counting how many particles are in each specific condition of the eleven states, and by taking their weights into account. The number of particles in a specific condition indicates the situation of the pedestrian. For example, if the number of particles that are in the "totally drunk" condition of the disorientation state is more than the others, then the pedestrian is most probably "totally drunk".

Inferring high level states (situation) from low level data (location) marks the advance from solely location based services to situation-aware services with large potential for increased user benefit. For example, if the system knows from the pedestrian movement that he is tired, then the next chill out point is the best service that can be offered to him.

Chapter 4

System design

4.1 Problem statement

A simulator and a demonstrator for an indoor/outdoor positioning system for pedestrians were implemented using Particle filtering. The implementation was a part of the Mobile Simulation of Context Information Tool project (MOSCITO). The goal of this project is to investigate different location estimation techniques and analyze their performance. The objective of this work was to analyze particle filter as a location estimation algorithm and integrate it into the MOSCITO suite.

A real time implementation in which a pedestrian is trackable was required. Furthermore, real time inference of his situation was also required. The demonstrator shall facilitate an evaluation of the particle filtering approach and the functionality of the movement model.

Indoor navigation is challenging due to measurement noise caused by multipath and signal blocking. It is therefore of utmost importance to derive accurate estimations out of these noisy measurements that encompass the entire available knowledge about the pedestrians. In addition, the inference of the pedestrian's situation out of his movement may improve the quality of location based services significantly.

4.2 Requirements

The application of particle filtering for improvements of indoor/outdoor navigation accuracy was required. Inferring the pedestrian situation from his movement was an additional requirement. It was also required to implement a GUI which shows the measurements, the estimations and the inferred pedestrian situation. Accuracy calculations of the particle filtering algorithm should also be done.

4.3 System setup

Figure (4.1) illustrates the needed setup to achieve the pre-mentioned requirements.

The setup shows a pedestrian carrying a backpack that is equipped with position and direction sensors (implementation of more sensors is going on). The backpack is

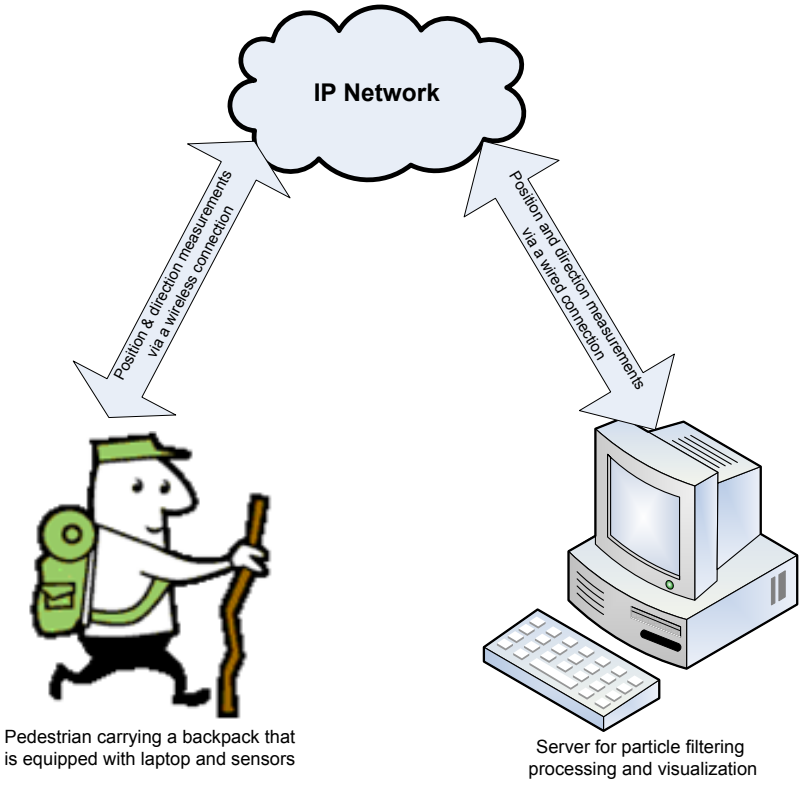

Figure 4.1: Setup functionality

also equipped with a laptop that collects the sensors readings and transmits them over an IP network to a server. This client-server environment was needed because this is the situation in most location based services, where the providers collect the data, process it and accordingly response to the user with the appropriate services.

The sensors are connected to the laptop either wireless or wired. Example of wireless connection is bluetooth, while serial port connection is an example of a wired connection. A wireless connection is used to connect the laptop to the IP network. Examples of such wireless connection could be WLAN or 3G/GPRS.

Data that are provided by the sensors are noisy. The server processes the noisy data using particle filtering algorithms and visualizes the estimates for the positions and the directions. Situation Inference from the pedestrian's movement is also done at the server side. For this purpose, a separate display is created for each of the eleven states described in section (3.1). Each display shows the current inference of that state of the pedestrian. For example, if the disorientation display is turned on, it shows what is the inferred pedestrian disorientation from his movement. Inference accuracy is also analyzed. The following is visualized at the server side:

- A Map of the pedestrian's vicinity.
- The received measurements for the position and the direction of the pedestrian.
- The estimated position and direction of the pedestrian.

- Soft Location estimation for the position of the pedestrian using probability density functions (PDFs).
- Probability density function is visualized in gray scale.
- Error and accuracy calculations.

As the pedestrian moves around DLR premises, measurements keep arriving and are visualized at the server side. Soft and point estimates are visualized accordingly. The Accuracy and the performance of the particle filtering algorithm for location estimation were investigated.

4.4 Simulation setup

A simulator for the positioning system has been implemented. The functionality of the particle filtering in positioning, the implemented movement model and inference were simulated before applying them in the real world. This was helpful during the development stage, since it offers the chance to test the functionality of the current step before going to the next. Other benefits of the simulator are as follows:

- Testing purpose, since it is easy in the simulation to change many parameters that are hard to play with in the real time test.
- Test some behaviors of the pedestrian which are hardly generated in the real world.
- Error calculations, due to the lack of reference measurements in the real system. This offers a better evaluation of the accuracy of the particle filtering algorithm.
- Debugging purpose, since the whole implementation is under control in the simulation mode.
- Checking the functionality of the new ideas and thoughts without the need of building them, which saves time and effort.

Next is a summary of the considered scenario in the simulation mode. A simulated pedestrian who walks with some speed and in some direction is considered. He starts at time t with random speed, direction, position, activity, disorientation, activeness, etc. His position, activity, activeness, disorientation, etc. at time $t + 1$ are calculated according to the developed movement model. The pedestrian is equipped with GPS sensor and electrical compass. As a result, position and direction measurements should be simulated. The simulated measurements are calculated out of the known pedestrian information. Additive White Gaussian Noise is added to the position and direction to get the GPS-measurement (x, y) and the heading (α). The particle filter receives only the noisy measurements. Prediction is done using the developed movement model, while the simulated measurements are used to update the prediction. The particle filter estimates the simulated pedestrian position and direction from the noisy measurements.

Estimated position and direction are compared with the simulated ones. Different kinds of errors are calculated and visualized to allow a better estimation for the accuracy of the algorithm. Linear errors, squared errors and filtered errors are calculated. The simulated pedestrian situation (i.e. disorientation, activity, age, emotions, etc.) is inferred out of the measurements. In order to study the behavior and the accuracy of the particle filter for situation estimation the following is visualized:

- DLR satellite map which is used for the simulation.
- The simulated position and direction of the pedestrian.
- A soft estimation of the location of the pedestrian using probability density function (PDF).
- A point estimation of the location and the direction of the pedestrian.
- Error calculations are done and visualized in a separate panel.
- A separate display is generated for each of the states for the inference purpose. Each state display shows what is the inferred state at that time step. The real simulated state is also shown in the panel.

Comparisons are made between the real states and the inferred states and the accuracy of the inference was evaluated.

4.5 Software design

The noisy sensor measurements are transmitted from the sensor platform to the server via the IP network. At the server, the position and the direction measurements are processed using particle filtering. Estimations for position, direction, activity, activeness, disorientation, etc. are done. The received measurements for position and direction, the estimations and the inference are visualized. Figure (4.2) shows the system components and their interactions at the server side.

The main components of the MOSCITO core are the location server and the visualization server. Likelihood functions, fusion engine and movement model are the main components of the particle filter plugin. MOSCITO core, particle filter plugin and their dependencies are shown in figure (4.2).

A network transmitter transmits the noisy sensor measurements via the IP network to the server side. A network receiver receives the measurements at the server side and passes them to a location server. The location server passes the measurements to the particle filter plugin and specifically to the likelihood functions. The arrived noisy measurements are used to prepare likelihood functions ($p(z_k|x_k)$) which will be used in the update stage according to equation (2.13). The fusion engine starts the particle filtering process with the prediction stage according to equation (2.12). Prediction depends on the developed movement model in chapter (3). At one side, the fusion engine uses the movement model block outputs to predict pedestrian location and direction at time $t + 1$. At the other side, it uses the likelihood functions at time

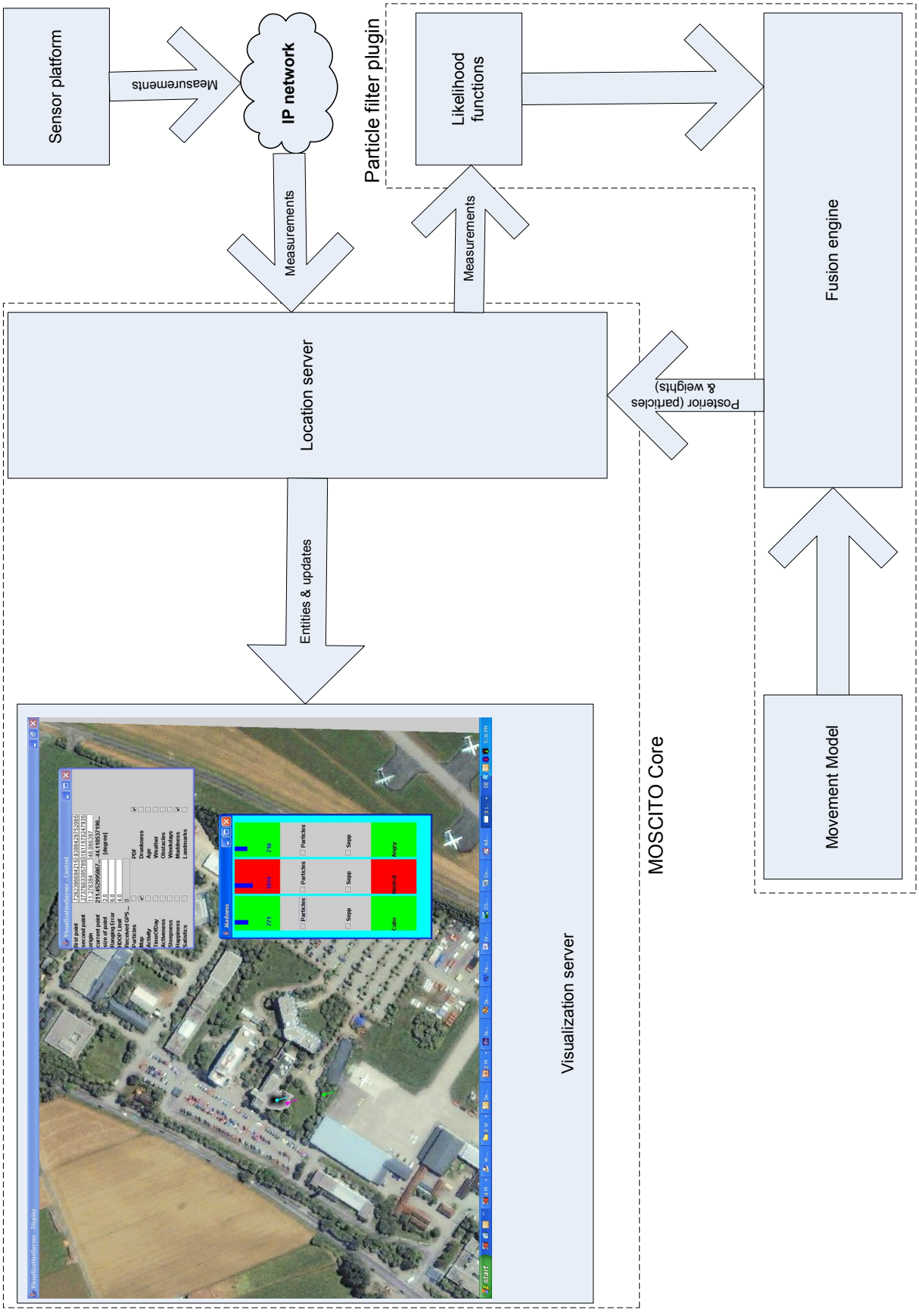

Figure 4.2: Software design at the server side

$t + 1$ to update the prediction. The fusion engine sends the posterior (particles and their weights) to the location server, whose main task is to manage, handle and store localized entities. It also delivers those entities to the visualization server in order to analyze them. Examples of **location server** tasks are as follows:

- Registers all the elements that are needed to be visualized like particles, measurements, persons and estimations. It also assigns an ID for each of them.
- Updates the registered entities if any change happens regarding any of their associated parameters.

As the visualization server receives the entities it starts analyzing them and visualizing the estimation results. Examples of the tasks assigned to the **visualization server** are as follows:

- Analyzes and visualizes the map data.
- Analyzes measurement entities and visualizes them.
- Analyzes particles and visualizes the particles cloud.
- Interpolates particles to create probability density function (PDF) of location. The PDF is visualized using a gray scaled cloud, where the probability of location increase with the increase of the color depth.
- Generates and visualizes a point estimate out of particle cloud.

Figure (4.2) shows also the display of the visualization server. The map, the simulated pedestrian (pink), the noisy measurement (green), the PDF (gray cloud) and the estimated position (cyan) are shown.

If no measurements are available, the fusion engine keeps predicting using the movement model and sends the posterior to the location server for analysis and visualization. As the measurements arrive, the likelihoods are prepared and used for update. By then, a more accurate posterior will be sent to the location server for visualization. Prediction and update continues as time evolves.

The fusion engine also uses the movement model for inference. According to the situation of the surviving particles, the fusion engine approximates the situation of the pedestrian. For example, if most of the particles that survived where the initially totally drunk particles, then the pedestrian is mostly totally drunk. Inspection panels are created by the particle plugin and inference results are shown on them. As an example, the arousal inference display is shown in figure (4.2).

Chapter 5

Implementation aspects

5.1 Hardware

This section discusses implementation aspects with respect to the hardware necessary to the system described in chapter 4. It gives brief information about each of the used hardware components. Additionally, it demonstrates where each of the hardware members fits in the system design and what functionality it contributes to achieve the implementation target.

5.1.1 Hardware components

1. Server host that has Windows XP operating system and Java Virtual machine.

2. Backpack.

3. Mobile host that has Windows XP operating system and Java Virtual machine.

4. 3G/GPRS data card. It is a Vodafone PCMCIA Type II Card [44] with a SIM card slot. The card supports UMTS 2100 MHz, GPRS/UMTS 850 MHz, 900 MHz, 1800 MHz and 1900 MHz. The card provides a link to the mobile network operator, and the connection is established through the operator to DLR's network.

5. Bluetooth adapter (3COM). Spontaneous wireless connections between a desktop or notebook PC and other Bluetooth devices is possible with this card. It provides reliable connections with speeds up to 720 kbps at up to 10 meters.

6. Vluetooth GPS receiver (RoyalTek) [45]. It provides location information in the form of longitude, latitude and height. The location information is sent to the coupled mobile device through bluetooth. The RoyalTek GPS receiver is a 12 channel GPS receiver that allows continuous tracking of all visible satellites. The GPS receiver measurements are send via bluetooth to the laptop in the NMEA format [46]. Details about the NMEA protocol are given in appendix (E).

7. Electronic compass. A Palm Navigator that is produced by Precision Navigation, Inc. [47] was used. If aligned and calibrated correctly it gives the heading of the user. The electronic compass provides four vectors from which the angle of the pedestrian related to north are calculated.

8. Wireless voice communication devices (walkie-talkie system or mobiles).

Photos of the equipment used are shown in appendix (D).

5.1.2 Hardware functionality

In this section the functionality of the hardware list is summarized. The bluetooth GPS mouse and electronic compass measure position and direction fixes respectively. Measurements are then collected by the Laptop. The bluetooth adapter is connected to the laptop to allow data transfer from and to the laptop through bluetooth medium. The Bluetooth GPS mouse is connected to the Laptop via the bluetooth adapter, while the electrical compass is connected to the Laptop via the R232 connector. The 3G/GPRS data card is used to connect the laptop to the server through the DLR network. The Laptop, the 3G/GPRS data card, the bluetooth adapter, the bluetooth GPS receiver and the electrical compass are equipped in the backpack. The backpack is carried by the user who starts his walk around the DLR premises. A photo of the backpack equipped and ready to be carried by the user is shown in figure (5.1)

Figure 5.1: The backpack equipped with the laptop, the 3G/GPRS data card, the bluetooth adapter, the bluetooth GPS receiver and the electrical compass.

As the measurements arrive to the server side through the mobile and DLR IP networks, the following should be done:

- Visualizing the received position and direction measurements on the map.
- Performing Particle filtering algorithms using the received measurements.
- Visualizing the estimated position and direction using soft estimation (PDF).
- Visualizing the estimated position and direction as a point estimate.
- Performing and visualize some accuracy calculations.
- Inferring each of the eleven states out of the movement in a separate display.

Wireless voice communication devices (walkie-talkie system or mobiles) are used to communicate between the user who is walking inside DLR premises and the other operator who is sitting at the server side. This communication medium gives the opportunity to check the accuracy of the particle filtering algorithm. It allows the server user to confirm the visualized estimated position with the real position of the other user. It also gives the opportunity for the server user to ask the walking user to act differently according to one condition of the eleven states described in section (3.1). This allows the server user to check the accuracy of our inference.

Hardware setup and its functionality is illustrated in figure (5.2).

5.2 Software

Java S2SE v1.4 [4] was the selected programming language for MOSCITO and was therefore also used for the particle plugin. This section describes a general overview of the main packages and classes of the the system. The relationships between these classes and how they interact to achieve the design goals are also discussed.

Unified Modelling Language (UML) diagrams are used for demonstrating the relationships between the classes, attributes, methods implemented in these classes and the properties of the attributes. UML [48] is a general-purpose visual modelling language that is used to specify, visualize, construct, and document the artifacts of a software design. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. It is intended for use with all development methods, lifecycle stages, application domains, and media.

As explained in section (4.5), the main task of the location server is to manage and store localized entities. This means that different kind of entities should be handled by the location server. Examples of such entities are particles, measurements, persons, estimations, buildings, barriers and landmarks. Some of these entities are having common associated attributes. For example, most of the mentioned entities are having position, direction of view, velocity, name and acceleration. It is convenient to generate a parent class in which all those common attributes are implemented and then generate the specific entities (instances) out of it. The specific entities may

implement more attributes. They may also implement some of the attributes differently compared to the parent class. Creating an interface that contains the needed methods for calculating these common attributes is also useful. With this interface it will be easier to change the way any of the required methods are implemented without affecting all the entities. `Abstract Localized Entity` was the constructed parent class, while `Localized Entity` was the constructed interface. Example of child classes that are derived from the `Abstract Localized Entity` class are `Particle, Barrier, Landmark, Person, Building, Measurement and Estimation.`

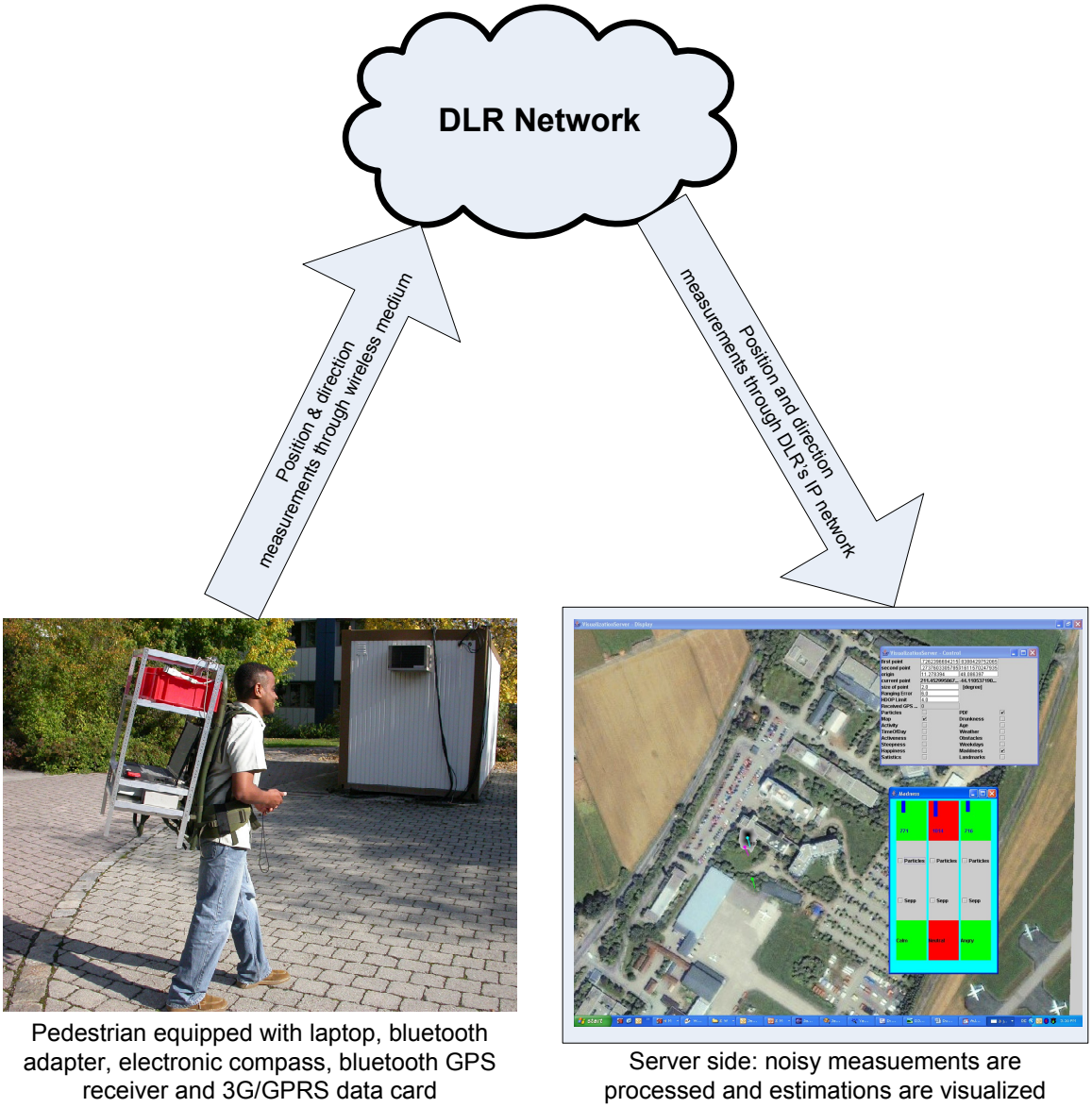

Figure 5.2: Hardware setup and functionality.

The UML diagram which shows the `Abstract Localized Entity` parent class, the `Localized Entity` interface class and the instantiated children is shown in figure (5.3). The `Location Server` class which register, manage and update those entities

is also visualized in the figure. It delivers all those entities to the Visualization Server for visualizing them.

The GPS receiver readings are in polar coordinates. All the visualization at the server side is done in rectangular coordinates. To transform a point from polar coordinates to rectangular, the radius of the circle and the angle of that point are needed. The difficulty with the earth is that it is not ideally spherical. That is why a special model for the earth that takes into consideration the ellipticality of the earth and other deviations from ideal shape was needed. The used model for the earth is the World Geodetic System 1984 (WGS84) model. It is the geodetic reference system used by GPS. A class that reads the polar coordinates of the position displayed by the GPS receiver and transfer them into rectangular coordinates using the earth model was implemented. A UML diagram for the transformation utility package is demonstrated in figure (5.4). The figure shows a General `Earth Model` class and the `WGS84 Earth Model` class which extends it. The `Translated Earth Coordinates` class is responsible of coordinates transformation. It extends an `Earth Coordinates` class in which the resultant rectangular coordinates will be written, while it uses `Polar Coordinates` as input.

Figure (5.5) shows a stack diagram for the particle filtering package. Starting the stack from bottom to top an explanation of the particle filtering package follows:

- A `Sensor` class extracts and provide the measurements that are read by the physical sensors.
- A `Measurement` class collects the measurements from the `Sensor` class and prepares them to be delegated to the `Likelihood Function` class.
- The `Likelihood Function` class use the measurements delegated by the `Measurement` class to build likelihood functions ($p(z_k|x_k)$). Likelihood functions will be used according to equation (2.13) in the update stage of particle filtering.
- `Entity Characterization` and the `Movement Model` will be used to predict the pedestrian position according to equation (2.12). `Entity Characterization` tells what kind of entity is tracked and give some details about it. It will be used by the `Movement Model` class to select the appropriate movement model for the entity. The selected movement model will be used in the prediction stage to calculate ($p(x_k|x_{k-1})$), which is the position and the direction at time $k+1$ given the position and the direction at time k. Details of the implemented movement model are explained in chapter (3).
- The `Fusion Engine` class represents the particle filtering engine. It uses the `Movement Model` class from one side for prediction and the `Likelihood` class from the other side for updates. The `Fusion Engine` calculates the posterior ($p(x_{1:k}|z_{1:k})$) every time that a measurement is received. If no measurements are received, the `Fusion Engine` will keep predicting the pedestrian's position and direction using the movement model. As measurements arrive and likelihoods get ready, the predicted pedestrian's position and direction will be updated according to the measurements.

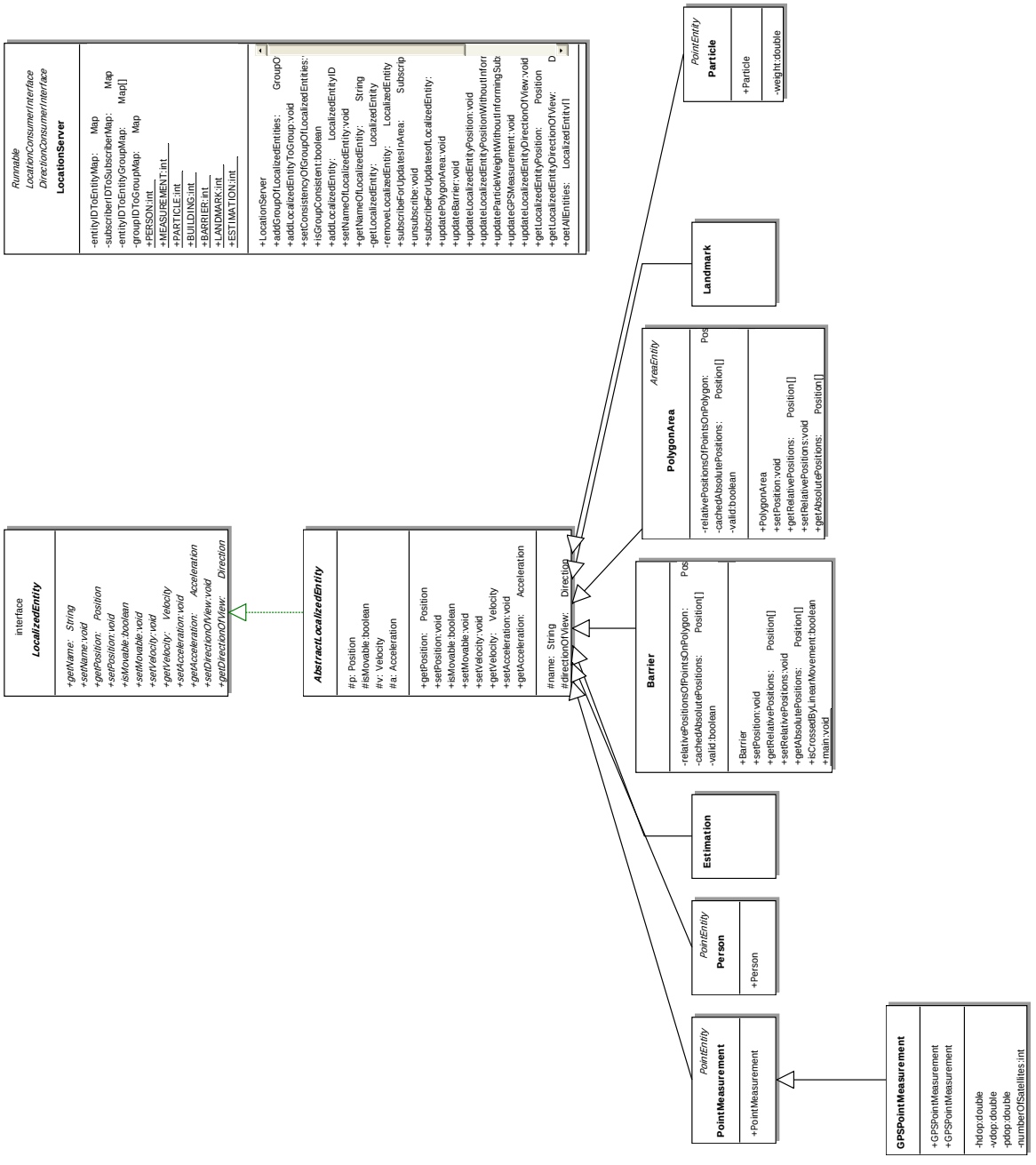

Figure 5.3: UML diagram for the `Abstract Localized Entity`, its parent and children classes

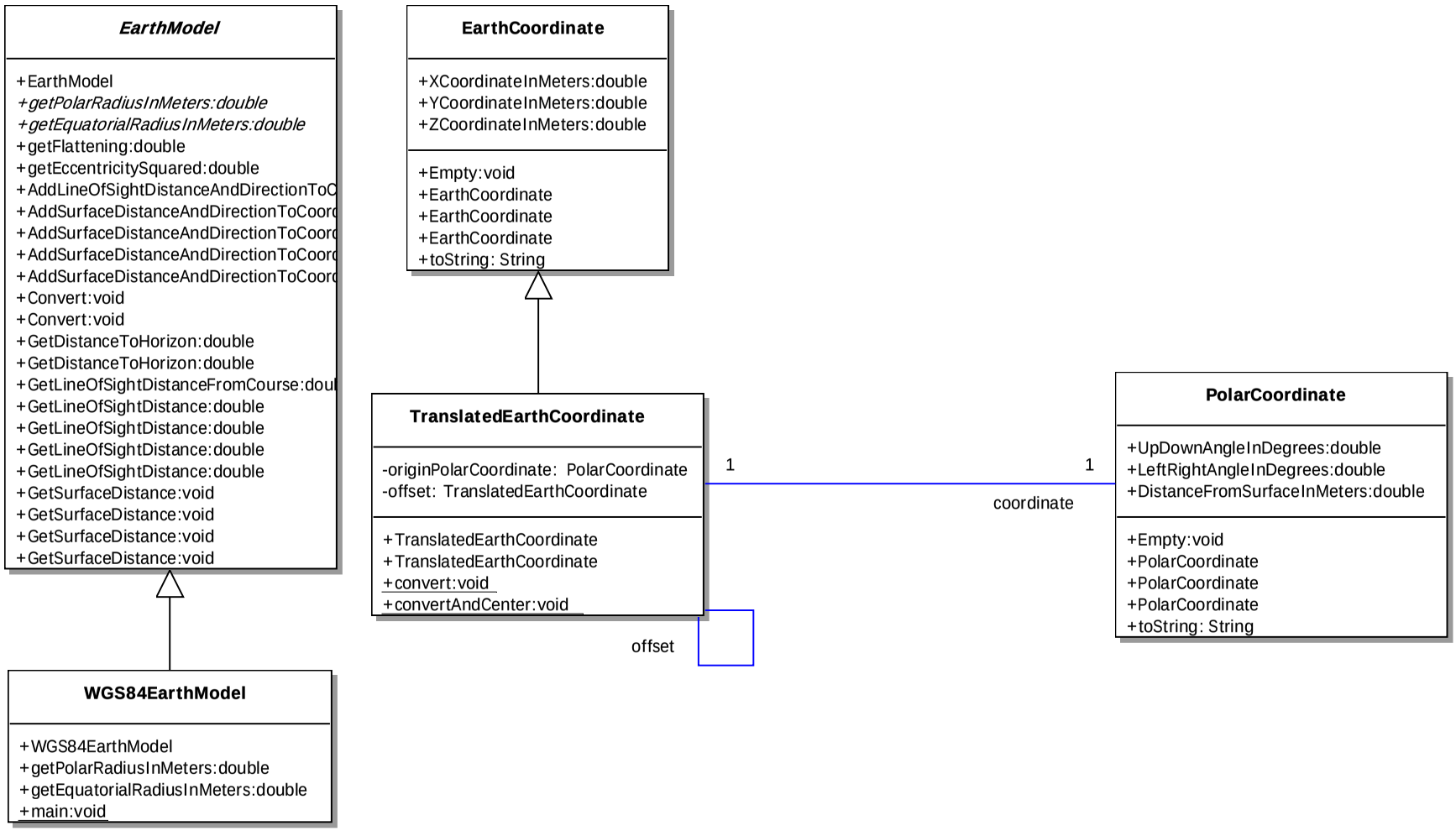

Figure 5.4: UML diagram of polar-rectangular location transformation utility package

- The `Entity Location Management` class act as a communication and management interface. The class that implements it has a location server, to which it delegates the management and storage of entities. The main roles of the `Location Management` class are as follows:
  - Registers the entities and assigns them IDs.
  - Collects measurements and estimations and passes them to the visualization server to show them.
  - Checks for entities updates and sends the updates for visualization.
  - Prepares the inference panels according to the number of the states considered in the movement model.
  - Updates the inference panels according to the particles states. Each state panel shows the number of the particles in each condition of that state. For example the age panel, shows the number of particles that are young, youth and old. If most of the particles are in the "youth" condition then the pedestrian situation could be inferred as youth.

- The `Context Fusion` class fuses the location data that are coming from the `Entity Location Management` class. It combines, refines and uses these data. For example location data that are coming from the `Entity Location Management` class could be combined with the some context information like the weather status to have a complete description of the situation of the pedestrian.

- The `Application` class uses the situation description of the pedestrian to offer him some location based service. The `Application` class represents the front end of the implementation. An example of an `Application` class could be a lunch advisory service.

Context information could also be propagated backwards from the `Context Fusion` class to the `Fusion Engine` to improve location estimation process. For example if a context information like "the pedestrian is in his car" is available, then passing this information to the particle filter will help it excluding some areas from location estimation process. This will improve the efficiency and the calculation complexity of the fusion process.

