Particle methods for indoor tracking in WiFi networks

Vilhelm von Ehrenheim

Abstract

This thesis treats the problem of positioning in WiFi networks and proposes a solution using hidden Markov models and particle filters based on sequential importance sampling with resampling. Hidden Markov models prove to be a powerful framework for this type of problem exhibiting both an intuitive and adaptive model structure. One of the difficulties encountered when filtering this model is that it is increasing in dimension over time. To get around this problem in this thesis, a method called sequential importance sampling is used which makes it possible to update the estimation sequentially using the previous estimate to calculate the new one. In addition, parameter inference is conducted using expectation-maximization-based likelihood inference in the proposed model. To estimate model parameters in a hidden Markov model smoothing is done to reconstruct the underlying state sequence given all data observations. These states are then used to calculate sufficient statistics for the sought parameters. Finding the smoothing distribution it might be necessary to use a large number of particles in the filter to overcome degeneration of the approximation. A method called fixed-lag smoothing is therefore investigated in order to get better estimates. Finally we illustrate the model and the estimation algorithms on data from a real world application.

Contents

Acknowledgements

I would first and most of all like to thank my supervisor for taking his time to sit with me and explain the theory and concepts involving this theses. His insightful comments have helped me get both a deeper understanding and greater passion for the subject of hidden Markov models.

Secondly I would like to thank Johan Westerborn for his company in the office and great discussions regarding hidden Markov models and last but not least my girlfriend Martina Byttner for her constant support.

Notations and abbreviations

The following notations and abbreviations are used in this paper.

$\mathbb{E}[X]$	the expectation of X
$\mathbb{V}[X]$	the variance of X
$\mathrm{P}(X \in A\|Y)$	the conditional probability of $X \in A$ occurring given Y
$f(x\|y)$	the conditional probability density function of x given y
$\mu \gg \lambda$	μ dominates λ
$\phi_{k\|k}$	the filtering distribution at step k
$\phi_{k\|n}$	the smoothing distribution given observations up to $n > k$
$\phi_{0:k\|n}$	the joint smoothing distribution for all states up to k
$X \sim f$	X is a stochastic variable distributed with density f
$\mathbb{1}\{x \in A\}$	The indicator function equal to 1 when $x \in A$ is fulfilled and zero otherwise

AP	802.11 WLAN access point
BSSID	Basic service set identification
EM	Expectation maximization
HMM	Hidden Markov model
MC	Monte Carlo
RSSI	Received signal strength

CHAPTER 1

Introduction

The *global positioning system* used in many mobile devices today makes it possible to get an accurate and reliable estimate of a device's current position on a map. Most people have used a GPS receiver at some point in their lives and therefore know why this might be of value in a large amount of different scenarios. One limitation of this system is that it depends on satellite communication to estimate the position. This works perfectly fine if the signal from the satellite can travel unhindered; however, as soon as you block this signal, for instance by using the GPS receiver indoors, it will not be able to deliver any reasonable estimate anymore. This makes it impossible to use GPS positioning in indoor environments where positioning might be of interest.

The need for indoor positioning might in a lot of scenarios still be as imminent as it is in the outdoor case, for instance in robotics where positioning is crucial for a large amount of context aware tasks. Another potential area of usage could be for automatic delivery of area specific information in museums and malls or just guidance for people in large buildings such as airports and train stations. It is therefore of interest to look for a different solution, i.e one that does not depend on satellite communication to function but rather some other local signal which preferably is already available in most indoor environments. One such signal could be the IEEE 802.11 signal used for WiFi communication.

Most mobile devices today rely on internet communication, which has created a need for good 802.11 WiFi coverage in public and commercial areas. These networks consist normally of a number of access points placed in different locations all over the building to increase the connection coverage. By measuring the signal strength of these access points it is possible to get a rough estimate of the distance between the same and and the receiver. Knowing a few of these distances should then make it possible to decide the current position of the device in the room. The problem is that this signal is heavily obfuscated by noise. Not only is the signal noisy by itself, moving people, other electric or radio transmitting devices, concrete walls and reflective glass windows all contribute to a very noisy signal environment. This makes it hard to use simple ad-hoc methods such as trilateration. If however the signal is covered in noise it should be possible to filter the signal and increase the accuracy using a more complex statistical model instead of simple measurement of distances.

Despite the interest from several researchers during the last few years, a final solution of this problem is hitherto lacking. Some models presented in different research papers include projection-based approaches such as multiple discriminant analysis described in [**FL09**], where the received signals are projected onto a discriminant space in order to classify the position of the receiver. This method requires a number of reference locations to be measured beforehand in order to be able to classify where in the reference space the current measurement is located. The paper in question focuses mainly on the extraction of information from the signal and compares the method with other similar methods such as principal component analysis. Even if this model shows promising performance it completely

leaves out the possibility to use the movement pattern of the receiver in order to increase the accuracy of its position estimate. Another approach, described in [**LPNM12**], involves classification using the Viterbi algorithm on a discrete *hidden Markov model* (HMM). To make sense this algorithm requires a completely discretized map, which makes the model somewhat blunt. The use of transition probabilities between adjacent squares does make use of movement information concerning the target but does not open for complex movement patterns. In addition, the accuracy is also highly dependent on the granularity of the discretization. The authors of [**SS00**] uses table-based measurements and neural nets for classification which again require some kind of fingerprinting of the signal space in order to classify the current position. Neural networks are known to exhibit good results in some classification problems and the method presented in the mentioned paper showed great accuracy for some specific areas in the data set. This model also require a large set of measurements to be done beforehand in order to classify the position. The lack of restrictions on movement patterns also makes it possible for the classification to move between different areas that are far from each other in the building. Some, as for example [**AMRE12**], have tried a different approach using the movement pattern of people to improve the estimate of position. This was achieved with Monte Carlo-based filters in generalized switching multiple model frameworks. The Markov switching model opened for modeling of a movement pattern with the possibility to switch between different motion sub-models and in order to estimate the position they used a Rao-Blackwellized particle filter where part of the estimate was computed exactly using a Kalman filter step.

Hidden Markov models form a powerful model framework when working with nonlinear, inhomogeneous or otherwise obscured stochastic processes. The basic idea of a Markov process hidden underneath a layer of noise is easily translated to a vast amount of processes present within several fields such as signal processing, finance and biology. The underlying Markov chain makes the process more manageable than models with longer dependence structures and it opens up for more straightforward simulation and approximation schemes. Although since the observations are not Markov the process can still be hard to handle depending on this obfuscating layer of noise.

This paper intends to take an approach similar to the one described in [**AMRE12**], using hidden Markov models and sequential Monte-Carlo based particle filters for proposing a solution to the positioning problem.

Adaptation of a hidden Markov model to this problem proves to be both intuitive and powerful. The model framework opens up for construction of complex models which are still both interpretable and efficient when conducting inference on both hidden states and parameters within the model.

HMM state inference mainly consist of two problems; filtering and smoothing. The first is the on-line problem of estimating the current state given observations up to the same time point, the second is reconstructing a previous states using all observations available. During the course of this thesis both these problems will be addressed in order to present a model which is fit for both state and parameter estimation.

As the number of possible positions increase exponentially with time, this problem puts a strict demand on the computational complexity of the algorithms involved during state inference. A Monte Carlo method that makes it possible to handle this increasing dimensionality is *sequential importance sampling* which plays an important role in this thesis. SIS uses a swarm of particles to estimate the sought distributions. This is done by attaching to each individual particle an importance weight measuring how well the particle estimate meets the target distribution. The

weights together morphs the swarm to fit the target which makes it possible to use more efficient ways of propagating the particles in the state space.

Depending on the model intricacy it however possible that a large amount of particles will be required in order to acquire a reasonable accuracy of the estimates. One way of decreasing the amount needed by adjusting the swarm to the approximated distribution is to use auxiliary weights. These weights can be used for nudging the swarm towards a more tractable area and hence increase the overall accuracy of the same.

In order to estimate model parameters the smoothing distribution is approximated using the same particle filter with some minor modifications. Instead of only looking at the current point, the whole trajectory of the particles is of interest. The complete set of particle trajectories and the corresponding weights can then be used to find a smoothed trajectory where all points have been estimated using all observations.The parameters to the model can then be estimated using this result. In order to find the parameters an algorithm called *expectation maximization* is used. This algorithm calculates estimations of the sought parameters iteratively using the estimate at each step to generate the next until convergence.

In order to avoid degeneracy looking at the whole trajectories, a method called fixed-lag smoothing is used, making it possible to estimate the model parameters for longer sequences and more complex model structures.

In chapter 2 some theory is discussed to remind and update the reader in the main areas approached in this thesis. The main subjects addressed are the theory behind HMMs and some inference theory needed for understanding of the EM algorithm. Chapter 3 goes into detail on the proposed model and the physical relations used to deduce its structure. Once the model is in place the problem of filtering using the SISR method is described in chapter 4. This chapter also discusses how the filter can be modified in order to produce more efficient estimates. The process of estimating the model parameters is described in chapter 5 where also further discussions on degeneracy in the smoothing setting and how ways around it can be found. Fixed-lag smoothing is therefore introduced taking advantage of a forgetting property of the latent chain when evolving conditionally on the observations. In this way the degeneration can be avoided by performing smoothing for shorter, truncated observation sequences. Chapter 6 shortly discusses the methods used for collecting data and chapter 7 presents the numerical results found using the proposed techniques. Finally chapter 8 points out the conclusions drawn from the results.

CHAPTER 2

Preliminaries

Some prior knowledge of probability theory and Monte Carlo methods will be required throughout this thesis. A brief introduction to a few of the central concepts is given in this section.

1. Markov chains

A Markov chain is a discrete time stochastic process that exhibits transitions in some *measurable state space* $(\mathsf{X}, \mathcal{X})$. We will here start by defining measurable, for further reference please confer [**Bil12**, section 2].

DEFINITION 2.1. *A class $\mathcal{A}$ of subsets of some space Ω is said to be a σ-algebra if it contains Ω and is closed under complements and countable unions.*

$$(i) \quad \Omega \in \mathcal{A} \tag{2.1}$$

$$(ii) \quad A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A} \tag{2.2}$$

$$(iii) \quad A_1, A_2, \ldots \in \mathcal{A} \Rightarrow A_1 \cup A_2 \cup \ldots \in \mathcal{A} \tag{2.3}$$

Any set in $\mathcal{A}$ is then said to be measurable.

If the process is memoryless it is said to hold the *Markov property* and a process exhibiting this property in discrete time is called a *Markov chain*. A Markov chain is more formally defined as follows; see [**Str05**, ch. 2] and [**CMR05**, p. 35–41] for further reference.

DEFINITION 2.2. *Let $\{X_k\}_{k=0}^{\infty}$ be a sequence of stochastic variables defined on some probability space $(\Omega, \mathcal{A}, P)$ taking values in a measurable space $(\mathsf{X}, \mathcal{X})$. This process is called a* Markov chain *if for every $k \in \mathbb{Z}^*$ the Markov property holds such that*

$$P(X_{k+1} \in A | X_k, X_{k-1}, \ldots, X_0) = P(X_{k+1} \in A | X_k), \quad \forall A \in \mathcal{X} \tag{2.4}$$

In addition, if the transition probability is independent of k, we call the chain time homogeneous.

Due to the flexibility paired with simplicity, Markov chains have a wide spread use in many different sectors such as physics, biology, economics, chemistry, linguistics, and information sciences. Common examples of use are changes in insurance bonus programmes, random walks in directed graphs, queuing models, and Brownian motion.

The transitions of a Markov chain can be described by a *Markov transition kernel*. This kernel is a function that expresses the probability of moving from one position to another within the state space. Formally we can express this as follows.

DEFINITION 2.3. *Let $(\mathsf{X}, \mathcal{X})$ and $(\mathsf{Y}, \mathcal{Y})$ be two measurable spaces. A* Markov transition kernel *is a function $Q : \mathsf{X} \times \mathcal{Y} \mapsto [0, 1]$ where $Q(x, \mathsf{Y}) = 1$, $\forall x \in \mathsf{X}$. $Q(x, A)$ then denotes the probability of a transition from x to the set $A \in \mathcal{Y}$.*

2. Hidden Markov Models

A *hidden Markov model* (HMM) is a two level stochastic process consisting of a Markov chain $\{X_k\}_{k\geq 0}$, which is not directly observable, and another observable process $\{Y_k\}_{k\geq 0}$, which depends conditionally on the underlying chain $\{X_k\}_{k\geq 0}$, see [**CMR05**, p. 42].

DEFINITION 2.4. *A hidden Markov model is a bivariate stochastic process $\{(X_k, Y_k)\}_{k\geq 0}$, defined on the product space $(\mathsf{X}\times\mathsf{Y}, \mathcal{X}\otimes\mathcal{Y})$, where $\{X_k\}_{k\geq 0}$ is a Markov chain taking values in X and $\{Y_k\}_{k\geq 0}$, defined on some state space Y, is, conditionally on $\{X_k\}_{k\geq 0}$, a sequence of mutually independent random variables such that for every k the conditional distribution of Y_k depends exclusively on X_k.*

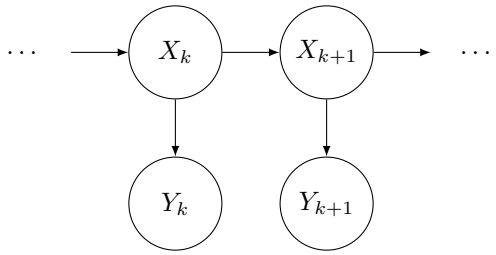

FIGURE 2.1. Schematic graph over the dependence structure of a hidden Markov model

A schematic figure over the dependence structure within a HMM is displayed in *Fig. 2.1.*

Hidden Markov models needs two different transition kernels to explain the transitions within the state spaces. In this paper the transition kernel of the hidden chain is denoted $Q(x,\cdot)$, the transition kernel from the hidden states to the observations is denoted $G(x,\cdot)$. Thus, for each k, $X_{k+1}|X_k = x \sim Q(x,\cdot)$ and $Y_k|X_k = x \sim G(x,\cdot)$. In addition, the probability density of each distribution $G(x,\cdot)$ is denoted by $g(x,y)$. We will frequently suppress the explicit dependence on the observation y_k, which often will be considered as fixed, and instead write $g_k(x) = g(x, y_k)$.

EXAMPLE 2.5. (Bearings only tracking) The problem of bearings only tracking arises in a number of different problems such as radar tracking or submarine detection. In the sense of WiFi positioning this example is a good starting point due to the close resemblance in problem statement with tracking of a moving target.

Consider a moving object for which we can estimate the position only in discrete time steps. The object will then change position between samples according to

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k \\ \dot{x}_{k+1} = \dot{x}_k + T\ddot{x}_k \end{cases}. \tag{2.5}$$

Here T denotes the sample period, $\dot{x}$ the velocity and x_k the position at time k. Assuming that the velocity does not change much over time, the acceleration can be modeled as stationary noise. Supposing this noise is Gaussian, we can model the transitions by

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} u_k \tag{2.6}$$

where u_k is an independent Gaussian noise vector, see [**SARGM04, Jak12**]. As stated above the acceleration is neglected here and variations in velocity are only modeled as Gaussian noise. This somewhat limits the model to handling only well

behaved targets, moving according to linear Gaussian dynamics; we will return to this later in the thesis.

Extending the model to motion in the plane yields the following state equations

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ \dot{x}_{1,k+1} \\ \dot{x}_{2,k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \dot{x}_{1,k} \\ \dot{x}_{2,k} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ 0 & \frac{1}{2}T^2 \\ T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} u_{1,k} \\ u_{2,k} \end{bmatrix}. \tag{2.7}$$

In addition to the state equation (2.7) the measurements at time k is given by the bearing of the tracked target, i.e. the angle between the observer's direction of movement and the position of the target. This leads to an additional equation describing the nonlinear relation between these measurements and the state equation above,

$$y_k = \arctan\left(\frac{x_{1,k}}{x_{2,k}}\right) + v_k, \tag{2.8}$$

where v_k is some, usually Gaussian, measurement noise.

This model describes an underlying unobservable Markov chain with states $X_k = (x_{1,k}, x_{2,k}, \dot{x}_{1,k}, \dot{x}_{2,k})$ and observations (2.8) and hence, clearly this model can be cast into the framework of HMMs. The bearings-only tracking problem is then to estimate the trajectory, $X_{0:k}$ given the observations up to time k as

$$\hat{X}_{0:k|k} = \mathbb{E}\left[X_{0:k}|Y_{0:k}\right] \tag{2.9}$$

As can be seen this problem is increasing in dimension over time and hence calls for some method which can compute the steps recursively.

3. Filtering

Estimation of the current underlying state given the observations up to this point is called *filtering* and this is normally done online, calculating each step as a new observation is aquired. It can be proved that the best estimator in terms of variance is given by the expectation,

$$\hat{X}_{0:k|k} = \mathbb{E}\left[X_{0:k}|Y_{0:k}\right] = \int x_{0:k}\phi_{0:k|k}(x_k)dx_{0:k}, \tag{2.10}$$

where $\phi_{k|k}(\cdot)$ is the kernel of each of the hidden states up to X_k given the observations up to the same time point, $Y_{0:k}$. This distribution is generally called the posterior distribution and finding a good approximation to this distribution is a key problem in solving the expectation of interest.

