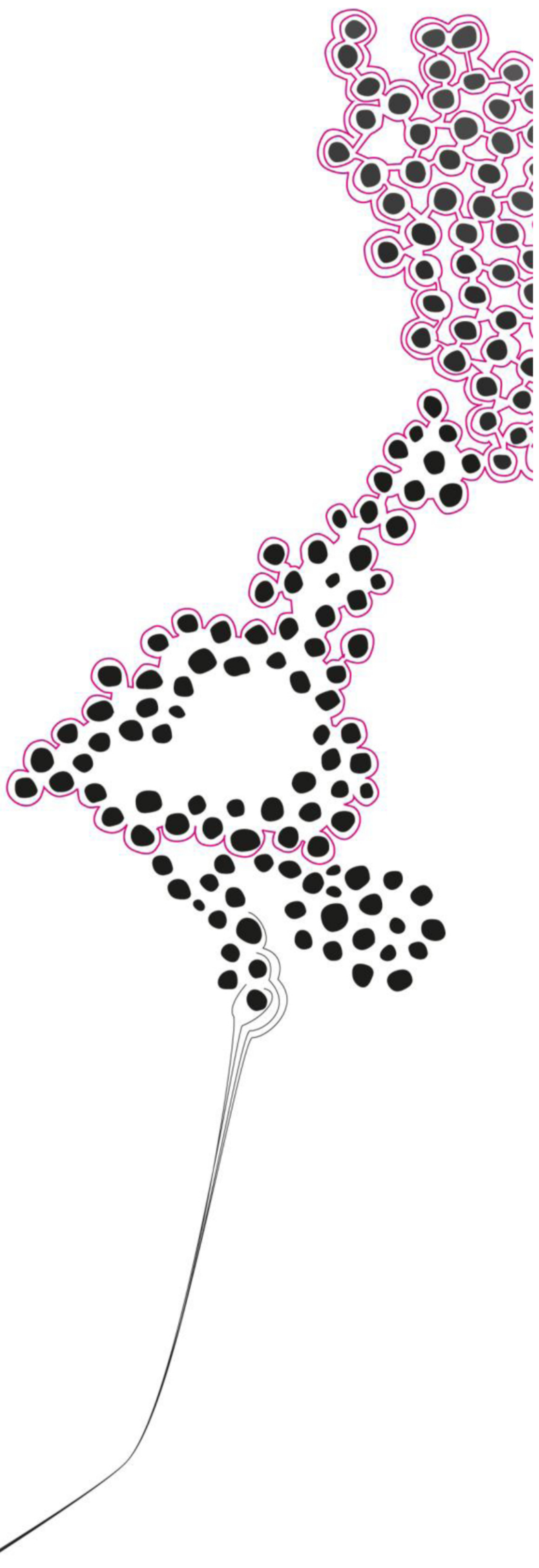
MASTER THESIS



DEALING WITH IMPERFECT KNOWLEDGE IN SENSOR PROCESSING

Menno Bootsveld

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE
CHAIR OF HYBRID SYSTEMS

EXAMINATION COMMITTEE
Prof. Dr. A.A. Stoorvogel
Dr. P.K. Mandal
Dr. M. Podt

19-12-2012

UNIVERSITY OF TWENTE.

Abstract

This thesis addresses the well-known problem of nonlinear target tracking. In its general form this problem can be solved by a state-of-the-art technique known as particle filtering. The solution can be made more accurate by applying constraints to the state space. In this research it is assumed that the constraints are not perfectly known. Given this assumption, we find a mathematically correct way of incorporating the imprecise hard constraints into a Bayesian filter.

First a discussion about information and imperfections is presented. Next a theoretical solution for incorporating imprecise hard constraints into a Bayesian filter is derived. For demonstration and analysis purposes, simulations are carried out. The theoretical method appears to be hard to apply in practice, therefore two methods to deal with imprecise knowledge in the form of geographical maps are presented. Finally, we make the first steps in optimizing the shortest distance method.

Acknowledgements

The research presented in this thesis is the result of my combined traineeship and final project for obtaining the degree Master of Science in Applied Mathematics at the University of Twente in Enschede. The project was carried out at Thales Nederland B.V. in Hengelo.

First of all, I would like to thank my supervisors Martin Podt, Yvo Boers and Anton Stoorvogel, for their guidance and support throughout the project. During the first half year both Yvo and Martin helped me with my research at Thales Nederland. After a half year, Yvo got ill and Martin took over the role as supervisor completely. I wish Yvo the best during his recovery and want to thank him once more for his help and useful instructions. During the last half year, Martin was my sole supervisor at Thales Nederland. He was always available for my countless questions and I had a lot of fun working with him. I would like to thank my colleagues Francesco Papi, Mélanie Bocquel and Fotios Katsilieris as well for their expertise and conversations they offered me. Thanks to my Paris roadtrip buddies Lionel Davies, Klaas Mussche, Michael ten Den and Gerard de Leeuw, with whom I spent many lunch breaks. And last but not least, thanks to my colleagues Klaasjan Wiersma, Perry Verveld, Linda Schipper and Hans Driessen.

Menno Bootsveld, december 2012

Contents

Chapter 1

Introduction

In this thesis dealing with imperfect knowledge in sensor processing is considered. This chapter deals with background information about radars and filtering. Also, the problem description and outline of the thesis is given.

1.1 Radar

The term *radar* is a contraction of the words *ra*dio *d*etection *a*nd *r*anging [15]. When the term was invented, radar engineers needed a device to detect the presence of a target and to measure its range. However, with modern techniques a radar can extract much more information from a target's echo than only its range.

Radar is an electromagnetic system for the detection and location of reflecting objects such as aircraft, ships, spacecraft, vehicles, people, and the natural environment. It operates by radiating energy into space and detecting the echo signal reflected from an object, or target. The reflected energy that is returned to the radar not only indicates the presence of a target, but by comparing the received echo signal with the signal that was transmitted, its location can be determined along with other target-related information. The basic principles of radar are illustrated in figure 1.1. A transmitter generates an electromagnetic signal that is radiated into space by an antenna. A portion of the transmitted energy is intercepted by the target and reradiated in many directions. The reradiation directed back towards the radar is collected by the radar antenna, which delivers it to a receiver. There it is processed to detect the presence of the target and determine its location. The range, or distance, to a target is found by measuring the time it takes for the radar signal to travel to the target and return back to the radar. The target's location in angle can be found from the direction the narrow-beamwidth radar antenna points when the received echo signal is of maximum amplitude. If the target is in motion, there is a shift in the frequency of the echo signal due to the doppler effect.

In the signal processing unit the electromagnetic signal is converted to a digital signal. The digital signal is at that point only raw data, it contains target signals, clutter signals and noise. The task of the target tracker is to produce the so-called *tracks*. Tracks are estimates of the state (position, speed and other variables of interest) of the target based on measurements subject to noise. Numerous filtering techniques are available to execute the task of *recursive state estimation*.

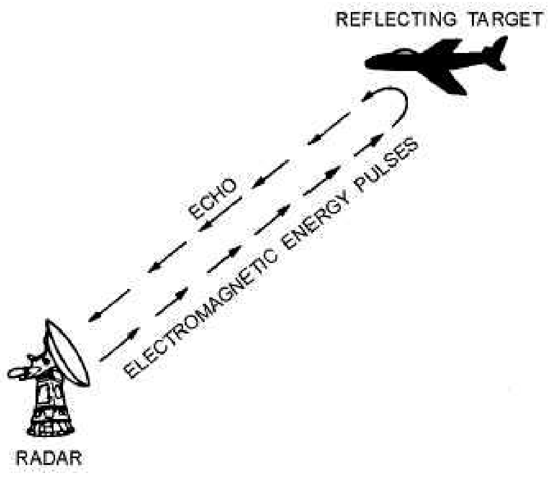


Figure 1.1: Basic principles of radar

1.2 Filtering

The fundamental building block of a tracking system is a filter for recursive state estimation [6]. A Kalman filter is the best known filter, a simple and elegant algorithm formulated more than 50 years ago. The filter computes the posterior distribution exactly for linear Gaussian systems by updating finite dimensional statistics recursively [9]. When systems tend to become nonlinear, the approach known as the Extended Kalman filter algorithm can be applied. Although it works well for mildly nonlinear systems, systems with 'big' nonlinearities cannot be filtered this way. Apart from the (Extended) Kalman filter there are other filters such as the unscented Kalman filter, Gaussian sum Kalman filter and point mass filter. The problem with any Kalman filter is that it can only handle Gaussian (sum) distributions. In this sense the point mass filter is an improvement, but for this filter the disadvantage lies in the grid: this is an approximation. The particle filter also provides a numerical approximation to the nonlinear filtering problem similar to the point mass filter, but uses an adaptive stochastic grid that automatically selects relevant grid points in the state space.

1.3 Problem description

We are familiar with models derived to deal with perfect knowledge in target tracking. However, in practice the knowledge we can use to improve target tracking is imperfect. This arises the question how to deal with imperfect knowledge in target tracking. Imperfections, in general, are a very broad class of errors applied to data. One of the research goals of this thesis is to categorize imperfections and make models for the different categories. Furthermore, we are interested in using these models to derive a way of dealing with imperfect knowledge in target tracking. Finally, we need practical algorithms to deal with imperfect knowledge in realistic situations.

1.4 Outline

In this thesis we treat the subject 'Dealing with imperfect knowledge in sensor processing'. In chapter 2 we look at what knowledge we use in sensor processing and in what way this knowledge can be imperfect. Imperfections appear to be heterogenous in nature; we distinguish different types of imperfections. One of the most common imperfections is imprecision, in chapter 4 we take a closer look at how to model imprecise data.

One of the tasks of sensor processing is to produce tracks out of target measurements. We can make the estimate of the state of targets more accurate by applying advanced filters to the data. A theoretically optimal filter is called the Bayesian filter, which is reviewed in a nutshell in chapter 3. It is already known that we can use external data to make filters more accurate. In the past, a model for perfect external knowledge was succesfully developed [8]. If we assumed the external knowledge to be perfect the model improved the accuracy of the tracks considerably. In this master thesis we take a deeper look into external data that is imprecise (chapter 4): we develop a model for incorporating imprecise external knowledge into a Bayesian filter.

One of the newest, most promising filters today is called the particle filter. This filtering technique is subject to research in this master thesis, we incorporate the filter with the model for imprecise external knowledge. In chapters 5 and 6 we perform simulations with the extended particle filter to test its performance. The external knowledge we use is in the form of digital charts which give us coastline position information. When we want to use this kind of information in practice, an approximation of the developed algorithm is needed. In chapter 6, we develop two methods that deal with this approximation: the shortest distance method and the grid method. Although the methods work alright, they have their disadvantages: the shortest distance method is slow and the grid method uses a lot of memory. In chapter 7, we set the first steps towards optimizing the shortest distance method. That is, using the low memory cost to its advantage and making the method faster to use. We conclude this master thesis with a discussion in chapter 8.

Chapter 2

Dealing with information and imperfections

Information can have different appearances. Data can be a geographical map, a table or spoken word for example. In our research we focus on using only structured data, i.e. data which can be represented by numbers. So in particular no video, text messages or similar (unstructured) data types.

When we are dealing with external information, we have to be very careful. In an ideal world all data is perfect, and can be used without hesitation. Unfortunately, we do not live in an ideal world and data is not perfect. If we would use all data without considering its possible imperfections, things may go wrong. Since data is inherently imperfect, we should take this into account when dealing with external data.

In this chapter we study the most important features of a digital map format: Electronic Navigational Charts (ENCs). Subsequently we take a look into imperfections in data: what causes imperfections and can we categorize them?

2.1 Electronic Navigational Charts (ENCs)

In our research we focus on targets constrained by its geographical surroundings which are visualized through geographical maps. Before we can use the information of geographical maps to the full extent, adjustments are needed. Regular geographical maps are nothing more but drawings indicating the various features of the terrain. In order to use these, we need a digital version of the conventional maps with actual data assigned to indicate the various features. This kind of digital map can be found in the form of ENCs: Electronic Navigational Charts. ENCs are vector maps being produced in the S-57 format of the International Hydrographic Organization (IHO) [10]. The ENC is internationally acknowledged as an official equivalent for the paper sea charts and is currently being used by ships for navigation purposes. An example of an ENC is depicted in figure 2.1.