Figure (5.6) shows a component diagram of the particle filtering package. It shows some extended components of the stack diagram shown in (5.5). The following should be noticed from the figure:

- The `Bayesian Filter` class is a `Fusion Engine` since prediction and update (fusion) are also needed in the Bayesian process. The `Fusion Engine` is a `Likelihood Listener` since it needs the likelihoods for updates. Additionally, a `Particle Filter` represents a `Bayesian Filter`.

- The `Likelihood Function` is a `Measurement Listener` since it senses measurements that will be used to build the likelihoods. It is also a `PDF` since the output of it will be in a form of probability density function.

- The `Runnable Measurement` class extends the `Measurement` interface and keeps running to collect measurements from the sensors .

- The `Measurement State` class which currently has three dimensions (x-position, y-position and direction) extends the `State` interface class. `Human State` which has more dimensions also extends the `State` interface. Examples of these additional dimensions are disorientation, arousal, activity, age and weather.

Figure (5.7) shows a UML diagram for the particle filtering package. From the figure we can notice the following:

- The `General Particle Filter` class implements the `Fusion Engine`. It is the center class of the implementation since it predicts using the `Movement Model`, updates using the `Likelihood Function` and delegate results to `Entity Location Management` class for demonstration.

- Three different types of particle filters are implemented. They extend the `General Particle Filter` central class. They are the `Sampling Importance Sampling (SIS)`, the `Sampling Importance Resampling particle filter (SIRPF)` and the `SIR with` $N_{\texttt{eff}}$ particle filters. A detailed explanation of each of them is given in section (2.5).

- The `General Particle Filter` uses a `Resampler` class for resampling. The reasons why resampling is needed in particle filtering processing, and how it works is explained in section (2.5).

- A `Systematic Resampler` class implements the `Resampler`.

- The `General Particle Filter` uses the `Movement Model` interface class for predicting the pedestrian location at each time step. A `Human Movement Model` class (not shown in the figure) implements the `Movement Model` interface class. Details of the `Human Movement Model` class specifications are explained in chapter (3).

- The `General Particle Filter` class uses a specific number of particles specified at run time. An equivalent amount of particles is needed by the `General Particle Filter` class for resampling. The needed particles are obtained from the `Particle` interface class by instantiating children classes out of it.

- The `General Particle` class implements the `Particle` interface class. All associated methods that are needed in the `Particle` interface are implemented in the `General Particle` class. The `Location Particle` class extends the `General Particle` class and adds some additional specific location methods.

- The `Particle` interface uses the `Discrete Particle PDF` class to calculate the particles weights. Particle weight is the posterior ($p(x_{1:k}|z_{1:k})$) value at the particle position.

- The `General State` class implements a `State` interface. The `State` interface includes common states that entities have and their associated methods. Example of such states are position and direction. `Human` and `Measurement States` are examples of classes that extend the `General State` class according to their need.

Soft Location Visualization

It was required to generate and visualize a probability density function (PDF) of the pedestrian's location at the visualization server display. The visualized PDF offers insight into the soft decision criteria at the area of interest. PDF generation and visualization were required in both simulation mode and real time system. The PDF was generated out of the particles cloud by interpolation. Following steps were done to generate the PDF:

- A gaussian distribution was assumed around each of the particles. The mean of the Gaussian function was set to the position of the particle. The amplitude of the Gaussian distribution was set to the weight of the particle. The variance of the Gaussian distribution was formulated to be dependent on the distribution of the particles. If the particles are spread the variance increases and vice versa.

- The PDF value at any pixel point of the display was calculated by summing the values of all the Gaussian functions of all the particles at that point.

- The overall PDF was created by calculating the PDF values for all the pixels in the panel.

- Calculating the PDF at all the pixels of the panel is time and memory consuming. To solve this problem, the PDF was calculated only around the area in which the particles had weights above an adjustable threshold.

- A gray scale was selected to visualize the PDF, where darker points represented pixels with higher densities. On the other hand, pixels with low PDF values (i.e. low densities) were visualized with lighter colors in the gray scale.

- A double buffer was used to draw the PDF in order to speed the visualization process. In such case, the PDF at the previous time step will be stored in the visual buffer till the new PDF is calculated. As soon as the new PDF gets ready, it will be passed to the visual buffer and displayed.

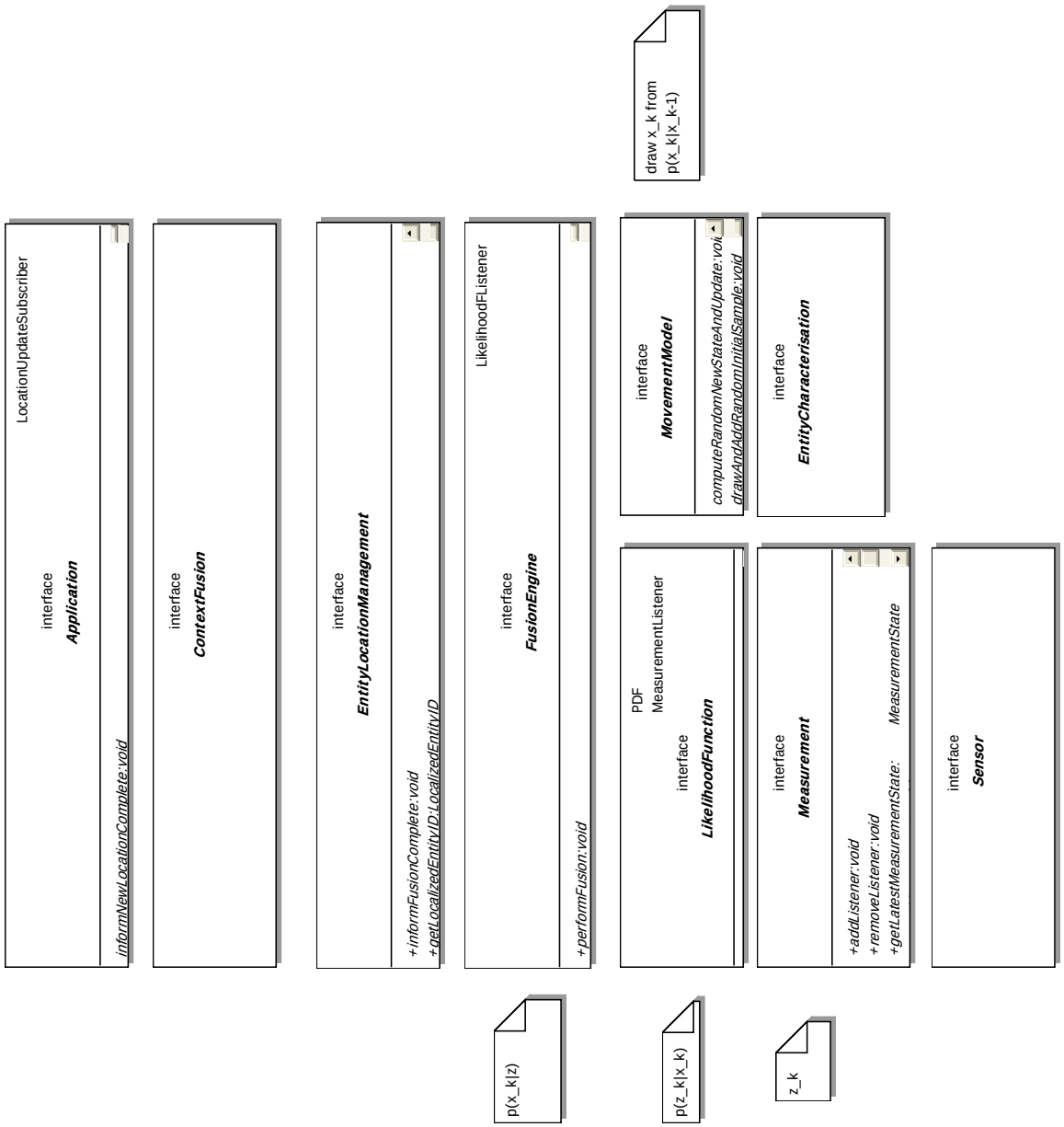

Figure 5.5: Stack diagram of the particle filtering package

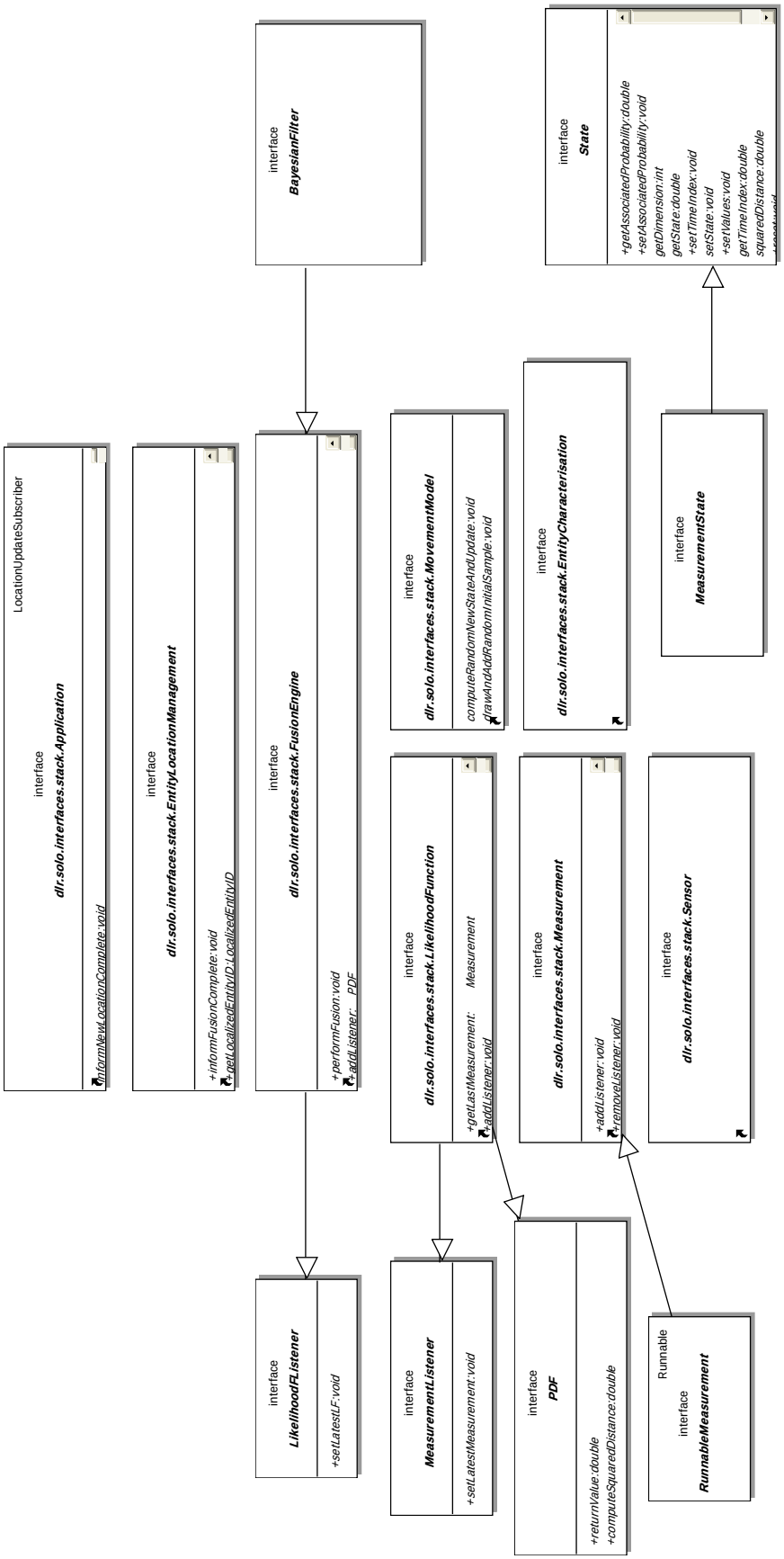

Figure 5.6: Component diagram of the particle filtering package

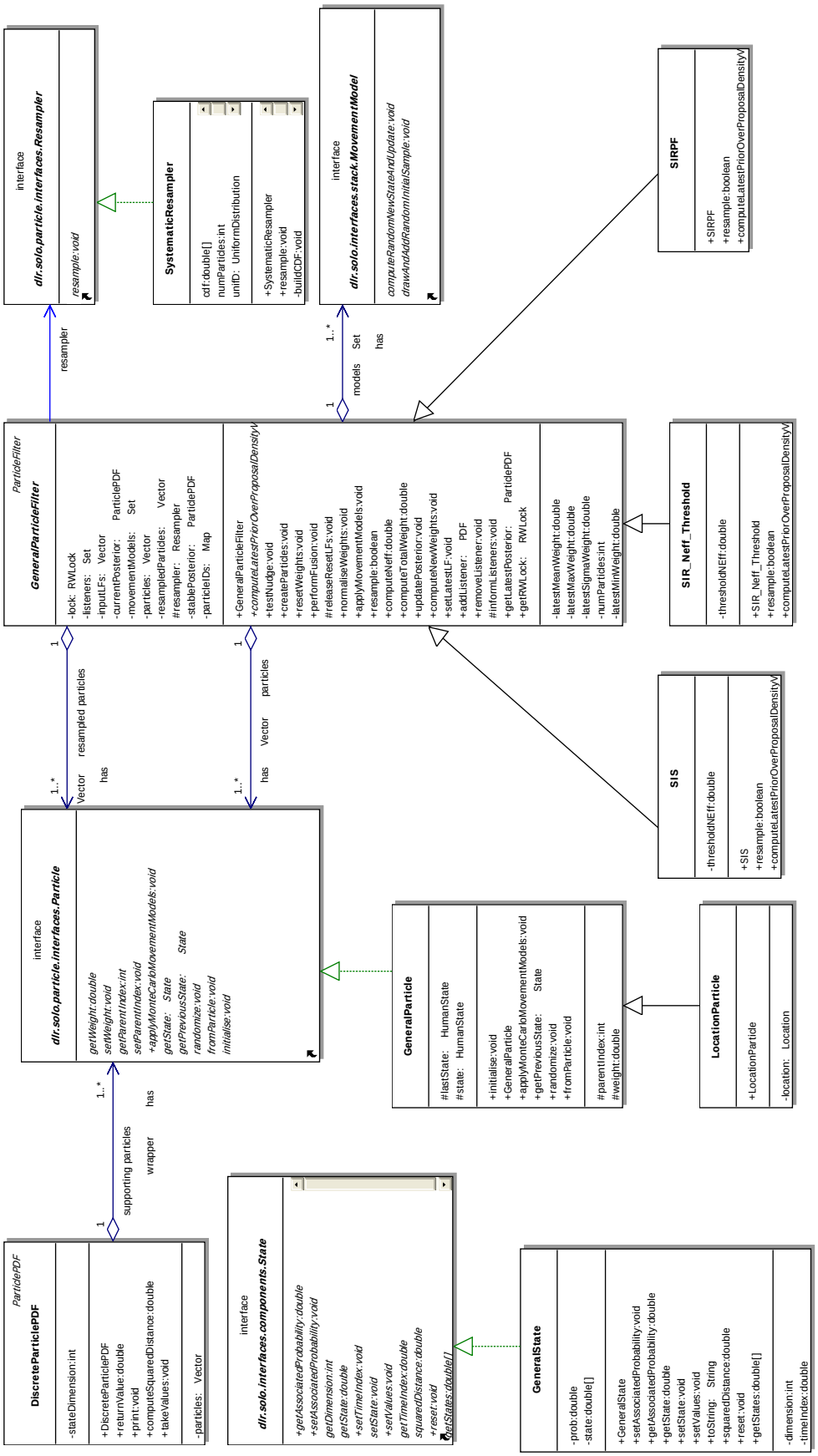

Figure 5.7: UML diagram for the particle filter implementation

Chapter 6

Test, Integration and Experiments

6.1 Simulation setup

The implementation started with the simulation setup. The simulation setup was an obvious starting point since it allows class wise test and integration. Additionally, it gives the opportunity to check the software functionality before connecting the hardware needed in the real system. Simulating different possible hardware components helps selecting among different possible vendors. Also, different new ideas might be tested without the need of the costly real system tests.

A summary of the test and integration steps in simulation mode is given next. First, classes and packages were tested separately before integrating them into the whole project. Then, modifications were done on them in order to achieve the required behavior and functionality. After that, related classes were integrated to form packages and simulations were run on these packages to confirm functionality and performance. Packages were integrated to achieve the system design demonstrated in figure (4.2). Next, simulations were done on the whole system and testing of some simulated behaviors of the pedestrian was performed. Finally, optimizations were done one class and package level in order to achieve minimal memory and time consumption.

For error and accuracy calculations of the implemented location estimation technique, pedestrian exact position would have to be known to the system for every time step k. This is not an easy task in both indoor and outdoor environments. Since no suitable reference measurement system was available, the mean square errors (MSE) and other accuracy calculations were performed using the simulation setup.

6.2 Real system integration

After test and integration in simulation mode, the real system test and integration was performed. A summary of the process is explained next.

The manuals and the data sheets of the RoyalTek bluetooth GPS receiver [45], the electronic compass [47] and the 3G/GPRS Vodafone data card [44] were reviewed in order to understand their functionality. The format of their output data and how such data could be read by a PC were investigated. Calibrations and test were done on

those devices to confirm their performance and functionality. Then, these devices were connected to the laptop which was in turn fitted to a backpack as shown in figure (5.1).

A GPS reader and a compass reader classes were implemented at the laptop side. The reader classes read the compass measurements from the serial port and GPS measurement from the bluetooth port. It also encapsulates the measurements to be send via the network. The two classes were tested by comparing positions and directions read by classes with some known values. A network transmitter class was needed at the laptop side in order to transmit the measurements over the IP network to the server. The reader class delegates the measurements to the transmitter class which in turns create a TCP/IP socket connection to the server. Connection to the server with the transmitter class was established and tested. First, the connection was tested via a normal LAN cable, then via GPRS and UMTS wireless connections.

A network receiver class was implemented at the server side. It receives the measurements from the transmitter class via the established socket connection. Measurements are collected, extracted and delegated to the location server class. The location server class passes the measurements to the likelihood class in the particle filter plugin. Likelihoods are used in the update stage as explained in section (2.5.3). The network receiver class was tested by comparing the transmitted measurements with the visualized ones by the class. Functionality of both transmitter and receiver classes were tested using a normal LAN connection and the wireless 3G/GPRS connection.

At this stage the real system integration process was completed and the system was ready for experiments and additional tests. Experiments and additional test were divided into different stages. First, the electronic compass and the GPS receiver were located at accurately pre-known positions and the visualized measurements at the server side were compared to these pre-known values. Tests were started at positions where the laptop was beside the server so as to get a full access and control of the whole system. Other positions were tested after that. This offers a functionality test of software and hardware at both laptop and server sides. The next step was to do stationary experiments where the electrical compass and the GPS receiver were in fixed positions and directions. Stationary experiments were needed in order to orient the map correctly on the visualization panel and to calibrate the panel so that it shows directions and positions correctly. After that, the GPS receiver was kept stationary while changing the direction of the compass. Estimated positions were compared with the real positions and initial accuracy measures were observed. After stationary experiments, then came the mobile experiments. In these experiments, the pedestrian was equipped with the backpack and a Walk Talky and tracked inside DLR premises. The estimated positions and directions using particle filtering were compared with the real ones. The accurate positions and directions were received from the pedestrian via the walkie-talkie. Accuracy and error calculations were performed in order to get an impression of the performance of the implemented technique.

The next step was to run experiments on inference in both stationary and mobile states. Experiments on inference of all the eleven states considered in the designed human movement model in chapter (3) were performed. This was done by asking the pedestrian to behave according to the different conditions of these states while monitoring their inference panels. An example of a stationary inference experiment

is to keep the GPS stationary and keep changing the direction of the compass. In these experiments the disorientation inference was of interest. An example of a mobile inference experiment is to ask the pedestrian to do a fast walk and check the activity panel. Accuracy and performance of the inference was investigated. Communication between the pedestrian and the server side user was done using the walkie-talkie.

Finally, mean square error (MSE) curves were generated. This was done in the simulation mode as explained in the previous section. The number of particles and the number of time steps were varied and the resulting error curves were investigated. Curves for the case where only one of the two sensors was functioning were also generated. The curves were generated by logging the calculated mean square errors to text files. Then, MATLAB® was used to generate the MSE curves out of these text files. MSE curves are a good measure of the accuracy and the performance of the particle filtering algorithm. The MSE curves are discussed in detail in the following simulation results chapter.

Chapter 7

Simulation results

Some of the simulation results will be visualized and commented in this chapter. They are divided into two categories: The first category includes screen captures of some illustrating visualizations at the server side for qualitative analysis (7.1). Example of such visualizations are the measurements, the particles cloud, the probability density function (PDF) of the location, the point estimation ("hard estimation") and the inference panels. The second category includes curves of mean errors over time for quantitative analysis (7.2. Several curves were generated by varying the number of particles, changing what the particle filter knows about the pedestrian or setting the pedestrian to a fixed situation (like totally drunk) over the entire simulation period.

7.1 Qualitative analysis

Category one figures offers a qualitative measures for the performance of particle filtering. They are screen captures of some important results at the server display. A summary of what might be visualized in these figures is given next:

- A map of the area of interest (DLR premises) is visualized in the background of the visualization server display. It gives a better sense regarding the position of the pedestrian and the accuracy of the estimations.
- The simulated pedestrian has a position and a direction at each time step. The pedestrian is visualized always using the pink color, where a small circle represent his position and an arrow represents his direction.
- The position and the direction of the simulated measurement are visualized into a single small circle with an arrow. Green color was used to visualize the measurements. The position of the small circle represents the simulated position measurement while the arrow represents the simulated direction measurement.
- The particles are visualized as small circles and the blue color is used to visualize the particle cloud.
- The probability density function (PDF) of location is visualized using the gray scale cloud. The dark gray represent points with high density.

- Hard estimations (point estimations) for position and direction were also calculated out of the particle cloud. Hard estimations are visualized using the cyan color. The small circle represent the point estimation for the position while the arrow represent the point estimation for the direction.

Figure (7.1) shows the simulated pedestrian versus the simulated received measurement. From the figure we can see that the measurements are really noisy in position and direction. The particle filter has to use these noisy measurements and produce more accurate estimates for position and direction.

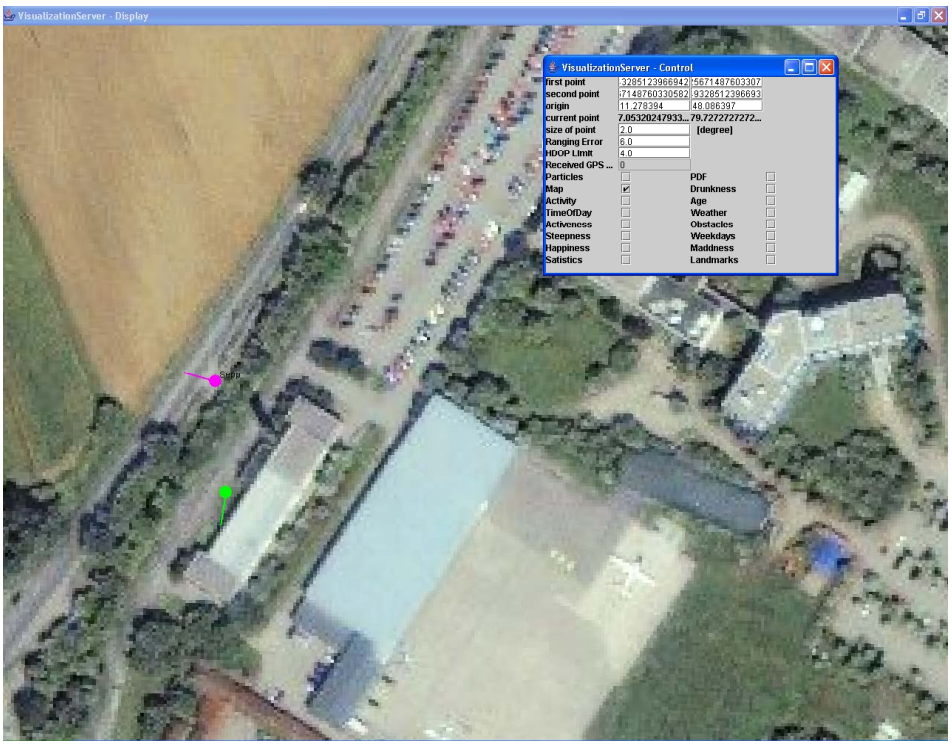

Figure 7.1: Simulated pedestrian(pink) and received noisy measurement(green).

Figure (7.2) shows the simulated pedestrian, the simulated received measurements and the particle cloud. It is visible that the particle cloud is concentrated around the simulated pedestrian position. More particles are accumulated closer to the pedestrian than to the measurement. This indicates the output of the particle filter is a more accurate estimate of the pedestrian's position and direction than the noisy measurement.

Figure (7.3) shows the simulated pedestrian, the simulated received measurements and probability density function (PDF) of location.

Figure (7.4) shows the simulated pedestrian, the simulated received measurements and a point (hard) estimate for the pedestrian position and the direction. Point estimate are generated out of the particle clouds by performing a weighted average over the particles position and direction.

Figure (7.5) shows the simulated pedestrian, the simulated received measurements, PDF of location, point estimate (hard estimate) for the pedestrian position and the

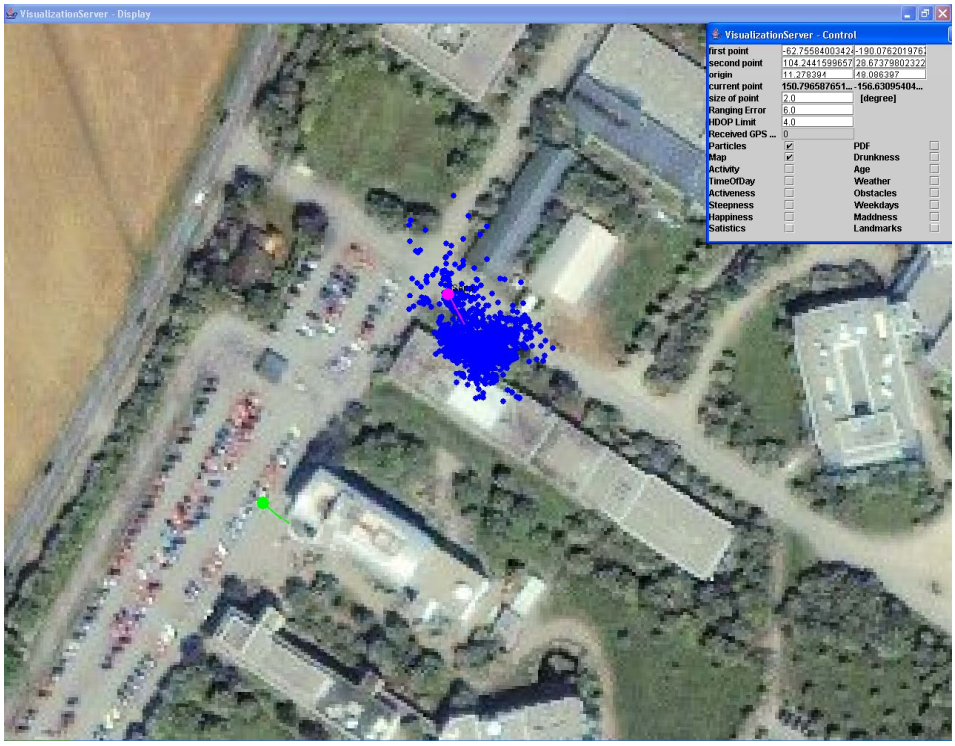

Figure 7.2: Simulated pedestrian(pink), received measurement(green) and particle cloud(blue).

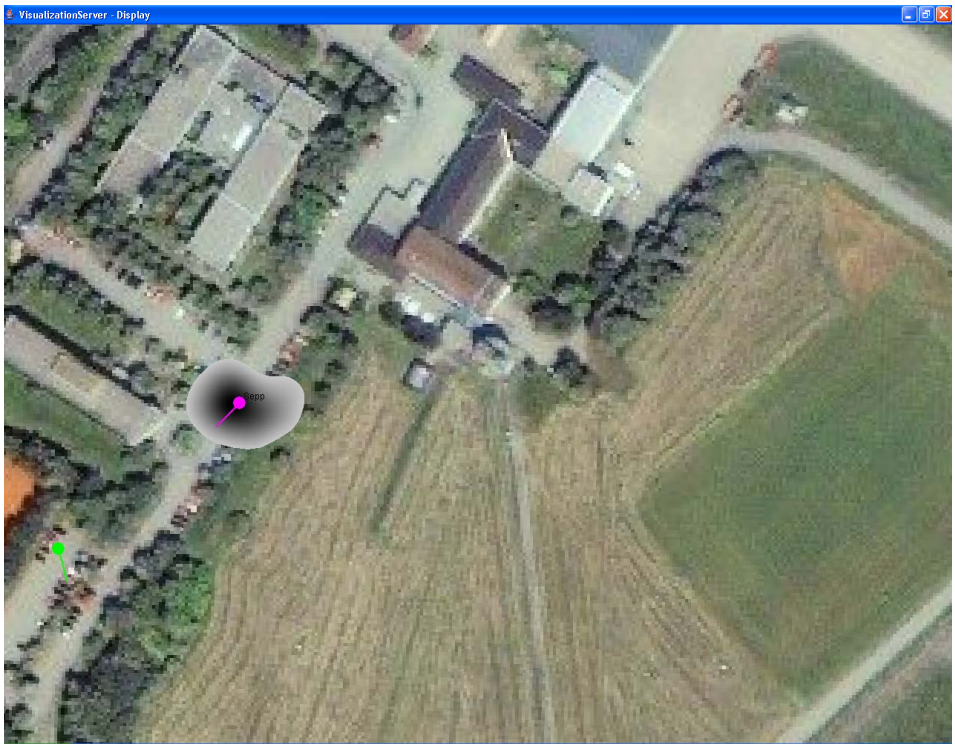

Figure 7.3: Simulated pedestrian(pink), received measurement(green) and PDF of location(gray cloud).

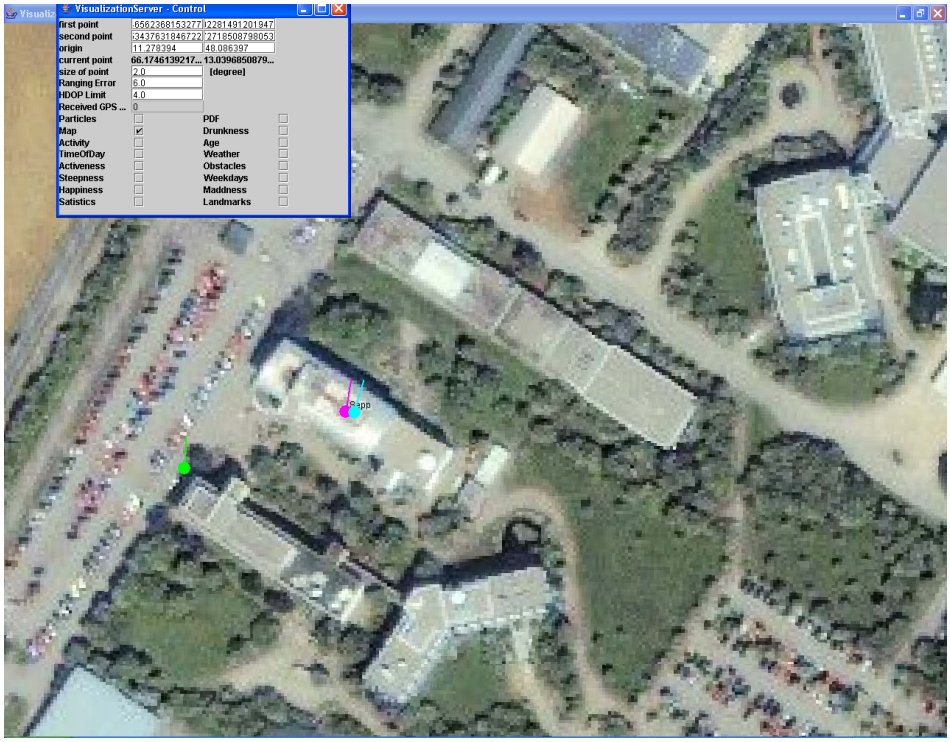

Figure 7.4: Simulated pedestrian(pink), received measurement(green) and point estimation(cyan).

direction. If a maximum likelihood (ML) estimation is required then the darkest position of the PDF should be the estimate. On the other hand, if a minimum mean square error (MMSE) estimation is required then the point estimate should be the one. If we compare these two estimations in the figure we can see that they are close to each other.

Figure (7.6) shows the simulated pedestrian, the simulated received measurements, particles cloud, PDF of location and point (hard) estimate for the pedestrian position and the direction. The qualitative results obtained from the simulations hint that the particle filter offers a reasonable approximation for Bayesian filters with a promising accuracy.

The following two figures are captured from the real time tests where a pedestrian is tracked while he is having a walk around DLR premises. They demonstrate the *exploring particles* phenomenon. This phenomenon shows that if any of the sensors fails for any reason, the particles will use the available sensors and the movement model and try to explore the area where the pedestrian might be.
Figure (7.7) shows the particle cloud behavior when the GPS receiver fails. The scenario is as follows: The GPS receiver fails after some time (manually switched off during the tests) while the compass is still on and reporting direction measurements. The particles will start from the last reported position, and using both the arriving direction measurements and the movement model to explore where most probably the pedestrian will be at the coming time steps. If the GPS receiver is switched on

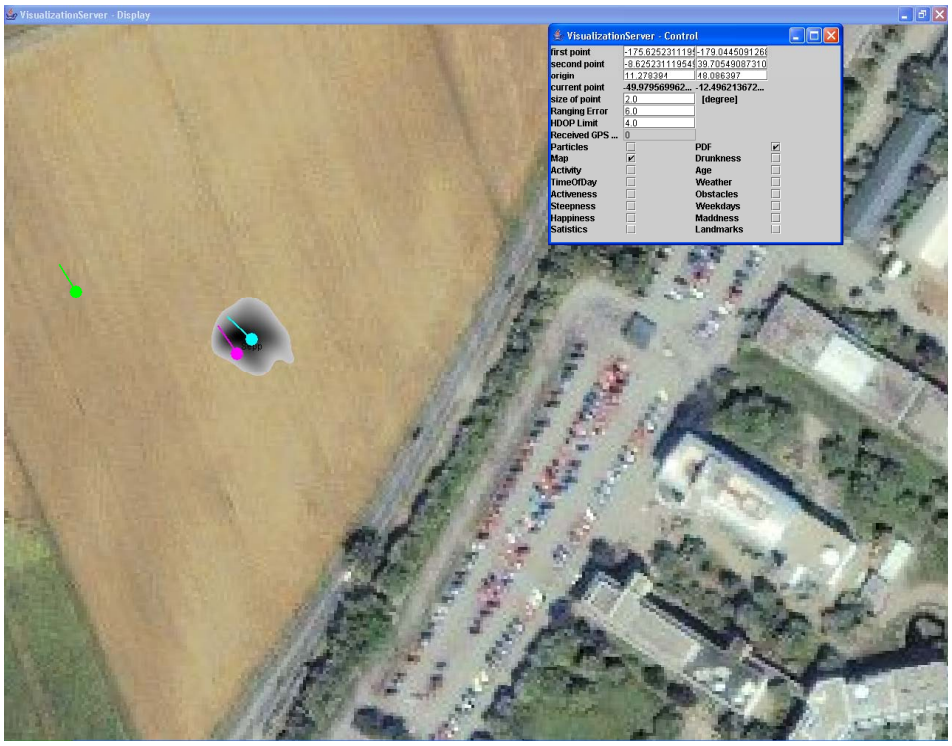

Figure 7.5: Simulated pedestrian(pink), received measurement(green), PDF of location(gray cloud) and point estimation(cyan).