Rewriting the posterior using Bayes formula it is apparent that this integral is unsolvable in most cases. The denominator of the posterior might not even exist as a closed form expression.

$$\phi_{0:k|k}(x_{0:k}) = \frac{\chi(x_0)g_0(x_0)\prod_{i=1}^{k} g_i(x_i)q(x_{i-1}, x_i)}{\int\cdots\int \chi(x_0)g_0(x_0)\prod_{i=1}^{k} g_i(x_i)q(x_{i-1}, x_i)dx_0\ldots dx_k} \tag{2.11}$$

Here $\chi(x_0)$ is the initial distribution and $g_k(\cdot)$ and $q(x, \cdot)$ is the same densities as defined in section 2.

Even if this integral is unsolveable analytically it might be possible to sample from some function $f_{0:k}(x_{0:k}) \propto \phi_{0:k|k}(x_{0:k})$ which opens up for approximating the expectation using Monte Carlo integration.

4. Smoothing

Inference on all the underlying states in a HMM given all observations up to a current point is called *smoothing*. The best estimator in terms of variance in this setting is the expectation of the states conditional on the observation, as in filtering but extended to cover all states

$$\hat{X}_{0:k|n} = \mathbb{E}\left[X_{0:k}|Y_{0:n}\right] = \int x_{0:k}\phi_{0:k|n}(x_{0:k})dx_{0:k}. \tag{2.12}$$

Here $\phi_{0:k|n}(\cdot)$ denotes the posterior joint smoothing distribution of the hidden states given all observations for each state. Finding this distribution is usually more intricate than in the case of filtering and will hence also require Monte Carlo methods to find a solution.

5. Monte Carlo integration

The basic idea behind Monte Carlo integration is to draw samples from the distribution of a random variable X and use the samples to estimate the expectation of a function $h(X)$, called the target function,

$$\mathbb{E}\left[h(X)\right] = \int_{\mathsf{X}} h(x)f_X(x)dx, \tag{2.13}$$

where $f_X(x)$ is the probability density function for X.

Let $\{\xi^i\}_{i=1}^N$ be N independent samples from the distribution of X, then it follows directly from the *strong law of large numbers*, see [**Gut09**, p. 182],[**CMR05**, p. 553], that the mean of these samples will converge to the true expectation,

$$\frac{1}{N}\sum_{i=1}^{N} h(\xi^i) \to \mathbb{E}\left[h(X)\right], \text{ a.s. as } N \to \infty. \tag{2.14}$$

Example 2.6. Suppose that you want to estimate the integral

$$\tau = \int_0^{2\pi} x \sin\left(\frac{1}{\cos\left[\log(x+1)\right]}\right) dx, \tag{2.15}$$

then we can rewrite it as an expectation

$$\tau = \int 2\pi x \sin\left(\frac{1}{\cos\left[\log(x+1)\right]}\right) f_X(x)dx, \tag{2.16}$$

where $f_X(x) = 1/2\pi, \quad 0 \leq x \leq 2\pi$ is the probability density function of the uniform distribution $Unif(0, 2\pi)$. Drawing N random variables $\{x_i\}_{i=1}^N$ from $Unif(0, 2\pi)$ it is possible to approximate the integral as

$$\tau_n = \frac{1}{N}\sum_{k=1}^{N} x_i \sin\left(\frac{1}{\cos\left[\log(x_i+1)\right]}\right). \tag{2.17}$$

Which for $N = 10^6$ gave the following result in 0.049113 seconds when entered into *MATLAB*,

$$\tau_n = 2.7369$$

As a comparison *WolframAlpha*®, solving it symbolically, gave the answer 2.73538.

6. Importance sampling

Simulating efficiently from arbitrary distributions may not always be so straightforward. For many known parametric distributions it is possible to sample directly using the inverse method but most often the posterior, or more generally the *target distribution*, is much more complicated and might only be known up to a normalizing constant. In these cases one might approximate the integral by simulating from another distribution that *dominates* the posterior distribution and weighing or alternatively discarding a certain amount of the results to approximate the real posterior. Let us define the concept of a dominating distribution [**LC98**, p. 14].

DEFINITION 2.7. *A probability distribution R defined on $(\mathsf{X}, \mathcal{X})$ is said to* dominate *another distribution Q, denoted $R \gg Q$, if the support of R is larger than that of Q, such that for every set $N \in \mathcal{X}$, $R(N) = 0 \Rightarrow Q(N) = 0$.*

This makes it possible to rewrite the sought integral as

$$\mathbb{E}\left[h(X)\right] = \int_{\mathsf{X}} h(x) \frac{f_X(x)}{\pi(x)} \pi(x) dx, \tag{2.18}$$

which is defined since $\pi \gg f_X$. This opens up the possibility to sample from the dominating density $\pi(x)$, called the *proposal density*, instead of $f_X(x)$ and then update the estimator accordingly, preserving the expectation, see [**RC04**, p. 84].

$$\mathbb{E}\left[h(X)\right] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f_X(\xi^i)}{\pi(\xi^i)} h(\xi^i). \tag{2.19}$$

It is easy to identify this estimator as a weighted average with weights $\omega^i = \frac{f_X(\xi^i)}{\pi(\xi^i)}$, corresponding to the difference in probability density between the two distributions. If the density $f_X(x)$ is known only up to a normalizing constant one might still use the same method but the weights will not sum up to N. It is easy to circumvent this problem by simply normalizing the weights, providing the estimator

$$\mathbb{E}\left[h(X)\right] \approx \sum_{i=1}^{N} \frac{\omega^i}{\sum_{k=1}^{N} \omega^k} h(\xi^i). \tag{2.20}$$

This method greatly simplifies solving complicated integrals but still does not solve the issue of increasing dimensionality as mentioned previously. In order to do this we need to change the method so that it is possible to update the estimate on-line, i.e. by using previous, already calculated estimates to estimate the new position as new data arrives.

7. Sequential Importance Sampling

Moving back to the original problem of estimating underlying states where we want to solve the integral in (2.12), it would be preferable if it were possible to update the estimates sequentially to refrain from using an estimator that grows in complexity with the number of time steps. This idea was originally proposed in [**GSS93**] and consists of drawing a swarm of so called particles $(\xi^i_{0:k})_{i=1:N}$ and using these to estimate the process recursively.

The key trick for achieving this sequential updating is to choose a proposal density such that

$$\pi_k(x_{0:k}) = \pi_k(x_k | x_{0:k-1}) \pi_{k-1}(x_{0:k-1}). \tag{2.21}$$

This way it is possible to sample each new step recursively conditionally in the old samples and then append these to the previous sequence.

Looking at the importance weights, these may also be computed recursively according to

$$\omega_k^i = \frac{\phi_k(\xi_{0:k}^i)}{\pi_k(\xi_{0:k}^i)} = \frac{\phi_k(\xi_{0:k}^i)}{\phi_{k-1}(\xi_{0:k-1}^i)\pi_k(\xi_k^i|\xi_{0:k-1}^i)}\omega_{k-1}^i, \tag{2.22}$$

such that $f_k(x) = \phi_k(x)/c_k$. As the weights are normalized this normalizing constant will cancel out. This makes it possible to sample from a larger set of functions and not only densities.

We then form a filter which simply draws different realizations from the proposal distribution, π_k, and calculates a weight for each of these depending on the difference between the selected distribution and the posterior. Identifying f_k in (2.11) we can see that the denominator is only a normalizing constant and we get

$$f_k(x_{0:k}) = \chi(x_0)\prod_{i=1}^{k} g_i(x_i)q(x_{i-1}, x_i) \tag{2.23}$$

resulting in the recursive weight update

$$\omega_k^i = \omega_{k-1}^i \times g_k(\xi_k^i)\frac{q(\xi_{k-1}^i, \xi_k^i)}{\pi_k(\xi_k^i|\xi_{0:k-1}^i)} \tag{2.24}$$

The ratio between the transition density and the proposal density is more formally called a Radon-Nikodym derivative denoted

$$\frac{dQ(\xi_{k-1}, \cdot)}{dR_k(\xi_{k-1}, \cdot)}(\xi_k^i),$$

which is defined for all $R_k \gg Q$. Seeing this as a derivative also helps the notion of weighing as depending on the relative changes in probability between proposal and transition kernels. The change of distribution can then be seen as a change of basis when integrating over probability spaces where the Radon-Nikodym derivative is corresponding to the Jacobian determinant in multivariate integration.

The method described above is called Sequential Importance Sampling (SIS), as described in [**CMR05**, p.217], and we define it in a more concise form in *Algorithm 1* below.

Algorithm 1 SIS

1: Draw an i.i.d sample $\left(\xi_0^1, \ldots, \xi_0^N\right)$ from the initial distribution ρ_0 and set

$$\omega_0^i = g_0(\xi_0^i)\frac{d\nu}{d\rho_0}(\xi_0^i)$$

2: **for** $k = 0, 1, \ldots$ **do**

3: Draw $\left(\xi_0^1, \ldots, \xi_0^N\right)$ conditionally, independently from the instrumental kernel $\xi_{k+1}^i \sim R_{k+1}(\xi_k, \cdot)$ given the previous particles $\{\xi_{0:k}^j,\ j = 1, \ldots, N\}$.

4: Append the new particles to the set of previous ones,

$$\xi_{0:k+1}^i = (\xi_{0:k}^i, \xi_{k+1}^i), \quad \forall i$$

5: Update importance weights

$$\omega_{k+1}^i = \omega_k^i \times g_{k+1}(\xi_{k+1}^i)\frac{dQ(\xi_k^i, \cdot)}{dR_{k+1}(\xi_k^i, \cdot)}(\xi_{k+1}^i), \quad \forall i$$

6: **end for**

8. Degeneration of importance weights

Optimally there will always be a large number of particles with a large likelihood and the average of these will give us a good approximation of the posterior. This will, however, not be the case in practise. Eventually, all particles will get small importance weights over time since they are drawn from another distribution than the actual posterior. When increasing the dimension of this distribution there will be a smaller and smaller chance of drawing a particle that actually falls within the main body of the posterior distribution. Because of this only a few particles will keep their large weights and the weighted average of the particles will form a poor approximation of the true posterior distribution.

This degeneration might not seem as a problem at first glance, since the estimator normalizes over all the weights. In this way the particles with a small weight will just not contribute to the estimate. But if this particle does not contribute to the swarm any more it also means the information is deteriorating within the particle swarm. Consider a random walk on $\mathbb{R}^2$. The further we walk the total number of possible ways increase exponentially and the probability of a particle walking the correct path is therefore decreasing at the same rate. Then after some time the particle swarm will only, if at all, hold a few particles that contain any sort of valuable information. This results in an inefficient estimator of the position that wastes a lot of computational power on calculating particles that do not provide any information, and since only a few particles hold any information the variance of the estimator will increase as well.

Since this degeneracy effectively deteriorates the estimation it might be of interest to be able to detect when the set of weights start to become imbalanced or in some other way measure the amount of information the swarm holds concerning the filtering distribution. One way is the *effective sample size* N_{eff} [**CMR05**, p. 235] given by

$$N_{\text{eff}}(k) = \left[\sum_{i=1}^{N} \left(\frac{\omega_k^i}{\sum_{j=1}^{N} \omega_k^j} \right)^2 \right]^{-1} \tag{2.25}$$

This measure approximates the amount of particles contributing to the approximation of the posterior. Another good measure is the *Shannon entropy* of the weights [**CMR05**, p. 235],

$$\text{Ent}(k) = -\sum_{i=1}^{N} \frac{\omega_k^i}{\sum_{j=1}^{N} \omega_k^j} \log_2 \left(\frac{\omega_k^i}{\sum_{j=1}^{N} \omega_k^j} \right) \tag{2.26}$$

which is the expectation of the self-information contained in the weights. This essentially measures the amount of uncertainty in the set of weights, reaching a maximum when the particles have uniform weights. Because of the logarithm the entropy makes it easier to spot differences when few particles have large weights.

These methods only provide ways of measuring the number of valuable particles. This makes it possible to know if the SIS algorithm still gives a good estimate but the degeneration is still imminent and prohibits the usage of the SIS algorithm for larger problems or longer periods of time.

9. Sequential Importance Sampling w. Resampling

The solution to degeneracy is to use something called resampling. The idea, originally presented in [**GSS93**], is simple but brilliant. After the calculation of weights in the SIS algorithm one simply draws a set of new particles with replacement from the distribution of particles with probability mass function given by the

normalized weights. This essentially multiplies the good particles and kills off the ones having very small weights.

The distribution of particles and their respective weights approximate the filtering distribution, resampling with replacement from this distribution will not change the expectation of the distribution itself, thus at worst only retaining the previous bias of the estimator. It will nevertheless introduce some additional variance [**CMR05**]. But this increase in variance is in general a small price to pay for a stable algorithm. Remember as well that the variance of the estimator still increases if fewer particles hold valuable information. In some cases the resampling might conduce to a need for a larger amount of particles in order to ensure the same speed of convergence.

Since resampling introduces additional variance and since it also increases the computational complexity of the algorithm, it might be of interest to resample as little as possible. One can use the measures described in the previous section to set up a limit on the number of effective particles and then resample once this limit is breached. This way it is possible to hold the increase in variance at bay and at the same time keep a stable algorithm.

This extension of the SIS algorithm is called Sequential Importance Sampling with Resampling [**CMR05**, p. 237], the full algorithm is stated in *Algorithm 2*.

Algorithm 2 SISR

1: Draw an i.i.d sample $\left(\xi_0^1, \ldots, \xi_0^N\right)$ from the initial distribution ρ_0 and set

$$\omega_0^i = g_0(\xi_0^i)\frac{d\nu}{d\rho_0}(\xi_0^i)$$

2: **for do** $k = 0, 1, \ldots$

3: Draw $(\tilde{\xi}_{k+1}^1, \ldots, \tilde{\xi}_{k+1}^N)$ conditionally independently from the instrumental kernel $\tilde{\xi}_{k+1}^i \sim R_{k+1}(\xi_k^i, \cdot),\ i = 1, \ldots, N.$ given the previous particles up to step k, $\{\xi_{0:k}^j,\ j = 1, \ldots, N\}$.

4: Update importance weights

$$\omega_{k+1}^i = g_{k+1}(\tilde{\xi}_{k+1}^i)\frac{dQ(\xi_k^i, \cdot)}{dR_{k+1}(\xi_k^i, \cdot)}(\tilde{\xi}_{k+1}^i)$$

5: Draw, conditionally independently given $\{(\xi_{0:k}^i, \tilde{\xi}_{k+1}^j),\ i, j = 1, \ldots, N\}$, a set of N indexes, $(I_{k+1}^1, \ldots, I_{k+1}^N)$, with probability mass function given by the respective normalized importance weights,

$$\frac{\omega_{k+1}^1}{\sum_{j=1}^N \omega_{k+1}^j}, \ldots, \frac{\omega_{k+1}^N}{\sum_{j=1}^N \omega_{k+1}^j}$$

6: Reset importance weights to some constant

7: Update the particle trajectory

$$\xi_{0:k+1}^i = (\xi_{0:k}^{I_{k+1}^i}, \tilde{\xi}_{k+1}^{I_{k+1}^i}), \quad \forall i$$

8: **end for**

10. Inference theory

10.1. Sufficient statistics. Consider a random variable X with a given parametric distribution P_θ. If one were to estimate the parameters θ using a number of realisations of X, $x_{0:n}$, it might be so that the observations could be reduced in some way without any loss of information about the specific parameter. Say it would be possible to extract the necessary information from the data using some

statistic $S(X)$. This statistic is said to be *sufficient* if it is containing all information about θ. A more rigorous definition is given below.

DEFINITION 2.8. *A statistic $S(X)$ is said to be* sufficient *for the parameter θ or for X distributed according to the family of distributions $\mathcal{P} = \{P_\theta, \theta \in \Omega\}$, if the conditional probability of X given $S = s$ is independent of θ for all s.*

This definition can be interpreted as the sufficient statistic S holding all the information about θ such that if S is given, the probability can no longer depend on θ as all information known. Yet, the definition given does not provide much help in finding sufficient statistics for a certain parameter. A convenient way to find a sufficient statistic for a distribution is *Fisher's factorization criterion.*

DEFINITION 2.9. (Fisher's factorization criterion) *A statistic $S(X)$ is sufficient for θ if and only if there exists non-negative functions h and g such that the probability density function of X can be factorized as*

$$f_\theta(x) = h(x)g_\theta(S(x))$$

For more information on sufficient statistics see [**LC98**, chapter 1.6].

10.2. Exponential families. A very common family of distributions is the *exponential family.*

DEFINITION 2.10. *A family of distributions $\mathcal{P} = \{P_\theta, \theta \in \Omega\}$ is said to form an s-dimensional* exponential family *if the probability densities of the distributions in $\mathcal{P}$ can be written on the form*

$$p_\theta(x) = \exp\left[\sum_{i=1}^{s} \eta_i(\theta)T_i(x) - B(\theta)\right] h(x)$$

where η_i and B are real valued functions.

Using this definition and the factorization criterion it is possible to deduce that there exists sufficient statistics for all distributions which are members within the exponential family. For further reference on exponential families and their use in inference theory the interested reader is referred to [**LC98**].