Although the ENC is widely used, it is not perfect [11, 12]. These imperfections have their origin in the way ENCs are set up. Before electronic navigational charts were used, ships would navigate with paper charts. With the introduction of electronic displays, navigation agencies in-

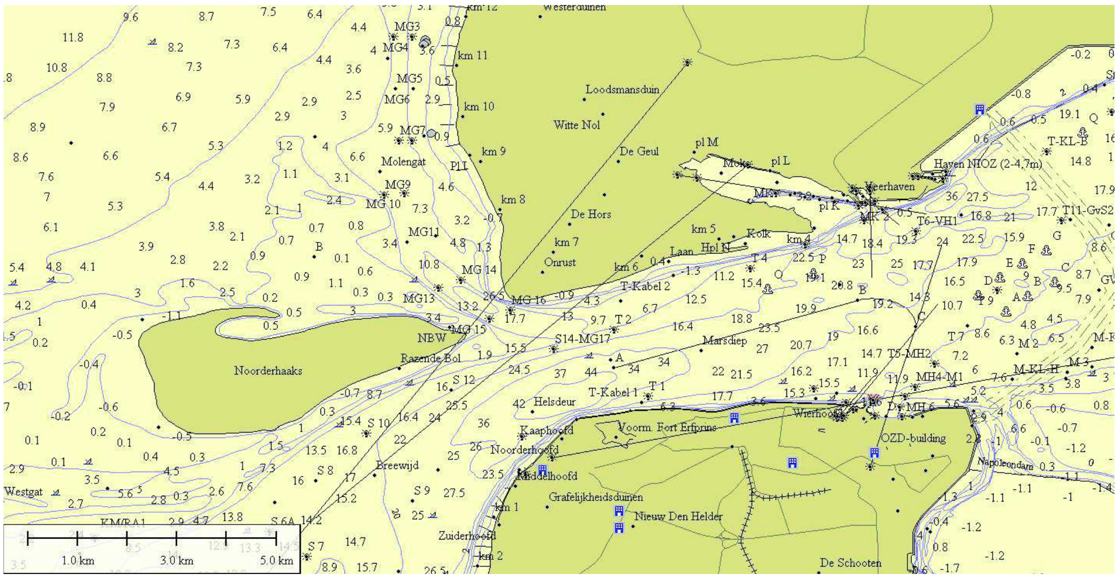


Figure 2.1: Electronic Navigational Chart of Marsdiep

troduced electronic charts. The ENCs were all built on the basis of paper charts [12]. One of the most important features of ENCs is that land masses are represented by polygons. Paper charts have been translated into a collection of connected lines which make up land areas. Standards have generally been such that surveys were conducted with a positioning accuracy of better than .75 millimeters [11]. Depending on the scale of the chart, this gives rise to a measurement error of a few meters for the smaller scales up until more than a kilometer for the larger scales. In our research, the ENC will be the basis for the geographical data.

2.2 Imperfections in data

When we are dealing with external data, we should realize that the data is often inherently imperfect. This is because the data was once retrieved from the 'real world'. And when one retrieves data, it has to be measured in one way or another.

2.2.1 Definitions

Measurements can be described according to [2], [3]:

$$y = \mu + \Delta + e, \tag{2.1}$$

with y the measurement, μ the accepted reference value or true value, Δ the bias or systematic error and e the random error. The random error e has a distribution with zero mean and variance σ_W^2 (the so-called 'within-laboratory variance') attached to it. Since Δ and e are in practice never zero, the true value cannot be exactly known. This gives us the idea that measurements

THALES

contain imperfections. Figure 2.2 gives an overview of imperfections related to measurements. The terminology in the diagram corresponds to the definitions below. The definitions are taken from ISO 3534-2, [4].

- *True value* (def. 3.2.5): a value which characterizes a quantity or quantitative characteristic perfectly defined in the conditions which exist when that quantity or quantitative characteristic is considered;
- *Measurement* (def. 3.2.1): a set of operations having the object of determining a value of a quantity;
- *Measurement result* (def. 3.4.2): a value of a quantity obtained by carrying out a specified measurement procedure;
- *Accuracy* (def. 3.3.1): the closeness of agreement between a test result or measurement result and the true value;
- *Trueness* (def. 3.3.3): the closeness of agreement between the expectation of a test result or a measurement result and a true value;
- *Precision* (def. 3.3.4): the closeness of agreement between independent test/measurement results obtained under stipulated conditions.

Imperfections can be very heterogeneous in nature due to different sources. The definition of true value, as given above is a formal definition for 'the actual value of that what we want to measure'. Whenever we want to gain information about the true value, we perform measurements. The actual values gained from the measurements are called 'measurement results'.

Every measurement result can be expressed in terms of bias and random error by (2.1). Together, $\Delta + e$ represent the accuracy of the measurement. Accuracy refers to a combination of trueness and precision, as can be seen in figure 2.2. All deviations from the true state are comprised of a systematic error (Δ) and a random error (e). The trueness of a measurement result is usually expressed in terms of bias, whereas precision only depends on the distribution of random errors. Precision is usually expressed in terms of imprecision and computed as a standard deviation of the test results or measurement results. In practice only discrete values can sometimes be measured with absolute accuracy (zero bias). When we want to takes measurements of continuous values, errors are involved. This means that accuracy plays an important role in virtually all measurements.

The Oxford dictionary [1] gives the following definitions for incompleteness and inconsistency:

- *Incompleteness*: the quality or state of being incomplete;
- *Incomplete*: not complete; not fully formed, made, or done; not whole entire or thorough; wanting some part; unfinished;
- *Inconsistency*: the quality, condition, or fact of being inconsistent;
- *Inconsistent*: not consistent, not agreeing in substance, spirit or form, not in keeping, not consonant or in accordance, at variance, discordant, incompatible, incongruous.

When a measurement result is not available or a measurement does not cover the complete true value, we say that the measurement is incomplete. Referring to equation (2.1), being incomplete

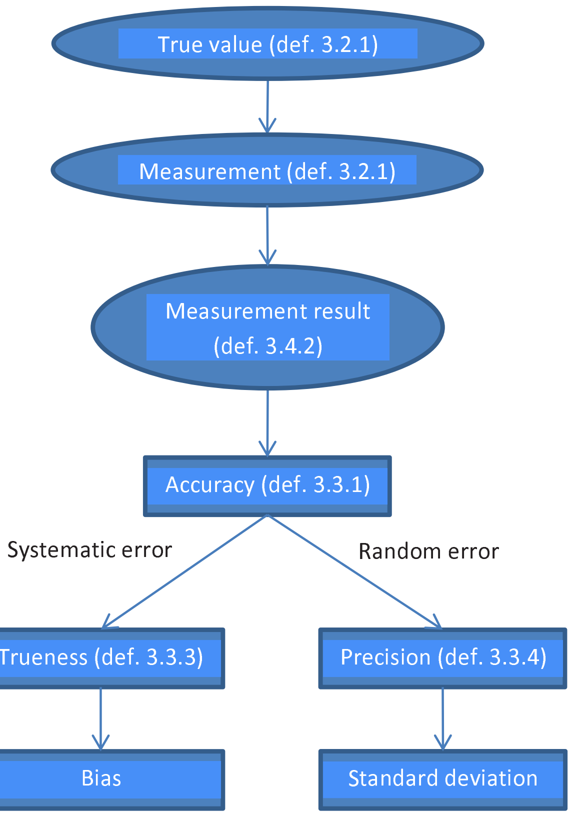


Figure 2.2: Overview of imperfections

means not having knowledge of the value of y. Also incompleteness can refer to no having knowledge of the value of Δ or the distribution of e. In this case we know that accuracy is present on measurements, but we do not know how the characteristics of the accuracy.

Finally, we distinguish the imperfection called inconsistency. We need more than one measurement for inconsistency to happen, suppose that we have n measurements of a true value:

$$\begin{aligned} y_1 &= \mu + \Delta_1 + e_1, \\ y_2 &= \mu + \Delta_2 + e_2, \\ &\vdots \\ y_n &= \mu + \Delta_n + e_n, \end{aligned}$$

with y_i the measurement, Δ_i the bias and e_i the random error distributed with zero mean and variance σ_i^2 for measurement i, $i = 1, \dots, n$. The true value is reflected by μ. Given this description of measurements, we call two or more measurements inconsistent if they have different bias. That

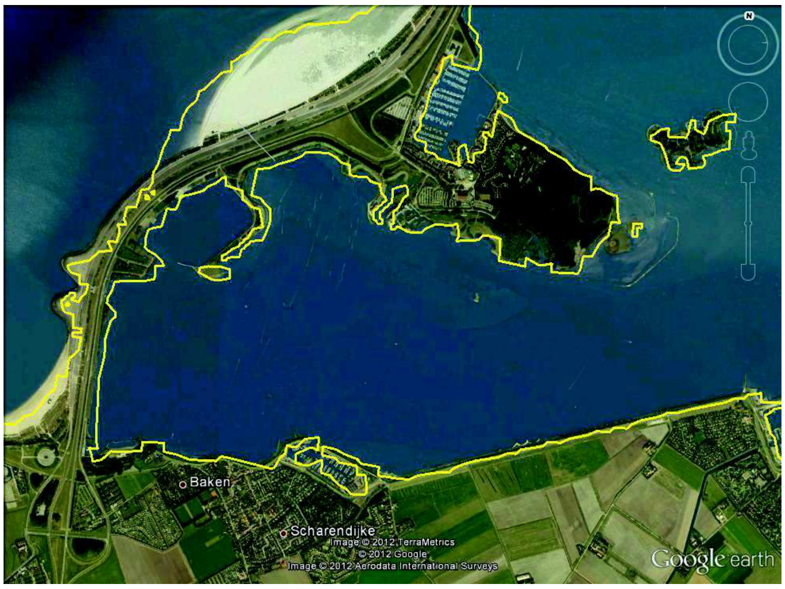

Baken
Scharendijke
Image © 2012 TerraMetrics
© 2012 Google
Image © 2012 Aerodata International Surveys
Google earth

Figure 2.3: Digital chart with accuracy (yellow line). True value depicted by the underlying photo. Source: Google Earth

is, a different random error does not make measurements inconsistent. Also, if one measurement is incomplete and another one is not we say that the measurements are inconsistent.

2.2.2 Examples

In this research we focus our attention to external knowledge in the form of geographical charts. Geographical charts are widely available in usable formats, for instance ENCs, and can be easily implemented in tracking software. If we consider ships as targets for our tracker, land masses form natural boundaries since ships cannot sail on land. Therefore geographical charts contain valuable information which can be translated into constraints for targets.

As we have already seen in section 2.1, digital charts contain imperfections. These imperfections can be related to the definitions in section 2.2.1 and can be best explained with the aid of examples. The first imperfection we addressed was accuracy. Accuracy consisted of trueness and precision; trueness was the systematic error measured in terms of bias and precision the random error. An example of accuracy on a digital chart is presented in figure 2.3. The digital chart is shown in yellow and the true value is shown by the underlying photo. We see that the digital chart does not follow the true situation perfectly. There is a bias on the digital chart: on average the chart is shifted to the southeast. Also, apart from the bias there is a random error present on the data, this embodies the precision of the data.

In the previous section, we talked about the imperfection known as incompleteness. The relation of incomleteness with digital charts is depicted in figure 2.4. The digital chart is shown in yellow and the true value is shown by the underlying photo. The photo shows the situation near the harbor of Den Helder, which is covered by an ENC. Now suppose we are interested in

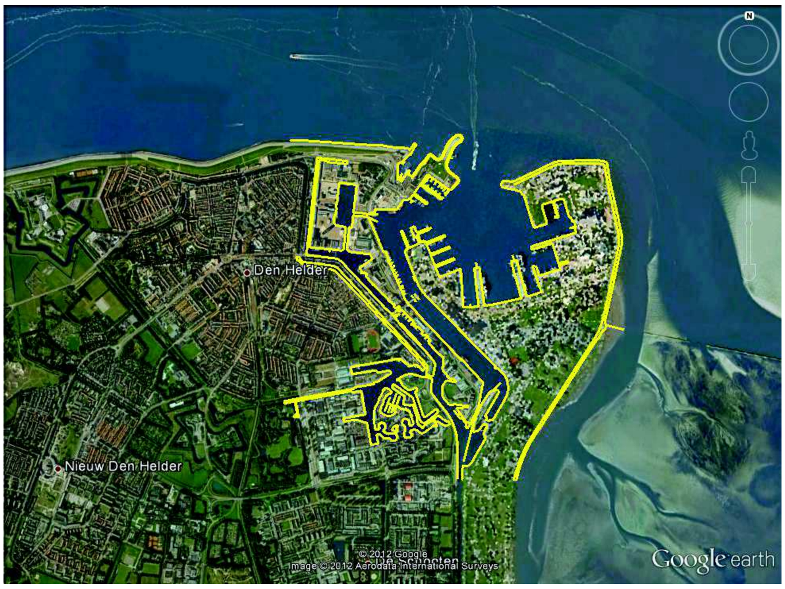


Figure 2.4: Incomplete digital chart (yellow line). True value depicted by the underlying photo. Source: Google Earth

a bigger surrounding than only the near environment. This imposes a problem: the chart we use is incomplete, meaning that there is vital information missing. This immediately shows that imperfections can be subjective: if we would have been interested in a smaller surrounding of the marine harbor, the digital chart might have sufficed and we would not have called it incomplete.

If we had two or more measurements, the measurements could be inconsistent towards one another. An example of inconsistency is displayed in figure 2.5. We see two different digital charts; the yellow chart is quite precise and follows the true state fairly good, the red chart is more imprecise and follows the true state more roughly. The underlying photo covers a section of the Rotterdam harbor and represents the true value. In section 2.2.1, we said that two digital charts are called consistent if they fall within the precision bounds. For figure 2.5 this means that the two digital charts are almost completely consistent with eachother. There is one exception: just northeast of the center of the area, there is a coastline construction which is covered by the yellow chart, but not by the red chart. It is most likely that incompleteness is involved in the red chart, which means the charts are inconsistent at that point.