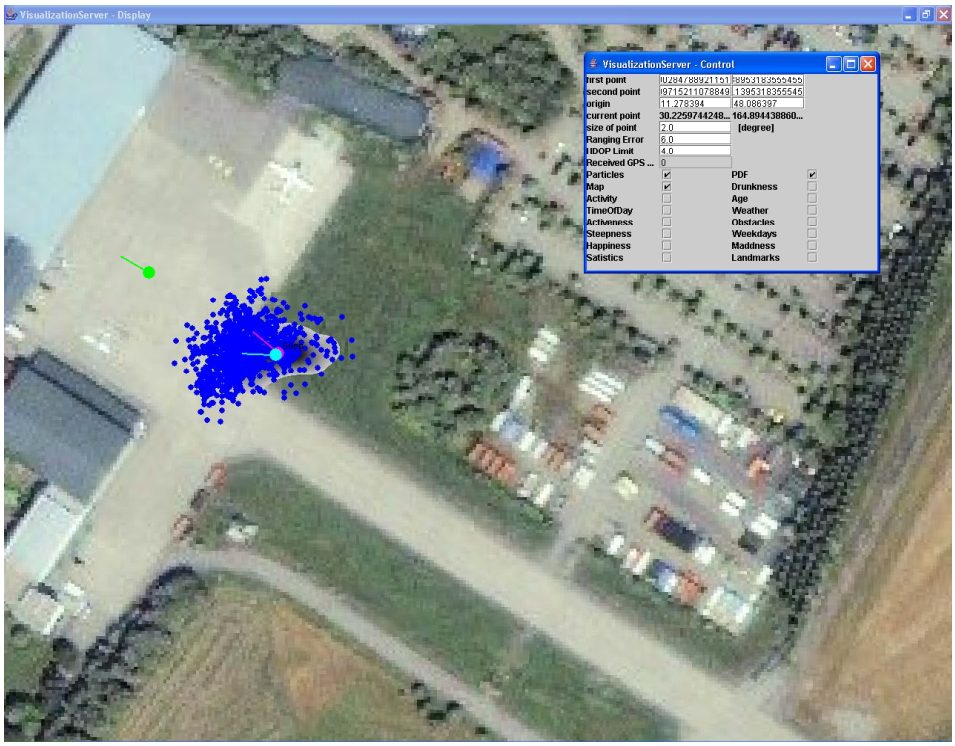

Figure 7.6: Simulated pedestrian(pink), received measurement(green), PDF of location(gray cloud), estimation(cyan) and particle cloud.

again, an update will happen and particle cloud will collapse around the position of the measurement.

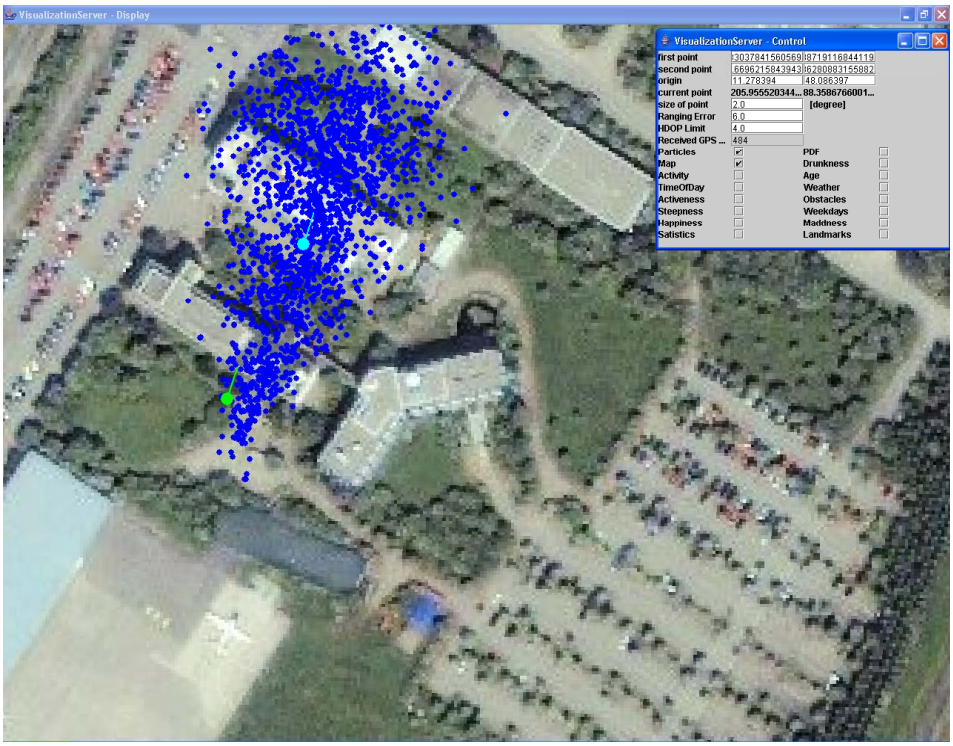

Figure 7.7: Particles cloud, measurement, estimation during a real time run when the GPS receiver fails.

Figure (7.8) shows the PDF behavior when the GPS fails. The same behavior and explanation of the particle cloud applies for the PDF.

Figure (7.9) shows the activity inference panel. This panel is captured during real time tests. The pedestrian was stopping somewhere inside the institute building. Most of the particles are also in the stopping state (1262 particles). This indicates a good accuracy of the inference and accordingly the designed movement model in chapter (3).

All the created panels for the considered eleven states in chapter (3) were tested during real time demonstrations. All the panels have shown very interesting and promising inference results.
The panels are designed in a way such that it is possible to tell the particle filter about the pedestrian situation directly. This can be imagined as if the pedestrian passes through a sensor that tells one of his states. An example of such situation aware services sensor could be an alcohol percentage sensor. Particle filtering performance and accuracy were investigated under such learning process.

In the simulation setup, inference panels are designed to allow one more testing possibility. It allows varying the simulated pedestrian states during the simulations. For example the simulated pedestrian could be set to be tired, drunk or any of the other conditions of the states during the simulations. This allows checking the inference accuracy and performance under different situation states.

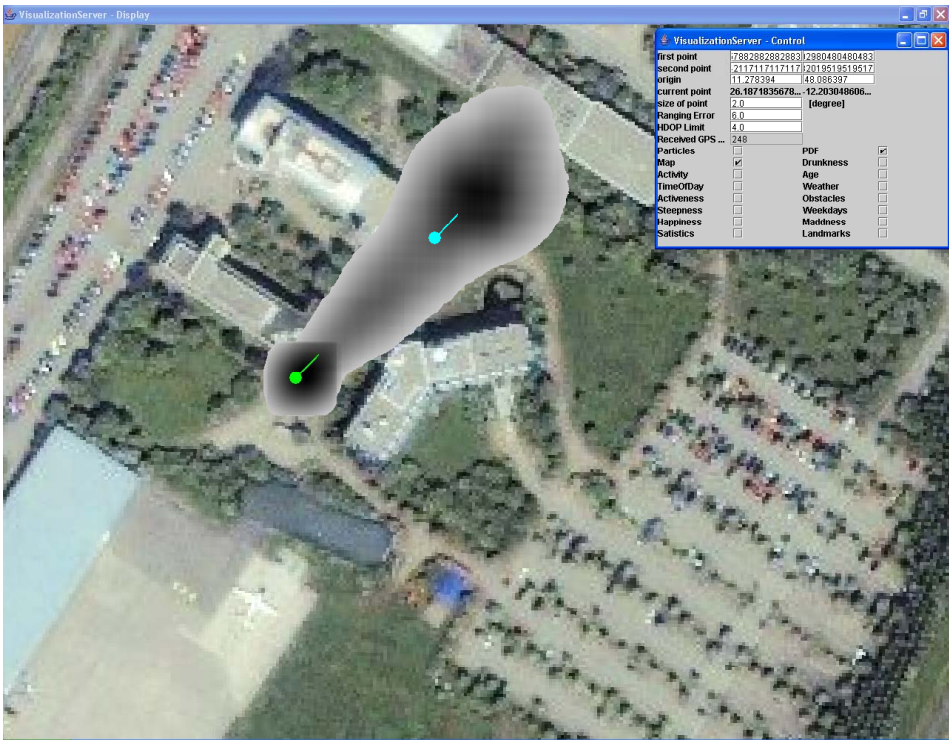

Figure 7.8: PDF, measurement, estimation during a real time run when the GPS receiver fails.

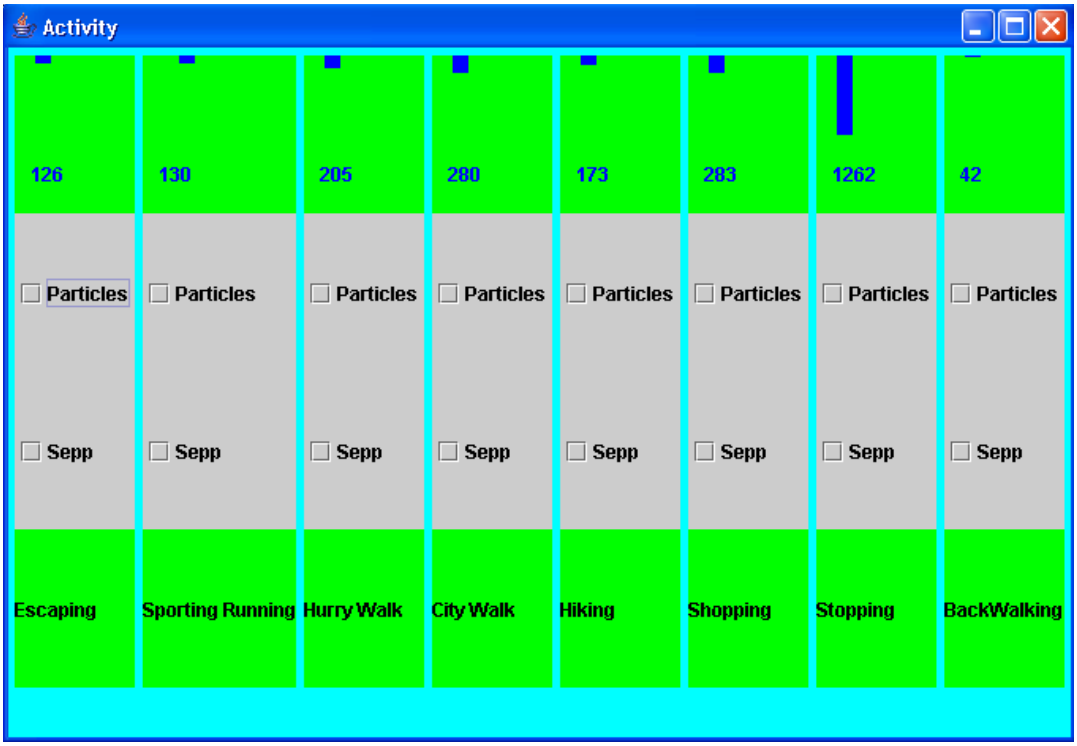

Figure 7.9: Inference panel of activity - pedestrian is stopping.

7.2 Quantitative analysis

Figures (7.10 - 7.15) include mean error curves of positions and direction and offer a quantitative measure of the particle filtering performance.

The simulation setup is used for the performance analysis of particle filters. The main reason of using it is the lack of a reference position and direction measurements in the real system. The knowledge of the reference measurements would allow numeric comparison with the estimations. Accordingly, mean errors of positions and directions could be calculated and the accuracy of particle filtering could be estimated.

As explained in section (4.4), the measurements in simulation setup are drawn from the simulated positions and directions of the pedestrian. A Gaussian noise is added to the simulated positions and directions to get the simulated measurements. Simulated position measurements are having x and y directions. Accordingly, position measurements are drawn from a two dimensional Gaussian with mean equal to the pedestrian position and with variance specified at the run time. Direction measurements are drawn from one dimensional Gaussian with mean equal to the pedestrian direction and with variance specified at the run time. For simplicity, x and y positions are assumed to be independent and accordingly, position samples are drawn from two separate Gaussian distributions.

In order to know how much improvement does the particle filter offer, mean errors should be compared with and without the particle filter. Mean errors without the filter can be calculated analytically or numerically. Measurements samples are normally distributed in both x and y directions. Accordingly, the error in any of the directions can be calculated by subtracting the measurement sample from the mean value of the Gaussian distribution in that direction (i.e. the simulated pedestrian position in that direction). The error is then calculated using Pythagorean theorem;

$$\varepsilon(k) = \sqrt{d_x^2(k) + d_y^2(k)} \tag{7.1}$$

where
d_x is the distance between the simulated measurement and the mean of the Gaussian distribution in the x direction at time k.
d_y is the distance between the simulated measurement and the mean of the Gaussian distribution in the y direction at time k.
And accordingly, the mean error will be:

$$ME = \frac{1}{N} \sum_{k=1}^{N} \varepsilon(k) \tag{7.2}$$

where N is the number of time steps.

In order to find the mean error for the case where no filtering is done, N samples are drawn from the Gaussian distribution and the ME is calculated for each sample. Then, the mean value of all these errors is evaluated. For example, if 10000 samples are drawn, and a standard deviation of 25 meters is considered, the mean error is found to be 31.3676 meters.

The pre-mentioned error evaluation technique is a numerical approach. For an analytical approach, the following well known statement is used.
As explained in [49], if two random variables $\mathbf{x}$ and $\mathbf{y}$ are normal, independent, with zero mean and equal variance, then the function:

$$\mathbf{z} = \sqrt{\mathbf{x}^2 + \mathbf{y}^2} \tag{7.3}$$

has a *Rayleigh* density. The mean of the Rayleigh distribution is $E\{\mathbf{z}\} = \sigma\sqrt{\frac{\pi}{2}}$ and variance $\sigma_z^2 = (2 - \frac{\pi}{2})\sigma_x^2 = (2 - \frac{\pi}{2})\sigma_y^2$.
Accordingly, the mean error of our Rayleigh distribution with the same previous standard deviation of 25 meters result in a mean error of $E\{\mathbf{z}\} = 25\sqrt{\frac{\pi}{2}} = 31.33285m$. It is clear that analytical and numerical solutions has led to approximately the same result.

The mean error for the case where particle filtering is used will be calculated next. A numerical approach is the appropriate method in this case, since the distribution in both directions is not known. For the selected number of time steps and the specified number of particles, a vector of mean errors is stored. Then, the mean value of this vector indicates the mean error for the specified number of particles. Finally, all the mean errors are averaged to get an indication of the performance of particle filtering. An example in which the number of particles was varied from 50 up to 10000 was considered. For each variation, 10000 errors were logged. The mean of the 10000 errors represents the average error for the specified number of particles. Running over 10000 values provides more accurate statistical analysis.
The average position errors versus the number of particles are plotted in figure (7.10). It can be observed in the figure that the mean error of position decreases as the number of particles increases. The explanation of this behavior is described in section (2.5.3). An increase of the number of particles from 50 to 2000, resulted in more than 10 meters error reduction. It can also be noticed that there is an optimal number of particles after which, no significant reduction in error was achieved. For the selected example this optimal number was 2000 particles at which the error was 9.2m. Compared to the 31.3676 meters obtained without particle filtering, more than three times improvement in location estimation was obtained due to particle filtering.

The average angle errors versus the number of particles are plotted in figure (7.11). The figure shows that the mean error of direction goes down as the number of particles increases. An increase in the number of particles from 50 to 450 particles, resulted in only 2.2° error reduction. Increasing the number of particles above 450 did not provide further significant error reduction.

Figure (7.12) shows the mean position errors for different standard deviations of the noise of the GPS receiver and fixed one of 25° for the compass. It is visible that as the standard deviation decreases, the average position error decreases also. As expected, using a more accurate GPS receiver resulted in more accurate location estimation. Additionally, in the region of large number of particles, when the noise standard deviation of the GPS receiver was decreased from 80m to 5m an improvement of only 15 meters was achieved. This means that having a larger number of particles compensates for the noisy GPS receiver since this large number will be exploring a wider area.

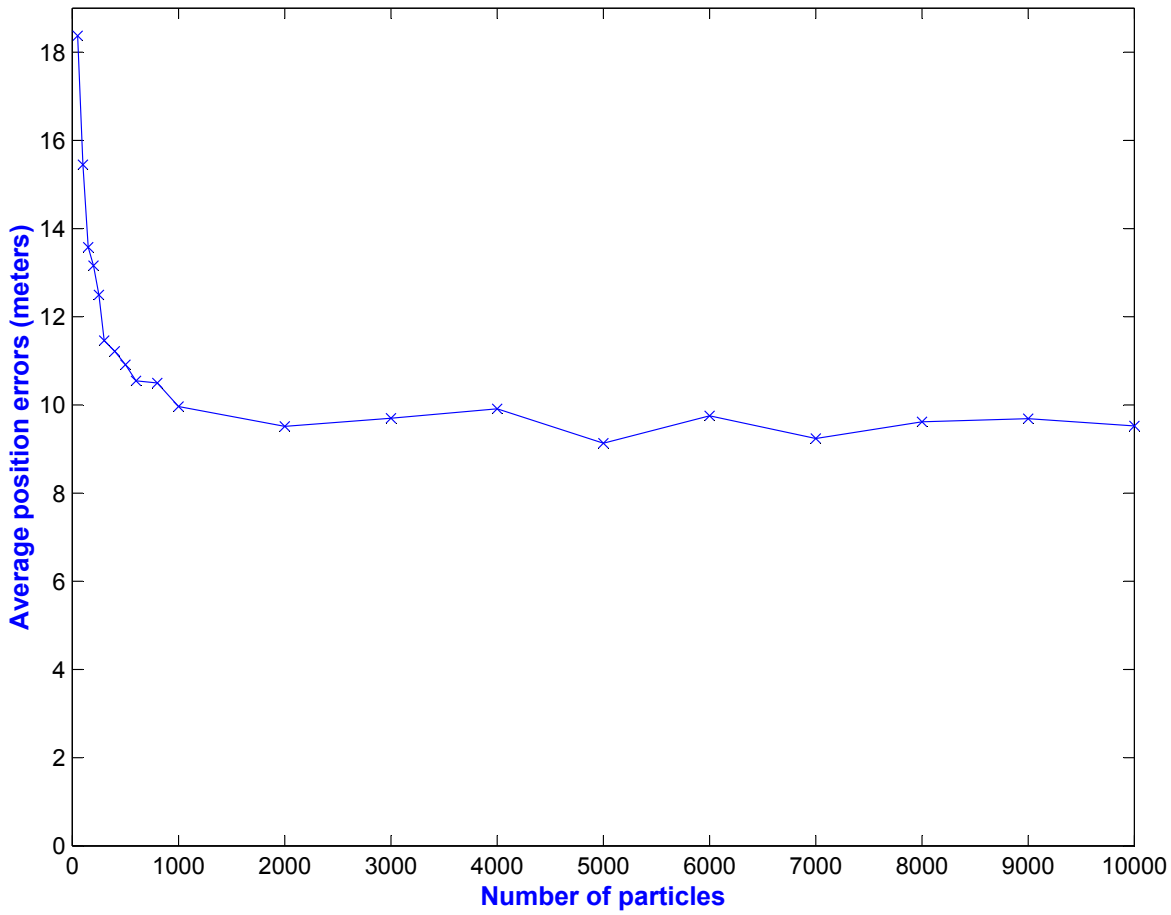

Figure 7.10: Mean position error vs. number of particles averaged over 10000 time steps, standard deviation of the compass = 20° and of the GPS receiver = 25m. An increase of the number of particles, results in a decrease in the mean error. For numbers of particles over 2000, no further significant error reduction can be achieved.

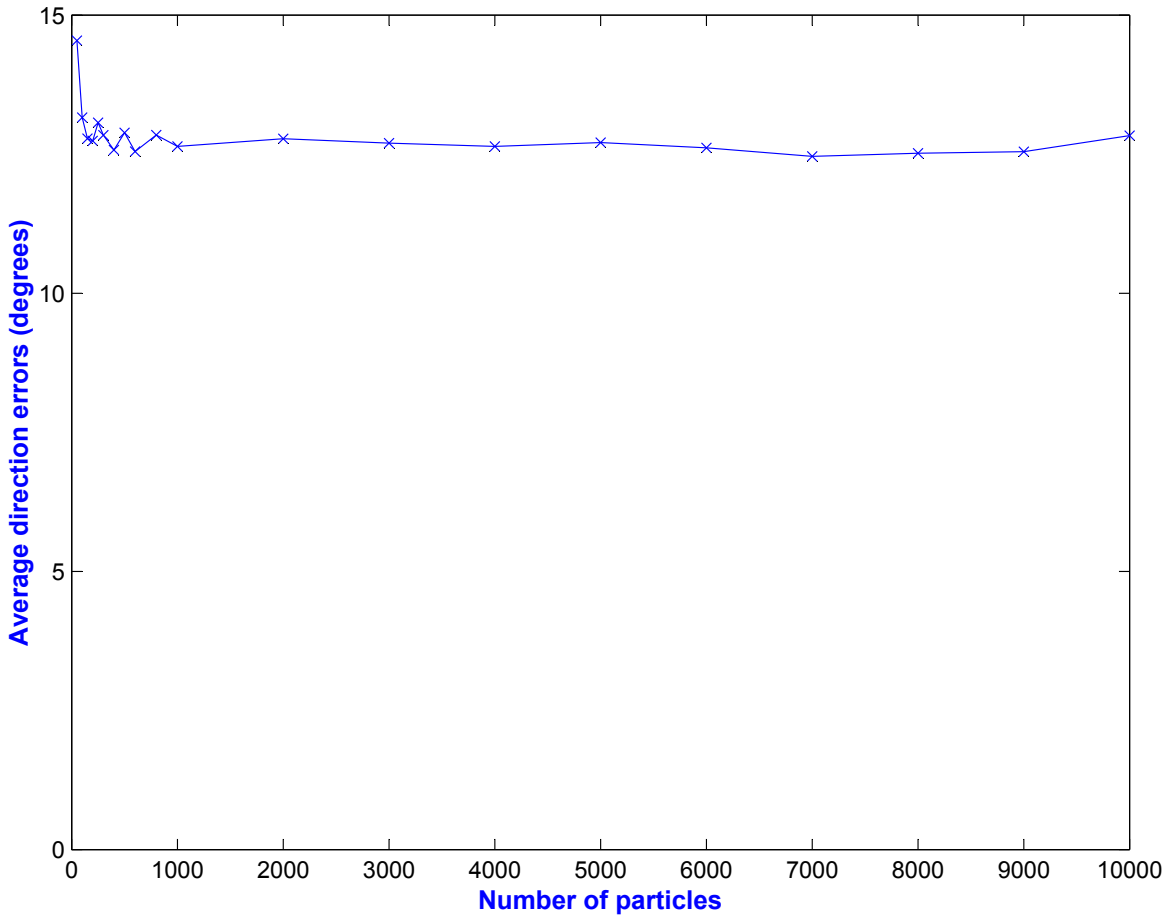

Figure 7.11: Mean angle error vs. number of particles averaged over 10000 time steps, standard deviation of the compass = 20° and of the GPS receiver = 25m. An increase of the number of particles, results in a small reduction in the mean error. For numbers of particles over 450, no further significant error reduction can be achieved.

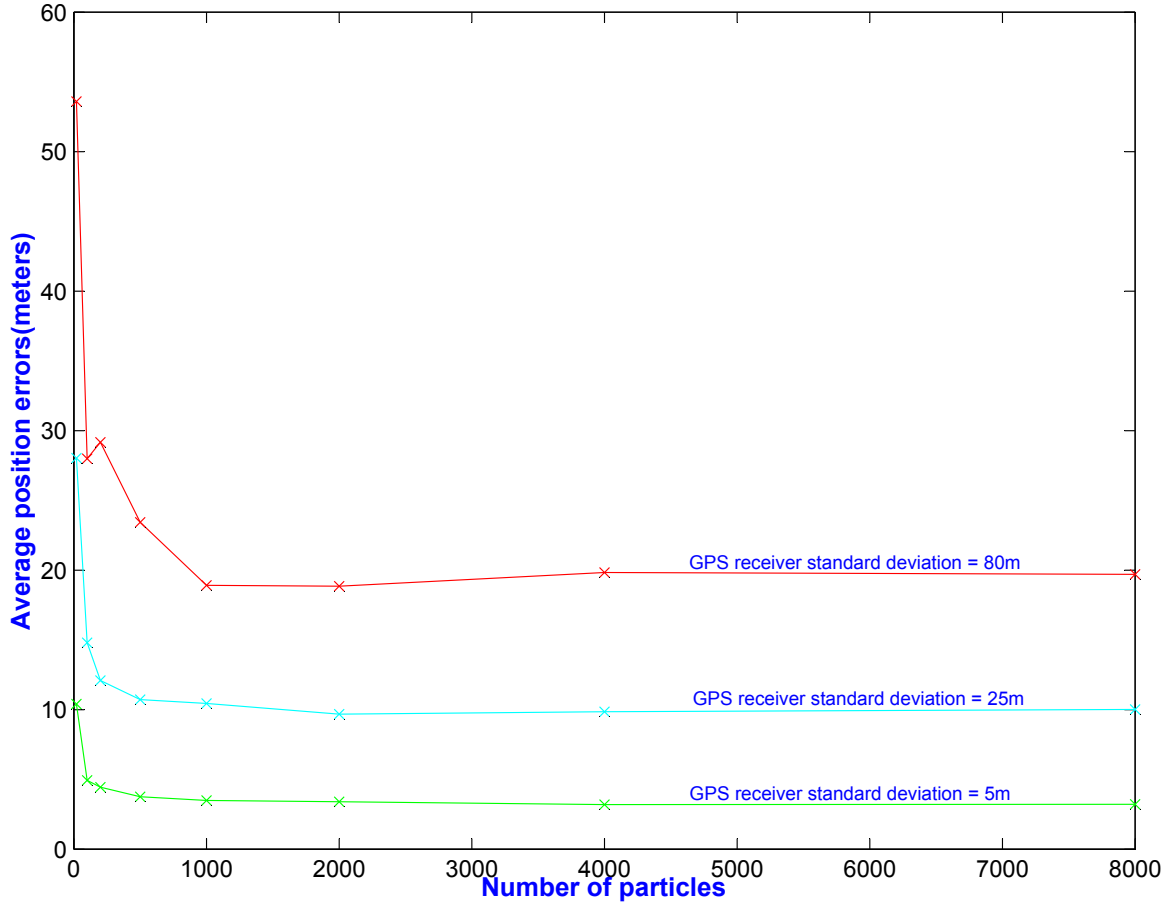

Figure 7.12: Mean position error vs. number of particles averaged over 5000 time steps, standard deviation of the compass = 25° and variable for the GPS receiver. It is obvious that as the noise standard deviation of the GPS receiver decreases, the average position error decreases also. Large number of particles compensates for having a noisy GPS receiver.

Figure (7.13) shows the mean angle errors for different standard deviations of the noise of the electronic compass and fixed one of 25m for the GPS receiver. It is visible that as the standard deviation decreases, the average angle error decreases also. As expected, using a more accurate compass results in more accurate direction estimation.

It has been found that the compass noise does not only affect the direction estimation, but it also affects position estimation. This is demonstrated in figure (7.14) which shows the mean position errors for different standard deviations of the noise of the electronic compass and fixed one of 25m for the GPS receiver. It can be observed in the figure that increasing the accuracy of the compass reduces position errors to some extent. A closer look to the curve of 0.5° standard deviation of compass noise shows that no average errors could be obtained for less than 2000 particles. The explanation of this phenomenon is as follows: due to the restricted standard deviation of the noise of the compass, few of the particles will have weights above the adjustable threshold. Only these particles will survive due to resampling. After certain number

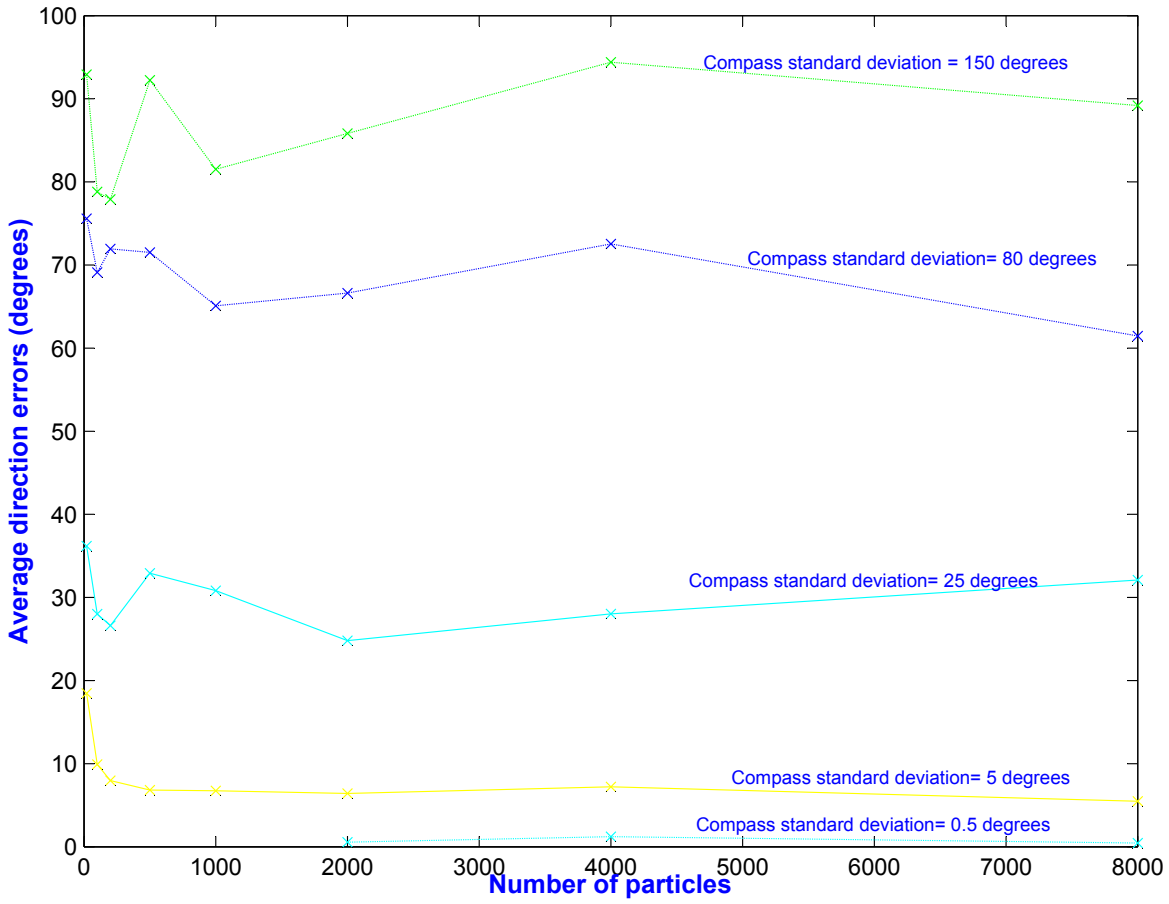

Figure 7.13: Mean angle error vs. number of particles averaged over 5000 time steps, standard deviation of the GPS receiver = 25m and variable for the compass. It is obvious that as the noise standard deviation of the compass decreases, the average direction error decreases also.

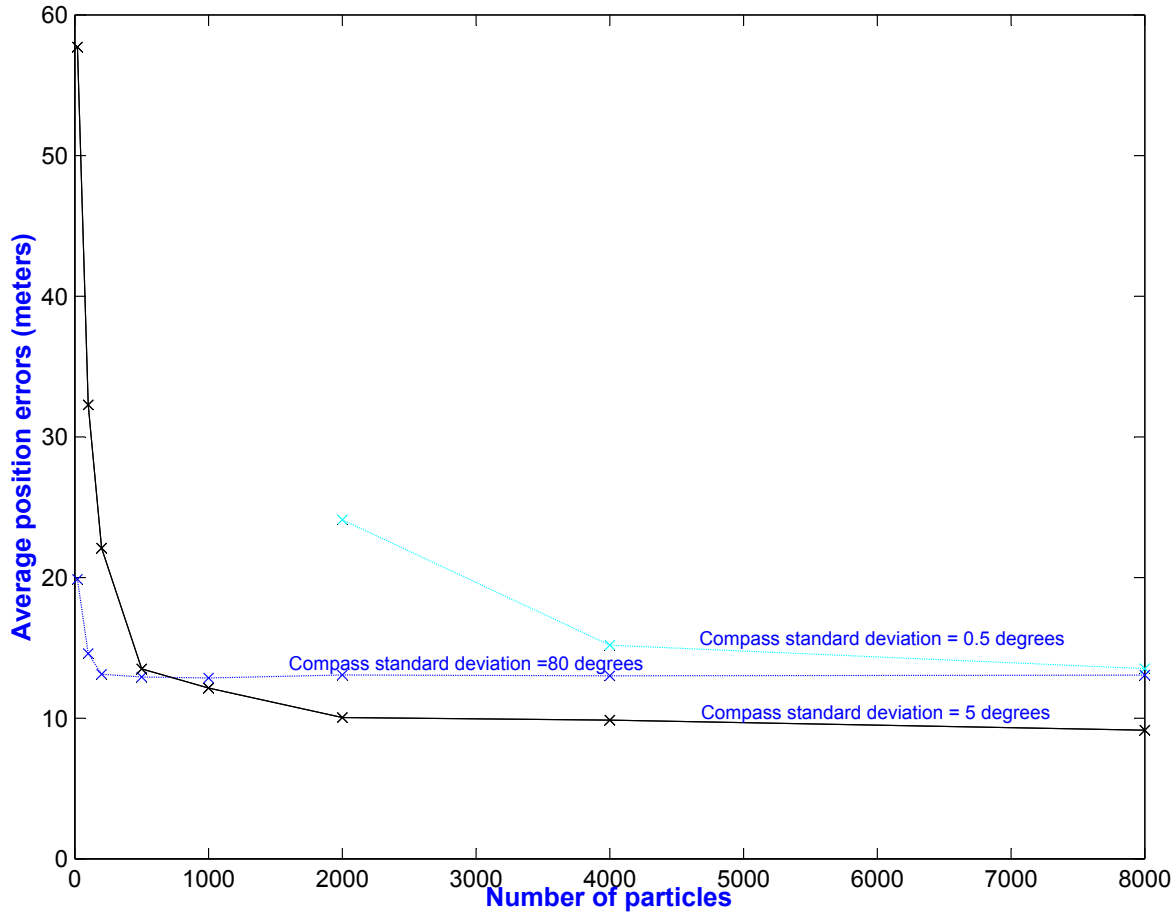

Figure 7.14: Mean position error vs. number of particles averaged over 5000 time steps, standard deviation of the GPS receiver = 25m and variable for the compass. An accurate compass results in not only accurate direction measurements, but also position measurements. More particles are needed if an accurate compass is used since it will cause many particles to vanish due to resampling.

of time steps, all the particles will be having lower weights than the threshold, and they will vanish. Above 2000 particles, there will always be particles that are having angles in this small noise variance range of the compass, and positions in the relaxed noise variance range of the GPS receiver. This is because more angles and positions ranges will be covered by the particles in the prediction stage. Accordingly, there will always be surviving particles and the algorithm will converge [1]. A costly result is the need of more particles for the error curve to reach the steady state.
If we compare the curve of the 5° standard deviation of compass noise with the 80° curve the following could be noticed:

- For a small number of particles, a non accurate compass shows better results compared to the accurate one. This is because, the 5° standard deviation compass causes many of the particles to get low weights and vanish due to resampling. On the other hand, the 80° standard deviation compass is a wider range that allow more particles to survive and accordingly shows better performance.