CHAPTER 3

Model for dynamical tracking in WiFi networks

1. The relation between position and RSSI

The received signal strength indication is a measure on IEEE 802.11 signal strength given in dBm. This unit is defined as

$$Y = 10\log_{10}(P) + 30 \tag{3.1}$$

where P is the power of the transmitted signal in watt and Y the received RSSI. The power of the signal decreases with the square of the distance from the source. This is easily seen by letting the power of the source be projected on the surface area of a sphere with the distance to the receiver being the radius. Indoor environments are not optimal settings for signals because of walls and people blocking and dampening the transmitted signal even further. It has been shown that the signal decreases at a power of 4 in a common indoor environment. The resulting relation between position and RSSI will then be

$$Y = 10\log_{10}\left(\frac{c_l}{\left(\sqrt{(x-\tilde{x}_l)^2 + (y-\tilde{y}_l)^2}\right)^4}\right) + 30 \tag{3.2}$$

where c_l is a constant corresponding to the transmit power of AP number l. Simplifying and combining constants we get

$$Y = -20\log_{10}\left((x-\tilde{x}_l)^2 + (y-\tilde{y}_l)^2\right) + D_l \tag{3.3}$$

where D_l denotes the combined constants in the RSSI domain. Note that the position of the access points, $(\tilde{x}_l, \tilde{y}_l)$ is considered to be known parameters in this expression.

One more thing to take into account is the height difference between the access point (AP) and the target. This height will not make much difference on longer distances but will definitely effect the result when close by. To account for this the height needs to be added to the distance part in (3.2), as easily derived from Pythagora's theorem, resulting in

$$Y = -20\log_{10}\left((x-\tilde{x}_l)^2 + (y-\tilde{y}_l)^2 + h_l^2\right) + D_l \tag{3.4}$$

2. The state space model

The problem of positioning in a WiFi-network is similar to the bearings only tracking problem described before. The major differences are that it is now the target itself that does the tracking and also that the measurements are not bearings but received signal strength indication (RSSI) per available access point. But more on this later.

The movement in the bearings only tracking example above assumes Gaussian changes in velocity. The changes in velocity of a walking person indoors will, however, most likely not change according to Gaussian noise. Furthermore the majority of the changes will be in direction or change to a sudden halt. So let us extend the model to include this.

A possible modification, to account for changes in direction, corresponding to turning, could be to include a state variable for the angular acceleration, α_k, and then update the velocity accordingly for each time step depending on this change in direction. This is done by extending the model in (2.7) by adding a rotation operator to the velocity parts in the transition matrix and the primitive of this operator handling the change in position. This gives the following dynamical system which was also discussed by [**AMRE12, SARGM04**].

$$\begin{aligned}\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ \dot{x}_{1,k+1} \\ \dot{x}_{2,k+1} \end{bmatrix} = &\begin{bmatrix} 1 & 0 & \frac{\sin(\alpha_k T)}{\alpha_k} & -\frac{1-\cos(\alpha_k T)}{\alpha_k} \\ 0 & 1 & \frac{1-\cos(\alpha_k T)}{\alpha_k} & \frac{\sin(\alpha_k T)}{\alpha_k} \\ 0 & 0 & \cos(\alpha_k T) & -\sin(\alpha_k T) \\ 0 & 0 & \sin(\alpha_k T) & \cos(\alpha_k T) \end{bmatrix} \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \dot{x}_{1,k} \\ \dot{x}_{2,k} \end{bmatrix} + \\ &+ \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix} \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ u_{3,k} \\ u_{4,k} \end{bmatrix}\end{aligned} \tag{3.5}$$

where T denotes the sample period and $u_{1:4,k}$ are Gaussian noise variables at time k accounting for other changes in acceleration. The matrix multiplied with the Gaussian vector derives from the changes in velocity due to the noise. This is given by $\dot{x}_{k+1} = T\ddot{x}_k$ and the same principle holds for the position yielding $x_{k+1} = \frac{1}{2}T^2\ddot{x}_k$. The angular acceleration is updated at each step by drawing from the conditional distribution $\alpha_k \sim p(\alpha_k|\alpha_{k-1})$.

It is important to point out that this turning model only handles a target that is turning. The matrix given by the limit where $\alpha_k \to 0$ is the matrix in (2.7) which is equivalent to walking straight. The matrix is however not defined in 0 and walking indoors will probably switch between these two patterns, that is periods of straight walking will be followed by sudden turns. It is therefore of interest to combine the model for constant velocity and the model for coordinated turns in some way. Since a person seldom moves indoors without a halt it would also be reasonable to include in the model a state of standing still.

A way to account for the switching between different movement patterns could be to add another dynamics to the model, namely a maneuver chain (M_k). Let this chain have Markovian evolution between the states, walking straight (W), turning (T), stopping (S), and starting to walk (I), shown in *Fig. 3.1.* The maneuver chain affects primarily the original model by changing the transition matrix depending on the current state in the maneuver chain, M_k. Resulting in $\alpha_k \sim f(\alpha_k|\alpha_{k-1}, M_k)$ and $M_k \sim p(M_k|M_{k-1})$.

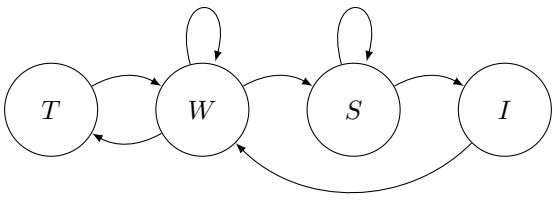

FIGURE 3.1. Schematic figure over the maneuver chain's state space.

Since all transitions between the states in our state space are Markovian and since the underlying states are not directly observable, this model fits a hidden Markov model. Denoting the states $(x_k, y_k, \dot{x}_k, \dot{y}_k, \alpha_k)$ as X_k, the maneuver chain

as M_k and the observations as Y_k we have a model structure as depicted in *Fig. 3.2*. This figure clearly shows the HMM structure where each observation Y_k only depends on the underlying position X_k, which in turn depends on the previous position X_{k-1} and the current maneuver M_k.

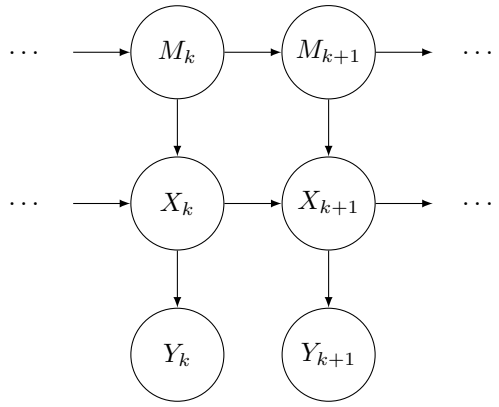

FIGURE 3.2. Schematic graph over the transitions in the Hidden Markov model

The generalized state space equations for this extended model are given below.

$$\begin{aligned} M_{k+1} &\sim p(M_{k+1}|M_k), \\ \alpha_{k+1} &\sim f(\alpha_{k+1}|\alpha_k, M_{k+1}), \\ X_{k+1} &= A(\alpha_{k+1}, M_{k+1})X_k + RU_k, \\ Y_{k+1} &= B(X_k) + SV_k \end{aligned} \tag{3.6}$$

Here, $p(M_{k+1}|M_k)$ is the transition probabilities for the maneuver chain, $f(\alpha_{k+1}|\alpha_k, M_{k+1})$ the conditional density for the angular acceleration, $A(\alpha_k, M_k)$ the conditional matrix taking the underlying state as a parameter. These are defined as

$$A(\alpha_k, M_k) = \begin{cases} \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & M_k = W \\ \begin{bmatrix} 1 & 0 & \frac{\sin(\alpha_k T)}{\alpha_k} & -\frac{1-\cos(\alpha_k T)}{\alpha_k} \\ 0 & 1 & \frac{1-\cos(\alpha_k T)}{\alpha_k} & \frac{\sin(\alpha_k T)}{\alpha_k} \\ 0 & 0 & \cos(\alpha_k T) & -\sin(\alpha_k T) \\ 0 & 0 & \sin(\alpha_k T) & \cos(\alpha_k T) \end{bmatrix}, & M_k = T, I \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & M_k = S \end{cases} \tag{3.7}$$

R is the matrix which transforms the acceleration noise to position and velocity noise, given by

$$R = \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix}, \quad M_k = W, T, I, S \tag{3.8}$$

The function $B(X_k)$ computes the signal strength of the access points Y_k given the current states, X_k, as given in (3.4), and V_k is a Gaussian variable representing additive measurement noise.

When in state S, both the position and the velocity state are held constant apart from some noise. This noise is needed to make sure that the complete likelihood does not collapse to a Dirac pulse. Even if the true velocity is zero when standing still the velocity is kept as the previous value so that it might be used to initialize the chain in a similar direction once the target starts moving again. This might seem as using an older state which would violate the Markov property but since the updating still takes place the transitions will only depend on their current values preserving the Markov property of the chain. Observe that the kernel itself is changed so that the velocity does not change while in the S state. The velocity will in that sense hold false information while the chain is in this mode.

3. Sub-model distributions and parameters

With a complete model structure the only missing part before it is possible to simulate from the model is to decide which distributions the underlying parameters should have. Not only should the transition probabilities in the maneuver chain be set, but also the distributions from where we draw the angular acceleration. These distributions corresponds to the prior knowledge of model parameters and should therefore be selected to mirror how a person moves indoors.

To decide which distributions to set as these priors one must look at the movement patterns of people. The model structure takes care of selection of the different sub-models but once the maneuver chain is in state (T) the distribution of angular accelerations should cover the possible turn rates a person moves according to. Looking at a map of a building it is quite reasonable to think that most turns

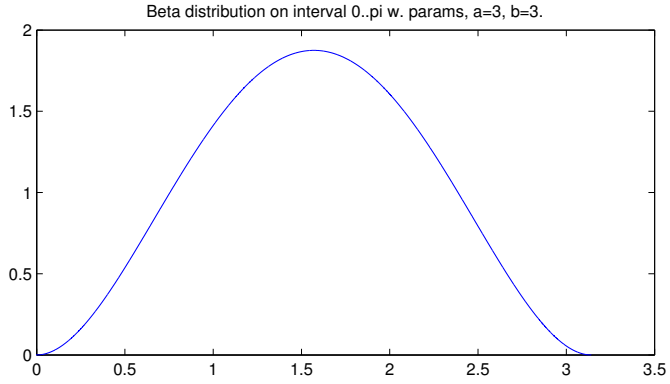

FIGURE 3.3. Beta distribution between 0 and π with $\alpha = 3$, $\beta = 3$.

will be sharp and usually with a steep angle close to $\pi/2$. After observing people moving indoors the mean turn rate of $\pi/2$ per 2 seconds was chosen. The different possible angles were selected to be everything between 0 and $\pm\pi$ radians with a maximum in $\pm\pi/2$ according to a $Beta(3, 3)$ distribution as displayed in *Fig. 3.3*. This distribution was selected due to the fact that it has a limited interval with good possibility for configuration of the mode.

This distribution of angular acceleration will in 1 second result in a change in velocity according to *Fig. 3.4* assuming the initial velocity was $(1, 0)$ and the resulting change in position is displayed in *Fig. 3.5*. In addition to this there will

be an additive Gaussian noise accounting for all other changes as described earlier.

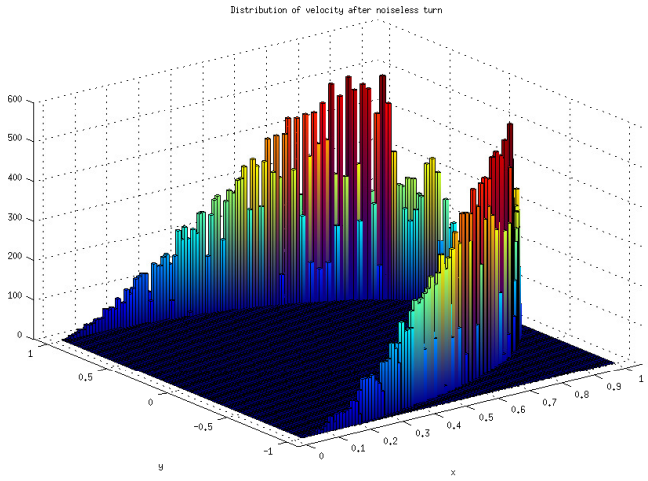

FIGURE 3.4. Distribution of velocities after one second of turning, given the starting velocity $(1, 0)$.

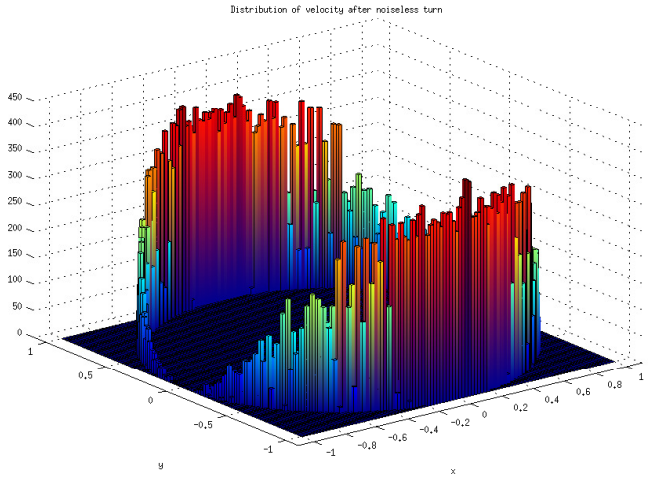

FIGURE 3.5. Distribution of velocities after two seconds of turning, given the starting velocity $(1, 0)$.

Walking straight will be modeled by the matrix in (3.7) with a Gaussian additive noise.

The (I) state will draw the angular acceleration uniformly, which opens up for the possibility of very sharp turns such as changing direction completely. The previous direction will however still be the expected direction.

The transition probabilities will be estimated using parameter inference discussed later in the thesis.

CHAPTER 4

On-line filtering algorithm

1. Initial implementation, the Bootstrap filter

The initial implementation of the SISR algorithm was done using the transition kernel of the hidden chain as the instrumental kernel for the filter. This type of filter is called a bootstrap filter, originally proposed in [**GSS93**]. The gain in sampling from the transition density itself is mainly two things. The first is that it is normally very easy to sample from this kernel since it only requires implementing the model structure. The other one is that the calculation of the incremental weights is greatly simplified as the Radon-Nikodym derivative will be 1, see line 4 *Algorithm 2*, resulting in the simple expression

$$\omega_{k+1}^i = \omega_k^i g_{k+1}(\tilde{\xi}_{k+1}^i) \tag{4.1}$$

It is then possible to implement a filter which for each step propagates each of the particles according to the model structure and then calculates importance weights using the local likelihood.

2. The auxiliary filter

The bootstrap filter may be easy to implement but it is far from optimal when it comes to variance of the estimation as it propagates the particles "blindly", without utilizing any of the information contained in the observations to move the swarm. This will affect the filter performance as an increased amount of particles will be needed in order to get a good approximation of the posterior distribution. Since the run-time complexity of the algorithm is dependent on the number of particles this will lead to a large increase in computational intensity. It might therefore be of interest to propagate the particles more efficiently using some dynamics different from that of the hidden chain. Better would be to use the *optimal kernel* given by

$$T_k(x,h) = \frac{\int h(x')q(x,x')g_{k+1}(x')dx'}{\int q(x,x')g_{k+1}(x')dx'}. \tag{4.2}$$

The optimal kernel comprises information concerning both the state dynamics and the current observations, see [**RC04**]. This results in a lower variance of the estimation. There is however one problem with using this kernel: it is in many cases not feasible to sample from it. The optimal kernel may still however offer ways to improve the propagation of the particles in order to lower the variance of the estimator. Modifying the proposal kernel to fit the model better is a way to adapt the filtering algorithm to the model under consideration. An early example of adaptation of particle filters is the auxiliary particle filter proposed by [**PS99**]. Auxiliary particle filters increase weights for particles more likely to do well, by calculation of an extra auxiliary weight for each particle. This weight, which is multiplied with the original weight before selection, is divided out in the calculation of the subsequent weights in order to keep the bias of the approximation unchanged.

Initially *pilot particles*, see [**OR11**], were used to move the swarm towards the more likely long term survivors. These pilot particles are an extra propagation of the particles testing the ground before the real propagation is performed. In this

way the particles that are more likely to do well at the next propagation are given a larger weight and will thus be more likely to survive the selection step. This means that most particles will be more similar to the posterior distribution and thus the variance of the filtering is reduced. The extra propagation will of course harm the running time of the algorithm but this improvement might still be of value due to the reduction in variance.

The auxiliary filter is defined Algorithm 3. For a more in-depth analysis of auxiliary particle filters consult [**OR11**].