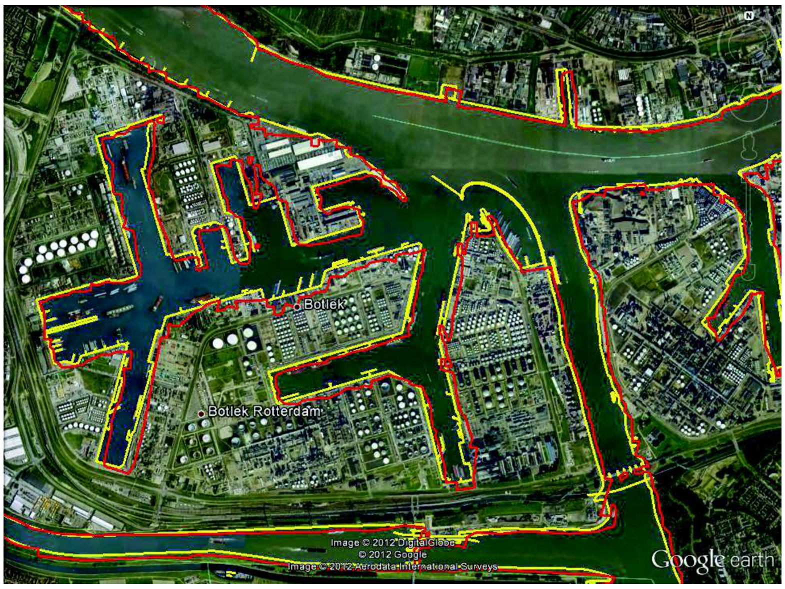

Botlek
Botlek Rotterdam
Image © 2012 DigitalGlobe
© 2012 Google
Image © 2012 Aerodata International Surveys
Google earth

Figure 2.5: Inconsistent digital charts (yellow and red lines). True value depicted by the underlying photo. Source: Google Earth

Chapter 3

Mathematical preliminaries

Before we dive into the theory of incorporating external information into a tracking filter, the theory of filtering is introduced. In this chapter, we set up a theoretical framework which is able to handle tracking a single target. This framework is called a Bayesian framework and is widely used in all kinds of filters. The Bayesian framework is based on Bayes' rule [13]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{3.1}$$

This formula is the key to succes for determining state estimates. One of the filters where it is used is the particle filter. Both Bayesian and particle filtering are reviewed in this chapter.

3.1 Bayesian filtering

We set up the Bayesian framework for the filtering problem we consider, following [6]. First of all, let us define the state vector $\mathbf{s}_k \in \mathbb{R}^{n_s}$, where n_s is the dimension of the state and $k \in \mathbb{N}$ is the time index. We assume that measurements $\mathbf{z}_k \in \mathbb{R}^{n_s}$ are available at every timestep k, where the time between k and $k+1$ equals T (so $t_{k+1} - t_k := T, \forall T$). Given the sequence of measurements $\mathbf{Z}_k := \{\mathbf{z}_i, i = 1, \dots, k\}$ at timestep k, the nonlinear filtering problem is deducing the state $\mathbf{s}_k$ from the given measurements. The nonlinear filtering problem is sometimes referred to as the Bayesian filtering problem. Suppose the target state evolves according to the dynamic model

$$\mathbf{s}_{k+1} = \mathbf{f}_k(\mathbf{s}_k, \mathbf{v}_k), \tag{3.2}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{s}_k, \mathbf{w}_k). \tag{3.3}$$

The equations above are respectively called *dynamical model* and *measurement model.* Here $\mathbf{f}_k$ is a function of $\mathbf{s}_k$ and $\mathbf{v}_k$, $\mathbf{h}_k$ is a function of $\mathbf{s}_k$ and $\mathbf{w}_k$, $\mathbf{v}_k \sim p_{\mathbf{v}}(\mathbf{v}_k)$ and $\mathbf{w}_k \sim p_{\mathbf{w}}(\mathbf{w}_k)$ are both noise vectors. Furthermore, we assume that our original target state $\mathbf{s}_0$ has a known probability density function $p(\mathbf{s}_0)$. The probabilistic model for the state evolution (often referred to as *transitional density*), $p(\mathbf{s}_k|\mathbf{s}_{k-1})$, is defined by the dynamical model (3.2) and the known statistics of $\mathbf{v}_{k-1}$. The *likelihood function* $p(\mathbf{z}_k|\mathbf{s}_k)$ is defined by the measurement model (3.3) and the known statistics of $\mathbf{w}_k$.

We are interested in $\mathbf{s}_k$, the state at time step k. To derive information about that state, we should make use of the measurements $\mathbf{Z}_k := \{\mathbf{z}_i, i = 1, \dots, k\}$ and the initial probability density function $p(\mathbf{s}_0)$. In particular, we are interested in the posterior probability density function $p(\mathbf{s}_k|\mathbf{Z}_k)$. This pdf may be obtained recursively in two steps: the prediction step and the update step. That is, given $p(\mathbf{s}_{k-1}|\mathbf{Z}_{k-1})$ we look for $p(\mathbf{s}_k|\mathbf{Z}_{k-1})$, next given $p(\mathbf{s}_k|\mathbf{Z}_{k-1})$ we look for $p(\mathbf{s}_k|\mathbf{Z}_k)$.

For a given $p(\mathbf{s}_{k-1}|\mathbf{Z}_{k-1})$, we obtain the prediction density through the Chapman-Kolmogorov equation:

$$\begin{aligned} p(\mathbf{s}_k|\mathbf{Z}_{k-1}) &= \int p(\mathbf{s}_k|\mathbf{s}_{k-1}, \mathbf{Z}_{k-1}) p(\mathbf{s}_{k-1}|\mathbf{Z}_{k-1}) \mathrm{d}\mathbf{s}_{k-1}, \\ &= \int p(\mathbf{s}_k|\mathbf{s}_{k-1}) p(\mathbf{s}_{k-1}|\mathbf{Z}_{k-1}) \mathrm{d}\mathbf{s}_{k-1}. \end{aligned} \tag{3.4}$$

This equation defines the prediction step.

When the new measurement $\mathbf{z}_k$ becomes available, we update our prediction with the new information. This is done with the aid of Bayes' rule (3.1).

$$\begin{aligned} p(\mathbf{s}_k|\mathbf{Z}_k) &= p(\mathbf{s}_k|\mathbf{z}_k, \mathbf{Z}_{k-1}), \\ &= \frac{p(\mathbf{z}_k|\mathbf{s}_k, \mathbf{Z}_{k-1}) p(\mathbf{s}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})}, \\ &= \frac{p(\mathbf{z}_k|\mathbf{s}_k) p(\mathbf{s}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})}, \\ p(\mathbf{z}_k|\mathbf{Z}_{k-1}) &= \int p(\mathbf{z}_k|\mathbf{s}_k) p(\mathbf{s}_k|\mathbf{Z}_{k-1}) \mathrm{d}\mathbf{s}_k. \end{aligned} \tag{3.5}$$

The term $p(\mathbf{z}_k|\mathbf{Z}_{k-1})$ is the normalization constant.

When we are in possesion of the posterior density function $p(\mathbf{s}_k|\mathbf{Z}_k)$, we can extract point estimators. Optimal state measures are the mimimum mean-square error (MMSE) and the maximum a posterior (MAP) estimators. The MMSE estimate is the conditional mean of $\mathbf{s}_k$:

$$\hat{\mathbf{s}}_k^{MMSE} := \mathbb{E}[\mathbf{s}_k|\mathbf{Z}_k] = \int \mathbf{s}_k p(\mathbf{s}_k|\mathbf{Z}_k) \mathrm{d}\mathbf{s}_k.$$

The MAP estimate is the maximum of $p(\mathbf{s}_k|\mathbf{Z}_k)$:

$$\hat{\mathbf{s}}_{k|k}^{MAP} := \arg\max_{\mathbf{s}_k} p(\mathbf{s}_k|\mathbf{Z}_k).$$

3.2 Particle filter

The theory presented in the previous section is merely a theoretical framework for solving the nonlinear filtering problem. In most practical situations, the recursive Bayesian state estimation will not suffice. The main problem is that for the solution to work, it requires the storage of the entire pdf, which is in general terms equivalent to an infinite dimensional vector [6]. To avoid this problem you could represent the required posterior density function by a set of random samples with associated weights and compute estimates based on these samples and weights. This is the key idea to a Sequential Monte Carlo (SMC) approach known as *particle filtering* [5]. In this section, the method of particle filtering is explained in a nutshell, where we will roughly follow [6].

Particle filtering is a filtering technique that uses Monte Carlo integration to avoid the problem of storing the entire pdf. The Importance Sampling method (a more general version of Monte Carlo integration) is applied to the recursive Bayesian state estimation to form the Sequential Importance Sampling (SIS) algorithm. The SIS algorithm forms the basis for most particle filters, in spite of the fact that it degenerates. Degeneration means that after a certain number of time steps, all but one particle have negligible weights. This problem is solved by making use of the resampling technique. Which is done in the Sampling Importance Resampling (SIR) filter, see algorithm 1; resampling is performed every time step. Here, $\mathbf{s}_k^i$ and w_k^i are respectively the state vector and corresponding weight for particle i and time step k. N_p indicates the number of particles. Other variables are defined accordingly.

Algorithm 1: SIR Particle Filter

```
Initialization:
Sample initial particles ${\mathbf{s}_0^{(i)}}_{i=1}^{N_p}$ from $p(\mathbf{s}_0)$;
Set the weights $w_0^{(i)}$ to $\frac{1}{N_p}$.
input  : $\{\mathbf{s}_{k-1}^{(i)}, w_{k-1}^{(i)}\}_{i=1}^{N_p}$ and a new measurement, $\mathbf{z}_k$.
output: $\{\mathbf{s}_k^{(i)}, w_k^{(i)}\}_{i=1}^{N_p}$.
1 - Prediction:
for $i = 1, \ldots, N_p$ do
    Sample $\mathbf{v}_{k-1}^{(i)}$ from $p_{\mathbf{v}}(\mathbf{v}_{k-1})$;
    Generate a new particle : $\hat{\mathbf{s}}_k^{(i)} = f(\mathbf{s}_{k-1}^{(i)}, \mathbf{v}_{k-1}^{(i)})$.
end
2 - Update:
for $i = 1, \ldots, N_p$ do
    Compute weights : $\tilde{w}_k^{(i)} = w_{k-1}^{(i)} p(\mathbf{z}_k|\hat{\mathbf{s}}_k^{(i)})$;
end
Normalize the weights : $w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{j=1}^{N_p} \tilde{w}_k^{(j)}}$.
3 - Resample:
Generate a new set of particles $\{\mathbf{s}_k^{(j)}\}_{j=1}^{N_p}$,
so that for any $j$, $P(\mathbf{s}_k^{(j)} = \hat{\mathbf{s}}_k^{(i)}) = w_k^{(i)}$;
Set the weights $w_k^{(i)}$ to $\frac{1}{N_p}$.
```

The SIR particle filter is the filter we use in our application. In practice, we see that a few adaptations are done but the general algorithm 1 is still applicable. For more in depth information about particle filtering, we refer to [6].

Chapter 4

Dealing with imprecise external data in a Bayesian framework

In chapter 2, we saw that information is often imperfect. We also saw examples of imperfections in geographical maps. As formula (2.1) indicates, measurements are not perfect descriptions of the true value. Every type of structured knowledge was once measured and thus is biased and/or imprecise. In this chapter we assume unbiased measurements ($\Delta = 0$) and look for a way to incorporate imprecise external data into a Bayesian filter.

In the past, research has been conducted to come up with a model for external knowledge and how to incorporate it into a Bayesian framework [8]. For modelling purposes, this knowledge was assumed to be perfect. Also in this research, it was assumed that the knowledge was available in the form of structured data so that it could be incorporated into the Bayesian framework by using hard constraints. If we replace the assumption that the external information is perfect, and we replace it by the assumption that the external information is imprecise, we arrive at the notion of *imprecise hard constraints.*

In this chapter, we introduce a way of modelling constraints as imprecise hard constraints. Next, it is described how we incorporate the model for imprecise hard constraints into the Bayesian filter. The last section discusses a model for constraints on position.

4.1 Incorporating imprecise hard constraints in a Bayesian filter

In our model, we follow the framework that has been introduced in [8] and extend it with imprecise external data. For starters, let us assume the system is described by the following state and measurement equations (3.2,3.3):

$$\begin{aligned} \mathbf{s}_{k+1} &= \mathbf{f}_k(\mathbf{s}_k, \mathbf{v}_k), \\ \mathbf{z}_k &= \mathbf{h}_k(\mathbf{s}_k, \mathbf{w}_k). \end{aligned}$$

where $\mathbf{s}_k \in \mathbb{R}^{n_s}$ is the system state, and n_s its dimension. $\mathbf{z}_k$ is the measurement vector, $\mathbf{v}_k \sim p_{\mathbf{v}}(\mathbf{v}_k)$ and $\mathbf{w}_k \sim p_{\mathbf{w}}(\mathbf{w}_k)$ are both noise vectors. By assumption, we impose Markov properties on our system, i.e.

$$p(\mathbf{s}_k|\mathbf{s}_{k-1},\mathbf{s}_{k-2},\dots,\mathbf{s}_0) = p(\mathbf{s}_k|\mathbf{s}_{k-1})$$
$$p(\mathbf{z}_k|\mathbf{z}_{k-1},\mathbf{z}_{k-1},\dots\,\mathbf{z}_0,\mathbf{s}_k,\mathbf{s}_{k-1},\dots,\mathbf{s}_0) = p(\mathbf{z}_k|\mathbf{s}_k).$$

Here, $p(\mathbf{s}_k|\mathbf{s}_{k-1})$ is known as the *transitional density* and $p(\mathbf{z}_k|\mathbf{s}_k)$ is known as the *likelihood function.*

We are looking for a way to incorporate imprecise hard constraint into the Bayesian framework. Consequently, we need a model for imprecise hard constraints. In [8], nonlinear hard constraints were modelled as:

$$\mathbf{a}_k \le \mathbf{C}_k(\mathbf{s}_k) \le \mathbf{b}_k,$$

where $\mathbf{C}_k(\mathbf{s}_k) : \mathbb{R}^{n_s} \mapsto \mathbb{R}^{n_s}$. Furthermore, the set of all states satisfying these constraints is modelled as

$$\mathcal{C}_k := \{\mathbf{s}_k|\mathbf{s}_k \in \mathbb{R}^{n_s}, \mathbf{a}_k \le \mathbf{C}_k(\mathbf{s}_k) \le \mathbf{b}_k\}. \tag{4.1}$$