- At some point, with the increase of the number of particles more of them will be having angles in the 5° small noise variance range of the compass. As a result, the average error of the 5° compass will decrease till both curves intersects.

- After the intersection point, there will always be enough number of particles in this range and the 5° curve will always show less average error compare to the 80° curve.

- It takes the restricted 5° standard deviation compass curve longer to reach the steady state.

The GPS receiver noise does not only affect the position estimation, but it also affects the direction estimation. This is demonstrated in figure (7.15) which shows the mean angle errors for two standard deviations of the noise of the GPS receiver and fixed one of 25° for the compass. It can be observed from the figure that increasing the accuracy of the GPS receiver results in reduction in the angle errors. If we compare the two curves in figure (7.15), we can see that for small number of particles the less accurate GPS receiver performs better and vice versa for large number of particles. Same explanations and observations that were given when comparing the 5° and the 80° curves in figure (7.14) are also valid here.

[1]The restricted standard deviation of the compass of 0.5° makes the compass like a restricted maths teacher who fails any student who does not really knows everything. On the other hand, the unrestricted standard deviation of the noise of GPS receiver of 25m makes the GPS receiver like a relaxed English teacher who requires less effort to pass his exam. With less than 2000 particles, there will be no enough of them to pass the two exams. While above 2000 particles, there will always be some of them that are perfect in math and just ok in English.

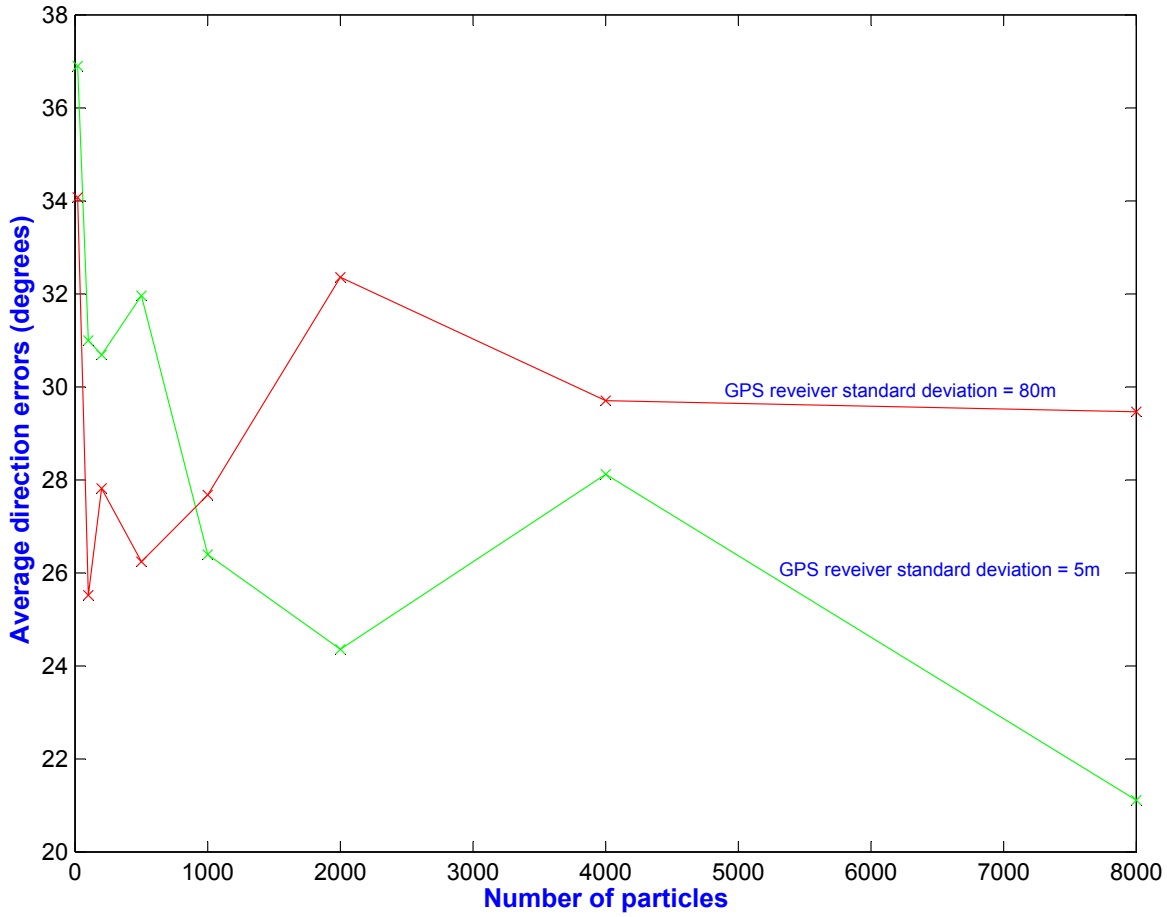

Figure 7.15: Mean angle error vs. number of particles averaged over 5000 time steps, standard deviation of the compass = 25° and variable for the GPS receiver. An accurate GPS receiver results in not only accurate position measurements, but also direction measurements. More particles are needed if an accurate GPS receiver is used since it will cause many particles to vanish due to resampling.

Chapter 8

Conclusions and Outlook

8.1 Conclusions

Bayesian filtering has been presented during this work as a general framework for location estimation. Our focus was on particle filtering as a selected approximation for cases where optimal Bayesian solution is intractable. This work has proven theoretically and from experimental results that particle filter is a good and reliable approximation for Bayes filter.

The applications of Bayesian filtering goes beyond location estimation. The generation of hierarchial models allows the seamless integration of location estimation into user activity estimation. During this work inference was done to estimate the situation of the pedestrian from his movement. Inference of many of the pedestrian situations was successfully shown. Bayesian techniques are considered to be extremely promising tool for situation aware services.
The main advantages of particle filtering are as follows:

- Particle filtering is able to represent arbitrary density.
- Particle filtering has no problem dealing with multi-modal distributions.
- Particle filtering converges to a true posterior even for non-Gaussian and nonlinear systems.
- Particle filtering is efficient in the sense that particles tend to focus on regions with high probability.
- Particle filtering is easy to formulate and easy to implement.

On the other hand particle filtering has two disadvantages. First, the complexity of the particle filter algorithm grows exponentially with the dimensions of the state space. Second, particle filtering is computationally expensive. However this computational burden could be handled by implementing particle filtering using parallel processors.

8.2 Outlook

Over the last few years, there has been a proliferation of scientific papers on particle filters and their application. This has opened the door for a wide range of possible further work in the direction of theocratical research and implementation.

Implementation wise more sensors like RF IDs, Bluetooth, WLANs, etc. are going to be added to the current implemented sensors. Other types of particle filters are going to be implemented in order to compare their accuracy and performance. Additionally, different particle filters are going to be combined with other Bayesian filters in order to get the advantage of each and avoid the disadvantages.
Some of the research directions are as follows:

- **High-level Representations**: The location of the pedestrian provides only very limited information about the person's current activity for the provider in order to improve their quality of service. Richer representations might include information such as the time of the day, the weather, the activity, the disorientation, the age, the mode of transportation, the destination of the current trip and the purpose of a specific location. During this work inference was done to estimate pedestrian situation from his movement in order to provide the service providers with more information. However some questions remain: What are important locations in a person life? How can they be described in a general way and learned from sensor data? How can we transfer experience gained from one person to another person? *Relational probabilistic models* [50], which can represent relations between classes of objects, provide a promising framework for addressing these problems.

- **Adaptive estimation**: Most current Bayes filters use the same, fixed representation of state space during the entire estimation process. However, in the context of location estimation, this is not appropriate. For example if GPS and compass, combined with a street-map were enough for the particle filter to estimate the pedestrian location accurately in an urban area, they will not be enough for sure in an indoor environment. Furthermore, even within the same building, different areas might be covered by completely different types of sensors requiring different representations of likelihoods. A key question is thus when and how to switch between different representations in a statistically sound way.

- **User Errors**: If the system learns the pedestrian behavior in some situations, then the offered service will be unreliable incase of unusual pedestrian's attitude. *Online model selection* [11] is a technique that can potentially solve this problem. Model selection aims at identifying the model that is best suited to explain the observed data. To apply model selection in the location context, one could generate generic and user specific Bayes models of activities. Both models are able to track a user's activities, but the specific model is tuned towards the typical actions of one particular user. The specific model additionally contains all errors that are typical for the user. The idea is that as long as the user

performs his usual activities, the tuned model will be much better in predicting these activities. Surprising actions, i.e. potential errors, however, are not well predicted by the specific model, in which case the generic model receives higher probability. For example if the pedestrian is having lunch everyday at the same time, then the specific model predicts this action with very high probability. If the pedestrian didn't has lunch at the usual time, then the general model predict it with higher probability, thereby triggering the detection of a potential user error. Another example is in the context of assisting cognitively impaired people, where the detection of when a person seems to be lost is an important aspect of location estimation. Obviously, such an approach can provide valuable information to user intervention modules.

This is in addition to wide range of other research directions in Bayesian filters and location based services.

Appendix A

Statistics tables

Activity	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Escaping	250	old direction	50	45
Sporting (Running)	200	old direction	34	15
Hurry walk	110	old direction	6	15
City walk	82	old direction	13	15
Hiking	75	old direction	8	15
Shopping	40	old direction	13	50

Table A.1: Activity statistics

Disorientation (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Sober	82	old direction	13	15
Half drunk	68	old direction	17	300
Totally drunk	43	old direction	22	600

Disorientaton (Running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Sober	200	old direction	34	15
Half drunk	180	old direction	37	400
Totally drunk	120	old direction	40	700

Table A.2: Disorientation statistics

Time of day (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Morning	70	old direction	16	20
Midday	82	old direction	13	15
Night	90	old direction	10	13

Time of day (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Morning	220	old direction	45	17
Midday	200	old direction	34	15
Night	170	old direction	39	13

Table A.3: Time of day statistics

Weather (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Very cold	100	old direction	10	15
Nice	82	old direction	13	15
Very hot	60	old direction	16	15

Weather (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Very cold	250	old direction	30	15
Nice	200	old direction	34	15
Very hot	160	old direction	38	15

Table A.4: Weather statistics

Activeness (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Tired	45	old direction	15	15
Just Ok	82	old direction	13	15
Active	105	old direction	11	15

Activeness (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Tired	120	old direction	38	15
Just Ok	200	old direction	34	15
Active	280	old direction	29	15

Table A.5: Activeness statistics

Obstacles (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
No obstacles	82	old direction	10	15
Quarter obstacles	72	old direction	11	22
Half obstacles	62	old direction	12	30
3Q obstacles	52	old direction	13	37
Full obstacles	42	old direction	14	45

Obstacles (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
No obstacles	200	old direction	32	15
Quarter obstacles	183	old direction	35	22
Half obstacles	165	old direction	37	30
3Q obstacles	149	old direction	39	37
Full obstacles	130	old direction	42	45

Table A.6: Obstacles statistics

Steepness (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Flat	82	old direction	13	25
Half steep	110	old direction	10	20
Tottaly Steep	140	old direction	7	15

Steepness (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Flat	200	old direction	34	20
Half steep	300	old direction	25	15
Tottaly Steep	450	old direction	15	10

Table A.7: Ground steepness statistics

Weekdays (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Saturday	70	old direction	16	20
Tuesday	82	old direction	13	15
Friday	90	old direction	10	13

Weekdays (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Saturday	220	old direction	45	17
Tuesday	200	old direction	34	15
Friday	170	old direction	39	13

Table A.8: Weekdays statistics

Emotions (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Sad	60	old direction	17	14
Normal	82	old direction	13	20
Happy	100	old direction	10	28

Emotions (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Sad	140	old direction	32	13
Normal	200	old direction	30	15
Happy	260	old direction	28	17

Table A.9: Emotions statistics

Arousal (walking)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Calm	60	old direction	17	14
Neutral	82	old direction	13	20
Angry	100	old direction	10	28

Arousal (running)	Mean speed (m/min)	Mean direction (degrees)	Sigma speed (m/min)	Sigma direction (degrees)
Calm	140	old direction	32	13
Neutral	200	old direction	30	15
Angry	260	old direction	28	17