Algorithm 3 APF with pilot sampling

for $i = 1..N$ **do**
  simulate $\xi_0^i \sim \rho_0$
  set $\omega_0^i \leftarrow g_0(\xi_0^i)\frac{d\nu}{d\rho_0}(\xi_0^i)$
  set $\vartheta_0^i \leftarrow 1$
end for
for $k = 0..n-1$ **do**
  for $i = 1..N$ **do**
    simulate $I_{k+1}^i \sim \left(\frac{\omega_k^\ell \vartheta_k^\ell}{\sum_{j=1}^N \omega_k^j \vartheta_k^j}\right)_{\ell=1}^N$
    simulate $\zeta_{k+1}^i \sim R_k(\xi_k^{I_{k+1}^i}, \cdot)$
    simulate $\xi_{k+1}^i \sim R_k(\xi_k^{I_{k+1}^i}, \cdot)$
    calculate $\omega_{k+1}^i = \frac{g_{k+1}(\xi_{k+1}^i)}{\vartheta_k^{I_{k+1}^i}} \frac{dQ(\xi_k^i, \cdot)}{dR_{k+1}(\xi_k^i, \cdot)}(\xi_{k+1}^i)$
    calculate $\vartheta_{k+1}^i = g_{k+1}(\zeta_{k+1}^i)\frac{dQ(\xi_k^i, \cdot)}{dR_{k+1}(\xi_k^i, \cdot)}(\zeta_{k+1}^i)$
    append $(\xi_{k+1}^i, \omega_{k+1}^i)$ to the set of particles and their weights
  end for
end for

The pilot particles act as estimates of the conditional expectation of the next hidden state and will when inserted into the local likelihood serve as an estimate of the next filtering distribution. This estimate will be unbiased but far from optimal. A way to reduce the variance of the estimate and at the same time reduce the computational complexity is to derive the expected position and let this serve as the base of the auxiliary weight calculation instead of the pilot particles. This expectation might not always be so simple to calculate but it might be possible to approximate it. This way a completely analytical expression is acquired, if possible, and no extra particle propagation is needed to calculate the auxiliary weights.

Using the model structure and the tower property we can find a good start for the calculation of the expectation.

$$\begin{aligned}
\mathbb{E}\left[X_{k+1}|X_k, M_k\right] &= \mathbb{E}\left[\mathbb{E}\left[X_{k+1}|X_k, M_k, M_{k+1}\right]|X_k, M_k\right] \\
&= \sum_j p_{M_k,j}\mathbb{E}\left[X_{k+1}|M_{k+1} = j, X_k\right] \\
&= \sum_j p_{M_k,j}\mathbb{E}\left[A(\alpha_{k+1}, M_{k+1})X_k|M_{k+1} = j, X_k\right] \\
&= \sum_j p_{M_k,j}\mathbb{E}\left[A(\alpha_{k+1}, M_{k+1})|M_{k+1} = j\right] X_k
\end{aligned} \tag{4.3}$$

A way to calculate an approximation of this expectation is to do a Taylor series expansion in the mean and then take the expected value of the first few terms.

$$\begin{aligned}
&\mathbb{E}\left[f(X)\right] = \mathbb{E}\left[f(\mu_X + (X - \mu_X))\right] = \\
&= \mathbb{E}\left[f(\mu_X) + f'(\mu_X)(X - \mu_X) + \frac{f''(\mu_X)}{2}(X - \mu_X)^2 + \mathcal{O}((X - \mu_X)^3)\right] \approx \\
&\approx f(\mu_X) + f'(\mu_X)\mathbb{E}\left[X - \mu_X\right] + \frac{f''(\mu_X)}{2}\mathbb{E}\left[(X - \mu_X)^2\right] = \\
&= f(\mu_X) + \frac{f''(\mu_X)}{2}\mathbb{V}\left[X\right]
\end{aligned} \tag{4.4}$$

where μ_X is the expectation of X.

The full calculation of (4.3) using this approximation is given in *Appendix 8*. The resulting approximation of the expected step serves as a prediction, giving additional weight to particles with expected high future fitness. Since the approximation is completely analytical it will be used instead of the pilot particles in order to reduce the execution time of the algorithm.

CHAPTER 5

Estimation of model parameters

The proposed model holds a fair amount of unknown parameters which most probably differs between different locations. In order to be able to use the model on real data it must be possible to calibrate the parameters of the model. This section will focus on an algorithm called Expectation Maximization (EM) and how this method can be used in order to estimate unknown parameters.

A first step is to identify which parameters the model consists of and to identify the parameters that can be considered known beforehand. Looking at the probability density functions used in the model we can easily identify the parameters listed in *Table 1*.

Parameter	Model property
D_l	AP signal strength
h_l	AP height over ground
$\tilde{x}_l$	AP x-coordinate
$\tilde{y}_l$	AP y-coordinate
σ_v^2	Measurement noise variance
σ_u^2	Acceleration noise variance
p_{ij}	Maneuver transition probability

TABLE 1. Table of model parameters

among the above parameters the AP locations and height over ground can be considered known. The remaining parameters will have to be estimated using the mentioned algorithm.

1. Expectation Maximization algorithm

A common algorithm used for parameter estimation in HMMs is the expectation maximization, or EM, algorithm originally suggested in [**DLR77**]. The idea behind the algorithm, given in a general form in *Algorithm 4*, originates from maximum likelihood estimation and consists of maximizing the likelihood through recursive maximization of another quantity called the *intermediate quantity* $\mathcal{Q}(\cdot, \theta^i)$.

DEFINITION 5.1. *The* intermediate quantity *of EM is the family of functions defined as*

$$\mathcal{Q}(\theta; \theta') = \mathbb{E}_{\theta'} \left[\log \mathcal{L}^c(\theta; x) \right] \tag{5.1}$$

where $\mathcal{L}^c$ is the complete data Likelihood.

Something which is of interest when conducting inference in hidden Markov models is the *complete data likelihood*.

DEFINITION 5.2. *The* complete data likelihood*, denoted $\mathcal{L}_n^c(\theta; x_{0:n}, y_{0:n})$, is the likelihood of the parameter θ given all data, including the hidden states $X_{0:n}$, up to*

point n. Strictly the complete data likelihood is the function defined as

$$\theta \mapsto p_\theta(x_{0:n}, y_{0:n}). \tag{5.2}$$

It can be shown that an increase in the intermediate quantity implies an increase of the log likelihood [**CMR05**, p. 351]. Since every step in EM maximizes the intermediate quantity the algorithm is a monotone optimization of the likelihood function.

A fact that should not be ignored is that a gradient-based method for optimization of the likelihood function might work just as well and will in some cases perform better than the EM algorithm [**CMR05**, p. 359]. The EM algorithm was chosen in this paper due to its robustness and easy implementation.

Algorithm 4 EM algorithm

1: **for** i=1... **do**
2: E-step: Determine $\mathcal{Q}(\theta;\theta^i)$
3: M-step: Maximize $\theta^{i+1} = \arg\max_{\theta\in\Theta}(\mathcal{Q}(\theta;\theta^i))$
4: **end for**

In order to implement this algorithm a fair amount of calculations need to be performed. Taking advantage of the fact that the model distributions belong to the exponential family it is however possible to simplify a few of the steps.

1.0.1. *Determination of sufficient statistics for model parameters.* In order to find sufficient statistics for the parameters in the model one need to look at the complete data likelihood. The complete data likelihood for our model is given by,

$$\begin{aligned}
&L_n^c(\theta; y_{0:n}, \mathbf{x}_{0:n}, \alpha_{0:n}, M_{0:n}) = \\
&= \chi(\mathbf{x}_0) g_0(\mathbf{x}_0) \prod_{k=0}^{n-1} p(\alpha_{k+1}) p(P; M_k, M_{k+1}) \left(\prod_{l=1}^{L} g(D_l, \sigma_v; \mathbf{x}_{k+1}, y_{k+1}) \right) q(\sigma_u; \mathbf{x}_k, \mathbf{x}_{k+1}) \\
&= \chi(\mathbf{x}_0) g_0(\mathbf{x}_0) \prod_{k=0}^{n-1} \Bigg[\frac{1}{2\pi} p_{M_k, M_{k+1}} \\
&\times \left(\prod_{l=1}^{L} \frac{1}{\sqrt{2\pi\sigma_v^2}} \exp\left\{ -\frac{1}{2\sigma_v^2} \left(y_{k+1} + 20 \log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2 \right] - D_l \right)^2 \right\} \right) \\
&\times \frac{1}{(2\pi)^2 \left|\mathbf{\Sigma}\right|^{1/2}} \exp\left\{ -\frac{1}{2} \left[(\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k) \right] \right\} \Bigg]
\end{aligned}$$

where $\mathbf{\Sigma} = R(\sigma_u^2 I_4) R^T$, $R = \begin{bmatrix} \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix}$.

Expanding the quadratic expressions and using $\left|\mathbf{\Sigma}\right|^{1/2} = \frac{T^6 \sigma_u^4}{4}$ yields

$$
\begin{aligned}
&= \chi(\mathbf{x}_0) g_0(\mathbf{x}_0) \prod_{k=0}^{n-1} \left[\left(\prod_{l=1}^{L} \frac{1}{\sqrt{2\pi\sigma_v^2}} \right) \frac{4}{(2\pi)^2 T^6 \sigma_u^4} \frac{1}{2\pi} \right] \\
&\quad \times \exp\left\{ -\frac{1}{2\sigma_v^2} \sum_{k=0}^{n-1} \sum_{l=1}^{L} \Big(\big(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right]\big)^2 \right. \\
&\qquad \left. - 2D_l \big(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right]\big) + D_l^2 \Big) \right\} \\
&\quad \times \exp\left\{ -\frac{1}{2\sigma_u^2} \sum_{k=0}^{n-1} \left[(\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k)^T (RR^T)^{-1} (\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k) \right] \right\} \\
&\quad \times \exp\left\{ \sum_{k=0}^{n-1} \log_e \left[p_{M_k, M_{k+1}} \right] \right\}
\end{aligned}
$$

$$
\begin{aligned}
&= \chi(\mathbf{x}_0) g_0(\mathbf{x}_0) \times \left(\frac{1}{(2\pi)^{L/2+3}} \right)^n \times \left(\frac{2}{T^3} \right)^{2n} \times \left(\frac{1}{\sigma_u^2} \right)^{2n} \times \left(\frac{1}{\sigma_v^2} \right)^{nL/2} \\
&\quad \times \exp\left\{ -\frac{1}{2\sigma_v^2} \sum_{k=0}^{n-1} \sum_{l=1}^{L} \left(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right] \right)^2 \right\} \\
&\quad \times \exp\left\{ \frac{1}{\sigma_v^2} \sum_{k=0}^{n-1} \sum_{l=1}^{L} D_l \left(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right] \right) \right\} \\
&\quad \times \exp\left\{ -\frac{n \sum_{l=1}^{L} D_l^2}{2\sigma_v^2} \right\} \\
&\quad \times \exp\left\{ -\frac{1}{2\sigma_u^2} \sum_{k=0}^{n-1} \left[(\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k)^T (RR^T)^{-1} (\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k) \right] \right\} \\
&\quad \times \exp\left\{ \sum_{k=0}^{n-1} \log_e \left[p_{M_k, M_{k+1}} \right] \right\}
\end{aligned}
$$

Using the factorization criterion and membership in the exponential family we can identify the sufficient statistics as

$$
s_{1,i}(l) = \sum_{k=0}^{n-1} \big(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right]\big) \tag{5.3}
$$

$$
s_{2,i} = \sum_{k=0}^{n-1} \sum_{l=1}^{L} \big(y_{k+1} + 20\log_{10}\left[(x_{1,k+1} - x_l)^2 + (x_{2,k+1} - y_l)^2 + h_l^2\right]\big)^2 \tag{5.4}
$$

$$
s_{3,i} = \sum_{k=0}^{n-1} (\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k)^T (RR^T)^{-1} (\mathbf{x}_{k+1} - A(\alpha_{k+1}, M_{k+1})\mathbf{x}_k), \tag{5.5}
$$

where subscript i denotes particle trajectory number i.

These statistics are calculated each step using draws from the smoothing distribution. The smoothing distribution is approximated by the particle trajectories

and their respective weight. Using these we can form the statistics as

$$\hat{s}_1(l) = \sum_{i=1}^{N} \frac{\omega_n^i}{\sum_{j=1}^{N} \omega_n^j} s_{1,i}(l) \tag{5.6}$$

$$\hat{s}_2 = \sum_{i=1}^{N} \frac{\omega_n^i}{\sum_{j=1}^{N} \omega_n^j} s_{2,i} \tag{5.7}$$

$$\hat{s}_3 = \sum_{i=1}^{N} \frac{\omega_n^i}{\sum_{j=1}^{N} \omega_n^j} s_{3,i} \tag{5.8}$$

where N is the total number of particles and ω_n^i is the weight for the particle trajectory i at step n.

1.0.2. *Calculations for maximization.* Using these statistics we can form a condensed expression of the complete log-likelihood as

$$l_n^c(\theta; y_{0:n}, \mathbf{x}_{0:n}, \alpha_{0:n}, M_{0:n}) = \log_e \left(L_n^c(\theta; y_{0:n}, \mathbf{x}_{0:n}, \alpha_{0:n}, M_{0:n})\right) =$$

$$= \log_e \left[\chi(\mathbf{x}_0) g_0(\mathbf{x}_0)\right] - (\frac{nL}{2} + 3n) \log_e (2\pi) - \frac{nL}{2} \log_e \left(\sigma_v^2\right) - 2n \log_e \left(\frac{T^3}{4}\right) - 2n \log_e \left(\sigma_u^2\right) -$$

$$- \frac{1}{2\sigma_v^2} \left[s_2 - 2 \sum_{l=1}^{L} D_l s_1(l) + n \sum_{l=1}^{L} D_l^2 \right] - \frac{1}{2\sigma_u^2} s_3 + \sum_{k=0}^{n-1} \log_e p_{M_k, M_{k+1}}$$

With this expression we can find estimators of the parameters maximizing the complete data likelihood

$$\frac{\partial l_n^c}{\partial D_l} = -\frac{1}{2\sigma_v^2}(-2s_1(l) + 2nD_l) = 0$$

$$\Rightarrow D_l^* = \frac{\hat{s}_1(l)}{n} \tag{5.9}$$

$$\frac{\partial l_n^c}{\partial \sigma_v^2} = -\frac{nL}{2\sigma_v^2} + \frac{1}{2(\sigma_v^2)^2}(s_2 - 2\sum_{l=1}^{L} D_l s_1(l) + n \sum_{l=1}^{L} D_l^2) = 0$$

$$\Rightarrow (\sigma_v^2)^* = \frac{1}{nL}\hat{s}_2 - \frac{2}{nL} \sum_{l=1}^{L} D_l^* \hat{s}_1(l) + \frac{1}{L} \sum_{l=1}^{L} (D_l^*)^2 = \frac{1}{nL}\hat{s}_2 - \frac{1}{n^2 L} \sum_{l=1}^{L} \hat{s}_1(l)^2 \tag{5.10}$$

$$\frac{\partial l_n^c}{\partial \sigma_u^2} = -\frac{2n}{\sigma_u^2} + \frac{1}{2(\sigma_u^2)^2} s_3$$

$$\Rightarrow (\sigma_u^2)^* = \frac{1}{4n} s_3 \tag{5.11}$$

The transition probabilities are estimated by the following estimator

$$p_{ij}^* = \frac{\sum_{k=0}^{n-1} \phi_{k,k+1|n}(i, j; \theta')}{\sum_{k=0}^{n-1} \phi_{k|n}(i; \theta')} \tag{5.12}$$

where $\phi_{k|n}(i; \theta')$ and $\phi_{k,k+1|n}(i, j; \theta')$ are the smoothing probabilities $\mathrm{P}_{\theta'}(M_k = i | y_{0:n})$ and $\mathrm{P}_{\theta'}(M_k = i, M_{k+1} = j | y_{0:n})$, respectively. These probabilities can be estimated using the draws from the smoothing distribution according to

$$\mathrm{P}_{\theta'}^*(M_k = i | y_{0:n}) = \sum_{\ell=1}^{N} \frac{\omega_n^\ell}{\sum_{j=1}^{N} \omega_n^j} \mathbb{1}\{M_{k,\ell} = i\}$$

and

$$\mathrm{P}^*_{\theta'}(M_k = i, M_{k+1} = j|y_{0:n}) = \sum_{\ell=1}^{N} \frac{\omega_n^\ell}{\sum_{j=1}^{N} \omega_n^j} \mathbb{1}\{M_{k,\ell} = i, M_{k+1,\ell} = j\}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and N is the number of particles used.

This way the transition probabilities can be estimated using the following relation

$$p_{ij}^* = \frac{\sum_{k=0}^{n-1} \sum_{\ell=1}^{N} \omega_n^\ell \mathbb{1}\{M_{k,\ell} = i, M_{k+1,\ell} = j\}}{\sum_{k=0}^{n-1} \sum_{\ell=1}^{N} \omega_n^\ell \mathbb{1}\{M_{k,\ell} = i\}} \tag{5.13}$$

Calculating the sufficient statistics and inserting them into the maximizing estimators is all that is needed for the EM algorithm. The calculated parameter estimates are then used for generating the next MC sample used in the expectation step. Iterating these steps until the values converge will give estimates of the model parameters.

2. Degeneration

The particle estimation of the smoothing distribution might not always be easily obtained. The resampling step solved degeneration in the case of filtering but when approximating the smoothing distribution there is an imminent risk that the resampled particle trajectories after some time will originate from only a single particle. This problem is illustrated in *Fig. 5.1–5.2* showing clearly the collapse in origin to the same ancestor.