In contradiction to hard constraints, imprecise hard constraints are unknown apart from their distribution. That is, the exact value is unknown, while the distribution of the constraints is known. By assumption, we impose a probability distribution onto the constraints, i.e.

$$\mathbf{a}_k \le \mathbf{C}_k(\mathbf{s}_k) \le \mathbf{b}_k, \tag{4.2}$$
$$\mathbf{a}_k = \overline{\mathbf{a}}_k + \boldsymbol{\alpha}_k, \quad \boldsymbol{\alpha}_k \sim \phi_k, \tag{4.3}$$
$$\mathbf{b}_k = \overline{\mathbf{b}}_k + \boldsymbol{\beta}_k, \quad \boldsymbol{\beta}_k \sim \phi_k, \tag{4.4}$$
$$\phi_k = \phi_k(\mathbf{s}_k). \tag{4.5}$$

The distribution function ϕ_k is responsible for the imprecision on the constraints. The dependence on $\mathbf{s}_k$ can be best explained by looking at a more specific model. This specific model is inspired by the use of multiple ENCs for an area: in practice adjoint ENCs often have different precision. Suppose that the state space we consider is split up into n non-intersecting subspaces S_i, i.e.

$$S = S_1 \cup S_2 \cup \dots \cup S_n, \quad S_i \cap S_j = \varnothing, \quad i \ne j,$$

and suppose that the following constraints hold:

$$\begin{gathered}
\mathbf{a}_{1,k} \le \mathbf{C}_{1,k}(\mathbf{s}_k) \le \mathbf{b}_{1,k}, \quad \forall \mathbf{s}_k \in S_1 \\
\mathbf{a}_{2,k} \le \mathbf{C}_{2,k}(\mathbf{s}_k) \le \mathbf{b}_{2,k}, \quad \forall \mathbf{s}_k \in S_2 \\
\vdots \\
\mathbf{a}_{n,k} \le \mathbf{C}_{n,k}(\mathbf{s}_k) \le \mathbf{b}_{n,k}, \quad \forall \mathbf{s}_k \in S_n \\
\mathbf{a}_{i,k} = \overline{\mathbf{a}}_{i,k} + \boldsymbol{\alpha}_{i,k}, \quad \boldsymbol{\alpha}_{i,k} \sim \phi_{i,k}, \quad i = 1,\dots,n \\
\mathbf{b}_{i,k} = \overline{\mathbf{b}}_{i,k} + \boldsymbol{\beta}_{i,k}, \quad \boldsymbol{\beta}_{i,k} \sim \phi_{i,k}, \quad i = 1,\dots,n
\end{gathered}$$

A more general way of writing this model is by letting $\phi_k = \phi_k(\mathbf{s}_k)$ as in (4.5).

In [8], external information was exploited in two ways: in the prediction step and in the update step. We follow the method of exploiting external information via the update step. The update step could be written as

$$p(\mathbf{s}_k|\mathbf{z}_{1:k},\mathcal{C}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{s}_k)p(\mathcal{C}_k|\mathbf{s}_k)p(\mathbf{s}_k|\mathbf{z}_{1:k},\mathcal{C}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1},\mathcal{C}_{1:k})p(\mathcal{C}_k|\mathcal{C}_{1:k-1})}. \tag{4.6}$$

In this update step, we identify the following terms: the likelihood function $p(\mathbf{z}_k|\mathbf{s}_k)$, the predictive density function $p(\mathbf{s}_k|\mathbf{z}_{1:k}, \mathcal{C}_{1:k-1})$ and the normalization factor $p(\mathbf{z}_k|\mathbf{z}_{1:k-1}, \mathcal{C}_{1:k})p(\mathcal{C}_k|\mathcal{C}_{1:k-1})$. The remaining factor $p(\mathcal{C}_k|\mathbf{s}_k)$ is called the hard constrained likelihood, and is given as

$$p(\mathcal{C}_k|\mathbf{s}_k) = \frac{p(\mathbf{s}_k|\mathbf{s}_{k-1}, \mathcal{C}_k)}{p(\mathbf{s}_k|\mathbf{s}_{k-1})} = \begin{cases} 1, & \mathbf{s}_k \in \mathcal{C}_k, \\ 0, & \text{otherwise.} \end{cases}$$

For imprecise hard constraints, $p(\mathcal{C}_k|\mathbf{s}_k)$ has to be defined differently, since we do not know when $\mathbf{s}_k \in \mathcal{C}_k$.

In equation (4.1), we gave the definition for the set $\mathcal{C}_k$ of all states which satisfy the constraints. This set can be seen as an intersection of two sets $\mathcal{C}_{a,k}$ and $\mathcal{C}_{b,k}$, being defined as

$$\begin{aligned} \mathcal{C}_{a,k} :&= \{\mathbf{s}_k|\mathbf{s}_k \in \mathbb{R}^{n_s}, \mathbf{C}_k(\mathbf{s}_k) \geq \mathbf{a}_k\}, \\ &= \{\mathbf{s}_k|\mathbf{s}_k \in \mathbb{R}^{n_s}, \boldsymbol{\alpha}_k \leq \mathbf{C}_k(\mathbf{s}_k) - \bar{\mathbf{a}}_k\}, \\ \mathcal{C}_{b,k} :&= \{\mathbf{s}_k|\mathbf{s}_k \in \mathbb{R}^{n_s}, \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k\}, \\ &= \{\mathbf{s}_k|\mathbf{s}_k \in \mathbb{R}^{n_s}, \boldsymbol{\beta}_k \geq \mathbf{C}_k(\mathbf{s}_k) - \bar{\mathbf{b}}_k\}. \end{aligned}$$

This gives us $\mathcal{C}_k = \mathcal{C}_{a,k} \cap \mathcal{C}_{b,k}$. For the set $\mathcal{C}_{a,k}$, by definition its conditional likelihood can be written as:

$$p(\mathcal{C}_{a,k}|\mathbf{s}_k) := \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k.$$

Similarly, the conditional likelihood of $\mathcal{C}_{b,k}$ can be written as

$$p(\mathcal{C}_{b,k}|\mathbf{s}_k) := \int_{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k}^{\infty} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k.$$

This implies that for $p(\mathcal{C}_k|\mathbf{s}_k)$, we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{s}_k) &= p(\mathcal{C}_{a,k} \cap \mathcal{C}_{b,k}|\mathbf{s}_k), \\ &= p(\mathcal{C}_{a,k}|\mathbf{s}_k)p(\mathcal{C}_{b,k}|\mathbf{s}_k), \\ &= \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k \int_{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k}^{\infty} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k, \\ &= \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k \left[1 - \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k.\right] \end{aligned}$$

The terms $\int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k = \Phi_{\boldsymbol{\alpha},k}(\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k)$ and $\int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k = \Phi_{\boldsymbol{\beta},k}(\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k)$ can be seen as the cumulative distributions of $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$. So we can write our expression for $p(\mathcal{C}_k|\mathbf{s}_k)$ as

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{s}_k) &= \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k \left[1 - \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\bar{\mathbf{b}}_k} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k\right], && (4.7) \\ &= \Phi_{\boldsymbol{\alpha},k}(\mathbf{C}_k(\mathbf{s}_k) - \bar{\mathbf{a}}_k)[1 - \Phi_{\boldsymbol{\beta},k}(\mathbf{C}_k(\mathbf{s}_k) - \bar{\mathbf{b}}_k)], && (4.8) \\ &= \Phi_k(\mathbf{s}_k). && (4.9) \end{aligned}$$

We name our function $\Phi_k(\mathbf{s}_k)$ the *factorization function*.

The factorization function has a nice interpretation: $\Phi_k(\mathbf{s}_k)$ appears to be equal to the probability that $\mathbf{s}_k$ satisfies the constraints.

$$\begin{aligned}
\Phi_k(\mathbf{s}_k) &= \Phi_{\boldsymbol{\alpha},k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k)[1 - \Phi_{\boldsymbol{\beta},k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k)], \\
&= P\{\boldsymbol{\alpha}_k \leq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k\}[1 - P\{\boldsymbol{\beta}_k \leq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k\}], \\
&= P\{\boldsymbol{\alpha}_k \leq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k\}P\{\boldsymbol{\beta}_k \geq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k\}.
\end{aligned}$$

By assumption the vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ are distributed independently, so

$$\begin{aligned}
\Phi_k(\mathbf{s}_k) &= P\{\boldsymbol{\alpha}_k \leq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k, \boldsymbol{\beta}_k \geq \mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k\}, \\
&= P\{\mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k), \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k\}, \\
&= P\{\mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k\}, \\
&= P\{\mathbf{s}_k \in \mathcal{C}_k\}.
\end{aligned}$$

Imprecise hard constraints can be seen as a general case of hard constraints. If we set $\phi_k = \delta$, the Dirac delta function, we have $\boldsymbol{\alpha}_k = \boldsymbol{\beta}_k = 0$ which gives us $\mathbf{a}_k = \overline{\mathbf{a}}_k$ and $\mathbf{b}_k = \overline{\mathbf{b}}_k$. This makes the constraints $\mathbf{a}_k$ and $\mathbf{b}_k$ perfectly precise. We see that the generalization of the hard constraints exactly coincides with the definition of hard constraints:

$$\begin{aligned}
p(\mathcal{C}_{a,k}|\mathbf{s}_k) &= \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{a}}_k} \delta(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k, \\
&= \begin{cases} 1, & \text{if } \mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k) \\ 0, & \text{otherwise} \end{cases} \\
p(\mathcal{C}_{b,k}|\mathbf{s}_k) &= \int_{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{a}}_k}^{\infty} \delta(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k, \\
&= \begin{cases} 1, & \text{if } \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k \\ 0, & \text{otherwise} \end{cases} \\
p(\mathcal{C}_k|\mathbf{s}_k) &= p(\mathcal{C}_{a,k}|\mathbf{s}_k)p(\mathcal{C}_{b,k}|\mathbf{s}_k), \\
&= \begin{cases} 1, & \text{if } \mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

The conditional likelihood function for the generalized case of hard constraints is equal to the conditional likelihood function as given in [8].

4.2 The model for imprecise hard constraints applied to position

A substantial part of research is devoted to position constraints, i.e. constraints on $[x_k\ y_k]$. When we apply the model for imprecise hard constraints, we should take into account a number of considerations. For (4.2-4.4), we assume that $\overline{\mathbf{a}}_k$, $\overline{\mathbf{b}}_k$ and ϕ_k are constant over time:

$$\begin{aligned}
&\mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k,\\
&\mathbf{a}_k = \overline{\mathbf{a}} + \boldsymbol{\alpha}_k, \quad \boldsymbol{\alpha}_k \sim \phi,\\
&\mathbf{b}_k = \overline{\mathbf{b}} + \boldsymbol{\beta}_k, \quad \boldsymbol{\beta}_k \sim \phi.
\end{aligned}$$

This means that although the mean and the distribution of $\mathbf{a}_k$ is constant over time, $\mathbf{a}_k$ itself is not, which is due to the fact that every time step a new value is drawn from the distribution ϕ. At first glance, it may seem like a rough model for imprecise hard constraints on position. That is, in reality position constraints, although unknown, tend to be fixed over time. Consider for example the coast of a river.

However, we can make this way of modelling imprecise hard constaints on position reasonable. In some cases, constraints on position do change over time. If we would measure the position of a coastline near a beach and we only measure a few times a day, we would see a changing coastline due to tides. Also, when we take a more detailed look and measure the coastline very often, we see a changing coastline due to waves crashing against the beach.

More importantly, the model is valid if we track a target which is moving fast enough. Consider a ship moving along a coastline. If the ship moves fast enough parallel to the coastline, it will encounter a 'new' piece of coastline every time a measurement is done. If the new position of the ship $\mathbf{s}_k$ is such that the correlation between the point on the coastline closest to the ship and the point on the coastline closest to the previous position of the ship is negligible, then we can safely draw new values for $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ from ϕ. This reasoning does not hold for stationary or barely moving targets. Using imprecise hard constraints with these type of targets requires a different model.

Chapter 5

Simulations with a theoretical scenario

In the previous chapter, we developed a general model for incorporating imprecise hard constraints into a Bayesian framework. Also did we develop a model for imprecise hard constraints on the position of a target which moves fast enough. For analysis and demonstration of the model for imprecise hard constraints, we perform simulations. We simulate a ship which sails through a canal and want to track its whereabouts. We do this with a radar which receives measurements of the ship, after that a particle filter performs the filtering and produces tracks. Tracks are estimates of the true state of the target for every time step. Of course, we want to make these estimates as accurate as possible. Although the particle filter already improves accuracy a lot, we can make the estimates even more accurate by incorporating external information. In our case, the external information is a geographical map of the surroundings showing water and land areas. This imposes constraints on the position of the ship. The analysis in chapter 2 indicates that this kind of information is rarely perfect. In the simulations we assume that this is also the case for our map: the map we use is imperfect. There is a certain degree of imprecision present on our map, which means that we do not know exactly where the coastlines are located. We assume that there is no bias present on this map.

In this chapter, we compare the performance of several particle filters to study the influence of imprecise hard constraints. We study a toy example, which has the nice property that the values of the factorization function Φ_k can be determined analytically. First we will explain the simulation setup: we zoom in on the used model, the simulation scenario and parameter settings. Next, we present the simulation results and draw conclusions where possible. In the next chapter, we will drop the property that the factorization function Φ_k can be calculated analytically and focus on methods which are able to deal with real ENCs.