Table A.10: Arousal statistics

Appendix B

Statistics figures

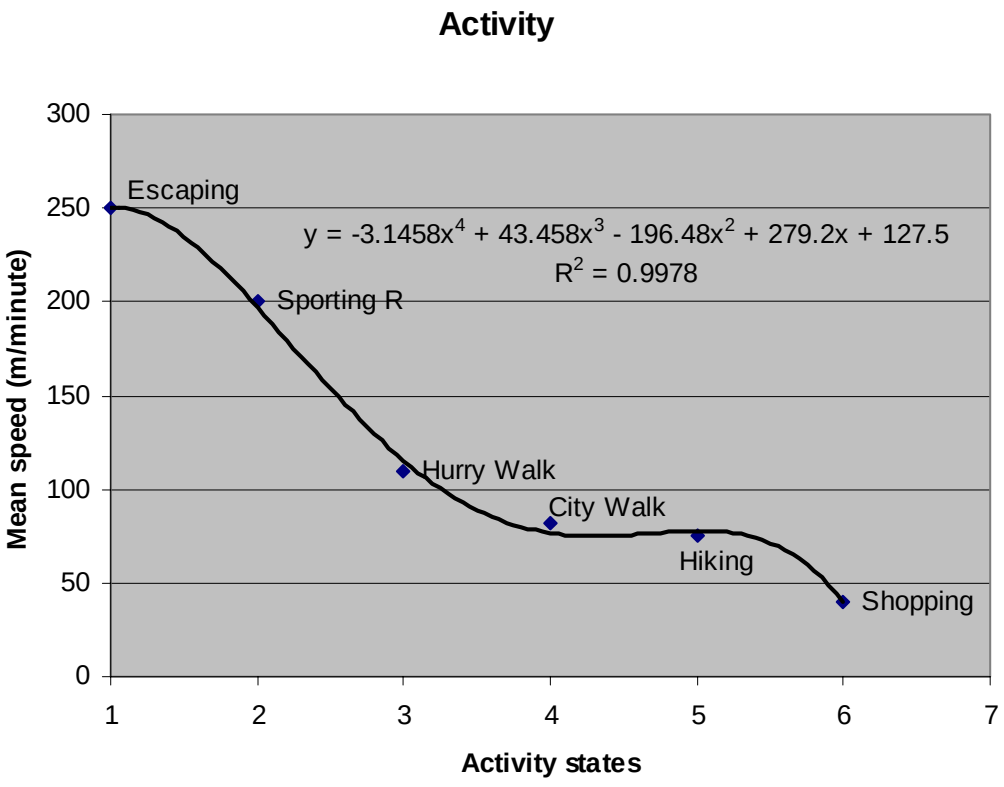

Figure B.1: Mean Speed as a function of the activity

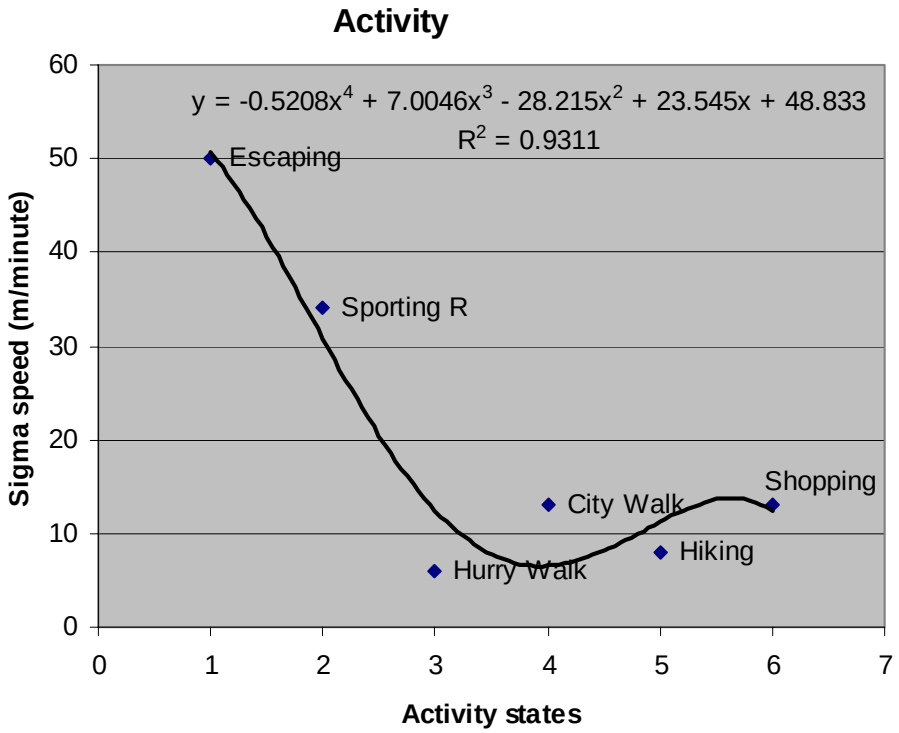

Figure B.2: Sigma speed as a function of the activity

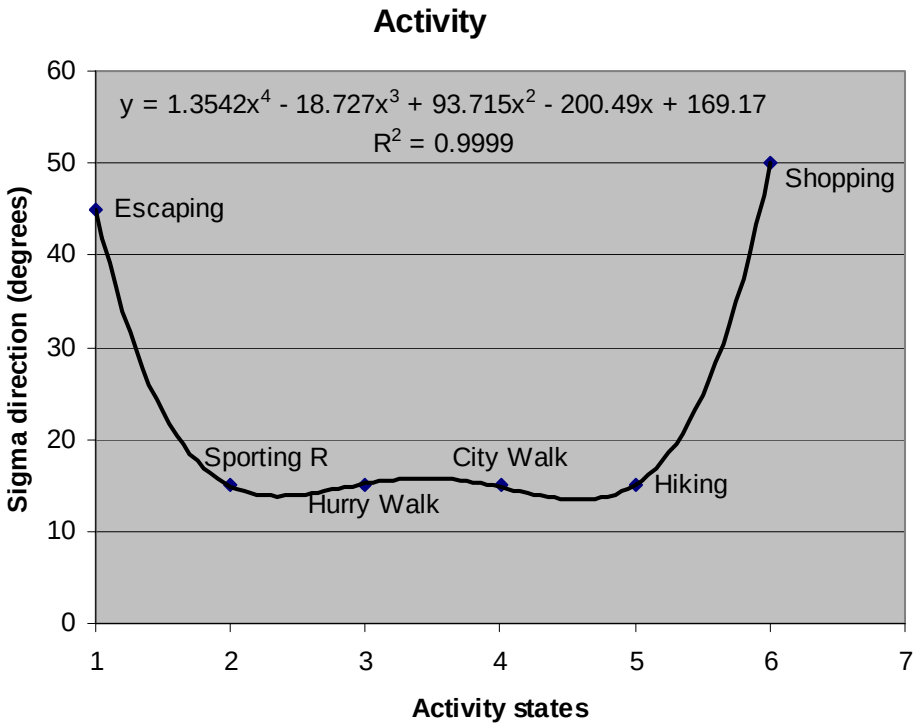

Figure B.3: Sigma direction as a function of the activity

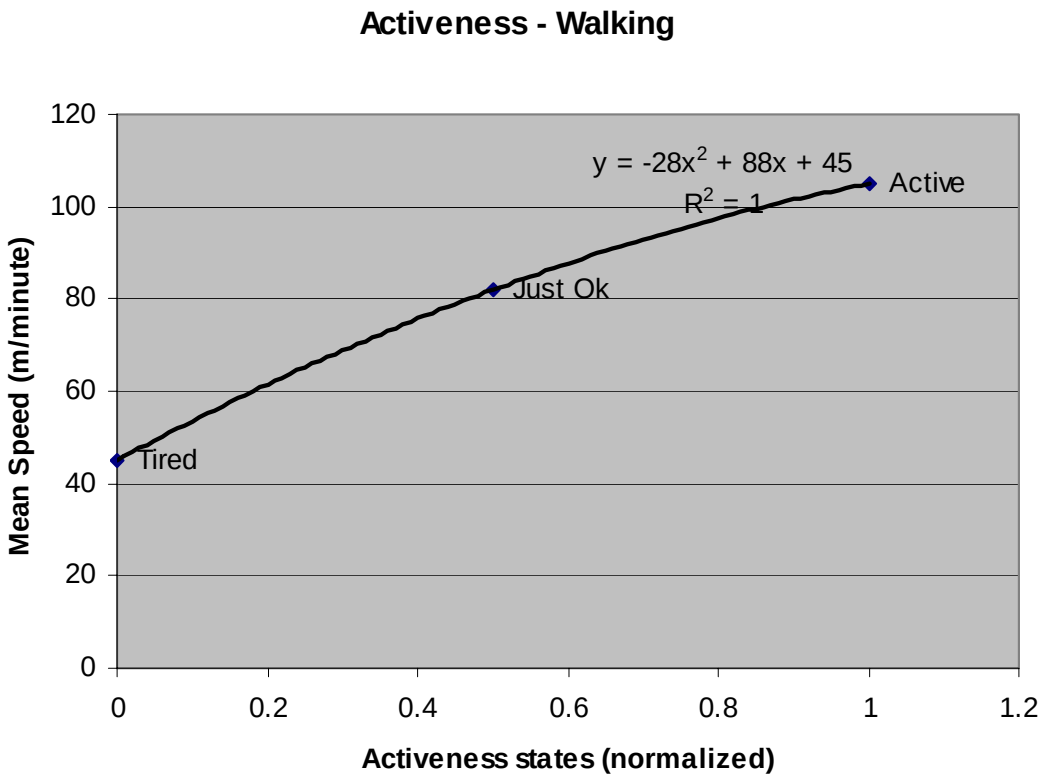

Figure B.4: Mean speed as a function of the activeness during walking

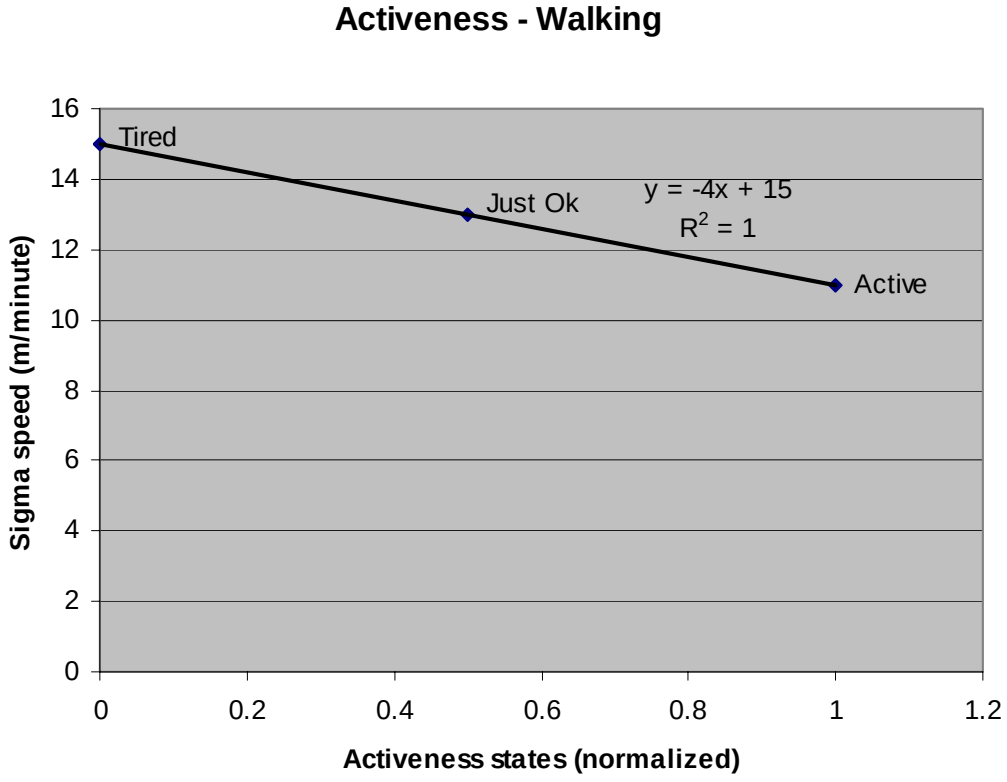

Figure B.5: Sigma speed as a function of the activeness during walking

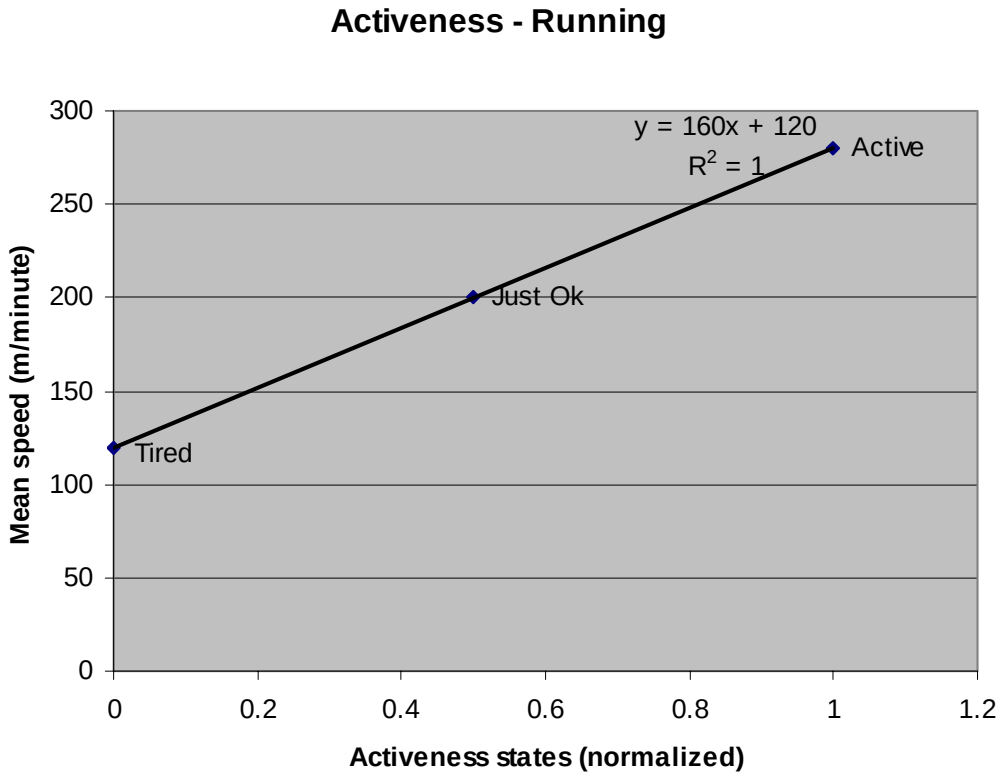

Figure B.6: Mean speed as a function of the activeness during running

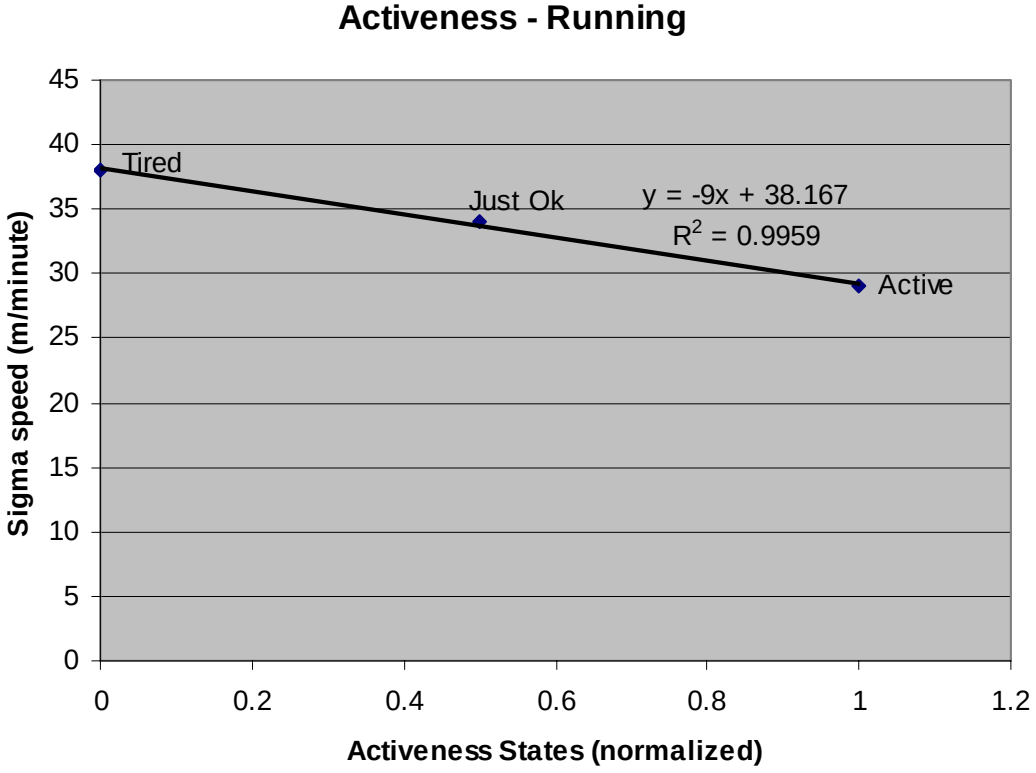

Figure B.7: Sigma direction as a function of the activeness during running

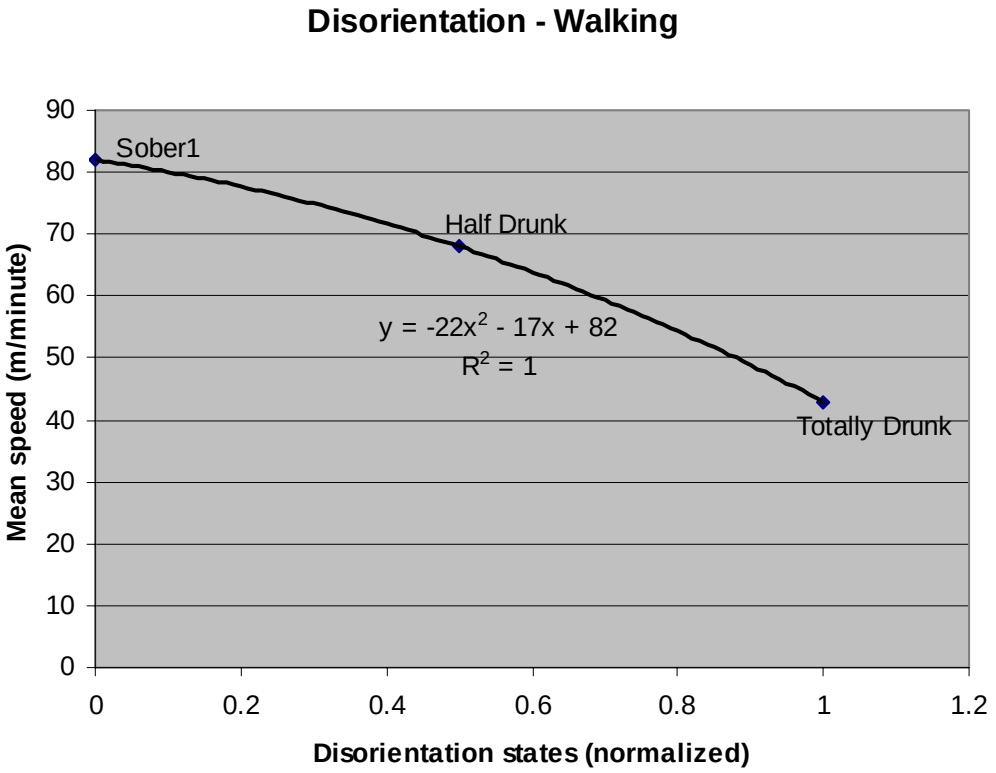

Figure B.8: Mean speed as a function of the disorientation during walking

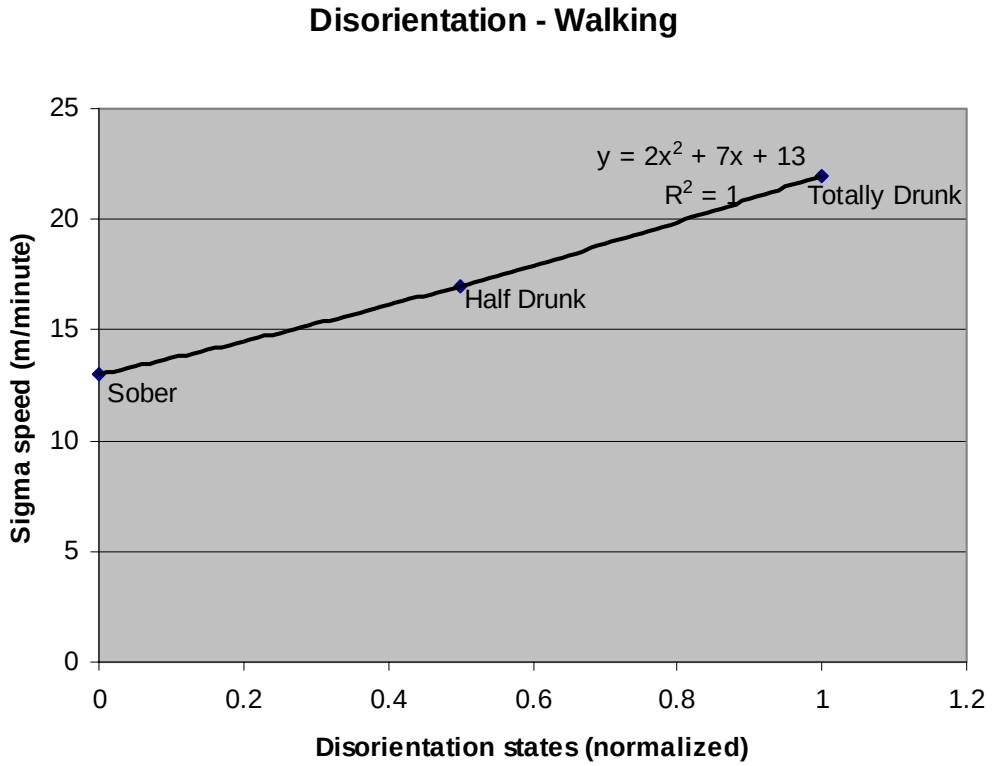

Figure B.9: Sigma speed as a function of the disorientation during walking

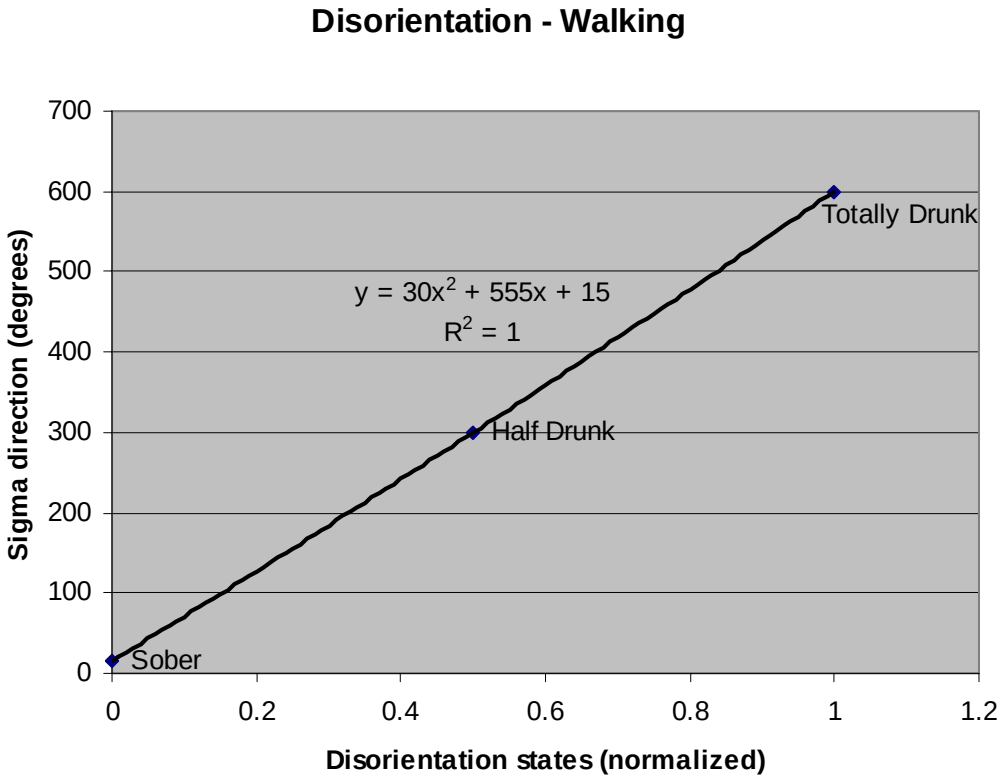

Figure B.10: Sigma direction as a function of the disorientation during walking

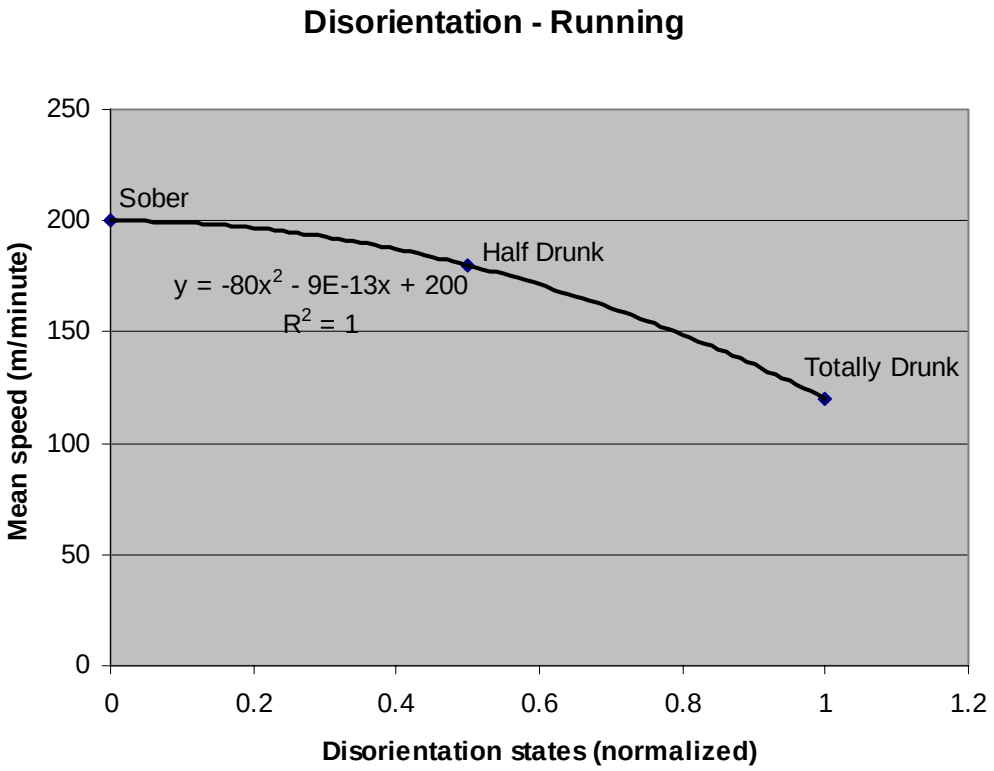

Figure B.11: Mean speed as a function of the disorientation during running

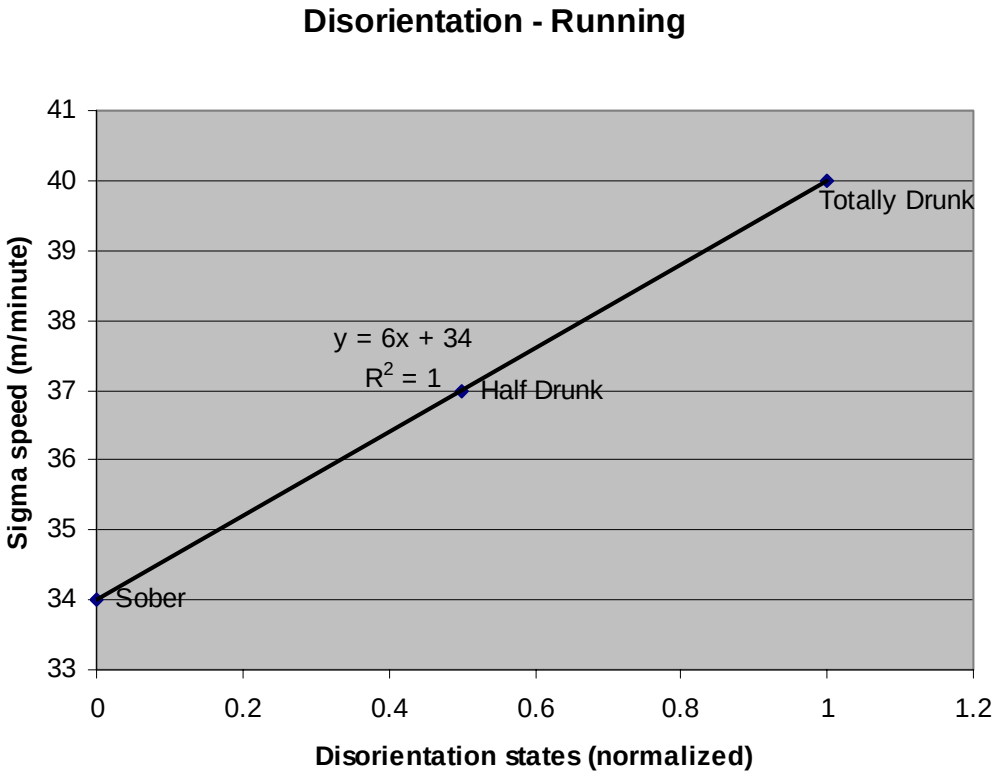

Figure B.12: Sigma speed as a function of the disorientation during running

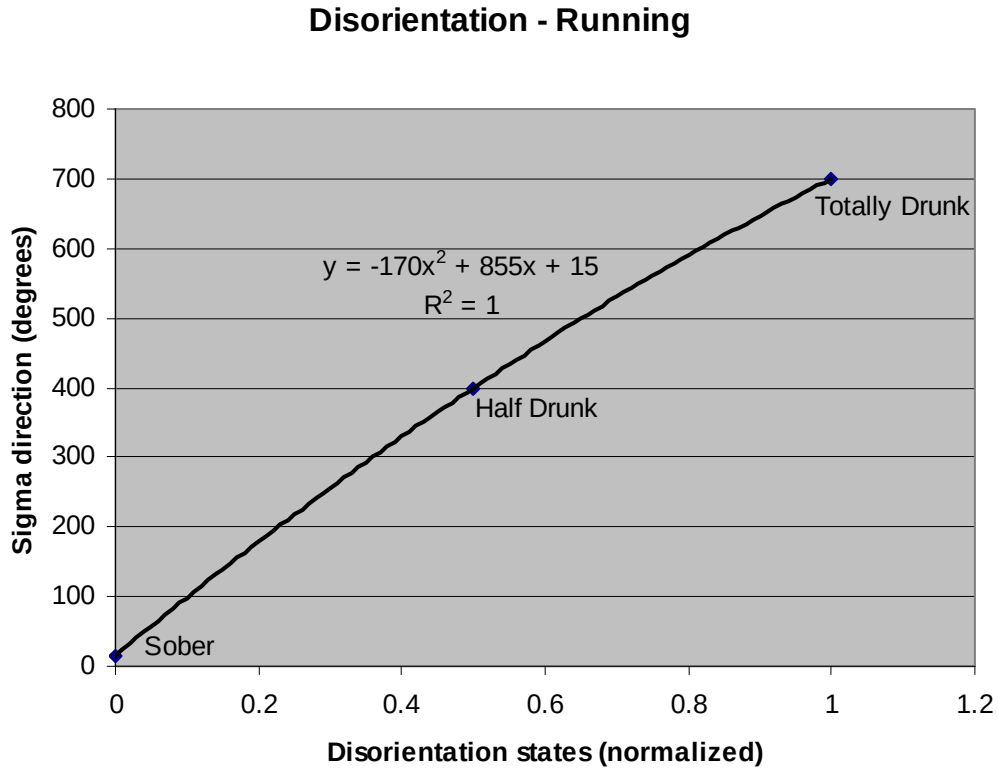

Figure B.13: Sigma direction as a function of the disorientation during running

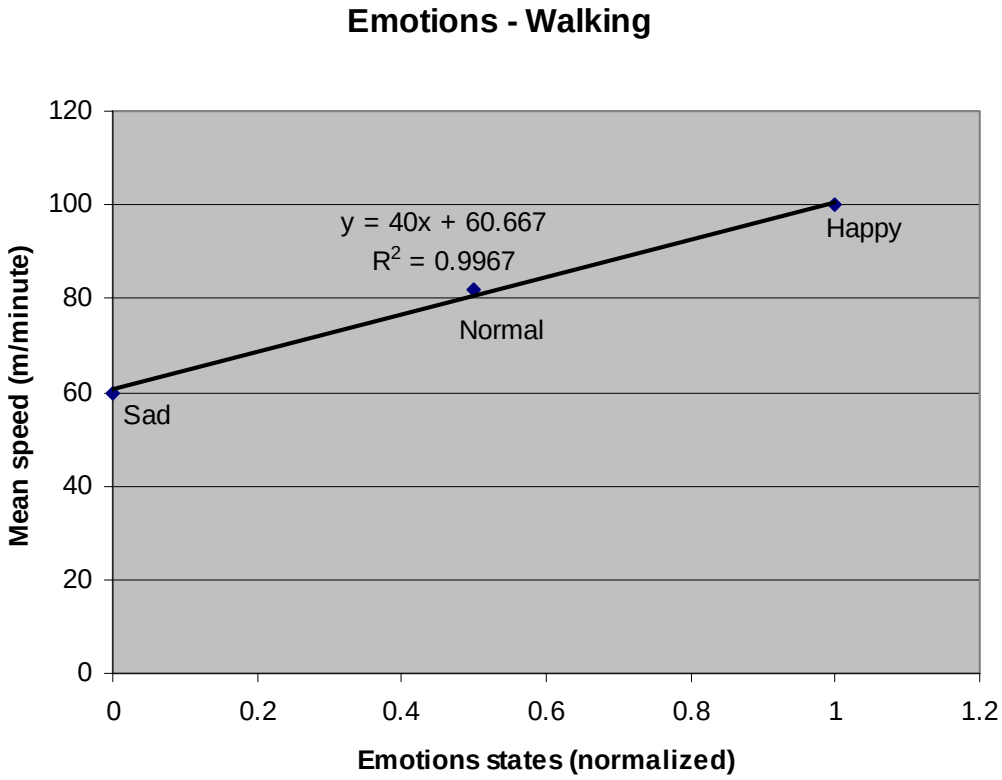

Figure B.14: Mean speed as a function of the emotions during walking

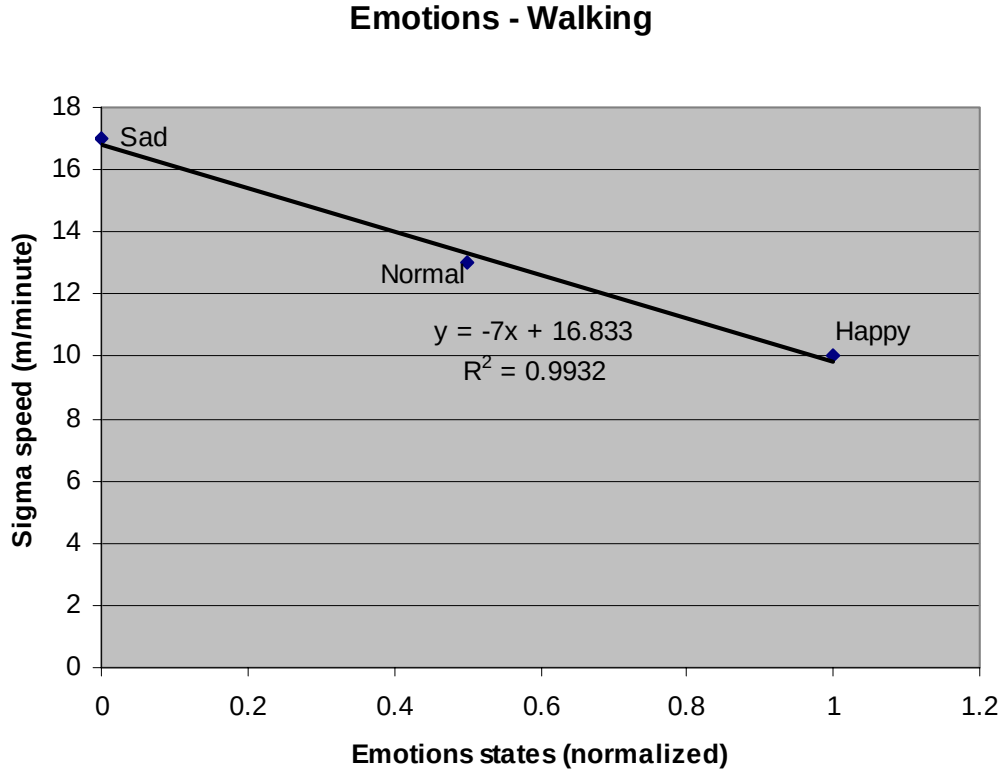

Figure B.15: Sigma speed as a function of the emotions during walking

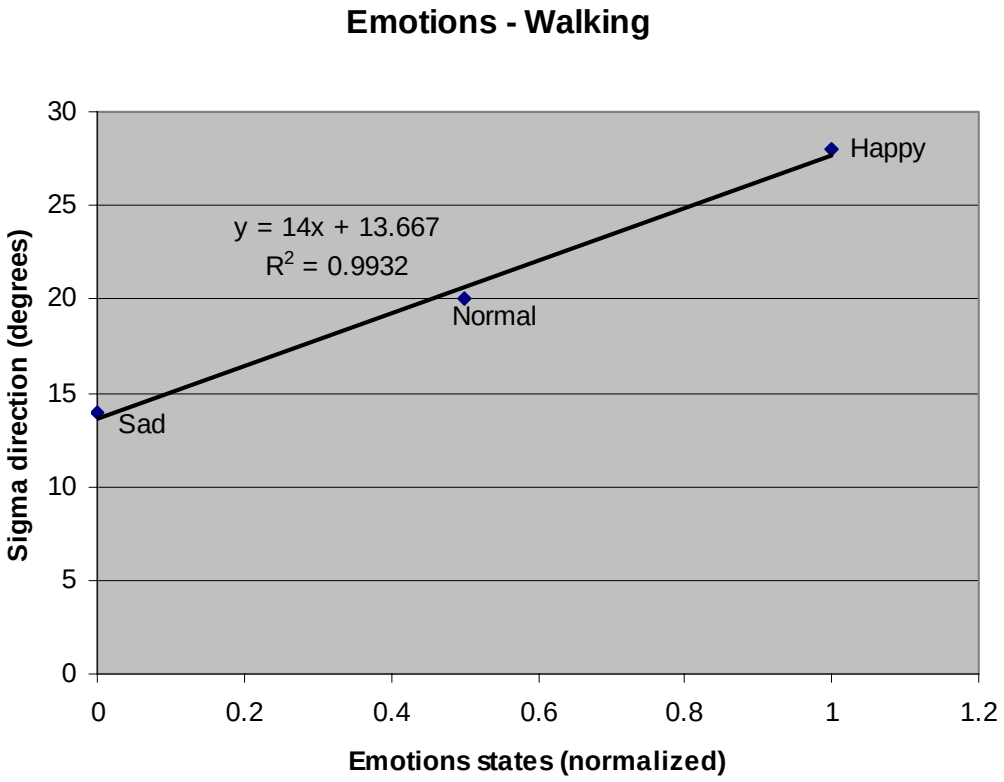

Figure B.16: Sigma direction as a function of the emotions during walking

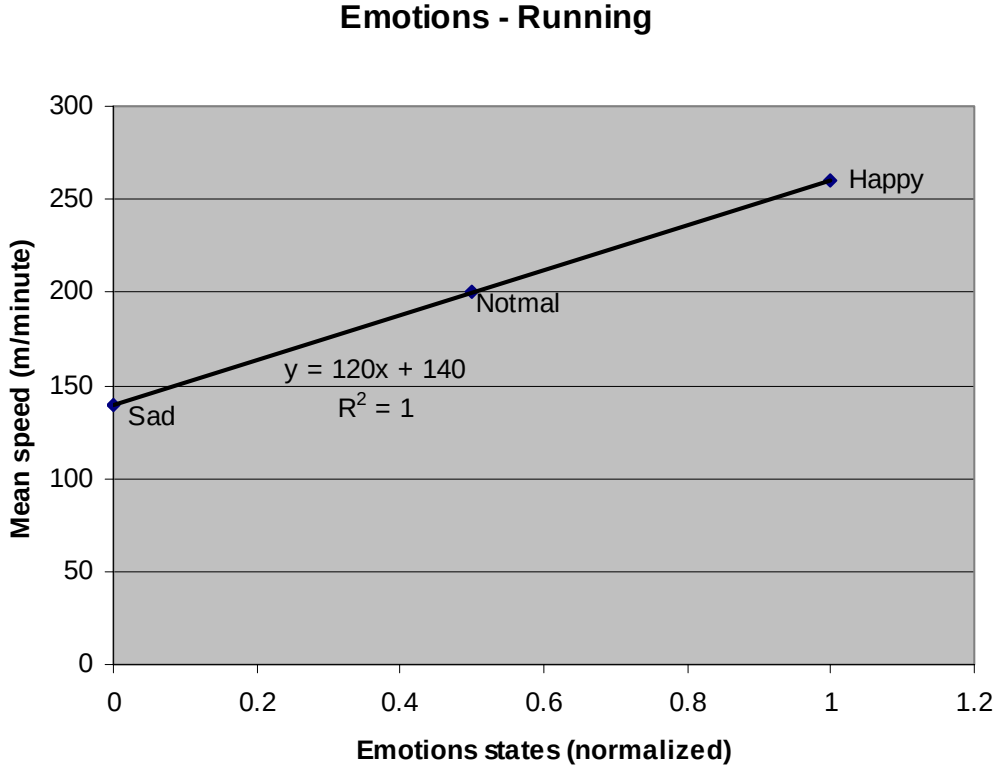

Figure B.17: Mean speed as a function of the emotions during running

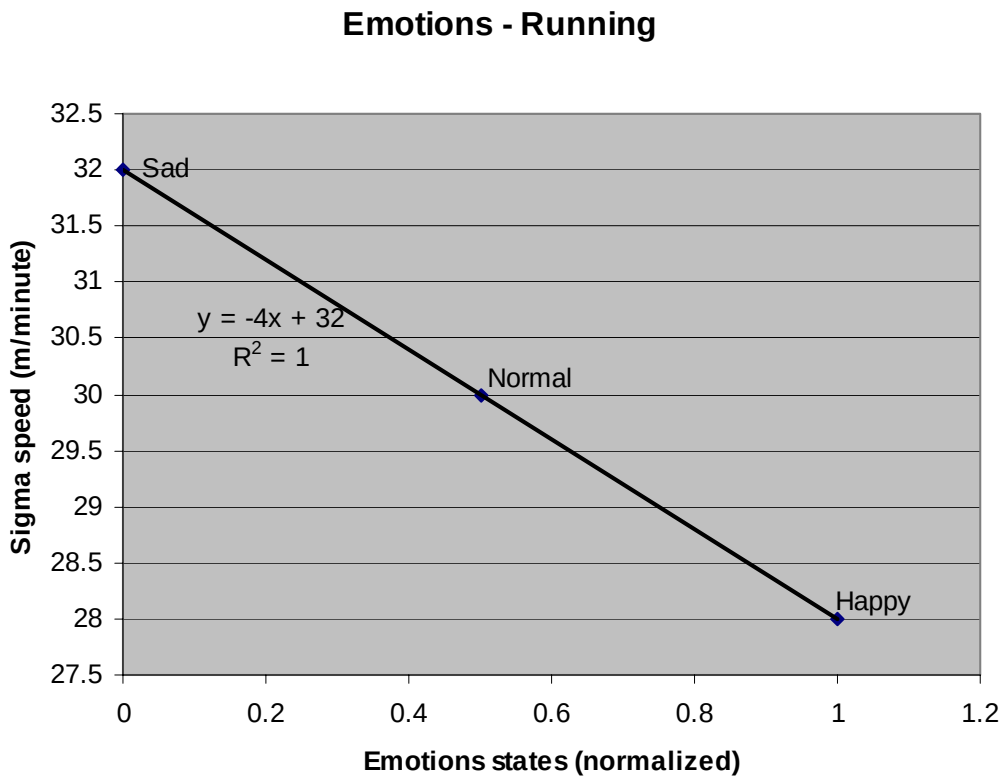

Figure B.18: Sigma speed as a function of the emotions during running

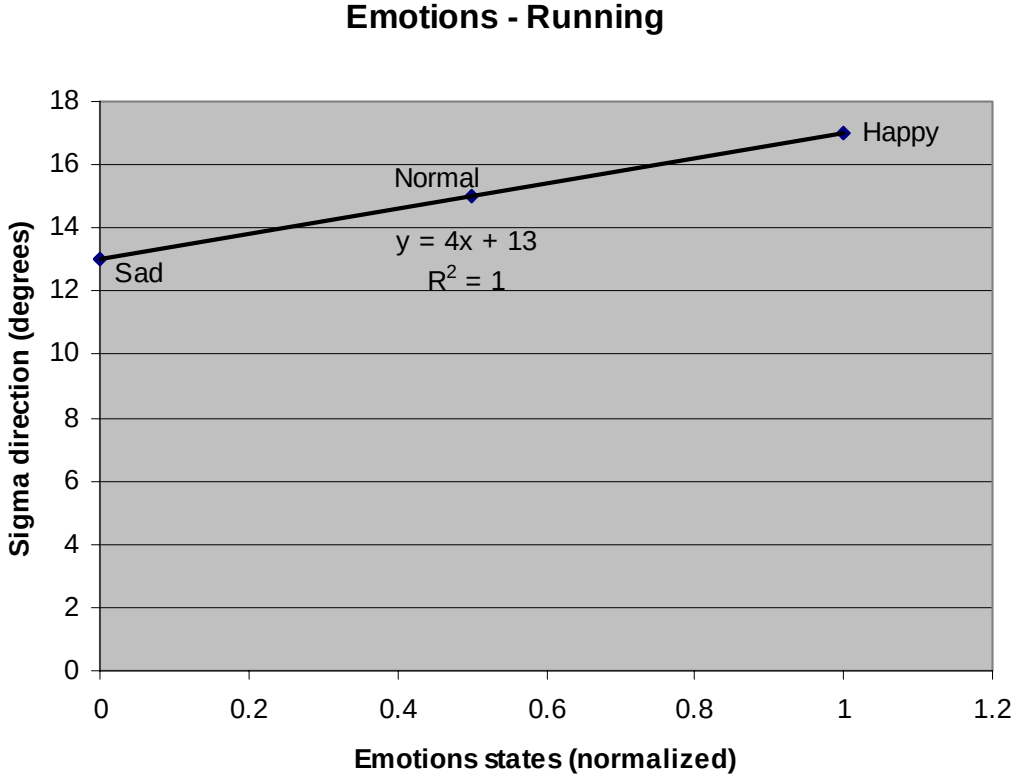

Figure B.19: Sigma direction as a function of the emotions during running

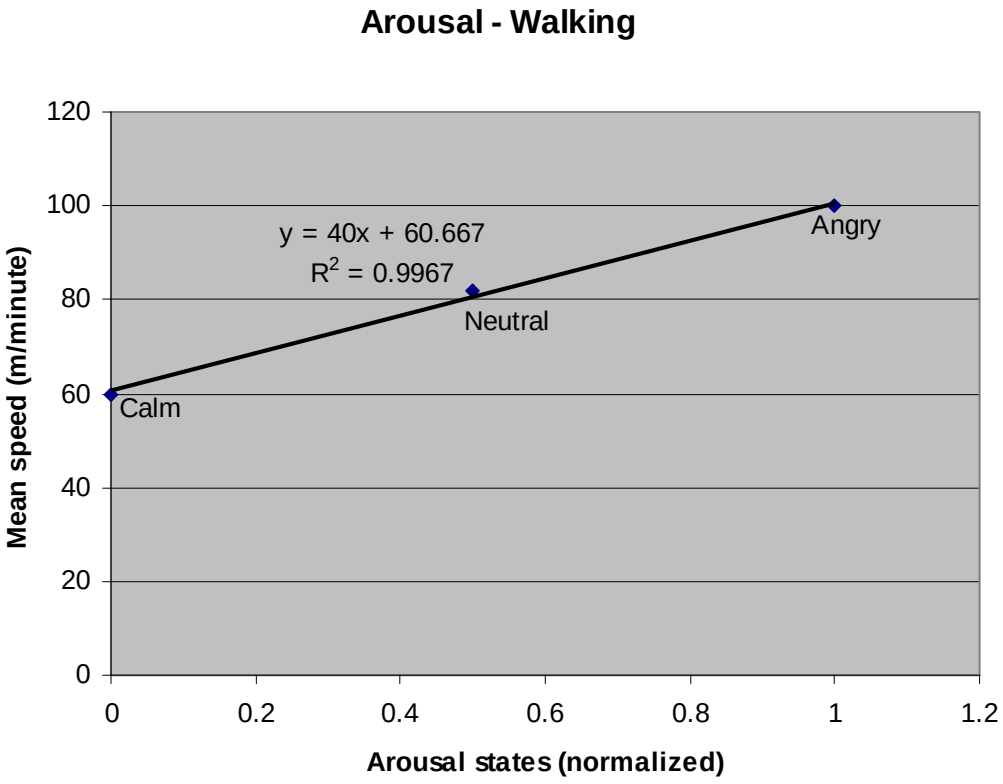

Figure B.20: Mean speed as a function of the arousal during walking

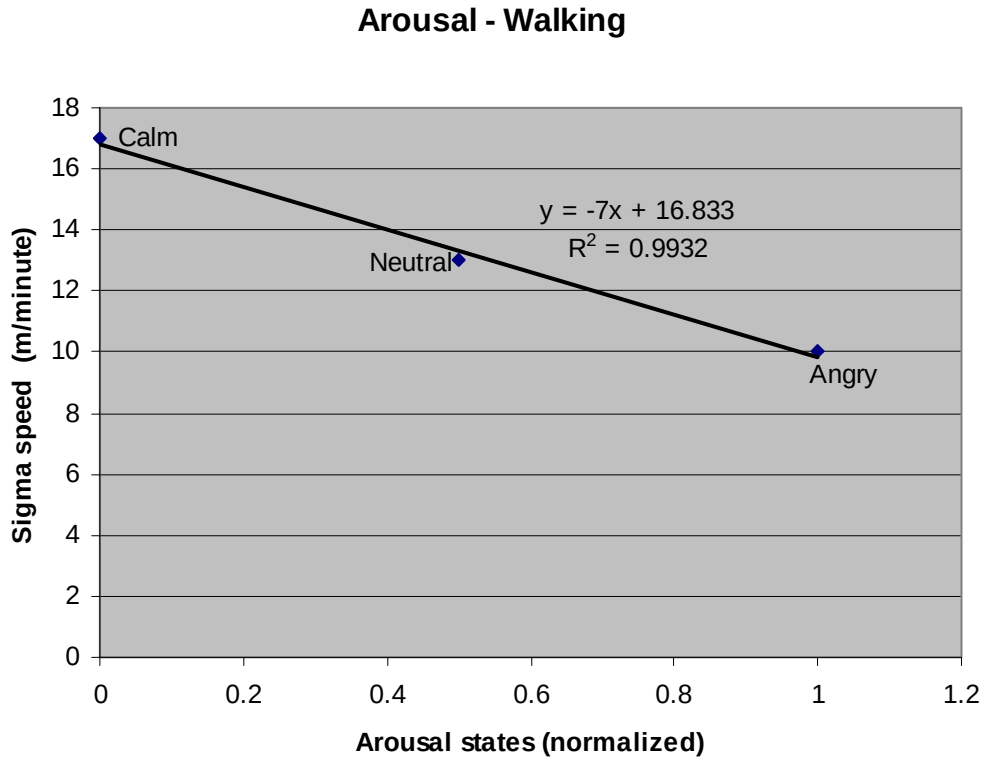

Figure B.21: Sigma speed as a function of the arousal during walking

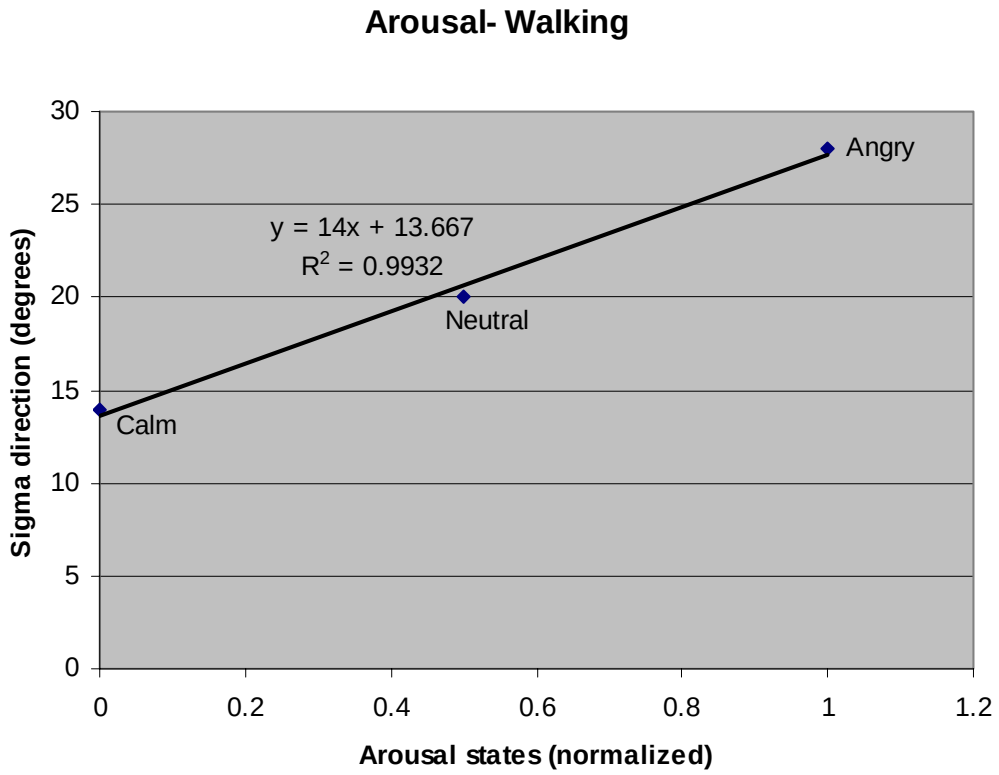

Figure B.22: Sigma direction as a function of the arousal during walking

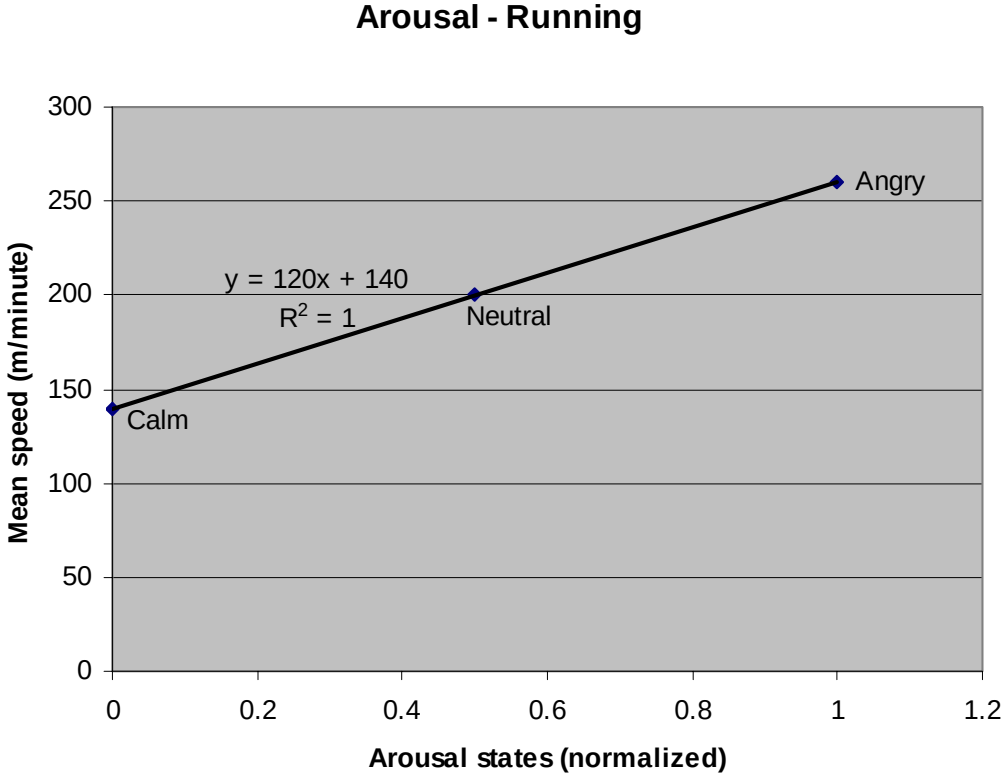

Figure B.23: Mean speed as a function of the arousal during running

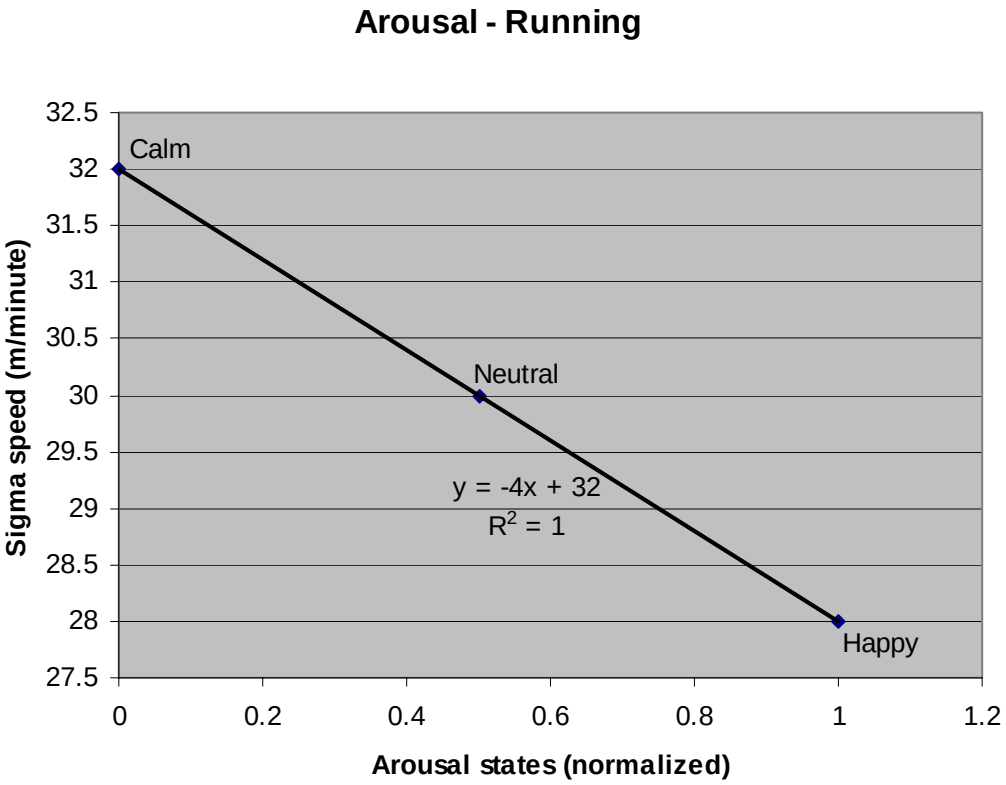

Figure B.24: Sigma speed as a function of the arousal during running

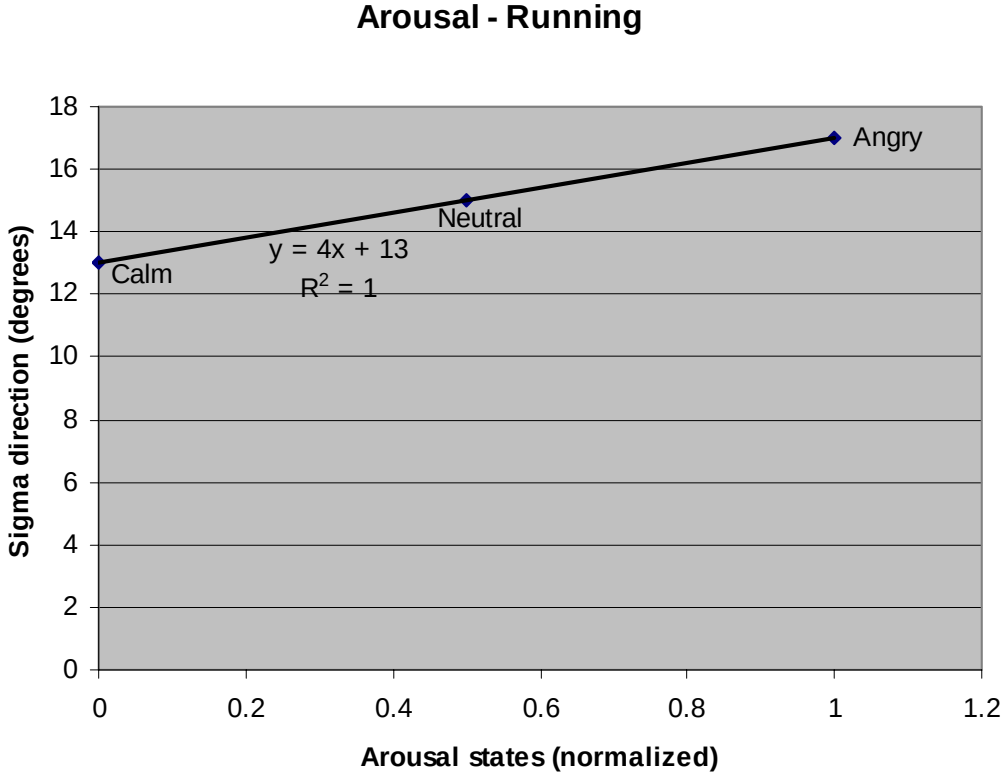

Figure B.25: Sigma direction as a function of the arousal during running

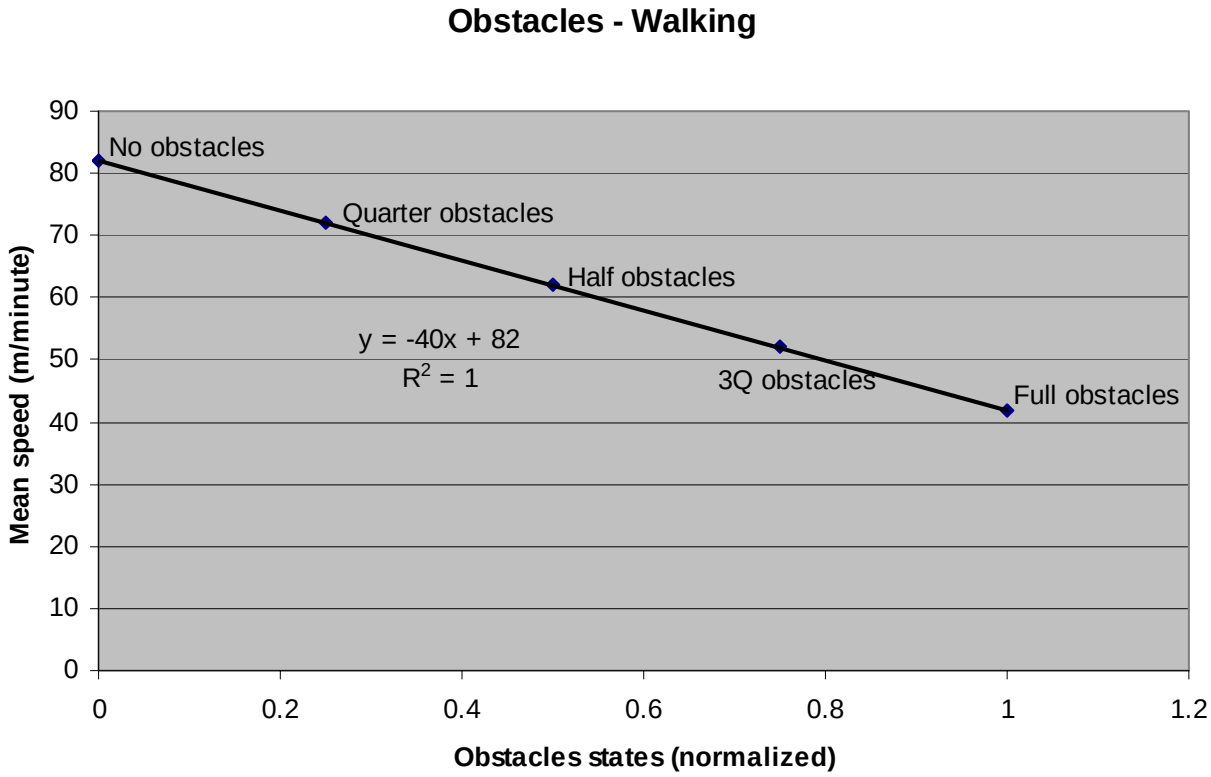

Figure B.26: Mean speed as a function of the obstacles during walking

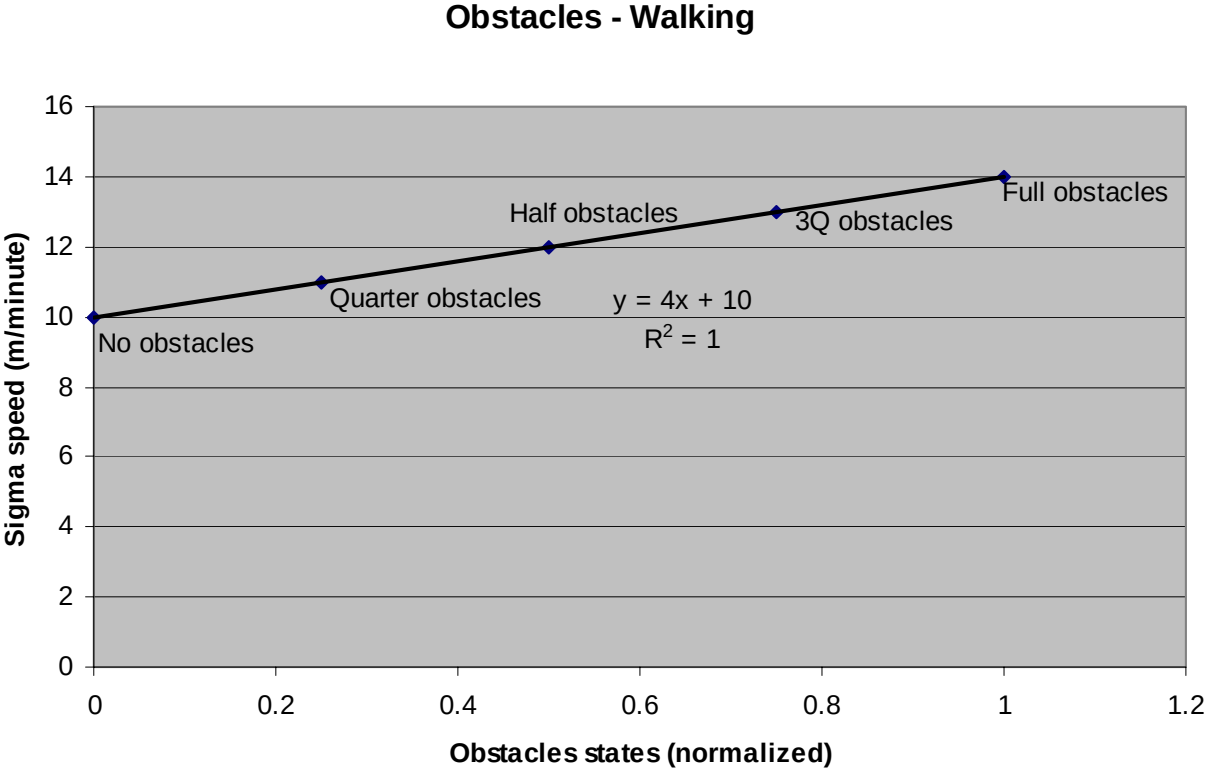

Figure B.27: Sigma speed as a function of the obstacles during walking

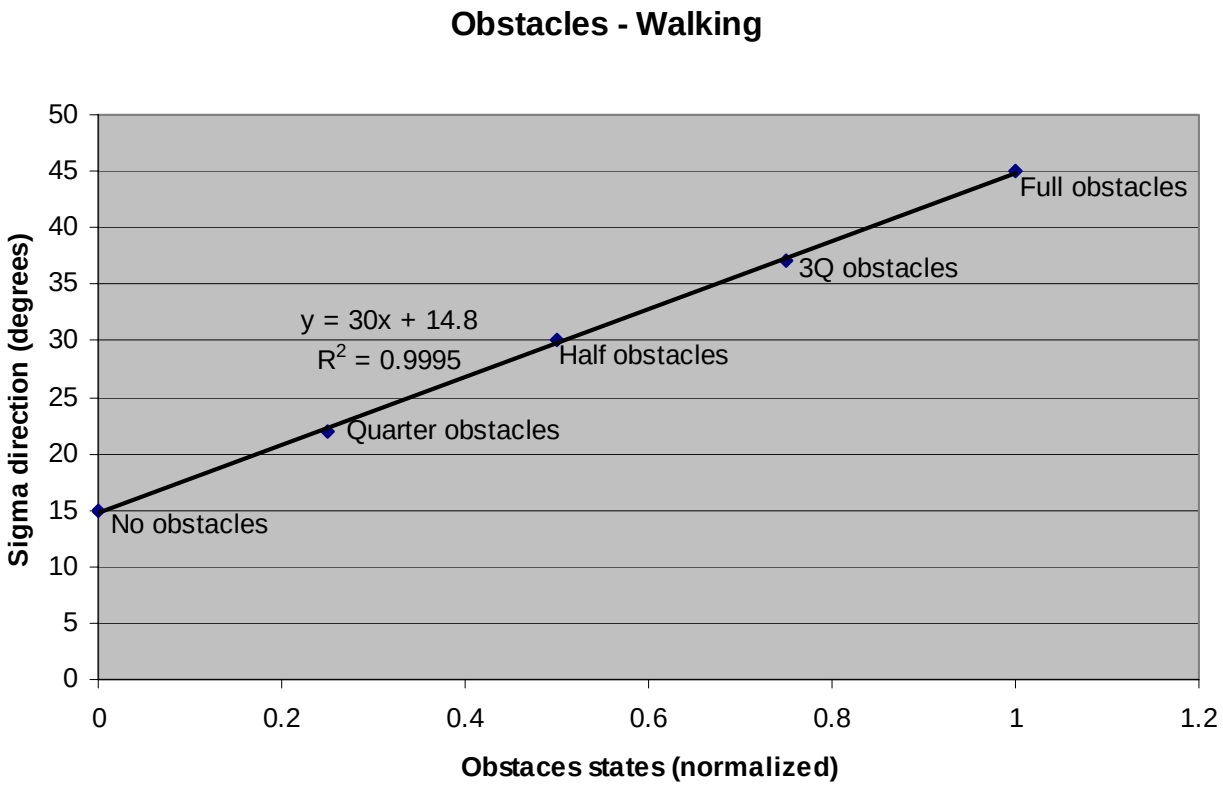

Figure B.28: Sigma direction as a function of the obstacles during walking

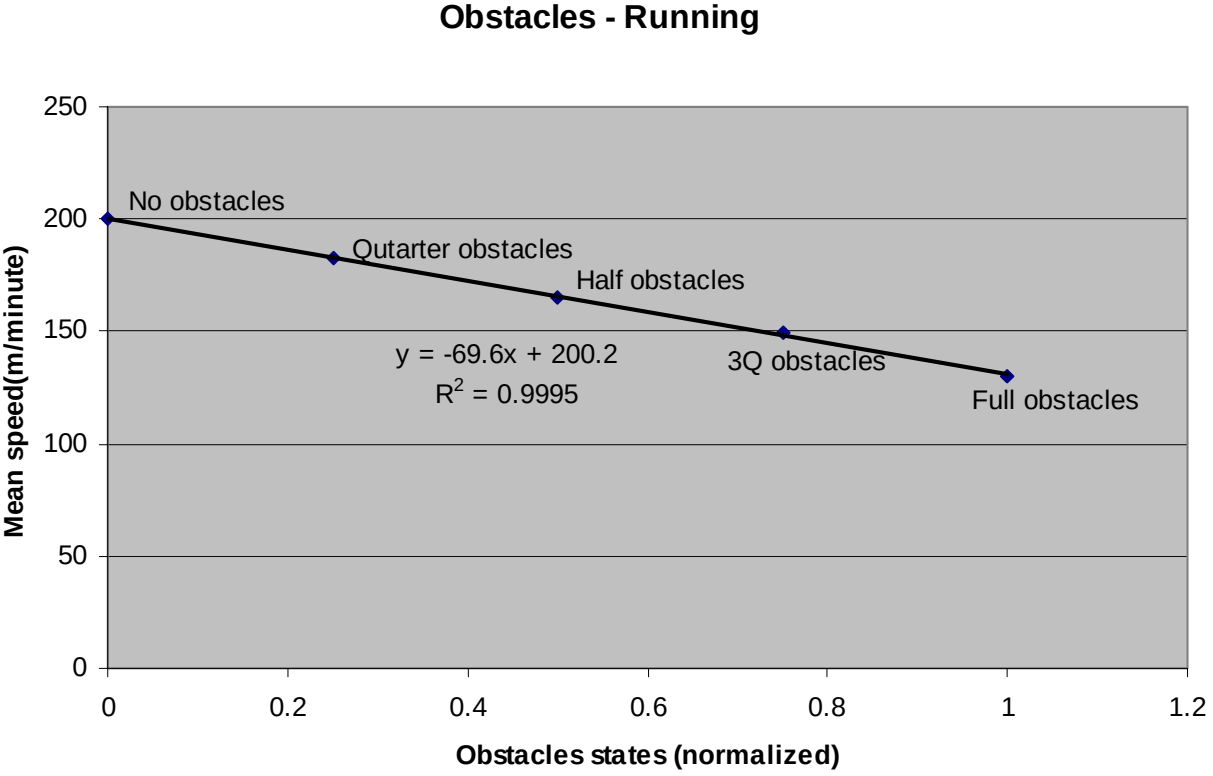

Figure B.29: Mean speed as a function of the obstacles during running

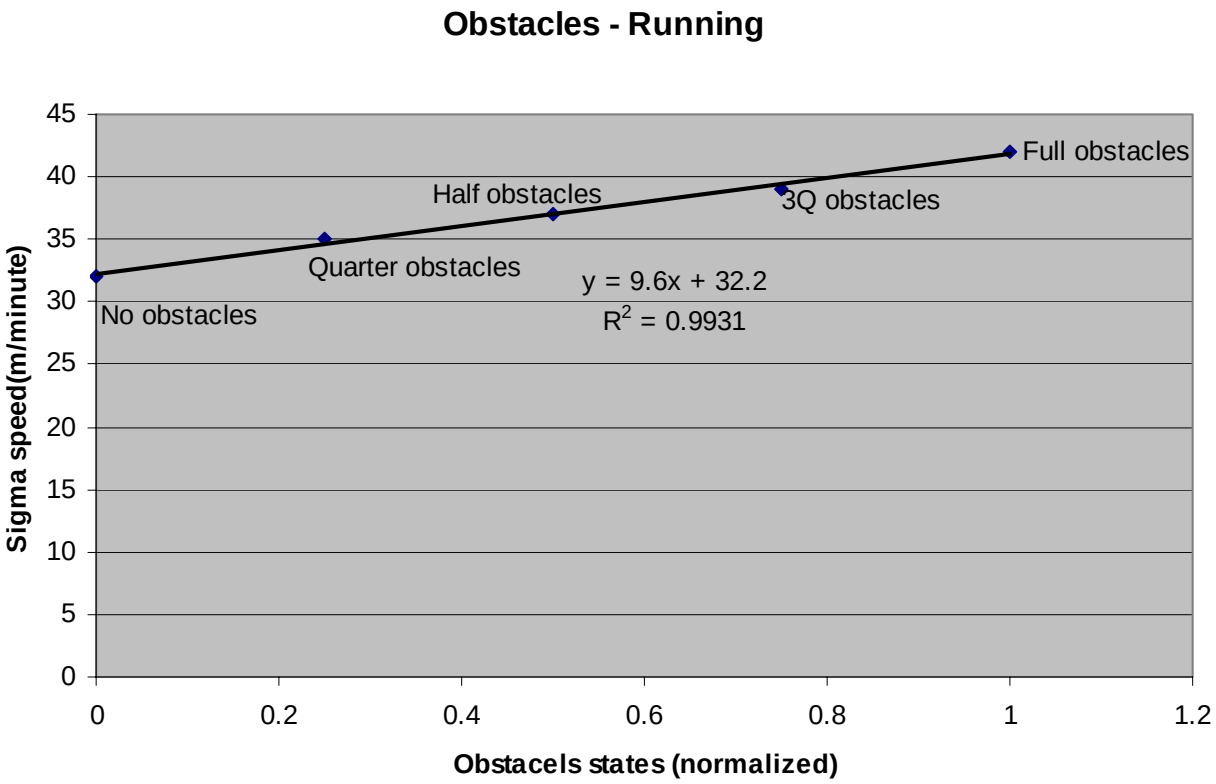

Figure B.30: Sigma speed as a function of the obstacles during running

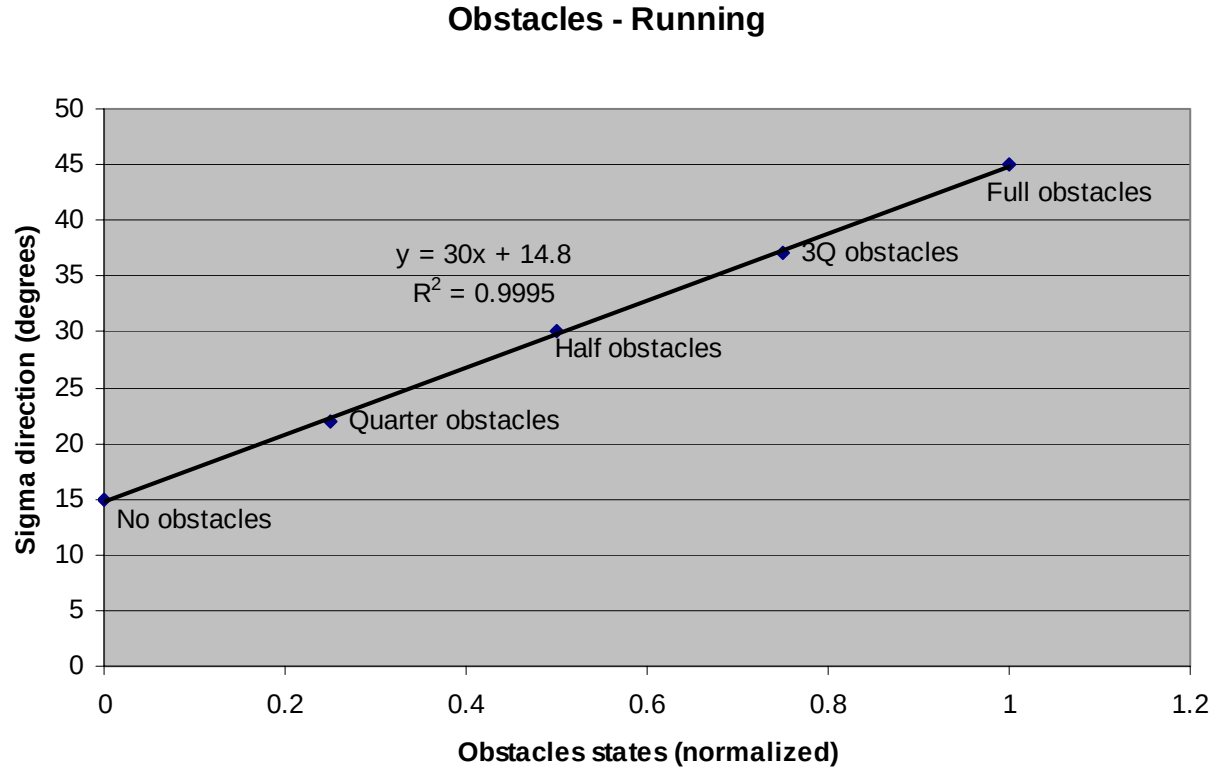

Figure B.31: Sigma direction as a function of the obstacles during running

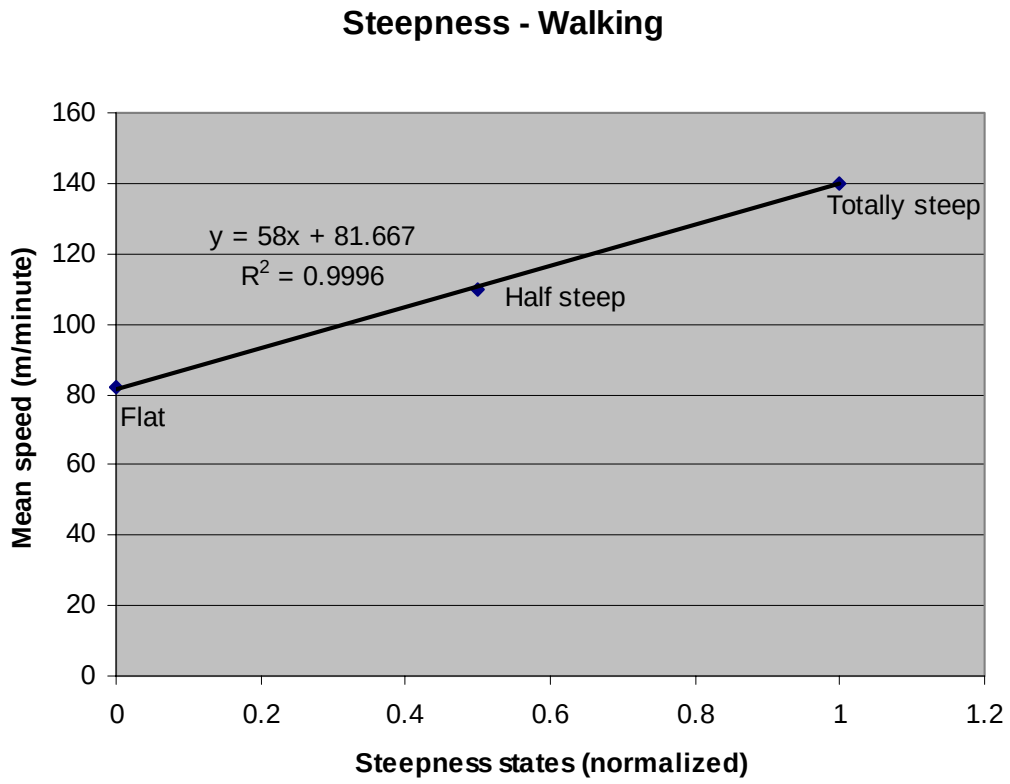

Figure B.32: Mean speed as a function of the steepness during walking

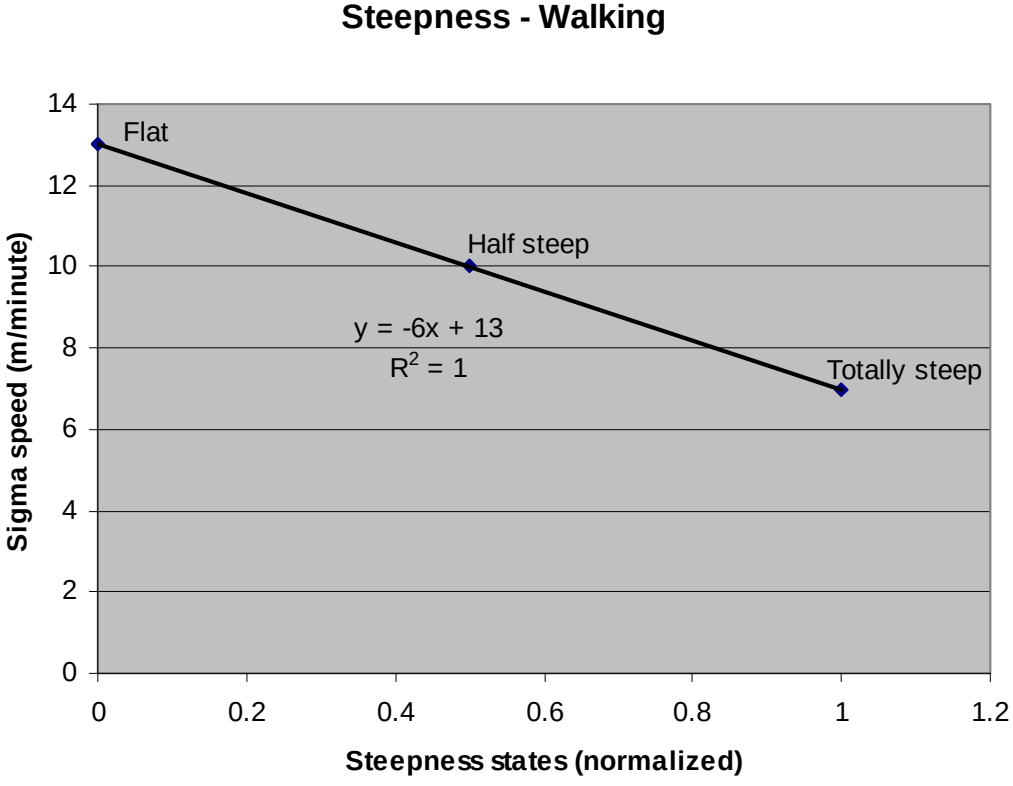

Figure B.33: Sigma speed as a function of the steepness during walking

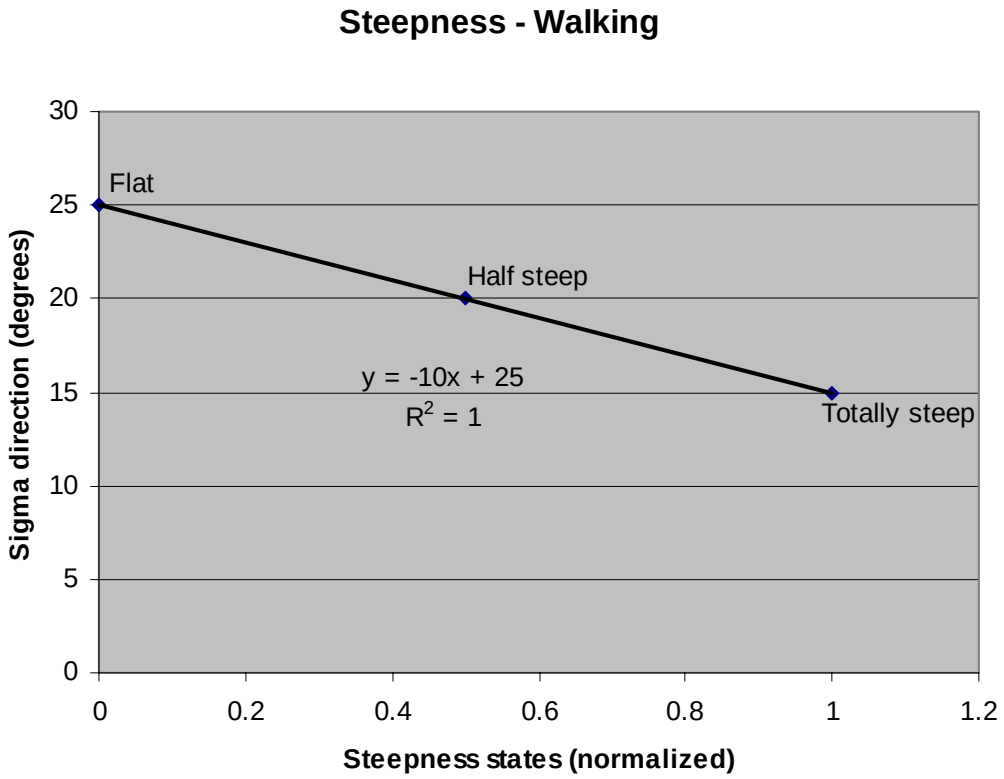

Figure B.34: Sigma direction as a function of the steepness during walking

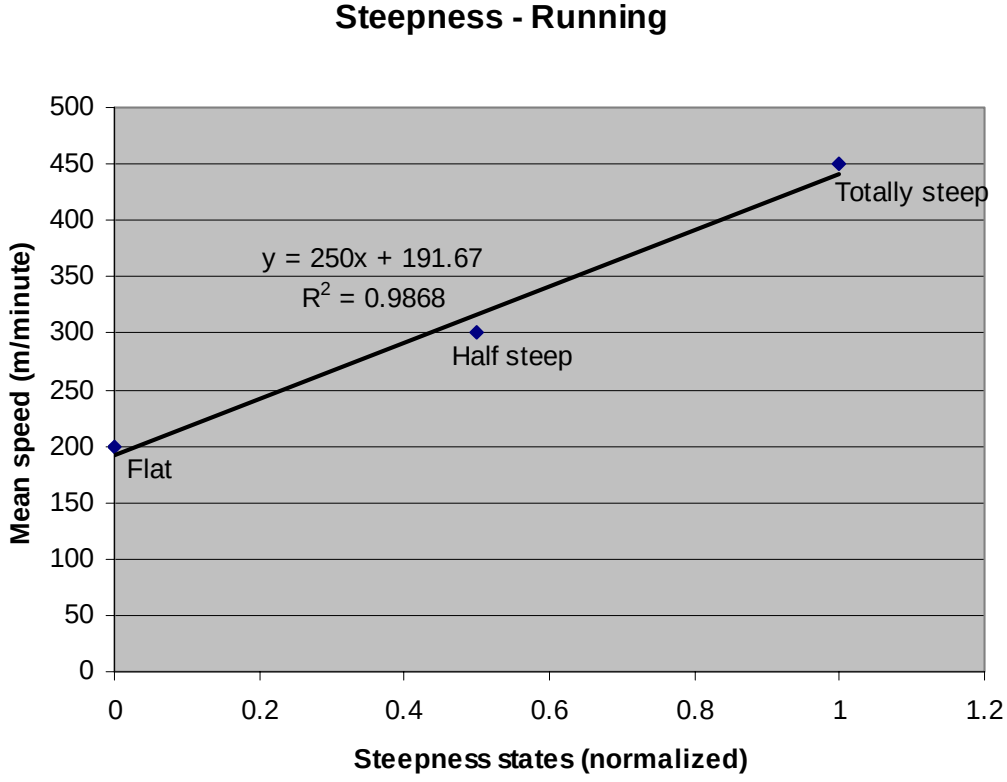

Figure B.35: Mean speed as a function of the steepness during running

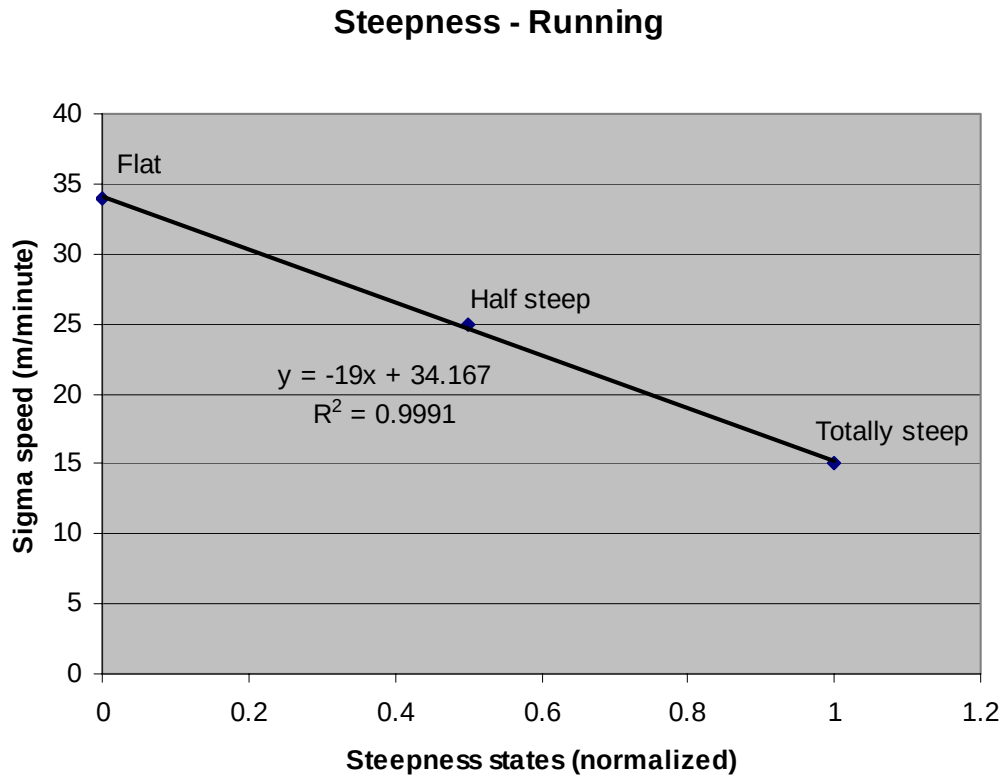

Figure B.36: Sigma speed as a function of the steepness during running

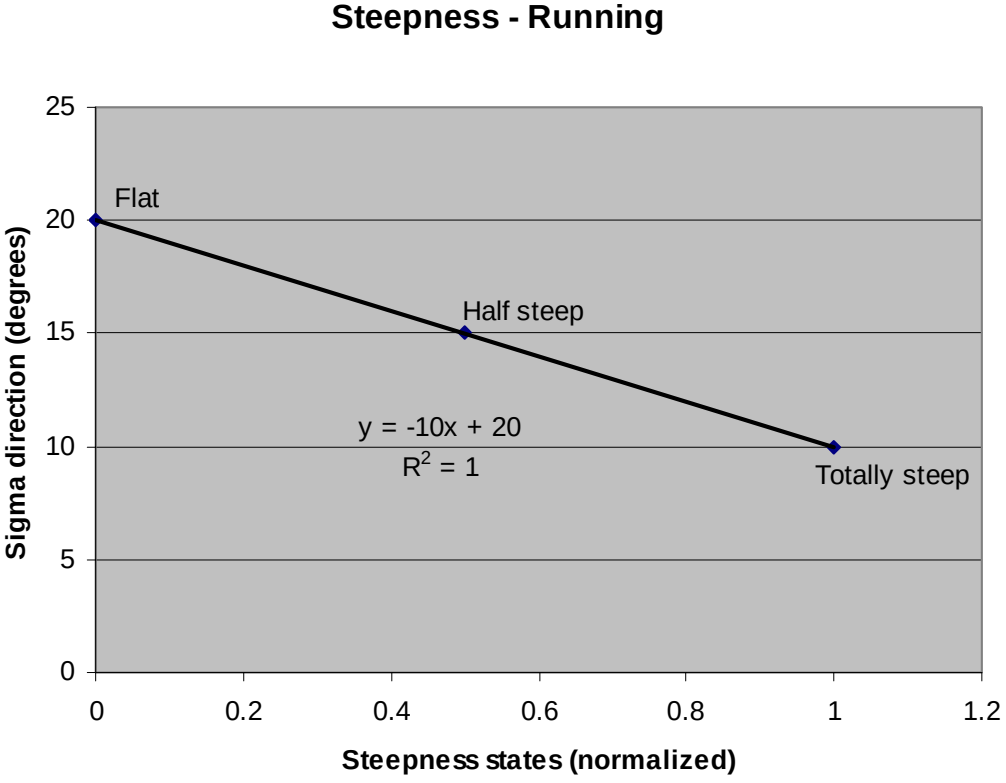

Figure B.37: Sigma direction as a function of the steepness during running

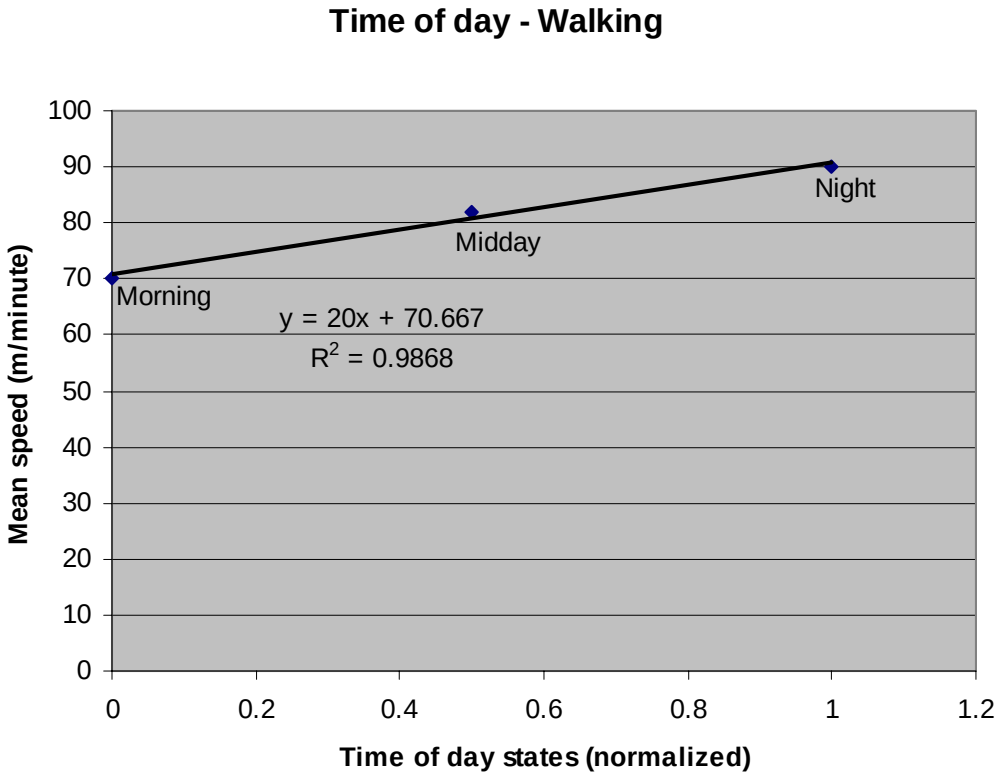

Figure B.38: Mean speed as a function of the time of day during walking

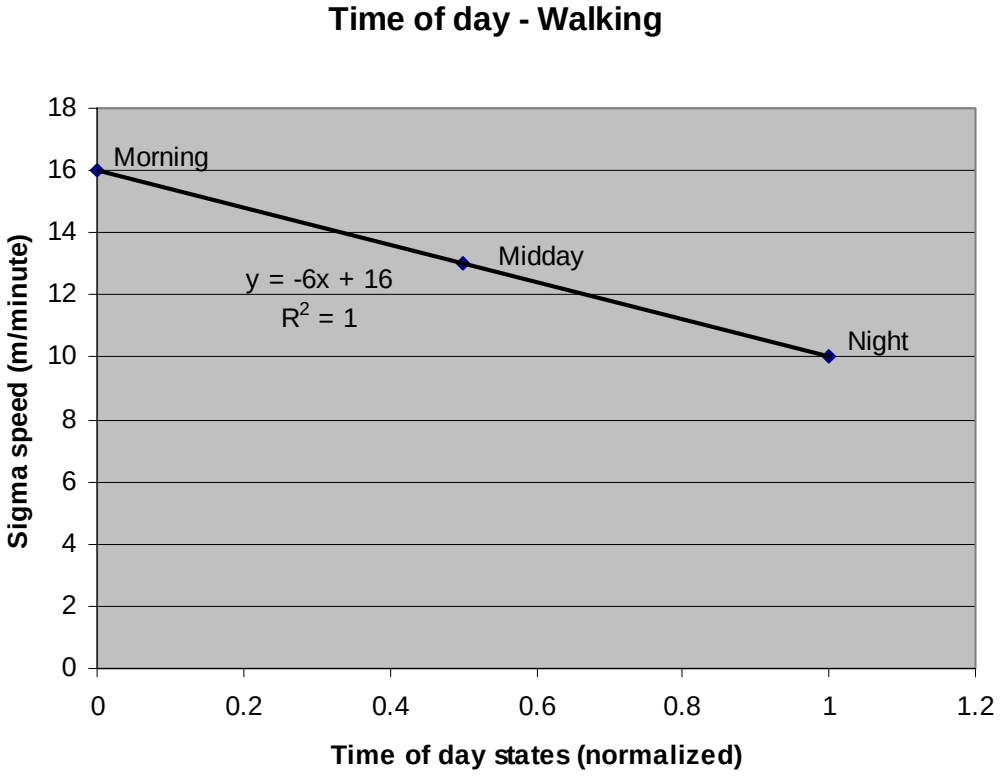

Figure B.39: Sigma speed as a function of the time of day during walking

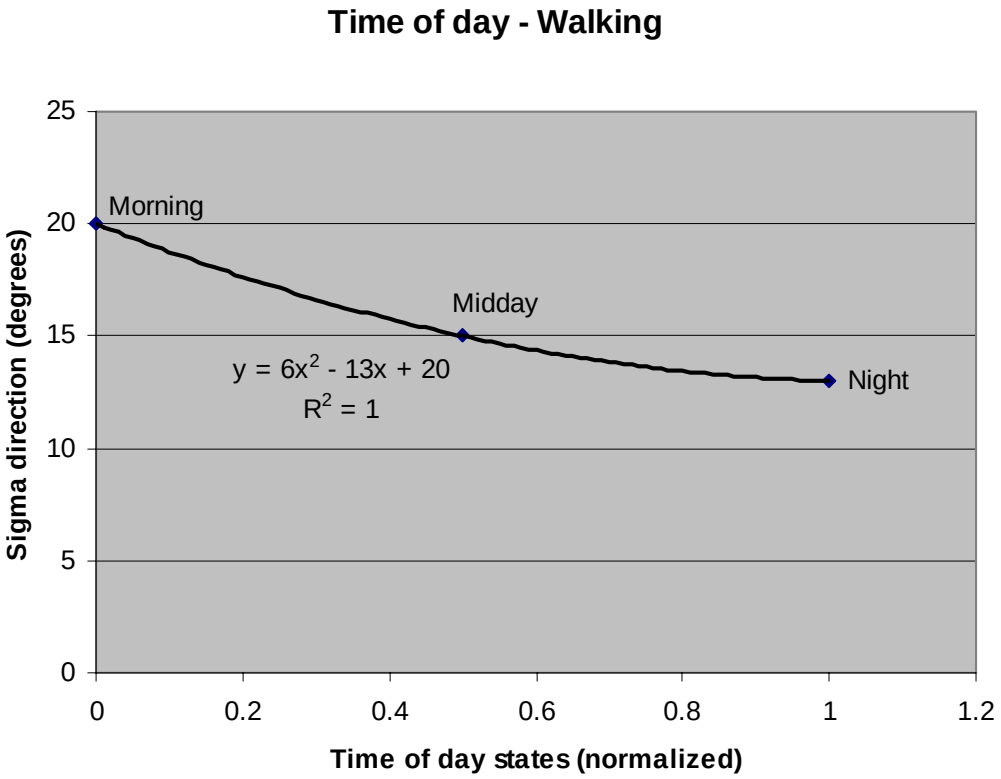

Figure B.40: Sigma direction as a function of the time of day during walking

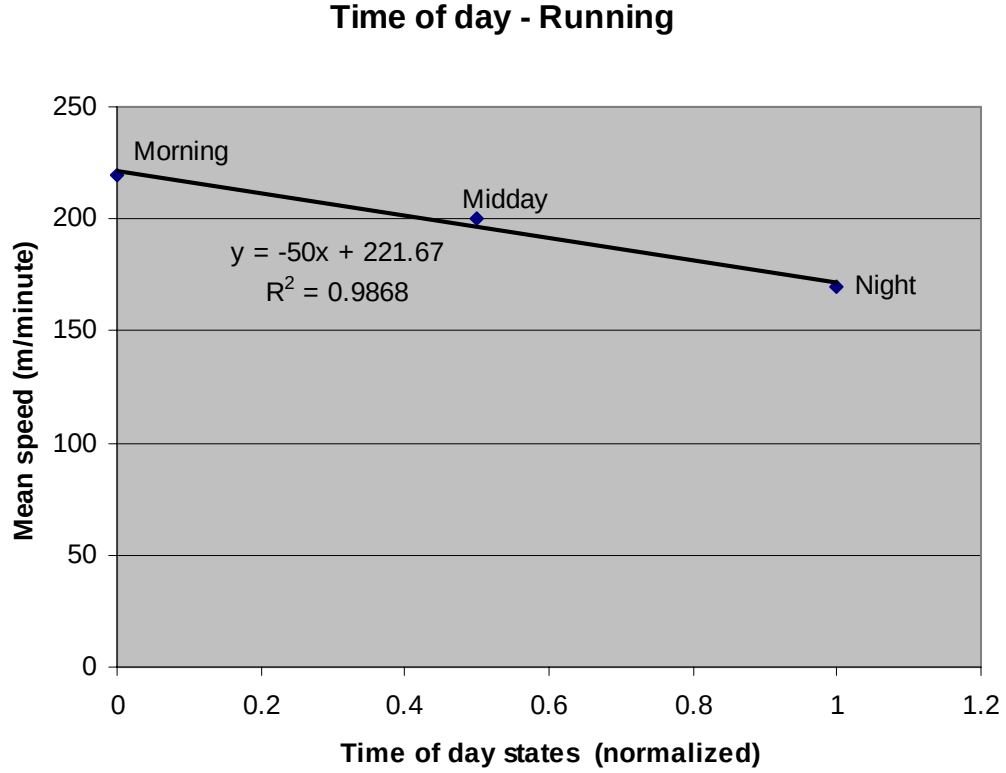

Figure B.41: Mean speed as a function of the time of day during running

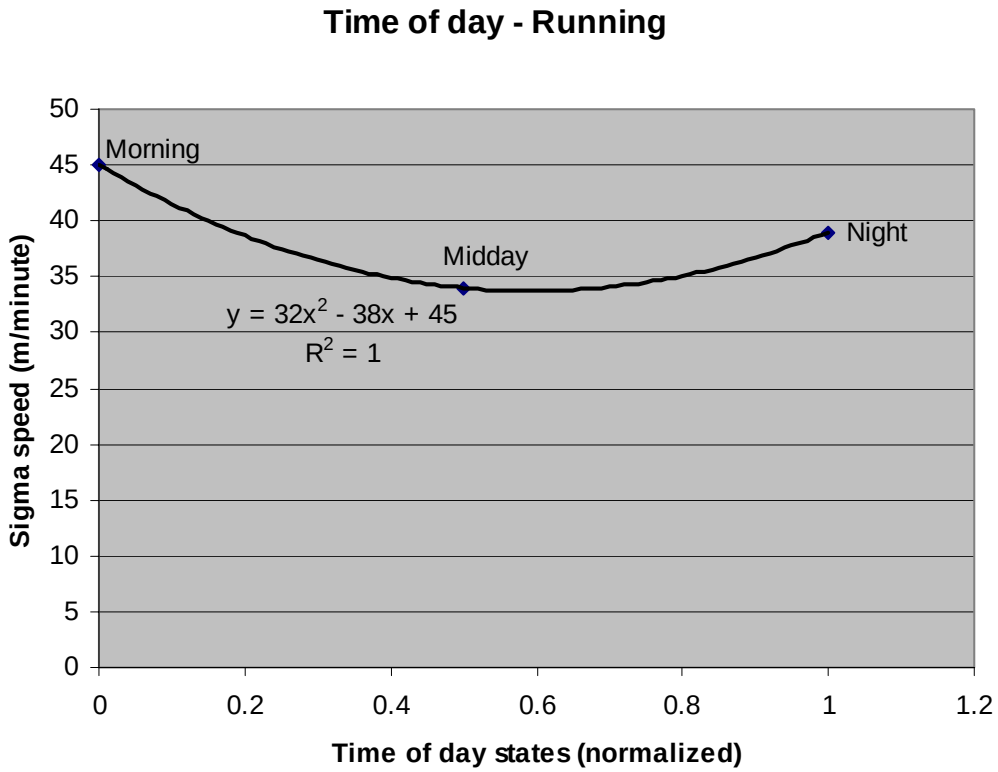

Figure B.42: Sigma speed as a function of the time of day during running

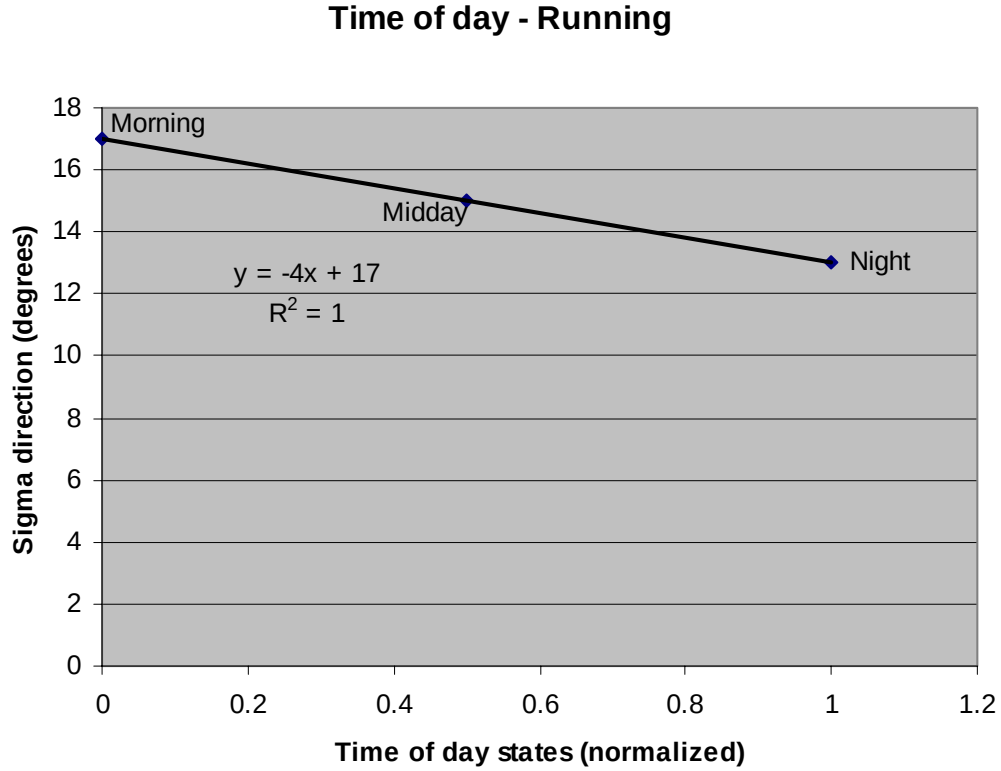

Figure B.43: Sigma direction as a function of the time of day during running

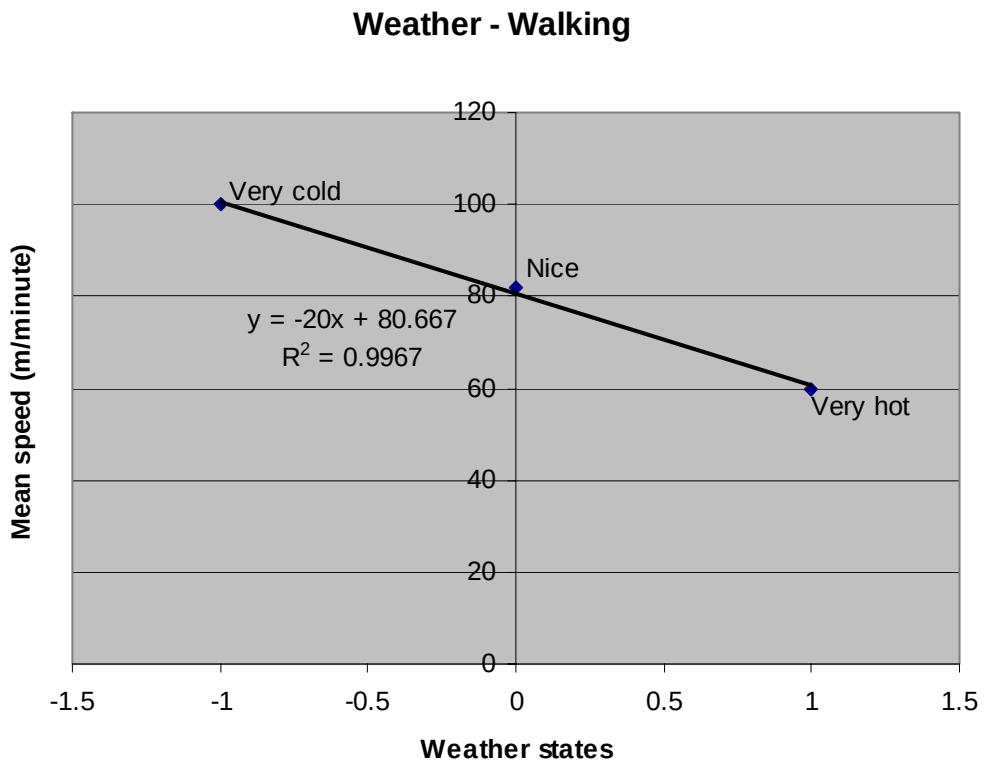

Figure B.44: Mean speed as a function of the weather during walking

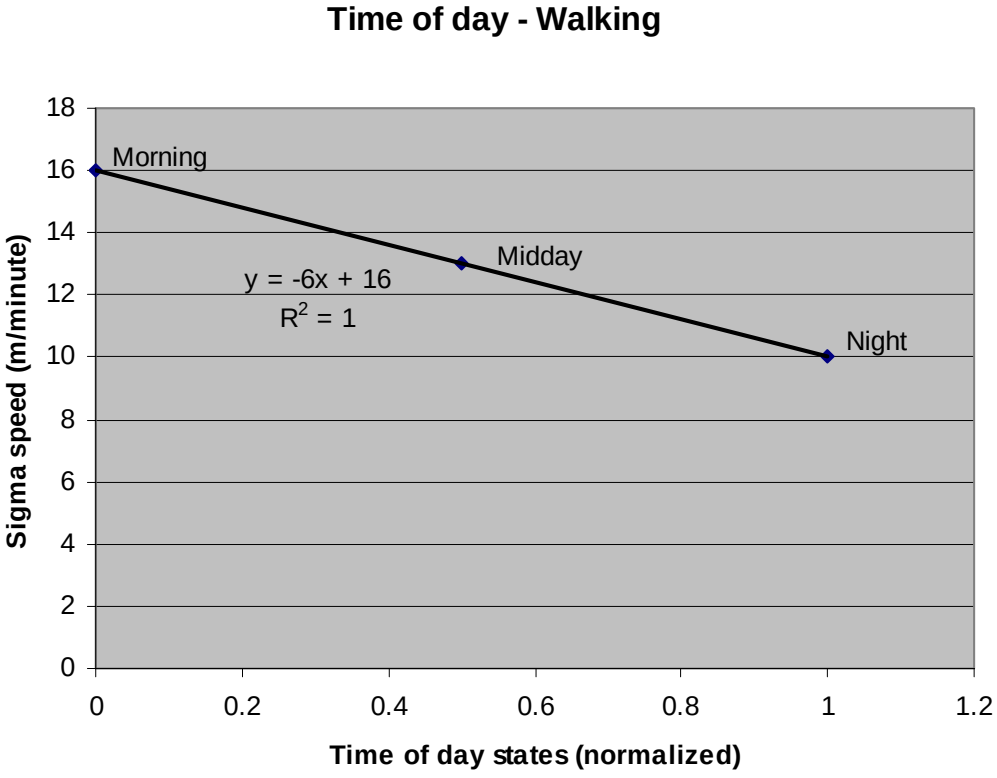

Figure B.45: Sigma speed as a function of the weather during walking

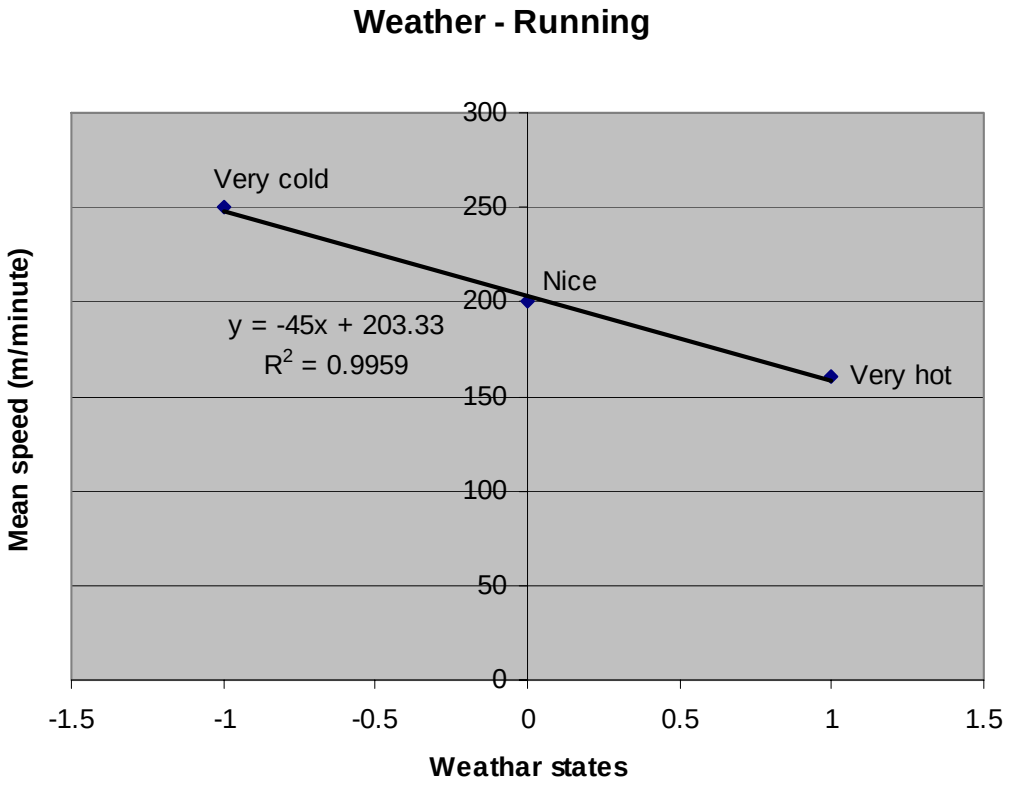

Figure B.46: Sigma speed as a function of the weather during running

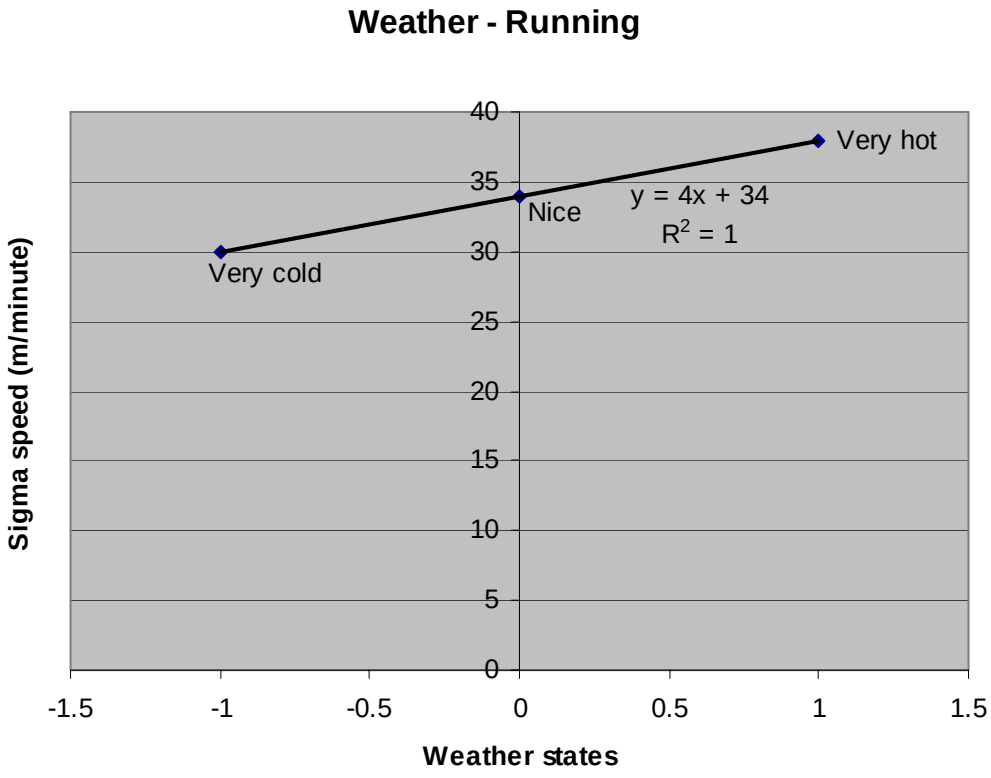

Figure B.47: Sigma direction as a function of the weather during running

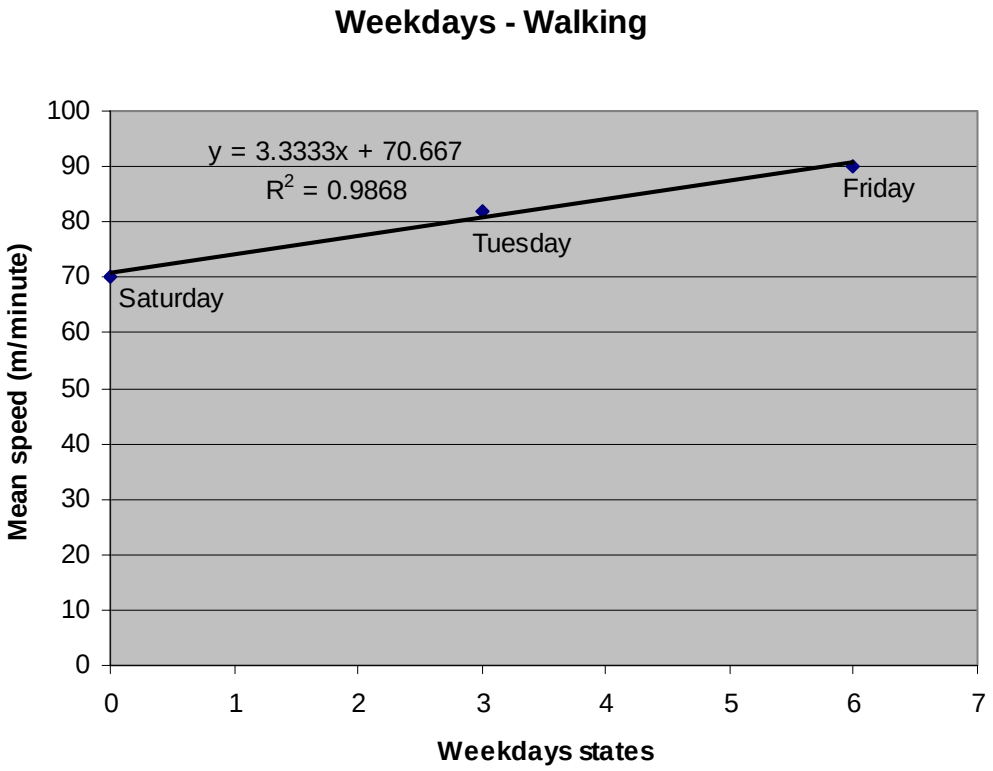

Figure B.48: Mean speed as a function of the week day during walking

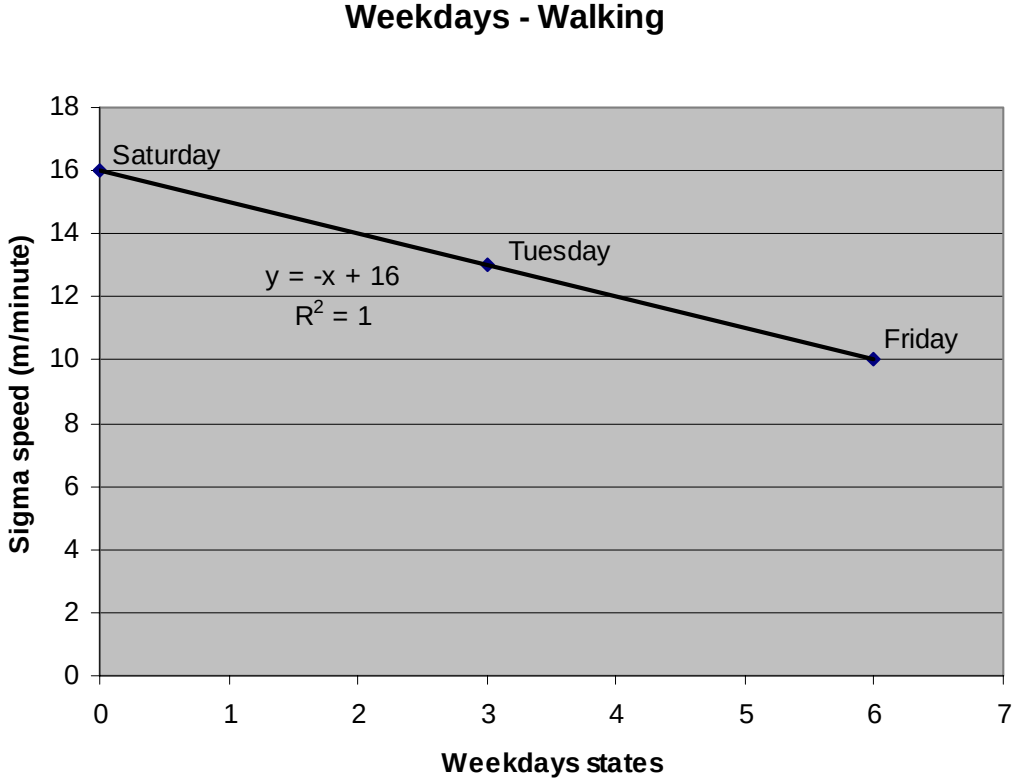

Figure B.49: Sigma speed as a function of the week day during walking

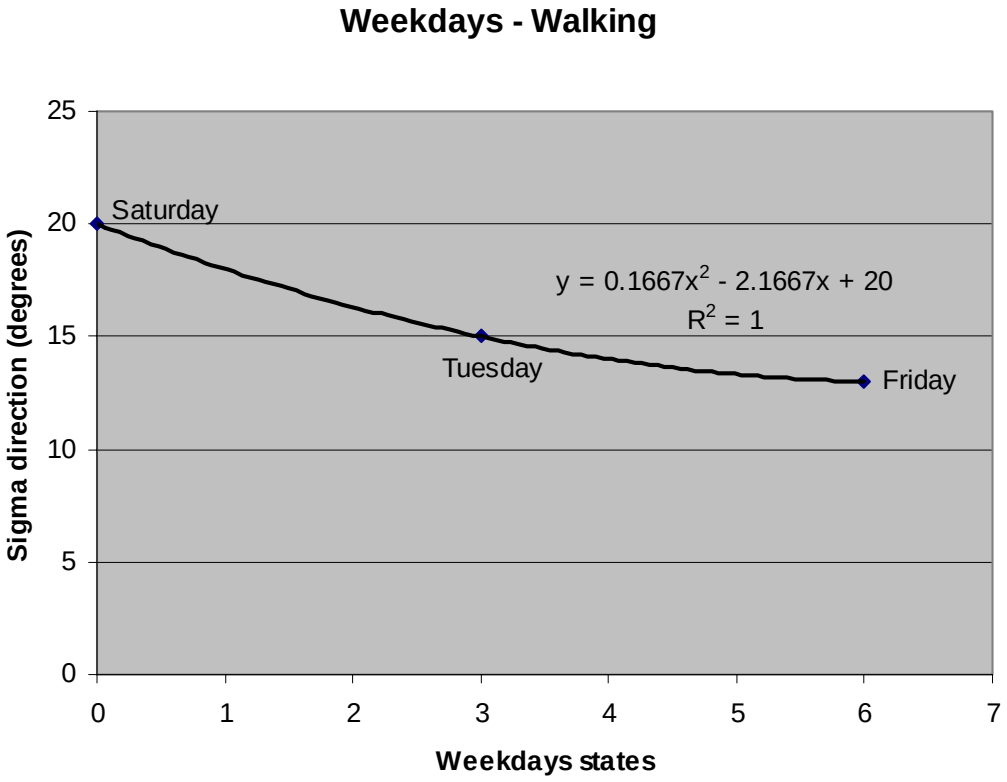

Figure B.50: Sigma direction as a function of the week day during walking

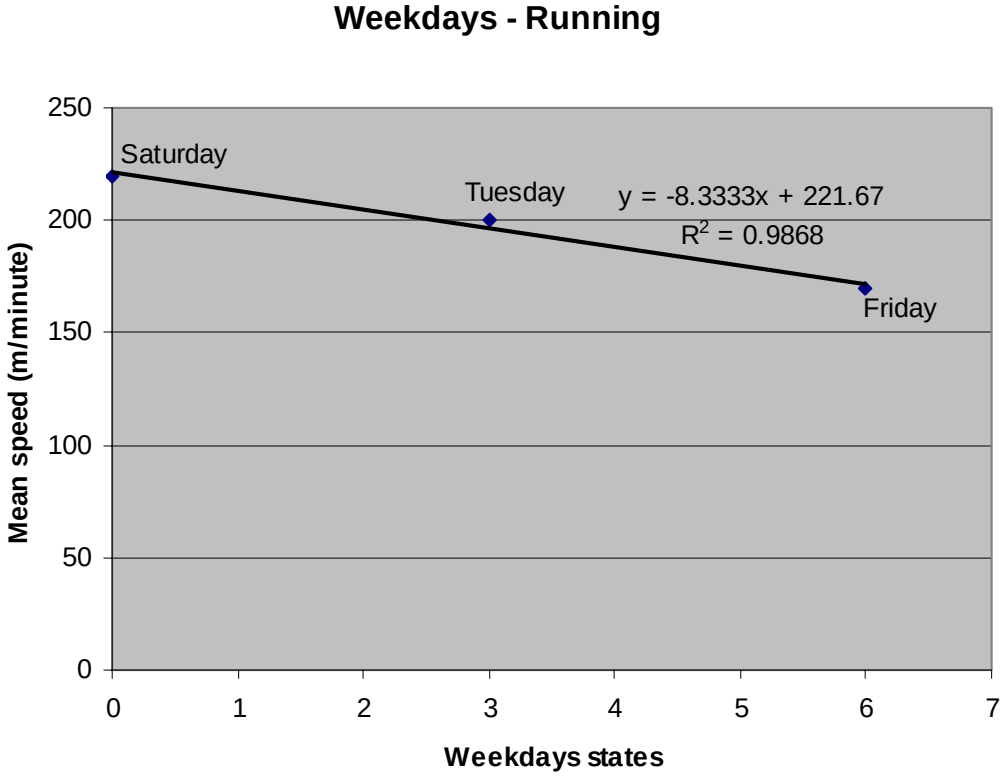

Figure B.51: Mean speed as a function of the week day during running

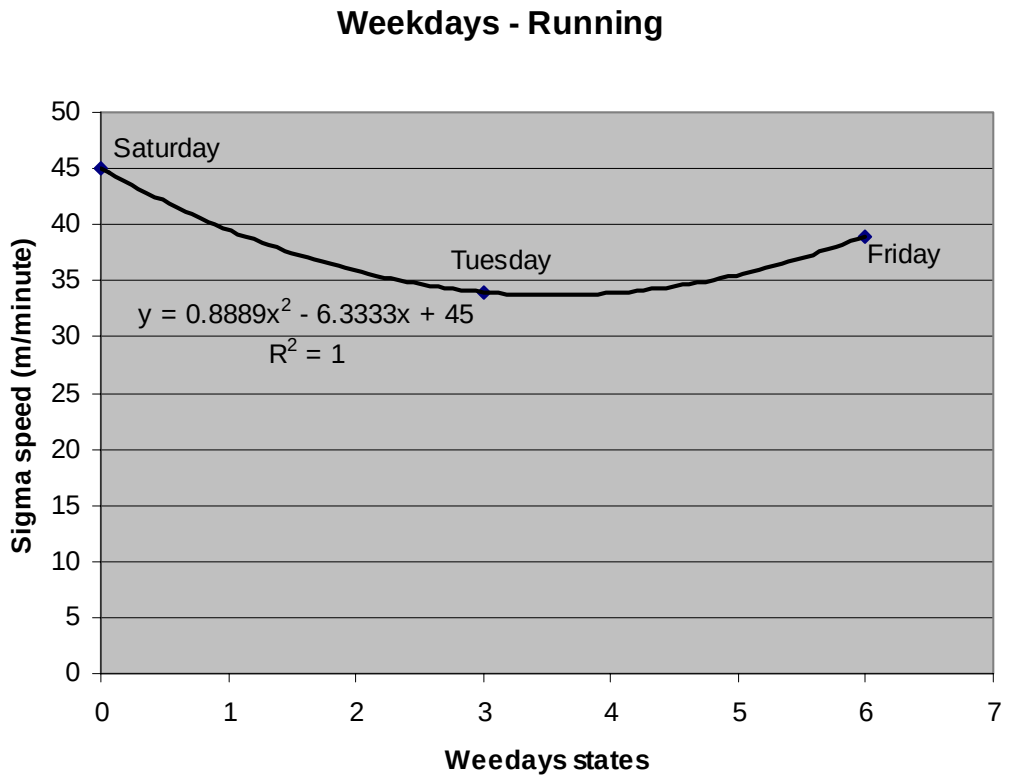

Figure B.52: Sigma speed as a function of the week day during running

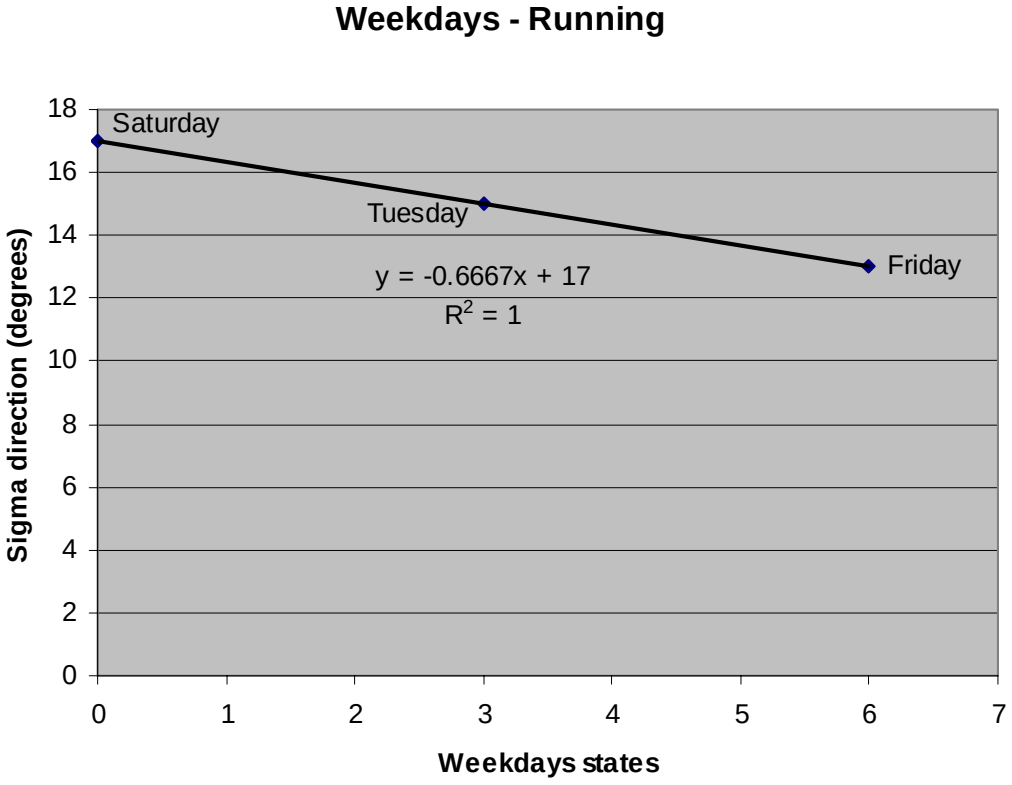

Figure B.53: Sigma direction as a function of the week day during running

Appendix C

States equations

I. Activity

$$\begin{aligned}\mu_v =& -3.1458(activity\ state)^4 + 43.458(activity\ state)^3 \\ & - 196.48(activity\ state)^2 + 279.2(activity\ state) + 127.5 \quad \text{(C.1)}\\ \sigma_v =& -0.52(activity\ state)^4 + 7(activity\ state)^3 \\ & - 28.2(activity\ state)^2 + 23.545(activity\ state) + 48.8 \quad \text{(C.2)}\\ \mu_\alpha =& \alpha_{old} \quad \text{(C.3)}\\ \sigma_\alpha =& 1.3542(activity\ state)^4 - 18.727(activity\ state)^3 \\ & + 93.7(activity\ state)^2 - 200.5(activity\ state) + 169.2 \quad \text{(C.4)}\end{aligned}$$

II. Disorientation

- Walking

$$\begin{aligned}\mu_v &= -22(disorientation\ state)^2 - 17(disorientation\ state) + 82 \quad \text{(C.5)}\\ \sigma_v &= 2(disorientation\ state)^2 + 7(disorientation\ state) + 13 \quad \text{(C.6)}\\ \mu_\alpha &= \alpha_{old} \quad \text{(C.7)}\\ \sigma_\alpha &= 300(disorientation\ state)^2 + 555(disorientation\ state) + 15 \quad \text{(C.8)}\end{aligned}$$

- Running

$$\begin{aligned}\mu_v &= -80(disorientation\ state)^2 - 9E^{-13}(disorientation\ state) + 200 \quad \text{(C.9)}\\ \sigma_v &= 6(disorientation\ state) + 34 \quad \text{(C.10)}\\ \mu_\alpha &= \alpha_{old} \quad \text{(C.11)}\\ \sigma_\alpha &= -170(disorientation\ state)^2 + 855(disorientation\ state) + 15 \quad \text{(C.12)}\end{aligned}$$

III. Age

- Walking