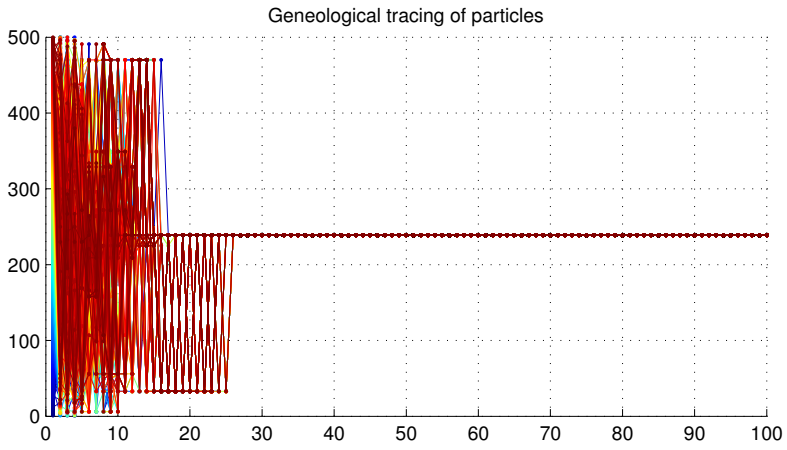

FIGURE 5.1. Genealogical tracing of 500 particles over time showing original ancestor index on the y-axis and EM iteration, or generation, on the x-axis. After just under 30 steps all particles origin from the same ancestor.

The collapse results in a loss of information and a bad approximation of the smoothing distribution. This in turn leads to bad estimates during parameter inference resultng in an expectation-maximization estimate with a large bias. In some cases the estimation algorithm will not even converge.

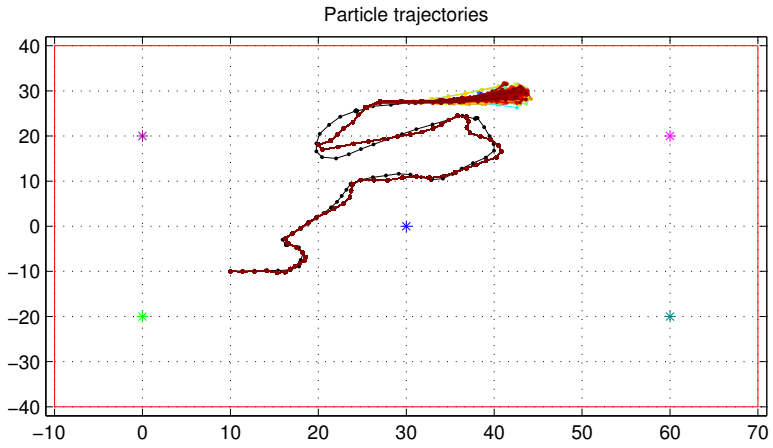

FIGURE 5.2. Degenerated sample of 500 particle trajectories showing a bad diversity which will lead to a poor approximation of the smoothing distribution.

3. Fixed-lag smoothing

The first and most simple solution of this problem is to increase the amount of particles used. Since it might be enough with a few particles to get a rough estimate it is possible to start the EM algorithm using a smaller amount of particles and then let the number increase with each iteration. This will allow for a considerable increase in particles but focusing the majority of the computations on the later iterations which require a better smoothing approximation to get a precise estimate. But solemnly increasing the amount of particles will not be a solution for longer sequences of data. Depending on the complexity of the HMM it might require such a large amount of particles to avoid degeneracy that the running time will suffer too much.

One method for avoiding degeneracy, discussed in [**OCDM08**], is called *fixed-lag smoothing*. This method uses the idea that it might be possible to approximate the smoothing distribution with a smaller amount of observations than the full trajectory. One might expect that the smoothing of a certain point would mainly depend on observations up to a certain point and that the later observations would affect the distribution less and less. This idea is based on something called the *forgetting property* of the hidden Markov chain. A geometrically ergodic chain will exhibit this property and forget its previous states geometrically and geometric ergodicity can be proven to hold under these circumstances. The interested reader is referred to [**OCDM08**] for an in depth discussion on the topic.

As the trajectories collapse after a certain amount of steps it would be tractable to use a fixed amount of observations, shorter in duration than the typical collapsing time. The fixed-lag smoothing uses the forgetting property to fight the degeneracy using trajectories that has not yet collapsed. Since chain forgets its previous states the data points further ahead will only provide very little information, if any, to the approximation of the smoothing distribution and can therefore be omitted.

The fixed-lag smoothing technique is mainly defined for additive functionals over the sequence of the form

$$t_n(x_{0:n}) = \sum_{i=0}^{n-1} s_k(x_{k:k+1}). \tag{5.14}$$

Here s_k is a series of functions defined on the product space $\mathsf{X} \times \mathsf{Y}$. Looking at the estimators used in EM they are additive functionals based on the sufficient statistics from the exponential family fitting exactly this format. One can then assume that the fixed-lag smoothing should suit our needs well.

So in the EM setting the expectations computed over the smoothing distribution can be approximated using observations up to a fixed lag for each function value in the sequence,

$$\mathbb{E}\left[s_k(X_{k:k+1}|Y_{0:n})\right] \approx \mathbb{E}[s_k(X_{k:k+1}|Y_{0:k(\Delta_n)})], \quad \forall k \tag{5.15}$$

where

$$k(\Delta_n) = (k + \Delta_n) \wedge n. \tag{5.16}$$

This function is used in the notation to ensure that the lag does not extend further ahead than the number of available observations. Using this notation and combining it with the particle and weight trajectories we can write the sufficient statistics in (5.6-5.8) as

$$\hat{s}_1 = \sum_{k=0}^{n-1} \sum_{i=1}^{N} \frac{\omega^i_{k(\Delta_n)}}{\sum_{j=1}^{N} \omega^j_{k(\Delta_n)}} \left(y_{k+1} + 20 \log_{10}\left[(\xi^i_{1,k+1} - \tilde{x}_l)^2 + (\xi^i_{2,k+1} - \tilde{y}_l)^2 + h_l^2\right]\right) \tag{5.17}$$

$$\hat{s}_2 = \sum_{k=0}^{n-1} \sum_{l=1}^{L} \sum_{i=1}^{N} \frac{\omega^i_{k(\Delta_n)}}{\sum_{j=1}^{N} \omega^j_{k(\Delta_n)}} \left(y_{k+1} + 20 \log_{10}\left[(\xi^i_{1,k+1} - \tilde{x}_l)^2 + (\xi^i_{2,k+1} - \tilde{y}_l)^2 + h_l^2\right]\right)^2 \tag{5.18}$$

$$\hat{s}_3 = \sum_{k=0}^{n-1} \sum_{i=1}^{N} \frac{\omega^i_{k(\Delta_n)}}{\sum_{j=1}^{N} \omega^j_{k(\Delta_n)}} (\xi^i_{k+1} - A(\alpha_{k+1}, M_{k+1})\xi^i_k)^T (RR^T)^{-1} (\xi^i_{k+1} - A(\alpha_{k+1}, M_{k+1})\xi^i_k) \tag{5.19}$$

Essentially this means that for each step $k : k + 1$ we update the functional values using the current particles and the associated smoothing weights at lag $k(\Delta_n)$ ahead. By using the additive structure of the functional it is possible to update the statistics recursively using previous values and just record the particle trajectories and weights as time progress. Apart from slightly higher storage requirements, this updating scheme will therefore not increase the computational complexity of the algorithm beyond the original smoothing algorithm.

CHAPTER 6

Data collection

The data needed for estimation and testing is the signal strengths from each AP, the id of these AP's and points in time when the measurement were made. In order to be able to control the correctness of the estimates one also needs to know a true trajectory and the correct AP locations. All existing data which was available failed on one or more of these aspects and it was therefore needed to construct some kind of data collection device. Smart phones are perfect for this task since they are both mobile and open for quick and easy application development. The Android platform is especially suitable because of the possibility of free development and easy access to the file system.

1. Development of a data mining application

Developing an Android application for data collection is pretty straightforward since all the needed libraries for reading the WiFi scans are contained within the Android SDK. A main Activity was created together with a BroadcastReceiver that listens on the completion of a WiFi scan. Another class, called DataPoint, was created to save and structure the output of each result. Since the default scan rate was a bit too slow, a Task scheduler initiating scans was set up on a more desirable frequency. Unfortunately there is no way to control the WiFi scans explicitly since these initiations of scans are only requests to the system. The OS seemed to sometimes ignore these requests completely resulting in somewhat irregular scan intervals. Unfortunately there does not seem to be solution to this problem. The Android SDK does not provide any other possibility to initiate WiFi scans which most probably is due to stability concerns from the OS developers. It would probably be possible to control this using kernel modules directly but since this would require a lot more complex programming it was not pursued further.

Two other buttons were created. One which starts and stops recording to file and one that clears the interface. The data is saved to file in CSV format for ease of import in other applications.

The source code of the resulting application can be found on `https://github.com/while/RSSIMiner` and is available for further usage and development under the MIT licence.

CHAPTER 7

Numerical results

1. Analysis of algorithm performance

The first tests performed were done on simulated data to assess the performance on the different algorithmic improvements. The first part was to compare the auxiliary filter with the bootstrap filter and auxiliary filter without the maneuvering state (S), i.e. the ability to stop completely. This test was done to assert the performance improvement introduced by the auxiliary filter and also the performance difference depending on inclusion or exclusion of the (S) state in the model.

As can be seen in *Fig. 7.1-7.3* and *Fig. 7.5* the auxiliary filter performs slightly better then the original bootstrap filter. The biggest difference is displayed during straight walks which seem to be because the pilot particles manage to predict this maneuver state better than others. One possible explanation here might be that the turns are more complex to predict and since these occur less frequently it might be so that the pilot propagation does not capture the turn that the actual particle will take. This is quite reasonable to expect since the pilot propagation is meant as an approximation of the expected value of the next step and as the walk is symmetric the expected step will not turn nor stand still. Non the less the accuracy of the filter increase with the use of pilot particles.

One can also conclude that the inclusion of the (S) state is of value since introducing this state leads to a significant improvement of the performance of the filter. The root-mean-square error of 64 sequential tests on the different test cases displayed in *Fig. 7.5* clearly show a significant difference in performance. As soon as the walk comes to a halt the model stripped of this state is incapable of predicting the position with any reasonable precision. Instead this model in many cases walks around the neighbouring area. As can be seen in right test in *Fig. 7.3* the no-(S) model finds a local max walking around the AP instead of standing still.

This result points to the fact that the model is not very sensitive to inclusion of motion sub-models that are somewhat uncommon but rather that they contribute if the pattern emerges in the data.

As can be seen in *Fig. 7.5* some of the models increase in error towards the end of the test walk. This might seem as the models are unstable after a large amount of steps but is rather due to the fact that some of the test walks leaves the area where the APs are located which results in a larger uncertainty in the estimate of position. The further away from all AP locations the receiver is located the flatter the local likelihood will be in the area of interest. The predictions will therefore also exhibit a larger uncertainty and error.

The simulated RSSI data belonging to each of the test cases are displayed in *Fig. 7.4*. In order to simulate the real RSSI measurements a cut off limit at $-110dBm$ was introduced. Real world devices cannot receive signals under a certain limit at which point no signal is received at all. The fact that the receiver needs to be at a certain distance in order for this to happen was not used in the model since no good way was found that did not affect the estimate in a negative way. Simply not using

the data in that point increases the uncertainty but even if as few as two AP's are in range for shorter amounts of time the model can handle it well. This is because of the models usage of previous movements which gives enough information to the model in order for it to single out where the most likely position regardless of the sometimes bi-modal local likelihood. The coloring in the graphs correspond to the AP with the same color in *Fig. 7.1-7.3*.

RMS errors of the bootstrap filter, the original auxiliary filter with pilot particles and the auxiliary filter using approximation by Taylor series expansion is displayed in *Fig. 7.6*. It is clear that the Taylor approximated auxiliary filter also performs better than the bootstrap filter and that it is very similar to the original one. The approximated expected value seem to coincide with the majority of pilot particles on the parts where the straight maneuver is in action. This strengthens the earlier hypothesis on the auxiliary filters performance on straights compared to the bootstrap filter. It is then natural that neither of the two should do any better than the bootstrap filter in areas where the walk is turning or standing still as the auxiliary weights mostly contribute information when walking straight.

The auxiliary filter with Taylor series expansions run in about half the time the original auxiliary filter. This is due to the fact that even though the same amount of approximations is calculated in the algorithms, the complexity of propagating particles and calculating weights is much less than that of a matrix-vector multiplication. This operation is both simpler computationally but also much more optimized in a vast majority of languages and architectures. There is of course a possibility that it might be possible to vectorize entirely the calculation of the auxiliary weights or in some other way parallelize the computation involved but the approximation should still provide faster execution times.

2. Estimation of model parameters

To ensure the effectiveness and convergence of the EM algorithm with fixed-lag smoothing, it was first run on simulated data. The result is displayed in *Fig. 7.7-7.16*. As can be seen all the D_l parameters converged beautifully within a few iterations together with the standard deviation of the noise, σ_v.

The transition probabilities generally converged but did show difficulty converging to the true value in some cases. This was mostly the walking straight and turning probabilities that seemed to interfere in some cases. This was probably due to the possibility for the turning state to actually capture the walking straight. This possibly introduced a chance to get particle trajectories with large weight using either of the two states which made them interfere during the EM estimation. Turning should however not be as likely as the actual straight walk but because of other parameters being estimated at the same time it might be so that they may cancel out each other or find other combinations that generates a high likelihood for just this specific case. There are definitely room for improvement here where some kind of regularization or cross-validation might improve the estimation and remove overfitting to one specific sequence. This was however not pursued any further in this thesis.

Estimation of σ_u did show some difficulties and estimation together with other parameters did not succeed. The estimated parameter did always fall off towards zero which indicates that measurement noise somehow takes over to be the single source of noise. In order to at least see that the suggested estimator converges, the parameter was estimated with all other parameters set for a longer duration. This estimate can be seen in *Fig. 7.15*. Even though it takes a long time it seems that the parameter at the near end converges to a value in in the near proximity of the

true value. Similar result was obtained for different walks so the problem does not seem to be related to the same problem as the transition probabilities.

3. Testing on real-world data

To see how well the model and suggested algorithms work in a real environment with real data, measurements were made in one of the buildings at Lund Institute of Technology using the previously described Android application. The resulting walks are displayed in *Fig. 7.12-7.13*. As can be seen the results are reasonably good. Even though a large bias is present the model manages to capture the movement pattern and predict the position to within a few meters. Due to the large amount of concrete pillars in the area, the signal was very intermittent and is switching a lot between different APs. This resulted in data from very few APs at the same time resulting in movement sideways and some other strange movements. Errors due to these were very consistent between different measurements as displayed in *Fig. 7.13*. The presence of people also made the signal deteriorate a lot. During some busy hours the D_l parameter changed due to dampening of the signal.

Parameter inference on the real data seems to work well for most of the parameters, just as in the simulated case. The estimated parameters increased the accuracy for both tests even if it is much more apparent in *Fig. 7.12* then in *Fig. 7.13*. Comparison is made with a manual estimate where the parameters were brought forth by manual iteration. Guessing is tedious and not very accurate but it works well here as a base case to see if there is any improvement introduced by the EM estimation.

A difficulty encountered when estimating the parameters on real-world data was the missing values. The filtering can easily handle the missing values without loosing much precision, as previously discussed, but the EM algorithm and smoothing was a lot more susceptible to this change. In the real setting it is however hard to ensure that this does not happen without using professional equipment and controlled environments. In order to make the estimation possible it was decided to set a base value in case of a missing value so that a value is present even though not entirely correct. This way it was possible to make the estimate but some kind of bias is most probably introduced due to this blunt fix. A missing signal does however tell us something and it might be of interest to investigate how to incorporate this into the model to enhance the estimates instead of bluntly introducing a bias.

The parameters the EM algorithm managed to estimate were acquired using 2000 particles and a fixed lag of 15 steps. A setting with runs reasonably fast on a modern computer.

4. Figures

The resulting figures are all collected under this section.