5.1 Simulation setup

This section describes the simulation setup: the scenario is explained and the derivation of the corresponding model is given. Furthermore we give attention to performance measures and describe the tests that are done in the scenario.

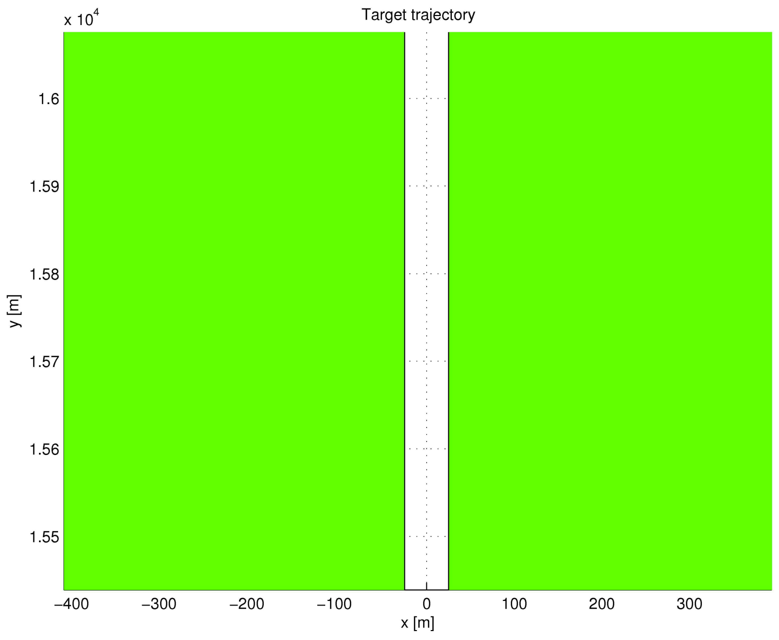


Figure 5.1: Simulation scenario: abstract canal

5.1.1 Simulation scenario

We consider the case of a vertical canal that has a width of 50 meters, it is shown in figure 5.1. The green rectangles denote land areas, the blanc area denotes water. In the canal we simulate the movement of a ship, which travels south to north with constant speed. A tracking radar receives measurements of the ship and converts them to tracks with the aid of a particle filter.

Following figure 5.1, this situation can be modelled as follows. The dynamical equations for the ship are set to

$$\mathbf{s}_{k+1} = F\mathbf{s}_k + \mathbf{w}_k, \tag{5.1}$$

$$\mathbf{s}_k = \begin{bmatrix} x_k & v_{x,k} & y_k & v_{y,k} \end{bmatrix}^T. \tag{5.2}$$

Here, x_k and y_k are the (x, y)-coordinates of the ship for time step k and $v_{x,k}$ and $v_{y,k}$ the velocities in the x and y directions respectively.

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.3}$$

In the above equation, T represents the update time. The Brownian motion model is applied, which means that the covariance matrix of the process noise $\mathbf{w}_k$ [7] is given by:

$$Q = \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 & 0 & 0 \\ \frac{1}{2}T^2 & T & 0 & 0 \\ 0 & 0 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ 0 & 0 & \frac{1}{2}T^2 & T \end{bmatrix} q. \tag{5.4}$$

Here, q represents the variance of acceleration (both in x and y direction). For normal distributions, we have that 99.9% of the distribution falls below 3σ. So by assuming $v_{x,k} \sim \mathcal{N}(0,\sigma)$ and $v_{y,k} \sim \mathcal{N}(0,\sigma)$, we could safely say that the maximum acceleration $a_{\max} = 3\sigma$. This results in $q = \sigma^2 = (\frac{a_{\max}}{3})^2$.

The available measurements at each time step are assumed to be the target range r_k, the bearing b_k and the Doppler velocity d_k.

$$\begin{aligned} r_k &= \sqrt{x_k^2 + y_k^2}, \\ b_k &= \arctan\left(\frac{x_k}{y_k}\right), \\ d_k &= -\frac{x_k v_{x,k} + y_k v_{y,k}}{r_k}. \end{aligned}$$

The measurement noise is zero-mean Gaussian noise with a standard deviation of σ_r, σ_b and σ_d for the range, bearing and Doppler, respectively.

Since the ship is not allowed to sail on land, the position of the ship is subject to constraints. Following figure 5.1, the constraints in the case of perfect knowledge can be formulated as

$$-25 \le x_k \le 25, \quad y_k \in \mathbb{R}, \quad v_{x,k}, v_{y,k} \in \mathbb{R}.$$

Formulating the constraints in the case of imprecise knowledge is not as easy. We can however, formulate the constraints for a fixed y_k:

$$\begin{aligned} a_k &\le x_k \le b_k, \quad y_k = \overline{y}_k, \quad v_{x,k}, v_{y,k} \in \mathbb{R}, \\ a_k &= -25 + \alpha_k, \quad \alpha_k \sim \phi, \\ b_k &= 25 + \beta_k, \quad \beta_k \sim \phi, \end{aligned}$$

We assume ϕ to be Gaussian distributed with zero mean and a standard deviation of $\sigma = 20$.

This gives us enough information to compute Φ_k in this instance:

$$\begin{aligned} \Phi_k(\mathbf{s}_k) &= \Phi_{\alpha,k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k)\left[1 - \Phi_{\beta,k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k)\right] \\ &= \Phi_{\alpha,k}(x_k + 25)\left[1 - \Phi_{\beta,k}(x_k - 25)\right], \end{aligned} \tag{5.5}$$

$$\begin{aligned} \Phi_{\alpha,k}(x_k + 25) &= \int_{-\infty}^{x_k+25} \frac{1}{20\sqrt{2\pi}} e^{-\frac{1}{800}\alpha_k^2} \mathrm{d}\alpha_k, \\ \Phi_{\beta,k}(x_k - 25) &= \int_{-\infty}^{x_k-25} \frac{1}{20\sqrt{2\pi}} e^{-\frac{1}{800}\beta_k^2} \mathrm{d}\beta_k. \end{aligned}$$

The imprecise hard constraints are incorporated into the particle filter through evaluating the function Φ_k for every particle. Let us return to previous equations to see what actually happens.

In chapter 4, we saw that the Bayesian filter with constraints consisted of two steps: the prediction step and the update step. We chose to incorporate the constraints in the update step and found formula (4.6), i.e.

$$p(\mathbf{s}_k|\mathbf{z}_{1:k},\mathcal{C}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{s}_k)p(\mathcal{C}_k|\mathbf{s}_k)p(\mathbf{s}_k|\mathbf{z}_{1:k},\mathcal{C}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1},\mathcal{C}_{1:k})p(\mathcal{C}_k|\mathcal{C}_{1:k-1})}.$$

We also found a way to express $p(\mathcal{C}_k|\mathbf{s}_k)$ in terms of the distribution function ϕ_k: $p(\mathcal{C}_k|\mathbf{s}_k) = \Phi_k(\mathbf{s}_k)$. This means that in practice, incorporating the imprecise hard constraints in the particle filter means evaluating the function Φ_k for every particle $\hat{\mathbf{s}}_k^{(i)}$ and multiplying it with the weights. In the case of the abstract canal, it is straightforward since Φ_k only depends on x_k. The result is the SIR particle filter with imprecise hard constraints as can be seen in algorithm 2.

Algorithm 2: SIR Particle Filter with imprecise hard constraints

```
Initialization:
Sample initial particles $\{\mathbf{s}_0^{(i)}\}_{i=1}^{N_p}$ from $p(\mathbf{s}_0)$;
Set the weights $w_0^{(i)}$ to $\frac{1}{N_p}$.
input  : $\{\mathbf{s}_{k-1}^{(i)}, w_{k-1}^{(i)}\}_{i=1}^{N_p}$ and a new measurement, $\mathbf{z}_k$.
output: $\{\mathbf{s}_k^{(i)}, w_k^{(i)}\}_{i=1}^{N_p}$.
1 - Prediction:
for $i = 1, \dots, N_p$ do
    Sample $\mathbf{v}_{k-1}^{(i)}$ from $p_{\mathbf{v}}(\mathbf{v}_{k-1})$;
    Generate a new particle : $\hat{\mathbf{s}}_k^{(i)} = f(\mathbf{s}_{k-1}^{(i)}, \mathbf{v}_{k-1}^{(i)})$.
end
2 - Update:
for $i = 1, \dots, N_p$ do
    Compute weights : $\tilde{w}_k^{(i)} = w_{k-1}^{(i)} p(\mathbf{z}_k|\hat{\mathbf{s}}_k^{(i)})\Phi_k(\hat{\mathbf{s}}_k^{(i)})$;
end
Normalize the weights : $w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{j=1}^{N_p} \tilde{w}_k^{(j)}}$.
3 - Resample:
Generate a new set of particles $\{\mathbf{s}_k^{(j)}\}_{j=1}^{N_p}$,
so that for any $j$, $P(\mathbf{s}_k^{(j)} = \hat{\mathbf{s}}_k^{(i)}) = w_k^{(i)}$;
Set the weights $w_k^{(i)}$ to $\frac{1}{N_p}$.
```

5.1.2 Performance measures

As mentioned before, we are interested in the performance of the particle filter under different circumstances. To be able to discuss the performance, we need performance measures, i.e. numbers which actually define the quality of the particle filter. We can acquire these measures by various means: by Monte Carlo simulations and by inspection of a single run.

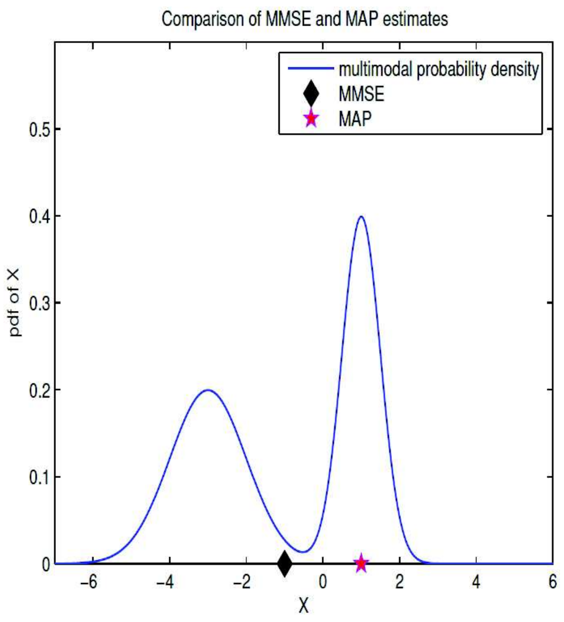


Figure 5.2: A bimodal Gaussian distribution. Source: [14]

A single run gives us an impression of the behaviour of the posterior density function for every time step. Although a single run is subject to randomness, we often gain fairly good insight. Also, point estimators can be extracted from a single run. A point estimator is a measure which assigns one value to the variable that has to be predicted. We distinguish between the mean estimator and the maximum a posteriori (MAP) estimator:

$$\begin{aligned}\hat{\mathbf{s}}_k^{mean} &:= \mathbb{E}[\mathbf{s}_k|\mathbf{Z}_k] = \int \mathbf{s}_k p(\mathbf{s}_k|\mathbf{Z}_k)\mathrm{d}\mathbf{s}_k;\\ \hat{\mathbf{s}}_k^{MAP} &:= \arg\max_{\mathbf{s}_k\in\mathbb{R}^{n_\mathbf{s}}} p(\mathbf{s}_k|\mathbf{Z}_k).\end{aligned}$$

For this research we choose to use the MAP estimator, because this estimator is better able to deal with nonlinearities [14]. Suppose we have a bimodal Gaussian distribution as in figure 5.2. The mean estimator is given by MMSE, the MAP estimator by MAP. The mean estimator gives little information about the true position of the target, while the MAP estimator gives you the top of the distribution and makes much more sense in this case.

A Monte Carlo simulation is a type of simulation that relies on repeated random sampling to compute results. Suppose we perform 100 runs and collect for every run and every time step in the simulation a state estimator. Then we can compute performance measures such as the mean bias and the standard deviation of the state estimates, i.e.

$$\begin{aligned}mean\ bias(\hat{\mathbf{s}}_k) &:= \mathbb{E}[\hat{\mathbf{s}}_k] - \mathbf{s}_k,\\ std(\hat{\mathbf{x}}_k) &:= \mathbb{E}[\hat{\mathbf{x}}_k^2] - (\mathbb{E}[\hat{\mathbf{x}}_k]^2),\\ std(\hat{\mathbf{y}}_k) &:= \mathbb{E}[\hat{\mathbf{y}}_k^2] - (\mathbb{E}[\hat{\mathbf{y}}_k]^2).\end{aligned}$$

The mean bias gives a measure for the deviation from the true state, whereas the variance gives a measure for the dispersion around the mean estimated state. We have to be very careful when using the mean bias of the state estimates. Againg, referring to figure 5.2: suppose that 70 MC runs place the state estimate around top on the right and 30 MC runs place the state estimate around the top on the left. Then the mean of the estimates is again somewhere in the middle. This means that we can only use the mean bias of the state estimates when we are dealing with a linear situation; a situation where the mean estimator coincides with the MAP estimator.

5.1.3 Trials and parameter settings

We perform two simulations to test the performance of various particle filters. The particle filters we compare are

- PF1: a particle filter in which the coastline position information is not taken into account;
- PF2: a particle filter in which the coastline position information is exploited as precise (perfect) hard constraints on the target dynamics;
- PF3: a particle filter in which the coastline position information is exploited as imprecise hard constraints on the target dynamics.