$$\mu_v = -118(age\ state)^2 + 133(age\ state) + 45 \quad \text{(C.13)}$$
$$\sigma_v = -4(age\ state) + 15 \quad \text{(C.14)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.15)}$$
$$\sigma_\alpha = 90(age\ state)^2 - 105(age\ state) + 45 \quad \text{(C.16)}$$

- Running

$$\mu_v = -240(age\ state)^2 + 280(age\ state) + 120 \quad \text{(C.17)}$$
$$\sigma_v = -46(age\ state)^2 + 41(age\ state) + 25 \quad \text{(C.18)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.19)}$$
$$\sigma_\alpha = 14(age\ state)^2 - 17(age\ state) + 20 \quad \text{(C.20)}$$

IV. Time of day

- Walking

$$\mu_v = 20(time\ of\ day\ state) + 70.667 \quad \text{(C.21)}$$
$$\sigma_v = -6(time\ of\ day\ state) + 16 \quad \text{(C.22)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.23)}$$
$$\sigma_\alpha = 6(time\ of\ day\ state)^2 - 13(time\ of\ day\ state) + 20 \quad \text{(C.24)}$$

- Running

$$\mu_v = -50(time\ of\ day\ state) + 221.67 \quad \text{(C.25)}$$
$$\sigma_v = 32(time\ of\ day\ state)^2 - 38(time\ of\ day\ state) + 45 \quad \text{(C.26)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.27)}$$
$$\sigma_\alpha = -4(time\ of\ day\ state) + 17 \quad \text{(C.28)}$$

V. Weather

- Walking

$$\mu_v = -20(weather\ state) + 80.667 \quad \text{(C.29)}$$
$$\sigma_v = 3(weather\ state) + 13 \quad \text{(C.30)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.31)}$$
$$\sigma_\alpha = 15 \quad \text{(C.32)}$$

- Running

$$\mu_v = -45(weather\ state) + 203.33 \quad \text{(C.33)}$$
$$\sigma_v = 4(weather\ state) + 34 \quad \text{(C.34)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.35)}$$
$$\sigma_\alpha = 15 \quad \text{(C.36)}$$

VI. Activeness

- Walking

$$\mu_v = -28(activeness\ state)^2 + 88(activeness\ state) + 45 \quad \text{(C.37)}$$
$$\sigma_v = -4(activeness\ state) + 15 \quad \text{(C.38)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.39)}$$
$$\sigma_\alpha = 15 \quad \text{(C.40)}$$

- Running

$$\mu_v = 160(activeness\ state) + 120 \quad \text{(C.41)}$$
$$\sigma_v = -9(activeness\ state) + 38.167 \quad \text{(C.42)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.43)}$$
$$\sigma_\alpha = 15 \quad \text{(C.44)}$$

VII. Obstacles

- Walking

$$\mu_v = -40(obstacle\ state) + 82 \quad \text{(C.45)}$$
$$\sigma_v = 4(obstacle\ state) + 10 \quad \text{(C.46)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.47)}$$
$$\sigma_\alpha = 30(obstacle\ state) + 14.8 \quad \text{(C.48)}$$

- Running

$$\mu_v = -69.6(obstacle\ state) + 200.2 \quad \text{(C.49)}$$
$$\sigma_v = 9.6(obstacle\ state) + 32.2 \quad \text{(C.50)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.51)}$$
$$\sigma_\alpha = 30(obstacle\ state) + 14.8 \quad \text{(C.52)}$$

VIII. Ground Steepness

- Walking

$$\mu_v = 58(steepness\ state) + 81.667 \quad \text{(C.53)}$$
$$\sigma_v = -6(steepness\ state) + 13 \quad \text{(C.54)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.55)}$$
$$\sigma_\alpha = -10(steepness\ state) + 25 \quad \text{(C.56)}$$

- Running

$$\mu_v = 250(steepness\ state) + 191.67 \quad \text{(C.57)}$$
$$\sigma_v = -19(steepness\ state) + 34.167 \quad \text{(C.58)}$$
$$\mu_\alpha = \alpha_{old} \quad \text{(C.59)}$$
$$\sigma_\alpha = -10(steepness\ state) + 20 \quad \text{(C.60)}$$

IX. Weekdays

- Walking

$$\mu_v = 3.3333(weedays\ state) + 70.667 \tag{C.61}$$
$$\sigma_v = -(weedays\ state) + 16 \tag{C.62}$$
$$\mu_\alpha = \alpha_{old} \tag{C.63}$$
$$\sigma_\alpha = 0.1667(weekdays\ state)^2 - 2.1667(weedays\ state) + 20 \tag{C.64}$$

- Running

$$\mu_v = -8.3333(weedays\ state) + 221.67 \tag{C.65}$$
$$\sigma_v = 0.8889(weekdays\ state)^2 - 6.3333(weedays\ state) + 45 \tag{C.66}$$
$$\mu_\alpha = \alpha_{old} \tag{C.67}$$
$$\sigma_\alpha = -0.6667(weedays\ state) + 17 \tag{C.68}$$

X. Emotions

- Walking

$$\mu_v = 40(emotions\ state) + 60.667 \tag{C.69}$$
$$\sigma_v = -7(emotions\ state) + 16.833 \tag{C.70}$$
$$\mu_\alpha = \alpha_{old} \tag{C.71}$$
$$\sigma_\alpha = 14(emotions\ state) + 13.667 \tag{C.72}$$

- Running

$$\mu_v = 120(emotions\ state) + 140 \tag{C.73}$$
$$\sigma_v = -4(emotions\ state) + 32 \tag{C.74}$$
$$\mu_\alpha = \alpha_{old} \tag{C.75}$$
$$\sigma_\alpha = 4(emotions\ state) + 13 \tag{C.76}$$

XI. Arousal

- Walking

$$\mu_v = 40(arousal\ state) + 60.667 \tag{C.77}$$
$$\sigma_v = -7(arousal\ state) + 16.833 \tag{C.78}$$
$$\mu_\alpha = \alpha_{old} \tag{C.79}$$
$$\sigma_\alpha = 14(arousal\ state) + 13.667 \tag{C.80}$$

- Running

$$\mu_v = 120(arousal\ state) + 140 \tag{C.81}$$
$$\sigma_v = -4(arousal\ state) + 32 \tag{C.82}$$
$$\mu_\alpha = \alpha_{old} \tag{C.83}$$
$$\sigma_\alpha = 4(arousal\ state) + 13 \tag{C.84}$$

Appendix D

Equipment photos

Figure D.1: Vodafone 3G/GPRS data card

Figure D.2: Bluetooth adapter

Figure D.3: Bluetooth GPS receiver

Figure D.4: Electronic compass

Appendix E

NMEA protocol

NMEA (National Marine Electronics Association) is a standard protocol, used by GPS receivers to transmit data. NMEA output is EIA-422A but for most purposes it can considered RS-232 compatible. It uses 4800 bps, 8 data bits, no parity and one stop bit. NMEA 0183 (which is the case in the used receiver) sentences are all ASCII. Each sentence begins with a dollar sign and ends with a carriage return linefeed. Data is comma delimited. A checksum is optionally added (in a few cases it is mandatory) The most important NMEA sentences include the GGA which provides the current Fix data (provides 3D location and accuracy data), the RMC which provides the minimum gps sentences information, and the GSA which provides the Satellite status data. For example if the following GSA sentence is read:
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
Then it can be explained as follows:

GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality
8	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude, Meters, above mean sea level
46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
*47	the checksum data, always begins with *

List of Figures

1.1 Situation Space - General Model, originally found in [3] 18

2.1 PDF of the location using time advance (estimate 1) 24
2.2 PDF of the location using multi-sector antenna (estimate 2) 24
2.3 PDF of the location after combining estimates 1 and 2 24
2.4 Problem statement in discrete time 26
2.5 First order Hidden Markov Model representing the dynamic system . . 27
2.6 Position estimation using Bayesian Filter, originally found in [13] . . . 31
2.7 Particles representation of continuous PDF 39
2.8 Position estimation using Particle Filter, originally found in [13] 48
2.9 Comparison between Bayesian Filter and Particle Filter, originally found in [13] 50

3.1 States that affect the human movement 52
3.2 Developed movement model 54
3.3 Satellite image of DLR premises 56
3.4 Gray scale map for DLR premises using 8 colors depth 57
3.5 Mean speed as a function of the age during walking 60
3.6 Sigma speed as a function of the age during walking 60
3.7 Sigma direction as a function of the age during walking 61
3.8 Mean speed as a function of the age during running 61
3.9 Sigma speed as a function of the age during running 62
3.10 Sigma direction as a function of the age during running 62
3.11 Acceleration -deacceleraation profile 68
3.12 Movement model evolution with time and its causal relationships . . . 70

4.1 Setup functionality 74
4.2 Software design at the server side 77

5.1 The backpack equipped with the laptop, the 3G/GPRS data card, the bluetooth adapter, the bluetooth GPS receiver and the electrical compass. 80