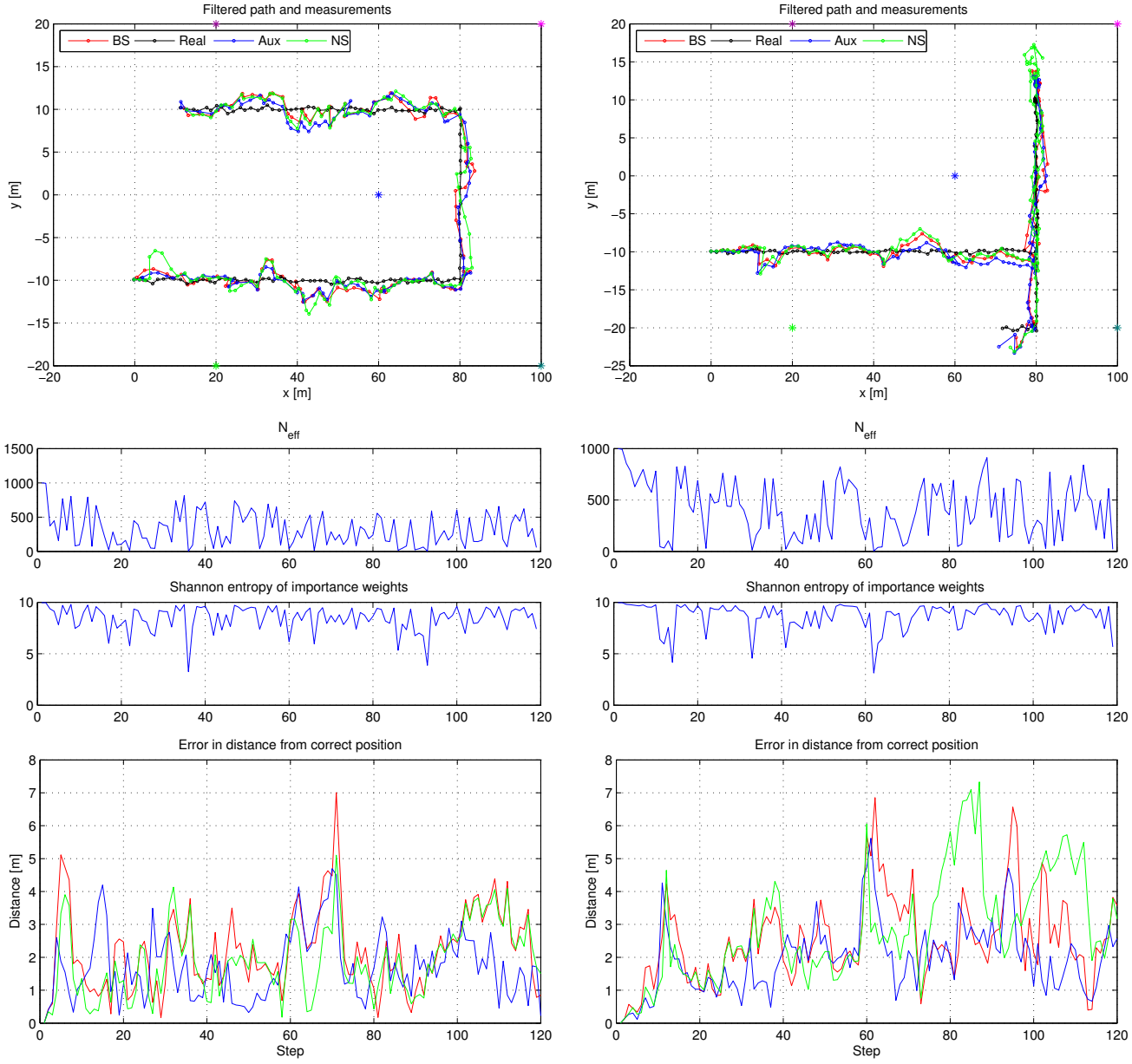

FIGURE 7.1. Test run 1 and 2, using both bootstrap and an auxilliary filter, together with the respective number of effective particles, the Shannon entropy and error distance for the two different methods. The step with the least amount of effective particles has been encircled in green and the access point locations have been marked with stars in different colors.

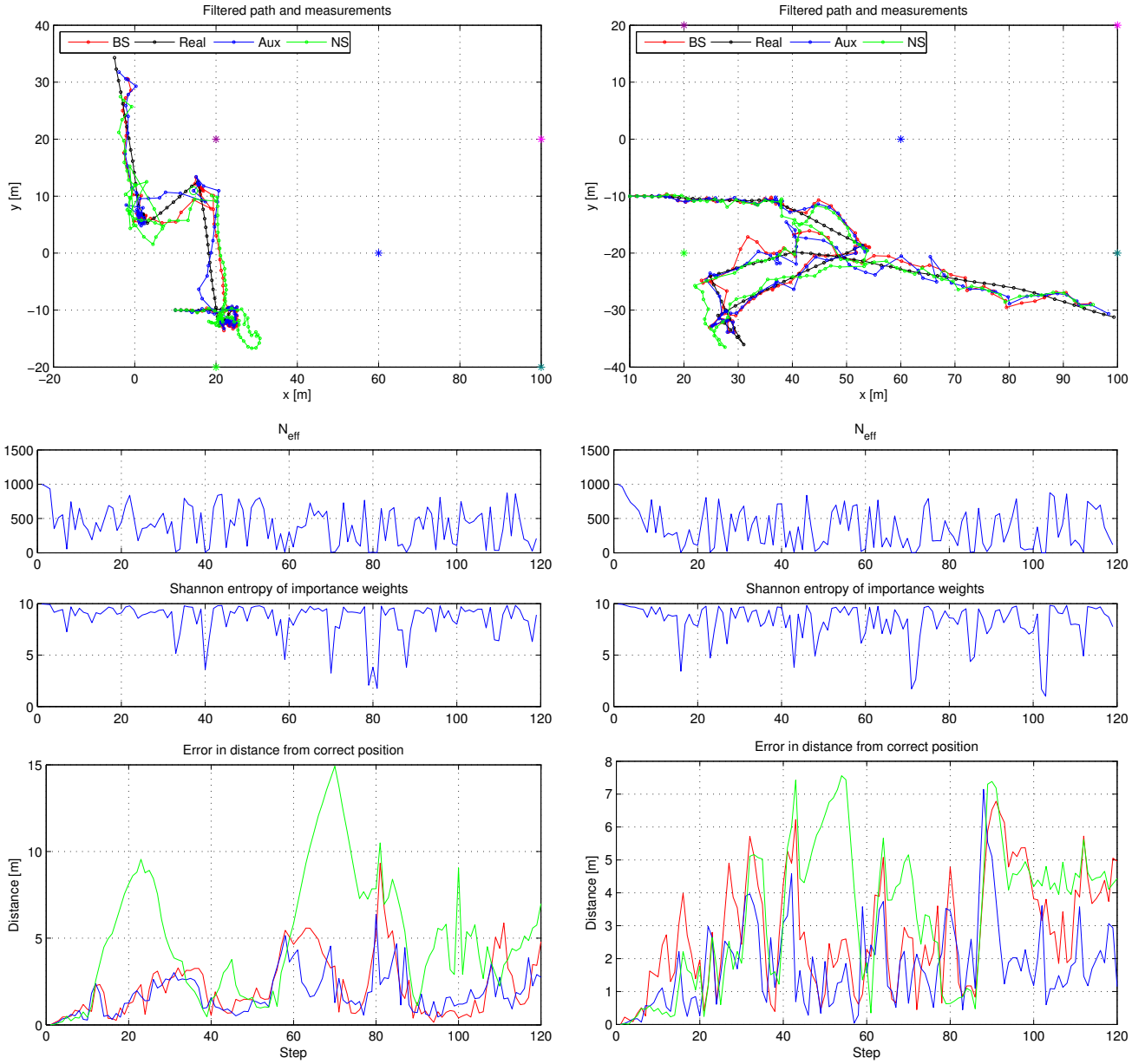

FIGURE 7.2. Test run 3 and 4, using both bootstrap and an auxilliary filter, together with the respective number of effective particles, the Shannon entropy and error distance for the two different methods. The step with the least amount of effective particles has been encircled in green and the access point locations have been marked with stars in different colors.

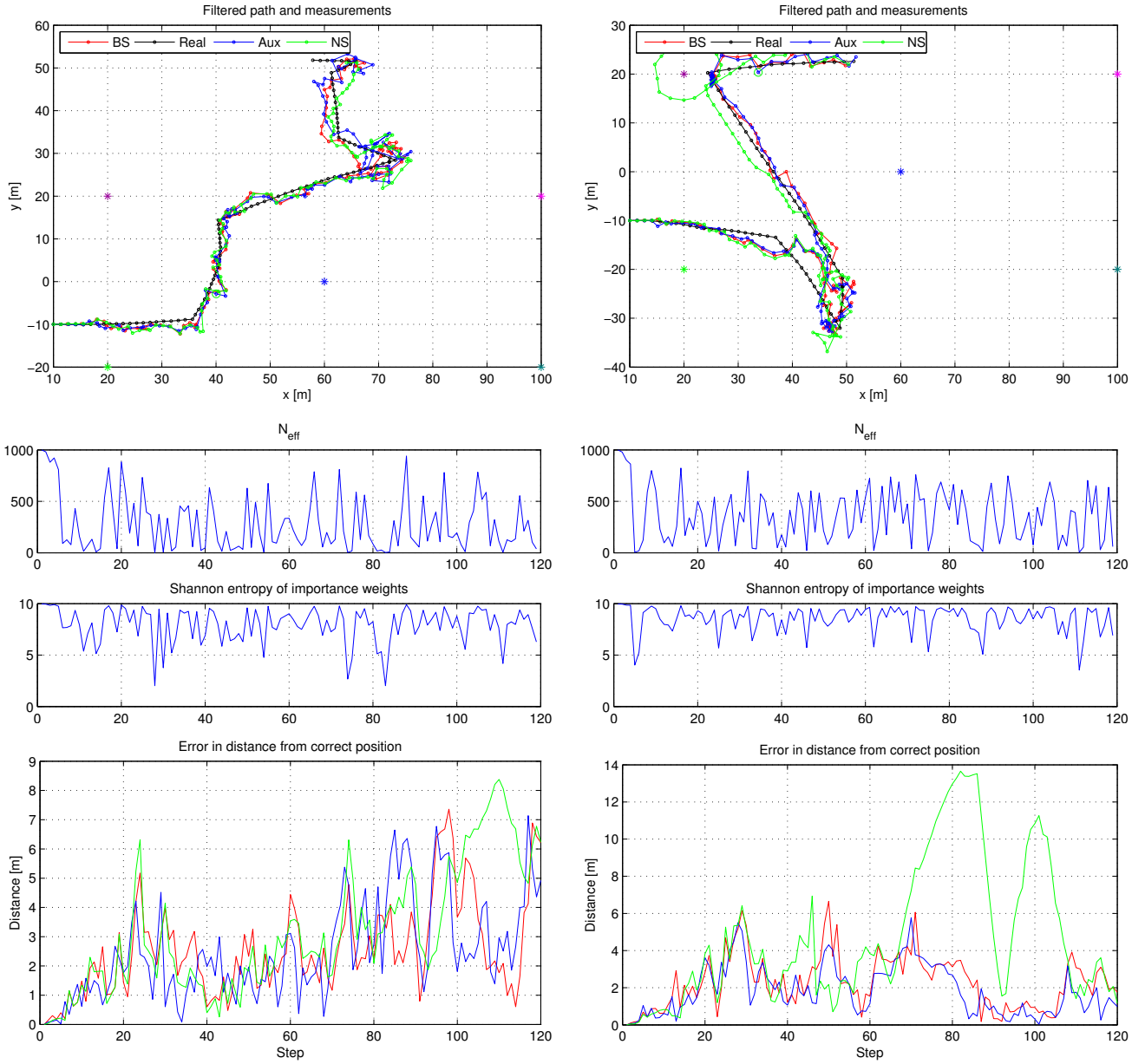

FIGURE 7.3. Test run 5 and 6, using both bootstrap and an auxilliary filter, together with the respective number of effective particles, the Shannon entropy and error distance for the two different methods. The step with the least amount of effective particles has been encircled in green and the access point locations have been marked with stars in different colors.

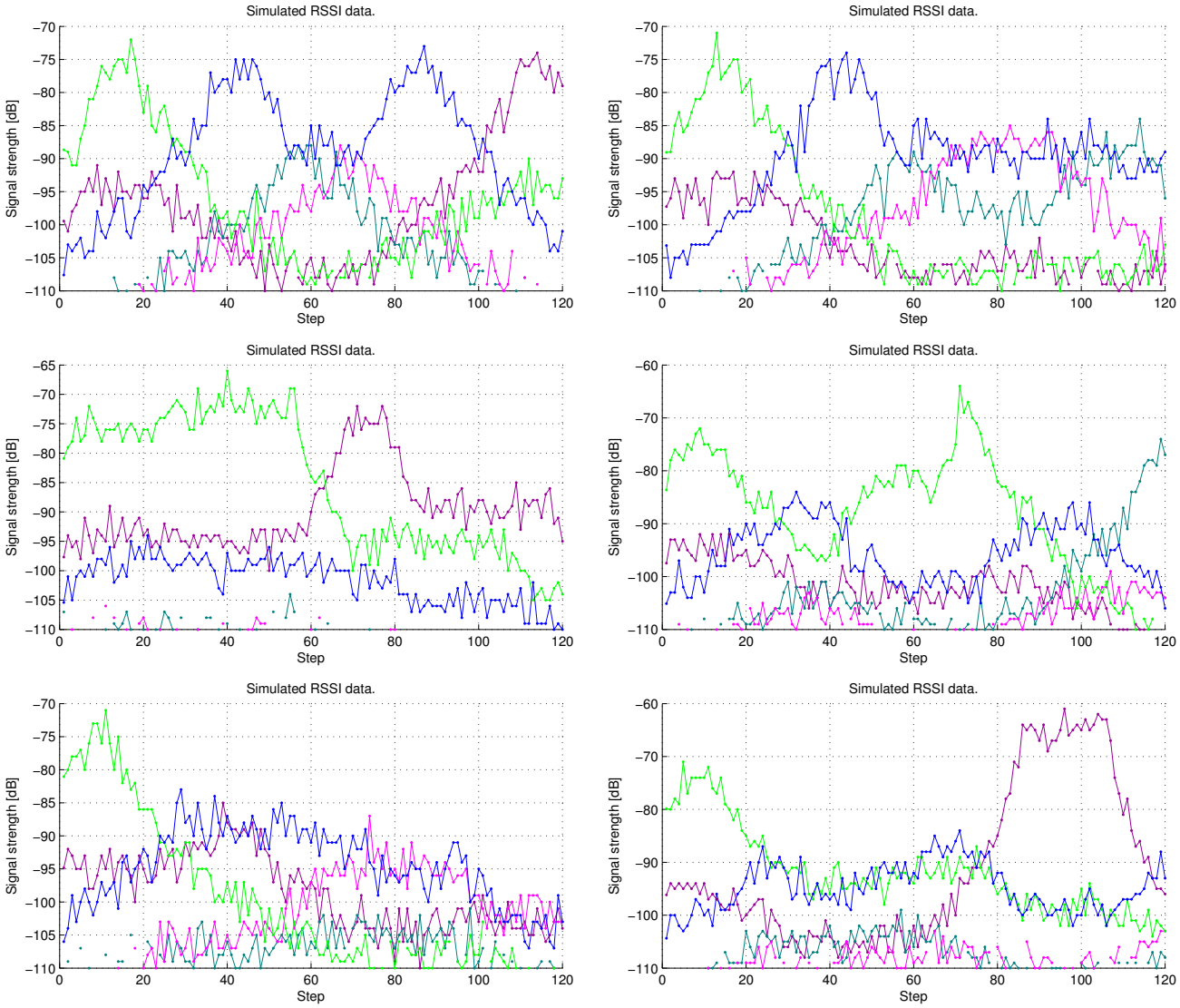

FIGURE 7.4. RSSI data for the different test cases.

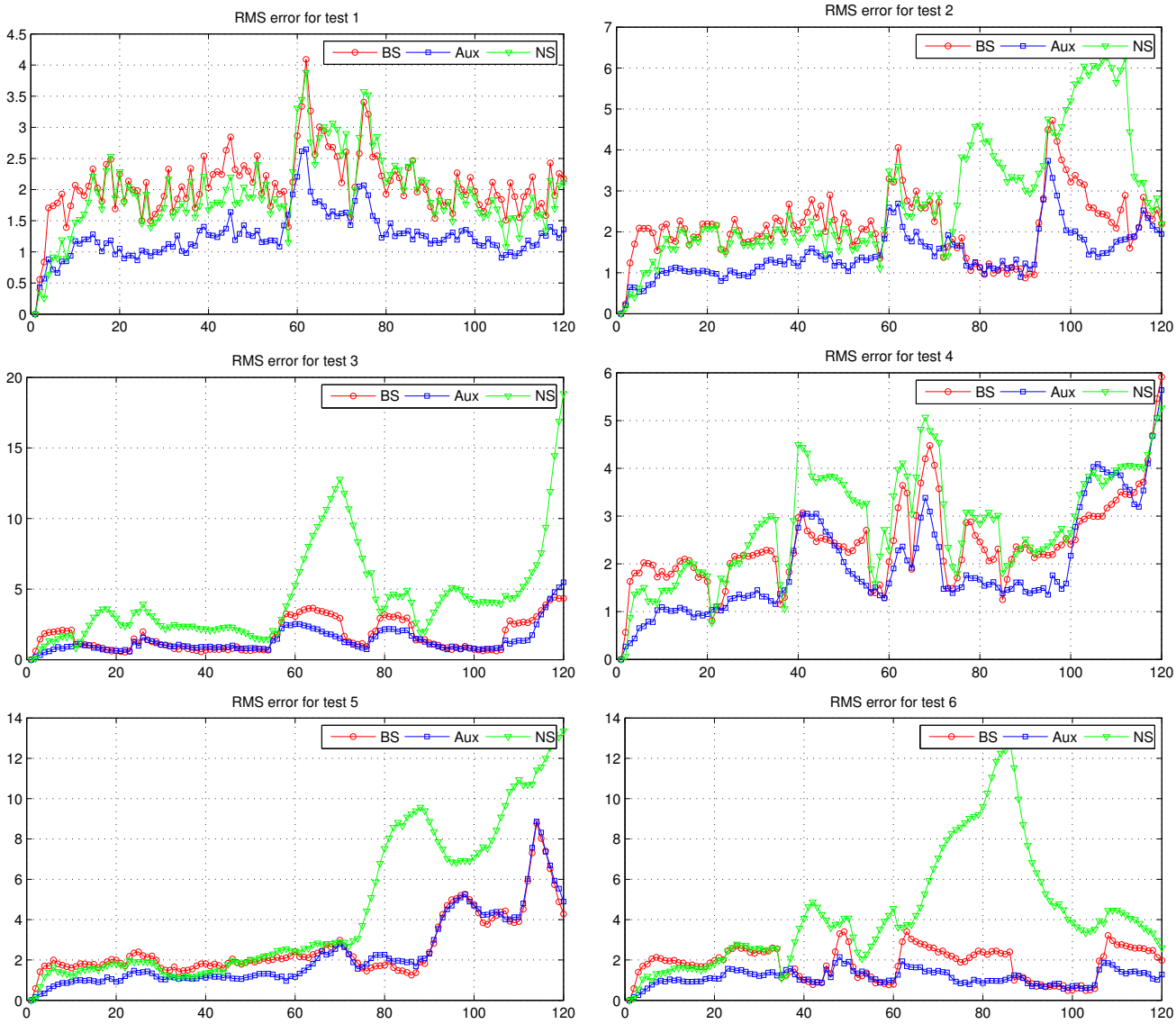

FIGURE 7.5. RMS error over 64 test runs for each of the different test cases.