The coastline position information is given by the map as shown in figure 5.1.

In the first trial, we consider the map to be perfect, i.e. it reflects the true values with absolute precision and no bias. On the perfect map, we consider two separate trajectories. They are shown if figure A.1 and figure A.2. In both figures, again the green rectangles represent land areas and the blank parts of the map represent water. The red lines are the trajectories, in both cases the ship sails from south to north. The performance of PF1 and PF2 is compared. In the case of perfect knowledge PF2 and PF3 would be exactly the same, therefore we leave PF3 out of the comparison.

In the second trial, we consider the map to be imprecise. We assume that we exactly know the distribution of the coastline position. On the imprecise map we consider the trajectory as shown in figure A.3. The imprecise coastline position information is shown by the green rectangles. The actual coast is shown in blue, we see that there is a certain deviation from the true value. Again the trajectory of the target is shown in red and the target moves from south to north. At first the trajectory may seem contraintuitive since it moves over the green area, which denotes land. One should remember that the green rectangles only represent our knowledge about the area and not the area itself. The performance of all three filters, PF1, PF2 and PF3, is compared.

We expect PF3 to give a trade-off solution between PF1 and PF2. In the case of the simulations with perfect information, PF2 is expected to give the best results. When information is not perfect, PF3 is expected to perform better than PF2.

A number of assumptions is imposed onto the model:

- There is only one target present;
- The target, in this case a ship, can only be present at water.

The following parameter settings have been used:

- The radar position is $(0, 0)$;

- The number of particles is 6000;
- The update time is $T = 1s$;
- The maximum acceleration is $a_{\max} = 5m/s^2$;
- The standard deviation on the measurement noise is $\sigma_r = 10m$, $\sigma_b = 1$mrad and $\sigma_d = 3m/s$, for range, bearing and Doppler respectively.

We have developed a MATLAB application that is able to perform the simulations. This application is focused on incorporating external information in the form of geographical maps into the filter.

5.2 Simulation results

This section covers the results of the trials as described in section 5.1.3. Furthermore, the results of the trials are analysed. All results can be found in appendix A.

5.2.1 Results for trial A

The trajectories in figures A.1 and A.2 were used to compare filter PF1 and PF2. We have performed single runs as well as Monte Carlo simulations (100 in total) to produce performance measures. For the single runs we have looked at the posterior density function for all time steps. The posterior density function is in practice reflected by the particle cloud. Although we miss information about the weights of the particles, we still get a good impression of what the posterior density function looks like. The Monte Carlo simulations provided us with the mean bias and the standard deviation of the point estimates. For this scenario, we have only looked at the performance measures over horizontal position x_k. Recall that we mentioned in section 5.1.2 that we have to be very careful when looking at the mean bias of point estimates. We could only do this when there were no nonlinearities involved. For the abstract canal, this is the case so we do not have to worry using these performance measures.

The results of the Monte Carlo simulations for both trajectories are presented in figures A.4 - A.7. Both trajectories were completed in exactly 100 time steps. Every time step a measurement becomes available. The red and blue lines represent the measures for PF1 and PF2 respectively.

Let us first take a look at the simulation results for the trajectory through the middle of the canal. The results of the Monte Carlo simulations are shown in figures A.4 and A.5. It can be seen that both filters perform similarly. This is the result of the constraints being 25 metres apart from the target trajectory, i.e. the influence of the constraints is minimal. This feature is clearly displayed when one takes a look at the particle clouds, see figures A.8 and A.9. Although the particle cloud is reduced in the hard constraints case, the reduction has no measurable effect. This is caused by the outer particles having negligible weights, complemented by the vast majority of the particles being located near the middle of the canal.

Secondly we discuss the results for the trajectory along the coast. The results of this simulation are depicted in figures A.6 and A.7. It appears that PF2 performs worse when one looks at the mean bias, which is higher thoughout the whole simulation. However, this is compensated by the standard deviation of the estimates which is lower. This phenomenon exists due to the fact that the weights of all particles located on land are set to 0, which results in a bias which is always located in the direction of the centre of the canal. Also for this simulation, a plot of particle clouds in a simulation is depicted, see figures A.10 and A.11.

5.2.2 Results for trial B

For the second trial, we compare particle filters PF1, PF2 and PF3. We compare their performance on the scenario depicted in figure A.3. In this scenario, we let the ship sail from south to north 1 meter from the actual coastline. It takes the ship 101 time steps to complete the trial.

The results of the simulations are shown in figures A.12 and A.13. Both the mean bias and the standard deviation are taken over the horizontal position x_k. The red, blue and yellow lines represent the simulations results for PF1, PF2 and PF3 respectively. Let us first zoom in on the results related to PF1 and PF2. Using external information as though it were perfect in the case it is actually imprecise can be harmful to the filter, as can be seen from the results. The trajectory is situated on land during time steps 1 to 45 and 87 to 101. During these time steps, the bias of the state estimates are in the case of PF2 larger than during the other time steps. This is a results of all weights of the particles on land reduced to 0. Figures A.14, A.15, A.16 and A.17 illustrate the situation. Note that the true state lies within the particle cloud of PF2 during time step 61, a time step in which the target is located in water according to the map. During time step 101, this statement is not true. This is the danger of using a filter with hard constraints in a scenario where information about constraints is imprecise. The filter throws away valuable information, resulting in a large state bias. This happens for instance in time step 101.

Let us now focus on comparing filters PF2 and PF3. When we look at figure A.12, we see that PF3 performs alternately better and worse than PF2. The simulation can be subdivided into four phases:

- time steps 1 - 10: PF3 performs better than PF2;
- time steps 11 - 42: PF2 and PF3 show similar performance;
- time steps 43 - 91: PF3 performs worse than PF2;
- time steps 92 - 101: PF3 performs better than PF2.

During time steps 1 - 10 and 92 - 101, the horizontal position of the target is more than 8 metres on land, according to the map. This indicates that when the target is far enough on land, the filter with imprecise hard constraints outperforms the conventional one. When the horizontal position of the target is less than 8 metres inland, the filter with imprecise hard constraints performs similar to or worse than the filter with hard constraints (see figures A.18 and A.19). This is a strange effect that we did not expect. It is caused by the fact that we are looking at a point estimator and not the complete posterior distribution. For more information on this topic, see section 5.3.

Compared to PF2, PF3 shows a more or less constant standard deviation (figure A.13). And, maybe more important, the standard deviation is found to be smaller than the standard deviation of PF1.

5.3 Influence of Φ

In section 5.2.2, we found that the filter incorporated with imprecise hard constraints sometimes seemed to perform worse than the filter incorporated with hard constraints. This seemingly strange

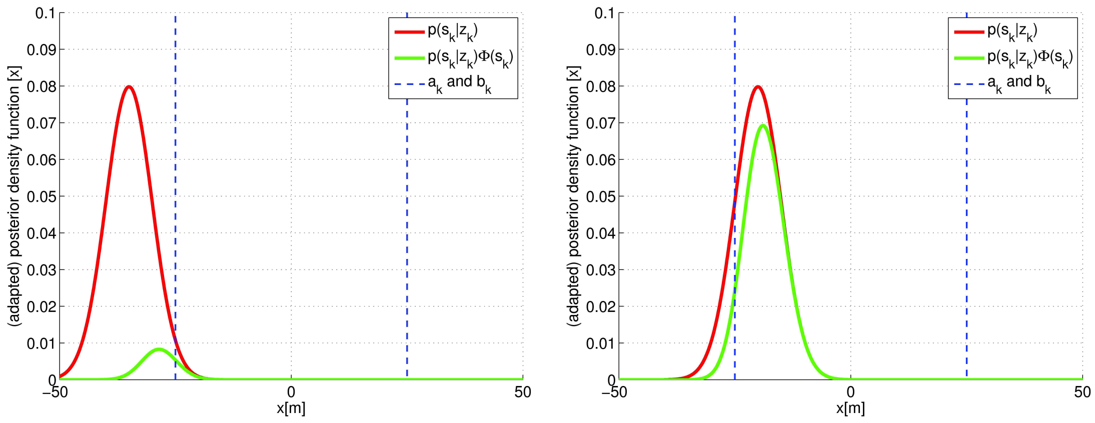


Figure 5.3: (Adapted) likelihood function for imprecise hard constraints: $\mu_T = -35$ (left) and $\mu_T = -20$ (right)

phenomenon can be explained. To incorporate external information into the particle filter, the likelihood function $p(\mathbf{z}_k|\mathbf{s}_k)$ is multiplied with the factorization function $\Phi_k(\mathbf{s}_k)$. Ultimately this has its effect on the posterior density function $\tilde{p}(\mathbf{s}_k|\mathbf{z}_k)$.

Now, suppose the posterior density function right before incorporating constraints has the form $p(\mathbf{s}_k|\mathbf{z}_k) \sim \mathcal{N}(\mu_T, \sigma_T^2)$, with μ_T the mean position of the target and σ_T its standard deviation, which we pick as 5. Figures 5.3 and 5.4 depict the situations where $\mu_T = -35$, -26 and -20. The red graph indicates the posterior density function right before incorporating the external information. The posterior density function including the external information (apart from a normalization factor) is indicated by the green graph. In all figures the canal is indicated by the blue lines.

If we deal with the MAP estimator, the point estimate that is chosen is the top of the graph. In the case of imprecise hard constraints, we see that this top is shifted towards the middle of the canal compared to the case of the filter without constraints. For targets far off-coast (figure 5.3, left), this gives us a nice trade-off solution with the state estimator halfway the coast and the target. But when targets are close to the coastline (figure 5.4, left), the point estimator performs worse than in the case of hard constraints (see figure 5.4, right). In the case that the target is located close the coast ($\mu_T = -26$), the MAP estimator is given as

$$\begin{aligned} \hat{\mathbf{x}}_k^{MAP} &= -25, \quad \text{for hard constraints,} \\ \hat{\mathbf{x}}_k^{MAP} &= -23.6, \quad \text{for imprecise constraints.} \end{aligned}$$

This means that according to the MAP estimator, the filter incorporated with imprecise hard constraints performs worse than the filter incorporated with hard constraints. This is an unwanted situation. However, if we look at the complete posterior distribution, we see that still a lot of information is preserved in the case of imprecise constraints. While in the case of hard constraints all information below the coastline is destroyed. So we can say that in general, when you take into account the complete posterior distribution, PF3 performs better than PF2.

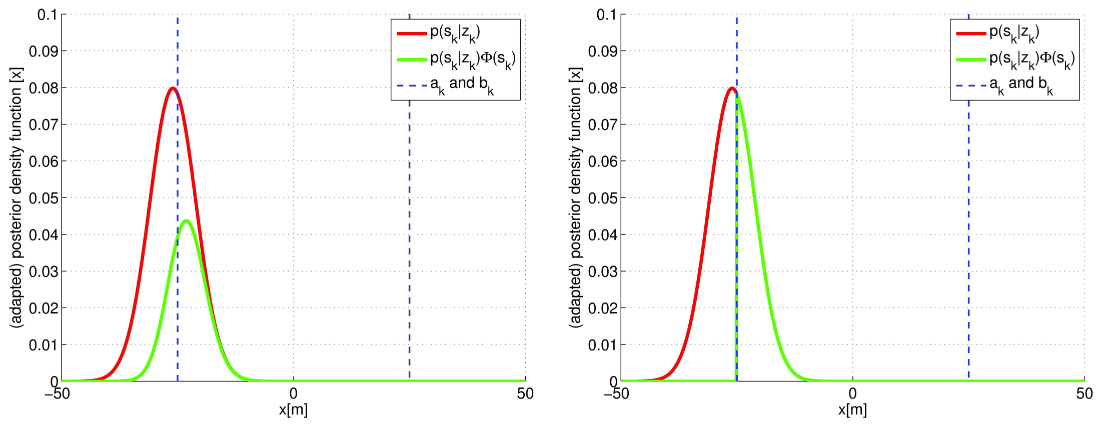


Figure 5.4: (Adapted) likelihood function for $\mu_T = -26$ for imprecise hard constraints (left) and hard constraints (right)

Chapter 6

Simulations with a realistic scenario

In chapter 5, simulations were performed on a theoretical scenario. We chose the scenario in such a way that it has nice properties: we could directly compute values of Φ_k and only had x_k as our variable of interest. We showed that the particle filter incorporated with imprecise hard constraints was succesfull in improving performance, when one considers the complete distribution. This arouses our interest in more realistic scenarios: what would happen if we performed the simulations with external information in the form of realistic maps, for example the map in figure 6.1?

In this chapter we develop methods for dealing with realistic imprecise digital charts. The main problem we face when we deal with realistic digital charts is computing the value of Φ_k for every particle. We address two methods which are able to deal with this problem in their own way: the *shortest distance method* and the *grid method*. In the remaining part of the chapter we perform simulations with both methods. In these simulations, we compare the performance of the shortest distance method with the grid method. Also, we compare the performance of the filter with imprecise hard constraints with the performance of the filter with and without hard constraints, just like we did in chapter 5.