5.2 Hardware setup and functionality. 82
5.3 UML diagram for the `Abstract Localized Entity`, its parent and children classes 84
5.4 UML diagram of polar-rectangular location transformation utility package 85
5.5 Stack diagram of the particle filtering package 89

5.6 Component diagram of the particle filtering package 90
5.7 UML diagram for the particle filter implementation 91

7.1 Simulated pedestrian(pink) and received noisy measurement(green). . . 98
7.2 Simulated pedestrian(pink), received measurement(green) and particle cloud(blue). 99
7.3 Simulated pedestrian(pink), received measurement(green) and PDF of location(gray cloud). 99
7.4 Simulated pedestrian(pink), received measurement(green) and point estimation(cyan). 100
7.5 Simulated pedestrian(pink), received measurement(green), PDF of location(gray cloud) and point estimation(cyan). 101
7.6 Simulated pedestrian(pink), received measurement(green), PDF of location(gray cloud), estimation(cyan) and particle cloud. 101
7.7 Particles cloud, measurement, estimation during a real time run when the GPS receiver fails. 102
7.8 PDF, measurement, estimation during a real time run when the GPS receiver fails. 103
7.9 Inference panel of activity - pedestrian is stopping. 103
7.10 Mean position error vs. number of particles averaged over 10000 time steps, standard deviation of the compass = 20° and of the GPS receiver = 25m. An increase of the number of particles, results in a decrease in the mean error. For numbers of particles over 2000, no further significant error reduction can be achieved. 106
7.11 Mean angle error vs. number of particles averaged over 10000 time steps, standard deviation of the compass = 20° and of the GPS receiver = 25m. An increase of the number of particles, results in a small reduction in the mean error. For numbers of particles over 450, no further significant error reduction can be achieved. 107
7.12 Mean position error vs. number of particles averaged over 5000 time steps, standard deviation of the compass = 25° and variable for the GPS receiver. It is obvious that as the noise standard deviation of the GPS receiver decreases, the average position error decreases also. Large number of particles compensates for having a noisy GPS receiver. . . . 108
7.13 Mean angle error vs. number of particles averaged over 5000 time steps, standard deviation of the GPS receiver = 25m and variable for the compass. It is obvious that as the noise standard deviation of the compass decreases, the average direction error decreases also. 109
7.14 Mean position error vs. number of particles averaged over 5000 time steps, standard deviation of the GPS receiver = 25m and variable for the compass. An accurate compass results in not only accurate direction measurements, but also position measurements. More particles are needed if an accurate compass is used since it will cause many particles to vanish due to resampling. 110

7.15 Mean angle error vs. number of particles averaged over 5000 time steps, standard deviation of the compass = 25° and variable for the GPS receiver. An accurate GPS receiver results in not only accurate position measurements, but also direction measurements. More particles are needed if an accurate GPS receiver is used since it will cause many particles to vanish due to resampling. 112

B.1 Mean Speed as a function of the activity 121
B.2 Sigma speed as a function of the activity 122
B.3 Sigma direction as a function of the activity 122
B.4 Mean speed as a function of the activeness during walking 123
B.5 Sigma speed as a function of the activeness during walking 123
B.6 Mean speed as a function of the activeness during running 124
B.7 Sigma direction as a function of the activeness during running 124
B.8 Mean speed as a function of the disorientation during walking 125
B.9 Sigma speed as a function of the disorientation during walking 125
B.10 Sigma direction as a function of the disorientation during walking . . . 126
B.11 Mean speed as a function of the disorientation during running 126
B.12 Sigma speed as a function of the disorientation during running 127
B.13 Sigma direction as a function of the disorientation during running . . . 127
B.14 Mean speed as a function of the emotions during walking 128
B.15 Sigma speed as a function of the emotions during walking 128
B.16 Sigma direction as a function of the emotions during walking 129
B.17 Mean speed as a function of the emotions during running 129
B.18 Sigma speed as a function of the emotions during running 130
B.19 Sigma direction as a function of the emotions during running 130
B.20 Mean speed as a function of the arousal during walking 131
B.21 Sigma speed as a function of the arousal during walking 131
B.22 Sigma direction as a function of the arousal during walking 132
B.23 Mean speed as a function of the arousal during running 132
B.24 Sigma speed as a function of the arousal during running 133
B.25 Sigma direction as a function of the arousal during running 133
B.26 Mean speed as a function of the obstacles during walking 134
B.27 Sigma speed as a function of the obstacles during walking 134
B.28 Sigma direction as a function of the obstacles during walking 135
B.29 Mean speed as a function of the obstacles during running 135
B.30 Sigma speed as a function of the obstacles during running 136
B.31 Sigma direction as a function of the obstacles during running 136
B.32 Mean speed as a function of the steepness during walking 137
B.33 Sigma speed as a function of the steepness during walking 137
B.34 Sigma direction as a function of the steepness during walking 138
B.35 Mean speed as a function of the steepness during running 138
B.36 Sigma speed as a function of the steepness during running 139
B.37 Sigma direction as a function of the steepness during running 139
B.38 Mean speed as a function of the time of day during walking 140

B.39 Sigma speed as a function of the time of day during walking 140
B.40 Sigma direction as a function of the time of day during walking 141
B.41 Mean speed as a function of the time of day during running 141
B.42 Sigma speed as a function of the time of day during running 142
B.43 Sigma direction as a function of the time of day during running 142
B.44 Mean speed as a function of the weather during walking 143
B.45 Sigma speed as a function of the weather during walking 143
B.46 Sigma speed as a function of the weather during running 144
B.47 Sigma direction as a function of the weather during running 144
B.48 Mean speed as a function of the week day during walking 145
B.49 Sigma speed as a function of the week day during walking 145
B.50 Sigma direction as a function of the week day during walking 146
B.51 Mean speed as a function of the week day during running 146
B.52 Sigma speed as a function of the week day during running 147
B.53 Sigma direction as a function of the week day during running 147

D.1 Vodafone 3G/GPRS data card 153
D.2 Bluetooth adapter 153
D.3 Bluetooth GPS receiver 154
D.4 Electronic compass 154

List of Tables

3.1 Activity Markov model 55
3.2 Disorientation Markov model 55
3.3 Emotions Markov model 55
3.4 Arousal Markov model 58
3.5 Activeness Markov model 58
3.6 Some statistics used to build the movement model 58
3.7 Age statistics 59
3.8 State weights according to their importance for human movement . . . 64

A.1 Activity statistics 117
A.2 Disorientation statistics 117