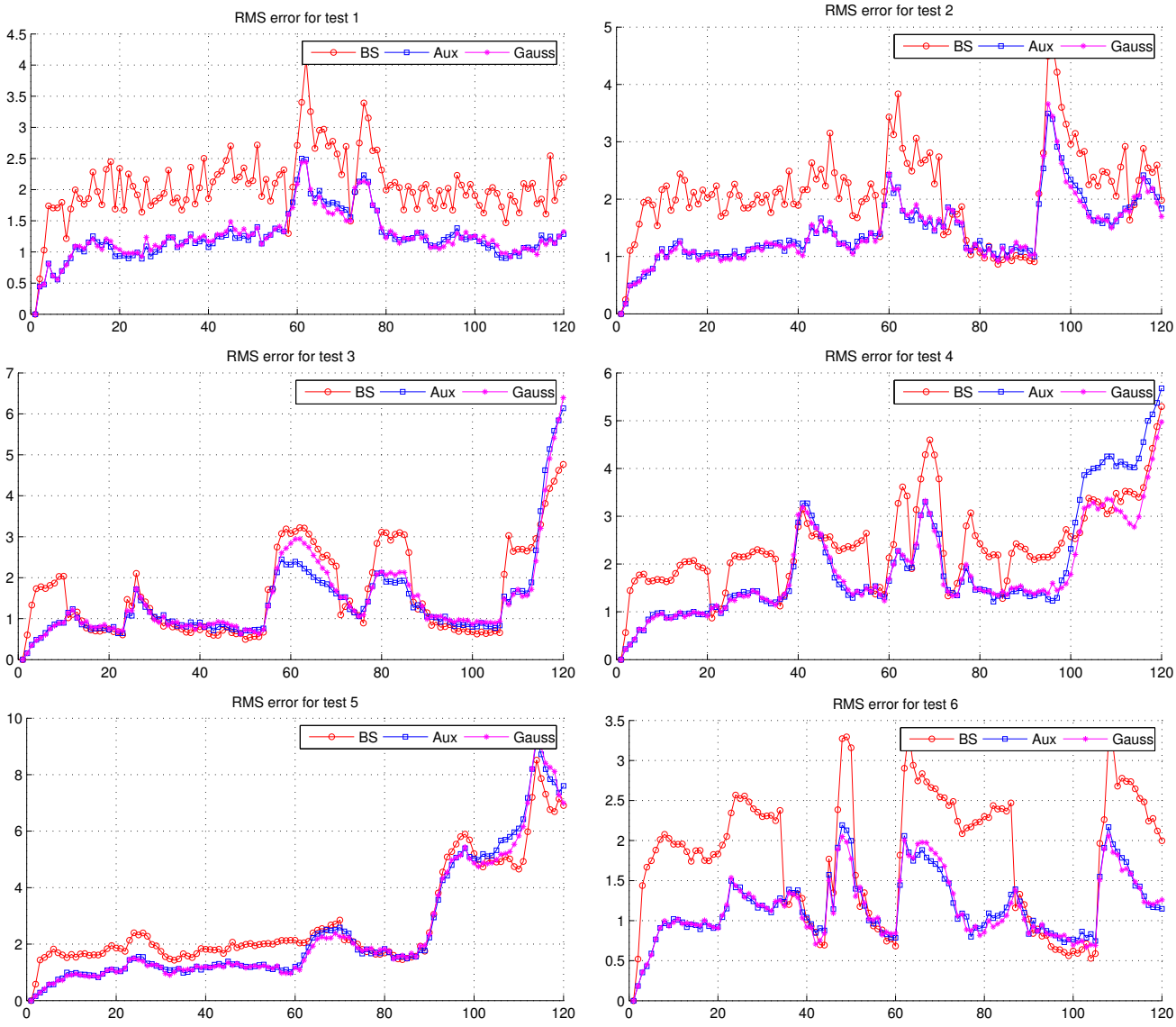

FIGURE 7.6. RMS error over 64 test runs for each of the different test cases. Gauss is the algorithm using the Taulor series expansions.

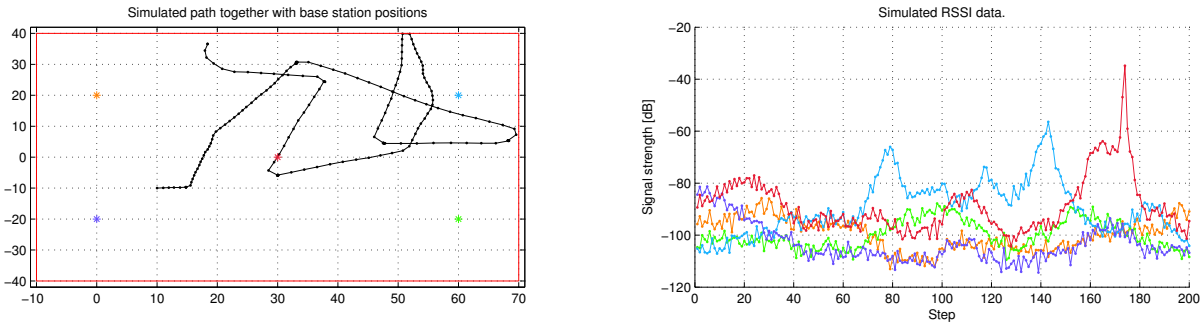

FIGURE 7.7. Simulated data used for estimation of parameters using the EM algorithm. The generated walk had 200 steps and 5 APs were used for generation of RSSI. The APs in the grid corresponds to the RSSI with the same color.

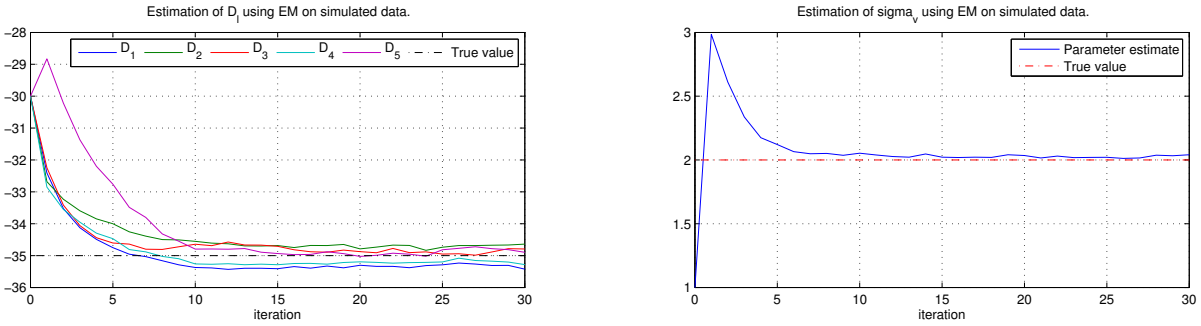

FIGURE 7.8. Estimation of Dl and σ_v parameters for simulated data using 30 iterations of the EM algorithm. As can be seen, all parameters converge quickly towards the correct value.

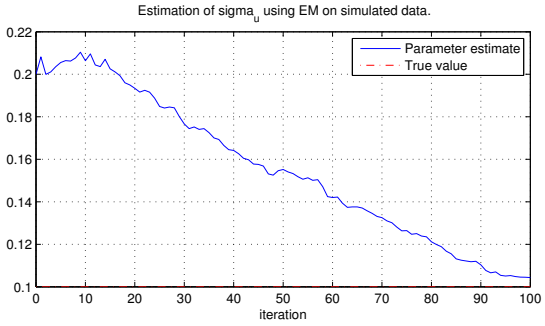

FIGURE 7.9. Estimation of σ_u for simulated data using the EM algorithm. Because of the convergence trouble this parameter exhibited, it was run separately with an increased number of iterations.

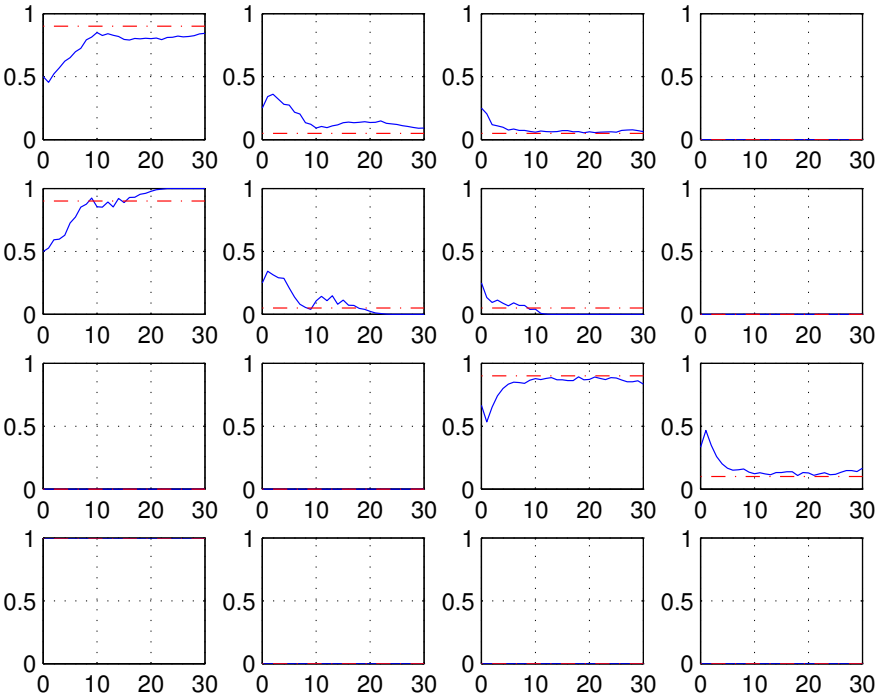

FIGURE 7.10. Estimation of transition probabilities on simulated data using 30 steps of the EM algorithm. As the probabilities in $P_{(2,2)}$ and $P_{(2,3)}$ were very small they droped off to zero. This might be avoided using loger walks for estimation or the average of several different walks.

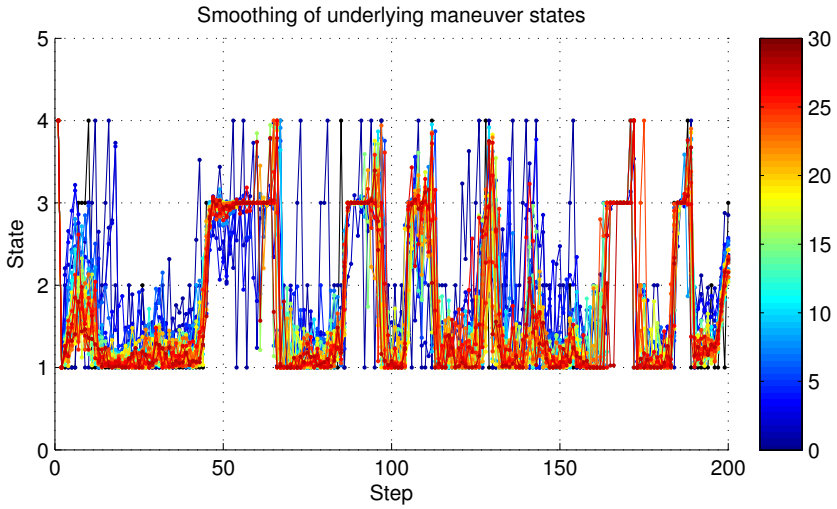

FIGURE 7.11. Smoothing of underlying maneuver chain for each of the 30 iterations. The coloring represents the EM iteration. As can be seen in the figure the smoothing reconstruction gets better and better with each iteration.

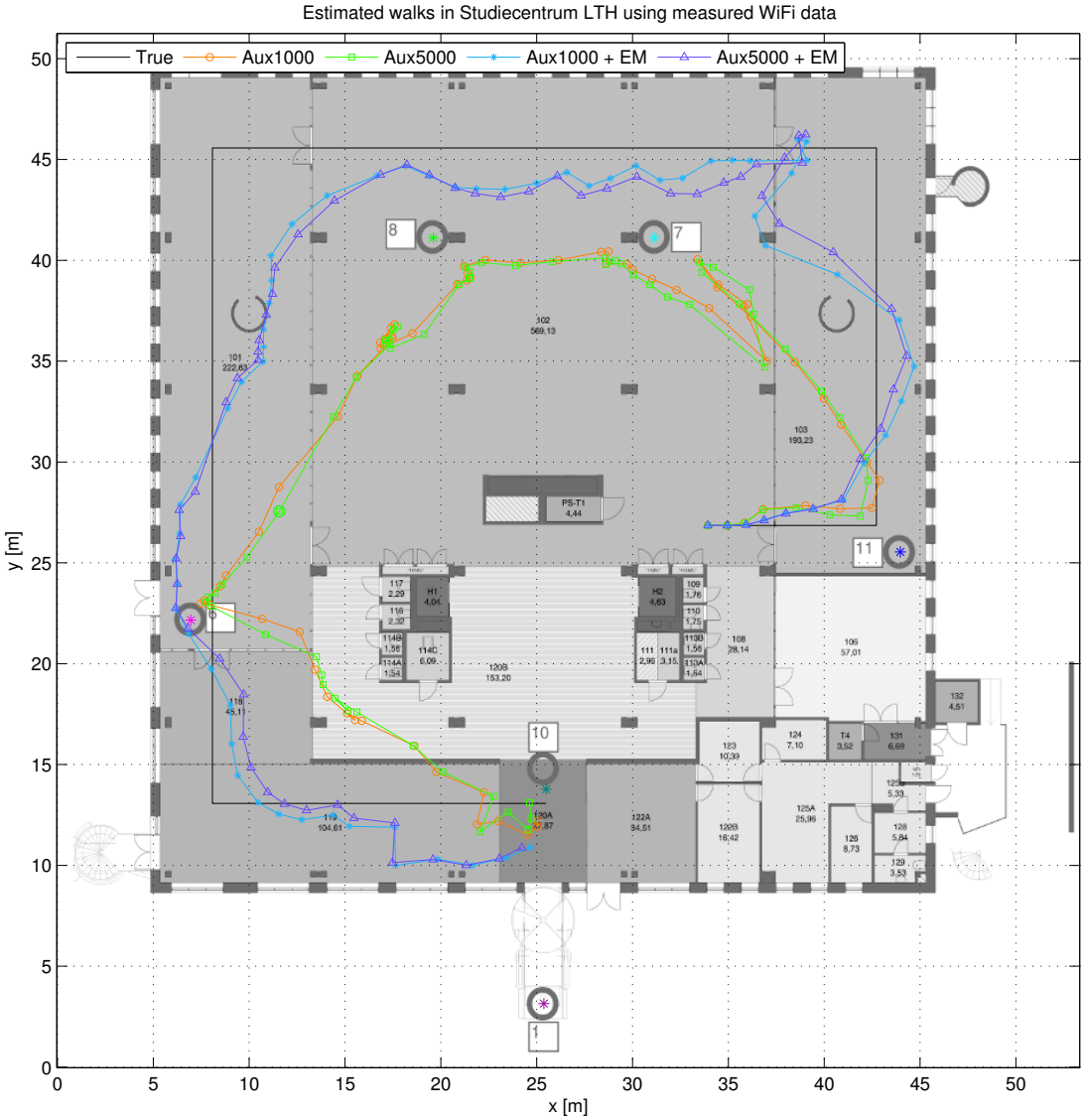

FIGURE 7.12. Estimated walks in Studiecentrum LTH based on real RSSI measurements. The figure displays a comparison between a qualitative guess on the parameters and the ones estimated using the EM algorithm. Two different estimations were done for both parameter sets using 1000 and 5000 particles respectively.