When we want to use imprecise knowledge in the form of nonlinear maps in a tracking filter, we need additional theory to be able to compute the value of Φ_k for every particle. Recall that the formula for Φ_k was (see equation (4.7))

$$\Phi_k(\mathbf{s}_k) = \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k \left[1 - \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{b}}_k} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k\right].$$

When we want to determine the value of $\Phi_k(\hat{\mathbf{s}}_k^{(i)})$, we need the values for $\mathbf{C}_k(\hat{\mathbf{s}}_k^{(i)})-\overline{\mathbf{a}}_k$ and $\mathbf{C}_k(\hat{\mathbf{s}}_k^{(i)})-\overline{\mathbf{b}}_k$. The problem is that we are in general not familiar with the values for $\mathbf{C}_k$, $\overline{\mathbf{a}}_k$ and $\overline{\mathbf{b}}_k$. To be able to use imprecise knowledge in a realistic situation, we need to tackle this problem somehow. We developed two methods that are able to do this approximately: the *shortest distance method* and the *grid method*.

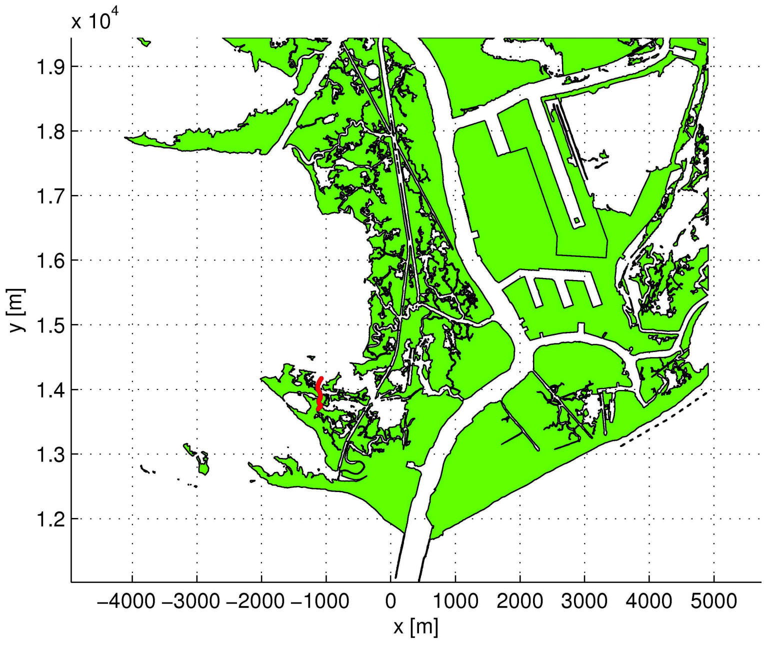


Figure 6.1: Simulation scenario: Port Fourchon and approaches in the Mississippi River Delta near New Orleans (ENC cell US5LA26M).

6.1 Shortest distance method

For the shortest distance method, we use the exact same particle filter as for the abstract canal (algorithm 2). The difficulty lies in computing $\Phi(\hat{\mathbf{s}}_k^{(i)})$ for every particle i, $i = 1 \dots N$. In the case of the abstract canal we could just input the variable x_k into formula (5.5) and extract this value. For the more realistic case of the map of New Orleans, this is not as easy. The shape of Φ_k is much more complex, by which it hampers these computations.

The main problem is that in fact, every piece of coastline contributes to the value of the function Φ_k. This means that we have an infinite amount of variables we have to take into account. Since this is impossible to implement we have to look for solutions. One of these solutions is the shortest distance method. The shortest distance method does not take into account all coastlines, but the point of all coastlines closest to the particle, i.e. the point with the shortest distance. It is an online method, so during the filtering the values of Φ_k have to be computed.

6.1.1 Assumptions

A number of assumptions are at the basis of the shortest distance method:

1. for the complete map and for all time steps, we have the same standard deviation σ;
2. the smallest shown waterways have size σ;
3. the influence of the closest coast is large enough to ignore the influence of the other coasts;
4. concave polygons are treated as though they are convex.

The assumptions will be illustrated below.

The same standard deviation for the whole map

If we only measure the shortest distance to the coast in order to find the value of $\Phi(\hat{\mathbf{s}}_k^{(i)})$, we implicitly assume that the distribution of every coast has the same standard deviation. If this was not the case, the shortest distance to a coast need not necessarily imply the biggest influence on the function Φ_k. In figure 6.2, a situation with two coasts containing different standard deviations is depicted. The coast on the left has a standard deviation of 10, while the coast on the right has a standard deviation of 100. This results in a skew function Φ_k, shown in green. If we try to mimick this Φ_k with the shortest distance method, the result is the function as shown in red. Clearly, this method is not sufficient for coasts with unequal standard deviations. But since in practice complete maps are built with the same standard deviations, we can safely assume that a complete map has the same standard deviation.

The smallest waterways equal σ

Geographical map are built with a certain imprecision. This leads to same standard deviation for the complete map, as mentioned in the previous paragraph. As a side effect, imprecision has its influence on the smallest waterways that can be printed on the map. The smallest waterways appear to be as small as the imprecision, or σ.

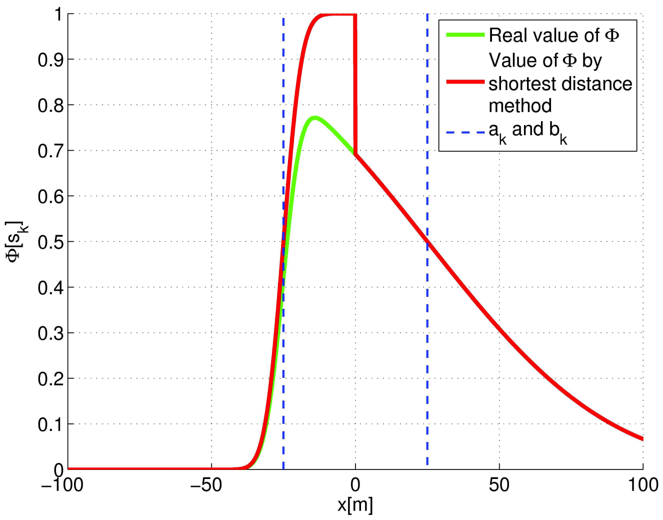


Figure 6.2: The function Φ for $\sigma_\alpha \neq \sigma_\beta$

The influence of the closest coast

In the shortest distance method, only the point on the coast closest to the particle is considered. This results in a slightly modified function Φ_k. In figure 6.3, the worst case scenario is sketched: a waterway with its width equal to the standard deviation. The blue dotted lines indicate the position of the coasts, whereas the green and red lines represent respectively the real function Φ_k and the function Φ_k constructed with the aid of the shortest distance method. We can see that the shape of the function Φ_k is more peaky, which results in more particles situated towards the middle of the waterway. For wider waterways, this effect is less dramatic and the function Φ_k constructed with the aid of the shortest distance method looks more like the real function Φ_k.

Concave polygons

The digital maps that are used in practice consist of polygons, see 2.1. In figure 6.4, a situation with a concave polygon is depicted. The shaded area represents the polygon, and the dotted lines represent the standard deviation. The only problem area, where the influence of the adjoining coast gets too large, is area A. In these areas, the approximation of Φ is somewhat rough, but we take this drawback for granted.

6.1.2 The method in detail

As mentioned in section 2.1, the maps described by ENCs are constructed out of polygons. Computing the shortest distance from a point to land is a matter of minimizing the distances to all polygons, which is done by minimizing the distance to every line in a polygon. Suppose that we want to know the value of Φ_k for particle $\hat{\mathbf{s}}_k^{(i)}$. First we compute the shortest distance d from $\hat{\mathbf{s}}_k^{(i)}$ to all polygons. Then we determine whether $\hat{\mathbf{s}}_k^{(i)}$ is inside a polygon or outside all polygons. If $\hat{\mathbf{s}}_k^{(i)}$ is

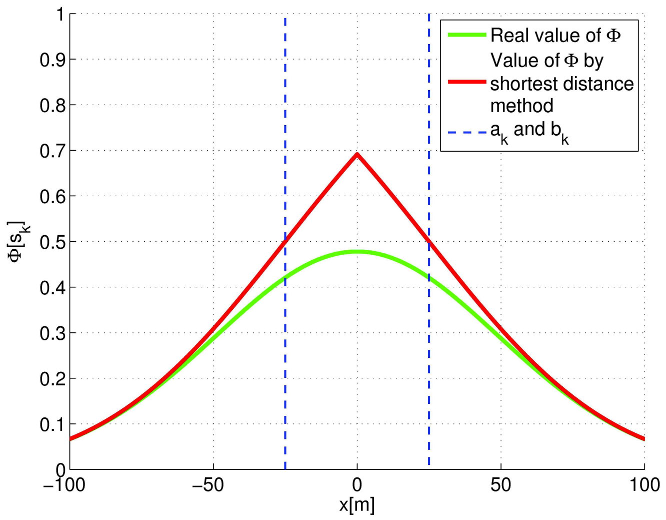


Figure 6.3: The difference between the smallest distance method and the real function Φ_k: worst case scenario

inside a polygon we set $D = -d$, otherwise $D = d$. We approximate the value of $\Phi(\hat{\mathbf{s}}_k^{(i)})$ as follows:

$$\Phi(\hat{\mathbf{s}}_k^{(i)}) \approx \int_{-\infty}^{D} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k.$$

This method is precise, but very slow as we will see in section 6.4.1.

6.2 Grid method

Another method for implementation of imprecise information into a particle filter for realistic maps is the grid method. This method relies on off-line computation of a grid, which is later used to easily access information about the value of Φ_k for a certain particle. The geographical map is divided into small cells which make up the grid. Each cell is then assigned a value of the function Φ_k. The grid is a discrete approximation of Φ_k.

First a $(0, 1)-$grid is constructed: a grid is put on top of the geographical map and for every central point $(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)$ in a cell it is checked whether the point is inside or outside a polygon. Inside a polygon generates a 0, outside all polygons generates a 1: we acquire a map with ones at sea and zeros at land. An example of such a map is shown in figures 6.5 and 6.6.

After the $(0, 1)-$grid is assembled, we perform a discrete convolution with a normal distribution. We use a two dimensional Gaussian with two independent variables x and y,

$$\phi(x, y) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\begin{bmatrix}x & y\end{bmatrix}\Sigma^{-1}\begin{bmatrix}x\\ y\end{bmatrix}\right),$$

with

$$\Sigma = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix}.$$

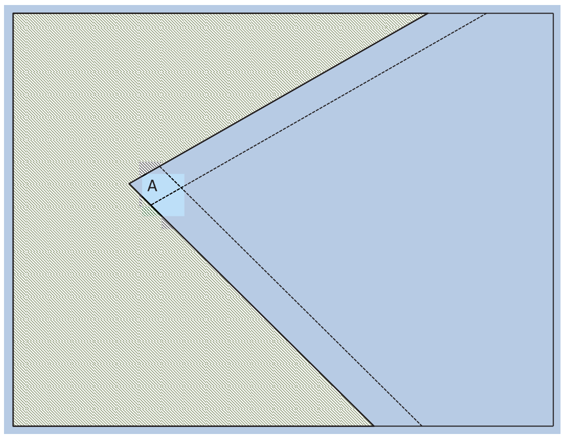


Figure 6.4: The drawback of sharp angles in coasts

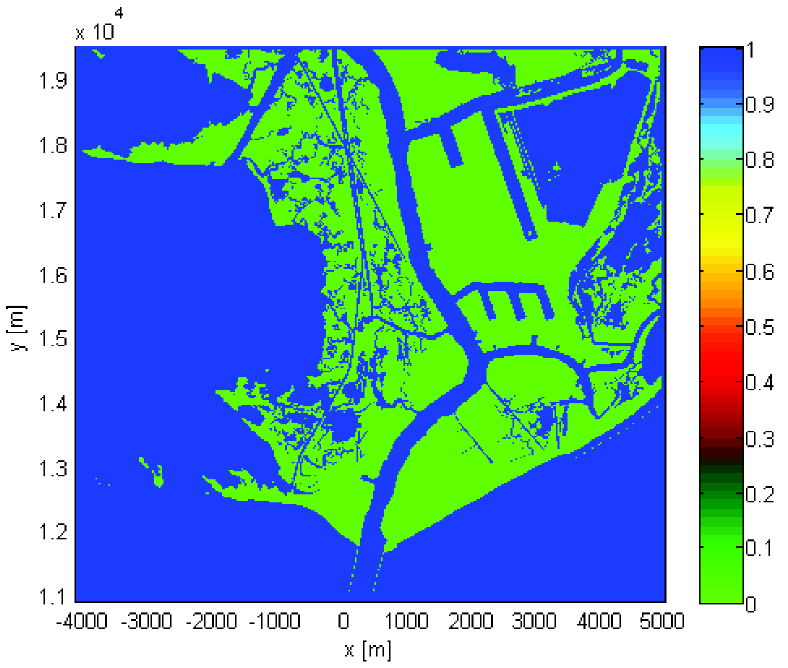


Figure 6.5: $(0, 1)$ grid of New Orleans

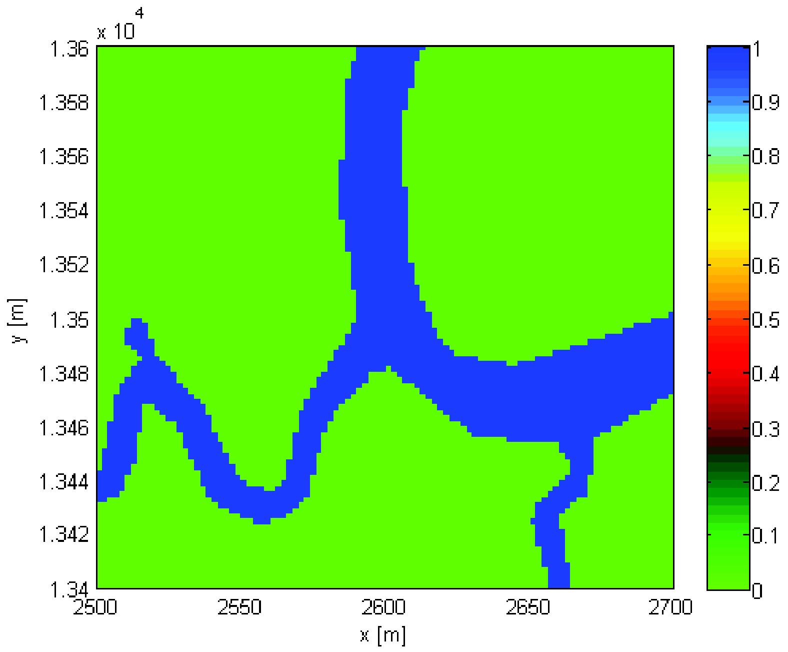


Figure 6.6: $(0, 1)$ grid of New Orleans zoomed in

If we name the $(0, 1)-$grid G, the convolution is performed as

$$(G * \phi)(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x-u, y-v)\phi(u, v)\mathrm{d}u\mathrm{d}v.$$

This is a continuous convolution. We use a discrete convolution, since it simplifies calculations. The discrete convolution can be written as

$$(G * \phi)(x, y) = \sum_{v=-\infty}^{\infty}\sum_{u=-\infty}^{\infty} G(x-u, y-v)\phi(u, v).$$

An example of a $(0, 1)-$grid after convolution with a normal distribution is shown in figures 6.7 and 6.8.