A.3 Time of day statistics 118
A.4 Weather statistics 118
A.5 Activeness statistics 119
A.6 Obstacles statistics 119
A.7 Ground steepness statistics 119
A.8 Weekdays statistics 120
A.9 Emotions statistics 120
A.10 Arousal statistics 120

Bibliography

[1] Lasertracker. [Online]. Available: http://www.gdv-systems.com/index.php?ID=10031

[2] The cricket indoor location system. [Online]. Available: http://nms.lcs.mit.edu/projects/cricket/

[3] M. Angermann, "Situation awareness for mobile information access in hetrogeneous wireless networks," Ph.D. dissertation, University of Ulm, 2005.

[4] S. M. Inc. (2005) Java 2 platform, standard edition (j2se). [Online]. Available: http://www.java.sun.com/j2se/index.jsp

[5] P. Robertson, A. Ouhmich, M. Angermann, and K. Wendlandt, "Implementation of soft location on mobile devices," in *International Symposium on Indoor Localisation and Position Finding, InLoc 2002, DGON, Bonn/Germany*, July 2002.

[6] M. Angermann, J. Kammann, P. Robertson, A. Steingass, and T. Strang, "Software representation for heterogeneous location data sources within a probabilistic framework," in *International Symposium on Location Based Services for Cellular Users - Locellus 2001*, February 2001, pp. 107–118.

[7] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, 1st ed., M. Jordan, S. Lauritzen, and J. Nair, Eds. Springer, 2001.

[8] F. Martinerie and P. Forster, "Data association and tracking using hidden markov models and dynamic programming," in *Conf. ICASSP*, 1992.

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," vol. 77, no. 2. IEEE, 1989.

[10] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 50, no. 2, pp. 174–181, February 2002.

[11] D. Fox, J. Hightower, H. Kauz, L. Liao, and D. J. Patterson, "Bayesian techniques for location estimation," in *Workshop on Location-aware Computing, part of UBICOMP Conference*, Seattle, Washington, October 2003.

[12] A. Leon-Garcia, *Probability and Random Process for Electrical Engineering*, 2nd ed. Addison-Wesley Publishing Company, 1994.

[13] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," presentation by: Honggang Zhang.

[14] Y. Ho and R. Lee, "A bayesian approach to problems in stochastic estimation and control," *IEEE Trans. Automat. Control*, vol. AC-9, pp. 333–339, 1964.

[15] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. New York:Academic, 1970.

[16] Y. Bar-Shalom and X.-R.Li., *Multitarget-Multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995.

[17] I. D. Coope and C. J. Price, "On the convergence of grid-based methods for unconstrained optimization," *SIAM Journal on Optimization*, vol. 11, no. 4, pp. pp. 859–869, 2001.

[18] S. Julier, "Comparison of the particle filter with range parameterized and modified polar ekf's for angle-only tracking," vol. 4048. SPIE, 2000.

[19] E. A. Wan and R. V. der Merwe, "The unscented kalman filter for nonlinear estimation," in *Symp. Adaptive Syst. Signal Process., Commun. Contr.*, Lake Louise, AB, Canada, October 2000.

[20] S. Arulampalam and B. Ristic, "A skewed approach to filtering," vol. 3373. SPIE, 1998.

[21] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear and non-gaussian bayesian state estimation," vol. 140. Inst. Elect. Eng., 1993.

[22] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter, Particle Filters for Tracking Applications*. Artech House, Boston, London, 2004.

[23] N. Bergman, "Recursive bayesian estimation: Navigation and tracking applications," Ph.D. dissertation, Linkoping University, Sweden, 1999.

[24] A. Doucet, "On sequential monte carlo methods for bayesian filtering," Dept. Eng., Univ. Cambridge, UK," Tech. Rep., 1998.

[25] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statist. Comput.*, vol. 10, no. 3.

[26] J. MacCormik and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," in *Int. Conf. Comput. Vision*, 1999.

[27] J. Carpenter, P. Cilfford, and P. Fearnhead, "Improved particle filter for nonlinear problems." Int. Elect. Eng. Radar, Sonar, Navig., 1999.

[28] D. Crisan, P. D. Moral, and T. J. Lyons, "Non-linear filtering using branching and interacting particle systems," *Markov Processes Realted Fields*, vol. 5, no. 3, pp. 293–319, 1999.

[29] K. Kanazawa, D. Koller, and S. J. Russel, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Eleventh Annual Conf. Uncertainty AI*, 1995.

[30] J. Liu and R. Chen, "Sequential monte carlo methods for dynamical systems," *J. Amer. Statist. Assoc.*, vol. 93, pp. 1032–1044, 1998.

[31] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump markov linear systems," *IEEE Trans. Signal Processing*, vol. 49, pp. 613–624, March 2001.

[32] P. D. Moral, "Measure valued processes and interacting particle systems. application to nonlinear filtering problems," *Ann. Appl. Probab.*, vol. 8, no. 2, pp. 438–495, 1998.

[33] B. Ripleu, *Stochastic Simulations*. New York: Wiley, 1987.

[34] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian non-linear state space models," *J. Comput. Graph. Statist.*, vol. 5, no. 1, pp. 1–25, 1996.

[35] S. Godsill, A. Doucet, and M. West, "Methodology for monte carlo smoothing with application to time-varying autoregressions," in *Int. Symp. Frontiers Time Series Modeling*, Lake Louise, AB, Canada, 2000.

[36] B. Carlin, N. Polson, and D. Stoffer, "A monte carlo approach to nonnormal and nonlinear state-space modeling," *J. Amer. Statist. Assoc.*, vol. 87, no. 418, pp. 493–500, 1992.

[37] W. R. Gilks and C. Berzuini, "Follwoing a moving target-monte carlo inference for dynamic bayesian models," *J. R. Statist. Soc. B*, vol. 63, pp. 127–146, 2001.

[38] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Computation*, vol. 11, no. 1, pp. 229–242, 1999.

[39] T. Zhao and R. Nevatia, "3d tracking of human locomotion: a tracking as recognition approach," in *16th International Conference on Pattern Recognition, 2002*, December 2002.

[40] A. Adam and S. Amershi, "Identifying humans by their walk and generating new motions using hidden markov models," The University of British Coloumbia, Topics in AI: Graphical Models and Computer Animation," Project, December 2004.

[41] G. A.Bekey, "Walking," *The Handbook of Brain Theory and. Neural Networks, MIT press*, pp. 1045–1049, 1995.

[42] Green, R.D., and L. Guan, "Quantifying and recognizing human movement patterns from monocular video images-part i: a new framework for modeling human motion," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 179–190, 2004.

[43] Fastest man in the world calculator. [Online]. Available: www.runnersweb.com/running/fastestm.html

[44] Vodafone Group. (2005) Vodafone 3g/gprs datacard - technical specifications. [Online]. Available: http://www.business.vodafone.com/mobileofficeroot/enuk/support/20_data%cards/10_3gdatacards/20_option_quad3g/30_spec/p_spec.jsp

[45] RoyalTek Company Ltd. (2005) Bluetooth gps receiver manual. [Online]. Available: http://www.royaltek.com/

[46] D. DePriest. (2005) Nmea data. [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm

[47] Precision Navigation INC. (2005) Palm navigator user manual. [Online]. Available: www.precisionnav.com/

[48] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[49] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, A. V. Balakrishnan, G. Dantzig, and L. Zadeh, Eds. McGRAW-Hill Book Comapny, 1965.

[50] M.West and J. Harrison, *Bayesian forecasting and dynamic models*, 2nd ed. Springer Series in Statistics, 1997.