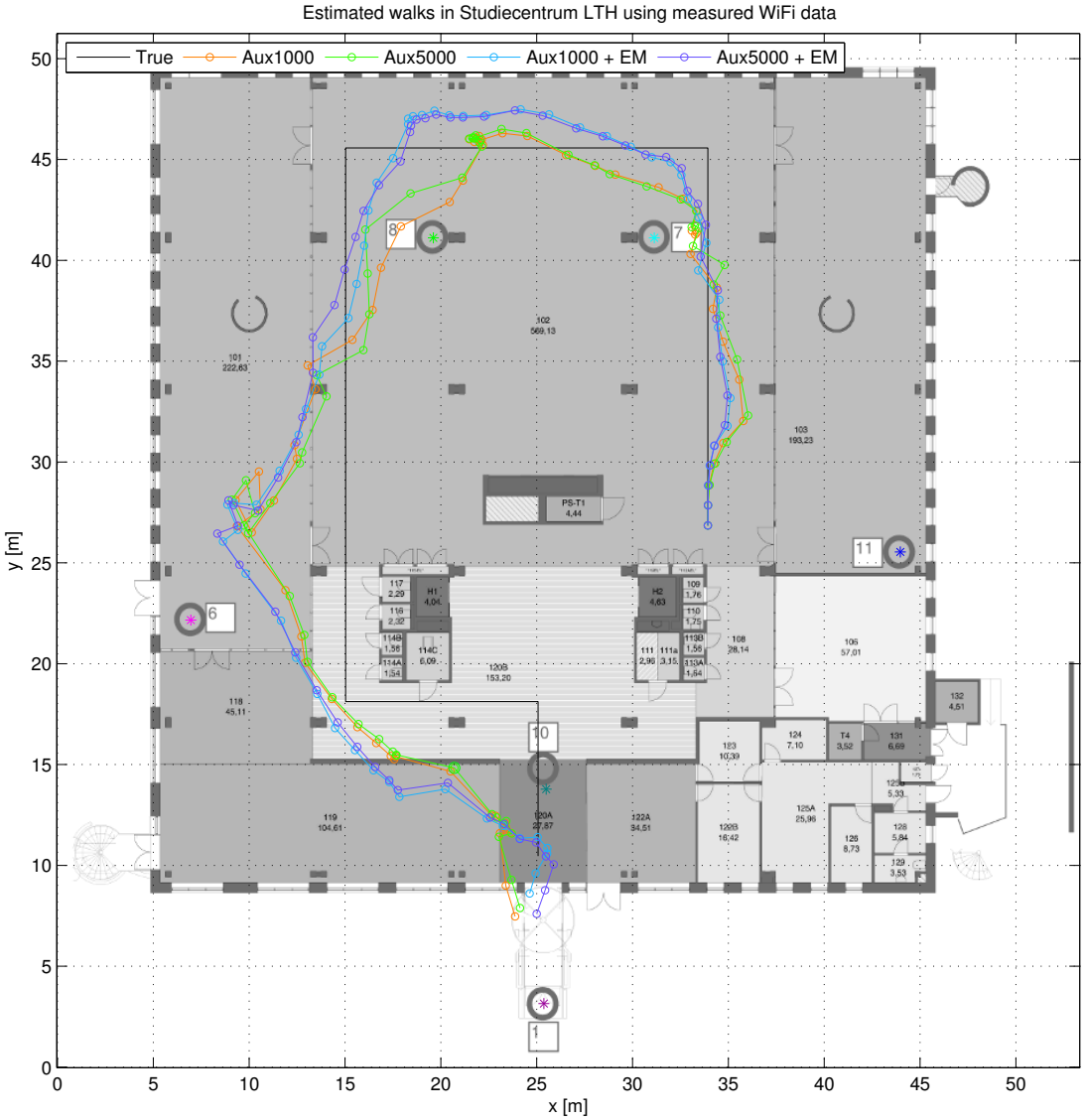

FIGURE 7.13. Estimated walks in Studiecentrum LTH based on real RSSI measurements. The figure displays a comparison between a qualitative guess on the parameters and the ones estimated using the EM algorithm. Two different estimations were done for both parameter sets using 1000 and 5000 particles respectively.

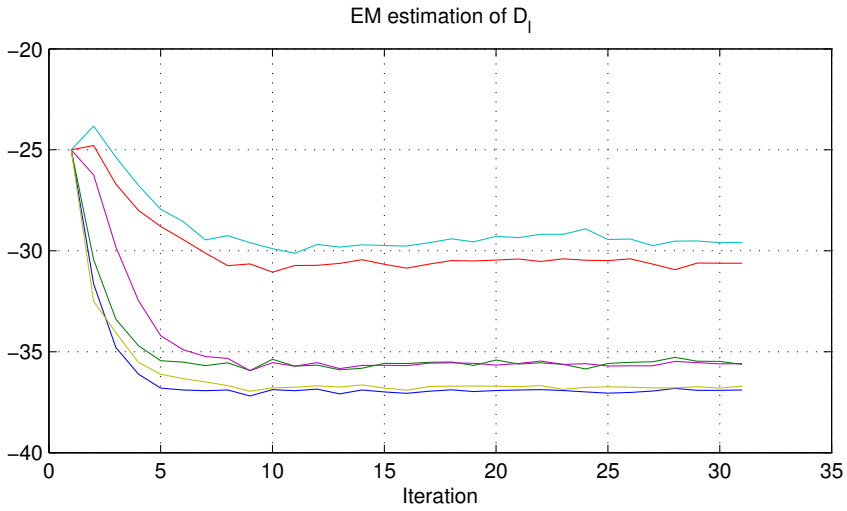

FIGURE 7.14. Estimation of D_l parameters using real-world measurments. The differences in level between the access points might be due to the many concrete walls in the building, other disturbances in the area or simply that the different stations had diferent transmit power.

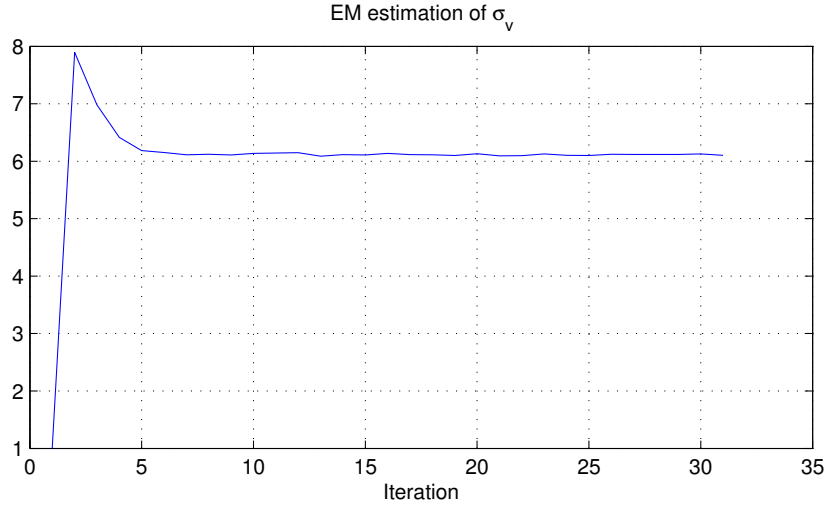

FIGURE 7.15. Estimation of the σ_v parameter using real-world measurments. This parameter exhibits a fast convergeance.

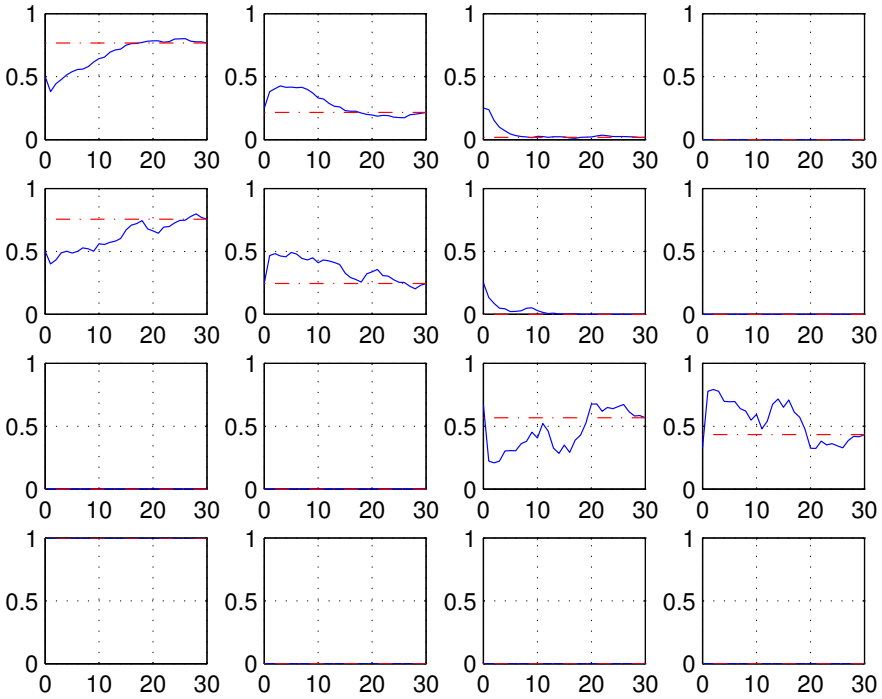

FIGURE 7.16. Estimation of transition probabilities on real-world data using 30 steps of the EM algorithm. The red lines are the last value obtained from the estimation which later was used in the filter.

CHAPTER 8

Conclusions

Experiments have shown that hidden Markov models in combination with auxiliary particle filters can form a powerful framework when estimating positions in WiFi networks. The model performs relatively well and manages to predict the trajectories even in real-world, uncontrolled, noisy environments. The estimations are maybe not good enough to be used in any application where good precision is required but it might serve well as a rough estimator when errors up to a few meters are acceptable. The poor precision might also be improved if APs were positioned with more than coverage in mind. At several times very few access points were in range which definitely affected the accuracy of the position estimates. Given these circumstances the model performed well once the parameters were tuned to their right value.

The estimation of parameters was slightly harder than expected. This is of course a negative aspect of the proposed model. One might argue that the convergence of the EM estimator might have been better if longer measurements would have been made. The short walks did not always give enough data to leave only a single optimal point in the parameter space. It is probably possible to improve the estimation procedure there. Some small changes to the model setup or change in the algorithm might be enough to fix these problems but has not been confirmed. This will require further investigation and is left for more in-depth studies of the problem.

Another problem encountered were the rapidly changing signal space in the real world environment. This is another area which is open for further investigation. It might call for some need to design an algorithm that learns and reconfigures the parameters as it is used. One idea here is to use something called on-line smoothing which is an area with a lot of ongoing research. Another idea might be to use some combination of machine learning algorithms and hidden Markov models in order to improve the estimation. It is possible that one can use the discriminatory power of machine learning algorithms to extract more information from the signal space before inserting it into the hidden Markov model. This might open up for making longer measurements in areas with poor coverage and use this to improve the position estimate. How this should work in details is left for further studies of the problem.

Finally the different maneuver sub-models might be investigated further. It is possible that there are other maneuvers that will make it possible to filter the process even better.

Bibliography

[AMRE12] K. Achutegui, J. Míguez, J. Rodas, and C.J. Escudero, *A multi-model sequential monte carlo methology for indoor tracking: Algorithms and experimental results*, Signal Processing **92** (2012), 2594–2613.

[Bil12] P. Billingsley, *Probability and measure*, Wiley Series in Probability and Statistics, Wiley, 2012.

[CMR05] O. Cappé, E. Moulines, and T. Ryden, *Inference in hidden markov models*, Springer Series in Statistics, Springer, 2005.

[DLR77] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the em algorithm*, Journal of the Royal Statistical Society. Series B (Methodological) (1977), 1–38.

[FL09] S. Fang and T. Lin, *Projection-based location system via multiple discriminant analysis in wireless local area networks*, IEEE Transactions on vehicular technology **58** (2009), no. 9, 5009–5019.

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, *Novel approach to nonlinear/non-gaussian bayesian state estimation*, IEE Proceedings-F **140** (1993), no. 2, 107–113.

[Gut09] A. Gut, *An intermediate course in probability*, Springer texts in statistics, Springer-Verlag New York, 2009.

[Jak12] A. Jakobsson, *Time series analysis and signal modeling*, Lund University, 2012.

[LC98] E.L. Lehmann and G. Casella, *Theory of point estimation*, Springer Texts in Statistics, Springer, 1998.

[LPNM12] T. Laursen, N.B. Pedersen, J.J. Nielsen, and T.K. Madsen, *Hidden markov model based mobility learning fo improving indoor tracking of mobile users*, Positioning Navigation and Communication (WPNC), 2012 9th Workshop on, march 2012, pp. 100 –104.

[OCDM08] J. Olsson, O. Cappé, R. Douc, and E. Moulines, *Sequential monte carlo smoothing with application to parameter estimation in nonlinear state space models*, Bernoulli **14** (2008), no. 1, 155–179.

[OR11] J. Olsson and T. Rydén, *Rao-blackwellization of particle markov chain monte carlo methods using forward filtering backward sampling*, IEEE Transactions on Signal Processing **59** (2011), no. 10, 4606–4619.

[PS99] K. Pitt and N. Shephard, *Filtering via simulation: Auxiliary particle filters*, Journal of the American Statistical Association **94** (1999), no. 446, 590–599.

[RC04] C. Robert and G. Casella, *Monte carlo statistical methods*, Springer Texts in Statistics, Springer, 2004.

[SARGM04] M Sanjeev Arulampalam, B Ristic, N Gordon, and T Mansell, *Bearings-only tracking of maneuvering targets using particle filters*, EURASIP Journal on Applied Signal Processing **15** (2004), no. 15, 2351–2365.

[SS00] Jason Small and Daniel P. Siewiorek, *Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure*.

[Str05] D.W. Stroock, *An introduction to markov processes*, Graduate Texts in Mathematics, Springer, 2005.

Appendix

The full calculation of (4.3) in matrix form.

$$\mathbb{E}\left[X_{k+1}|X_k, M_k\right] = \sum_j p_{M_k,j}\mathbb{E}\left[A(\alpha_{k+1}, M_{k+1})X_k|M_{k+1} = j, X_k\right] =$$

$$= p_{M_k,1}\mathbb{E}\left(\begin{bmatrix}1 & 0 & T & 0\\ 0 & 1 & 0 & T\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} X_k \middle| M_{k+1} = 1, X_k\right) +$$

$$+ p_{M_k,2}\mathbb{E}\left(\begin{bmatrix}1 & 0 & \frac{\sin(\alpha_{k+1}T)}{\alpha_{k+1}} & -\frac{1-\cos(\alpha_{k+1}T)}{\alpha_{k+1}}\\ 0 & 1 & \frac{1-\cos(\alpha_{k+1}T)}{\alpha_{k+1}} & \frac{\sin(\alpha_{k+1}T)}{\alpha_{k+1}}\\ 0 & 0 & \cos(\alpha_{k+1}T) & -\sin(\alpha_{k+1}T)\\ 0 & 0 & \sin(\alpha_{k+1}T) & \cos(\alpha_{k+1}T)\end{bmatrix} X_k \middle| M_{k+1} = 4, X_k\right) +$$

$$+ p_{M_k,3}\mathbb{E}\left(\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} X_k \middle| M_{k+1} = 3, X_k\right) +$$

$$+ p_{M_k,4}\mathbb{E}\left(\begin{bmatrix}1 & 0 & \frac{\sin(\alpha_{k+1}T)}{\alpha_{k+1}} & -\frac{1-\cos(\alpha_{k+1}T)}{\alpha_{k+1}}\\ 0 & 1 & \frac{1-\cos(\alpha_{k+1}T)}{\alpha_{k+1}} & \frac{\sin(\alpha_{k+1}T)}{\alpha_{k+1}}\\ 0 & 0 & \cos(\alpha_{k+1}T) & -\sin(\alpha_{k+1}T)\\ 0 & 0 & \sin(\alpha_{k+1}T) & \cos(\alpha_{k+1}T)\end{bmatrix} X_k \middle| M_{k+1} = 4, X_k\right) \approx$$

$$\approx p_{M_k,1}\begin{bmatrix}1 & 0 & T & 0\\ 0 & 1 & 0 & T\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} X_k +$$

$$+ p_{M_k,2}\begin{bmatrix}1 & 0 & \frac{\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]} & -\frac{1-\cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]}\\ 0 & 1 & \frac{1-\cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]} & \frac{\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]}\\ 0 & 0 & \cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T) & -\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)\\ 0 & 0 & \sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T) & \cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=2,X_k]T)\end{bmatrix} X_k +$$

$$+ p_{M_k,3}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} X_k +$$

$$+ p_{M_k,4}\begin{bmatrix}1 & 0 & \frac{\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]} & -\frac{1-\cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]}\\ 0 & 1 & \frac{1-\cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]} & \frac{\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)}{\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]}\\ 0 & 0 & \cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T) & -\sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)\\ 0 & 0 & \sin(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T) & \cos(\mathbb{E}[\alpha_{k+1}|M_{k+1}=4,X_k]T)\end{bmatrix} X_k$$