To verify whether a convolution of the $(0, 1)-$grid with a normal distribution gives indeed the desired function Φ we perform a small check. In figure 6.9, the convolution between a one dimensional $(0, 1)-$grid and a normal distribution with $\sigma = 5$ is depicted (the red graph). At the same time, the real values of Φ are plotted in blue. The difference between the two functions is shown in green in the other figure. We can see that the difference between the two is relatively small, with peaks of 0.04.

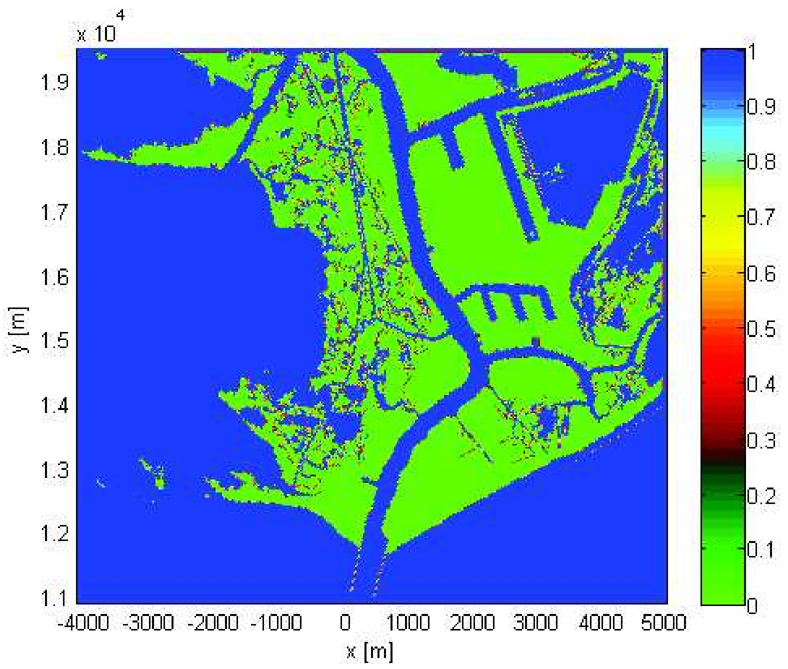


Figure 6.7: Φ grid of New Orleans

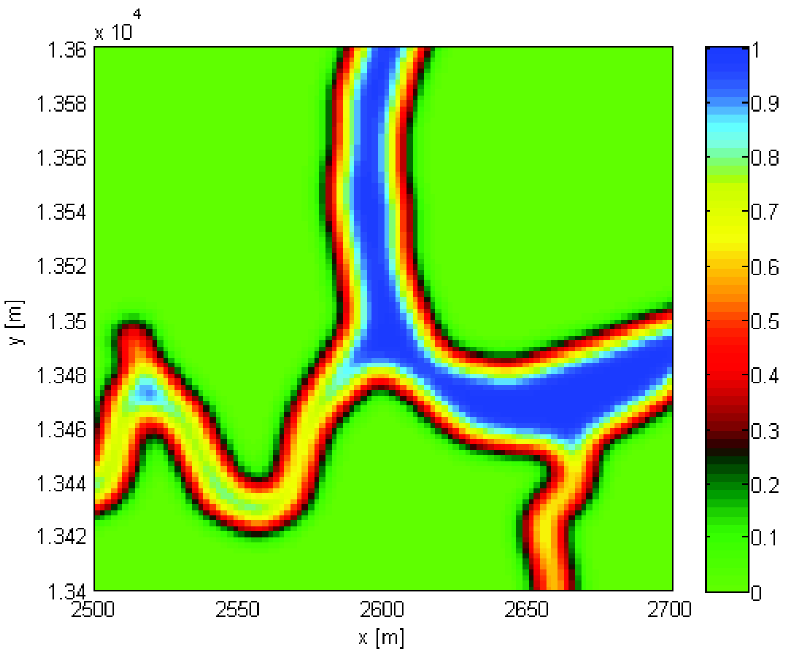


Figure 6.8: Φ grid of New Orleans zoomed in

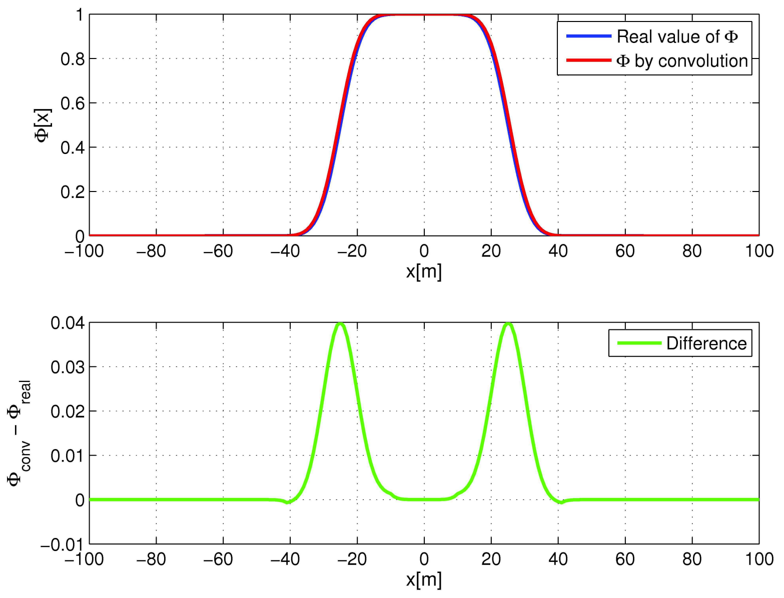


Figure 6.9: Deviation of the convolution method

6.3 Simulation setup

In sections 6.1 and 6.2, two methods for evaluating $\Phi_k(\hat{\mathbf{s}}_k^{(i)})$ in a realistic scenario were presented. We compare both methods and measure their performance. Furthermore, we want to measure the influence of imprecise hard constraints in a coastal environment.

6.3.1 Simulation scenario

We consider the case of the area as depicted by figure 6.1. The map depicts a coastal area near New Orleans. Again, the green areas denote land whereas the blank areas denote water. In this area we simulate the movement of a ship and again a tracking radar receives measurements of the ship and converts them to tracks with the aid of a particle filter.

The dynamical equations are unchanged from chapter 5, see equations (5.1, 5.2, 5.3, 5.4). The state is constrained by the general equations as given in section 4.2. This makes the filtering algorithm unchanged from the one presented in chapter 5, i.e. algorithm 2.

6.3.2 Trials and parameter settings

We perform three simulations to test the quality of the various particle filters. The particle filters we compare are:

- PF1: a particle filter in which the coastline position information is not taken into account;
- PF2: a particle filter in which the coastline position information is exploited as precise (perfect) hard constraints on the target dynamics;

- PF3: a particle filter in which the coastline position information is exploited as imprecise hard constraints on the target dynamics, the shortest distance method is used to compute values of Φ_k;
- PF4: a particle filter in which the coastline position information is exploited as imprecise hard constraints on the target dynamics, the grid method is used to compute values of Φ_k.

The coastline position information is given by the map as shown in figure 6.1. We use the performance measures as described in section 5.1.2.

In the first trial we assume that the coastline position information is imprecise. We simulate the trajectory as seen in figure B.2 and compare the performance of PF3 and PF4. The target moves from north to south. We do not perform Monte Carlo simulations, since the shortest distance method is too slow.

In the second trial we assume that the coastline position information is perfect. We simulate the trajectory as seen in figure B.1 and compare the performance of PF1 and PF2. The target moves from north to south

In the third trial we assume that the coastline position information is imprecise againg. Just as in the first trial, we simulate the trajectory as seen in figure B.2 and compare the performance of PF1, PF2 and PF4. The target moves from north to south. For reasons of computational time, PF3 is left out of the comparison.

The same MATLAB application is used, supplemented with the shortest distance method for PF3 and the grid method for PF4. Apart from the assumptions already discussed in this chapter, we impose a number of additional assumptions:

- There is only one target present;
- The target, in this case a ship, can only be present at water.

The following parameter settings have been used:

- The radar position is $(0, 0)$;
- The number of particles if 6000;
- The update time is $T = 1s$;
- The maximum acceleration is $a_{\max} = 5m/s^2$;
- The standard deviation on the measurement noise is $\sigma_r = 10m$, $\sigma_b = 1\text{mrad}$ and $\sigma_d = 3m/s$ for range, bearing and Doppler respectively.

6.4 Simulation results

In this section, simulation results are presented and analysed for the simulation scenarios mentioned in section 6.3.

6.4.1 Trial A

For this trial we compared a single run of PF3 and PF4. We only compare a single run, because a full Monte Carlo simulation of 100 runs would take weeks for the shortest distance method to complete the simulation. To be able to do a valid comparison, we use the same measurements for both filters. The results of the simulations are plotted in figures B.3 and B.4. We can see that although the methods produce different results, the results are comparable. For example, the big deviations on time step 11 and 54 are present in both filters. Time step 36 shows a relatively big difference between the two methods when one considers the bias on the horizontal position. Even for this time step the particle clouds behave similar, see figures B.5 and B.6. We can conclude that both methods act differently, but produce similar results in our case.

6.4.2 Trial B

For the second trial, we perform simulations as though we were using a perfect map. This is for comparing the performance of PF1 and PF2. In particular, we let the target move according to the trajectory as shown in figure B.1. The target follows the trajectory from north to south. For one particular run, we measure the posterior density function as reflected by the particle cloud. We measure the mean bias and the standard deviation of the estimates over 100 Monte Carlo runs. In chapter 5, we explained that using a mean bias can be very dangerous in case of a nonlinear system, see figure 5.2. Unfortunately, the coastline position information produces highly nonlinear constraints, which makes the use of the mean bias for measuring performance cumbersome.

Figures B.7, B.8, B.9 and B.10 show the Monte Carlo results for the second trial. In these figures, the mean bias and standard deviation are plotted for x_k and y_k. The results for PF1 are shown in red, while the results for PF2 is shown in blue. If we take a look at the results for the filter with no constraints we see, after a few initialisation steps, a constant standard deviation. Furthermore, we see that the bias fluctuates, these fluctuations occur whenever the target moves along a curve.

The result for the particle filter with hard constraints show a constant standard deviation as well when the target is located on broad waterways. This was to be expected, since most constraints are virtually inactive. The standard deviation of this filter in the direction of x_k shows a remarkable drop when waterways become smaller. This indicates that constraints become active and the posterior density function becomes peakier in those areas. The results for the filter with hard constraints show an increase in mean bias when the target moves along a coast. When the coast is situated on the left or right, the standard deviation of x_k increases or decreases respectively. And when the coast is situated above or below the target, the standard deviation of y_k decreases or increases respectively. These results comply to earlier results in the abstract case. Particle clouds for a single run are plotted in figures B.11, B.12, B.13 and B.14. We see that in these figures, the particle cloud for PF1 is a nicely shaped Gaussian both in time step 15 and time step 102. The particle cloud for PF2 is shaped according to the coastline in time step 102. In time step 15, the particle cloud of PF2 is not changed compared to the particle cloud of PF1: this is due to the fact that the constraints are barely active.

6.4.3 Trial C

For the third trial we assume that the map in figure 6.1 is imprecise. Like in the case of the abstract canal in chapter 5, the target trajectory moves over areas that are designated as land by our map.

However, in reality these areas are water and the target is able to move there. See figure B.2 for reference, again the target moves from north to south.

Figures B.15, B.16, B.17 and B.18 show the results for the second trial. In these figures, the mean bias and standard deviation are plotted for x_k and y_k. The results for PF1 are shown in red, the results for PF2 is shown in blue and the PF4 is shown in yellow. We do not use PF3 in this scenario because it is too slow. Moreover, we have already shown that it performs similar to PF4.

First of all, we notice that the results for PF1 are not any different than in the first trial. This is not very surprising, since the target follows roughly the same trajectory. The most important part of this trial is the comparison of the results of PF2 and PF4. When the target is on open sea, we see no significant differences between the two filters, see the particle clouds in figures B.19 and B.20. We do see differences when the target moves over area that is shown as land on our map. There are instances when the bias of PF2 larger and instances when the bias of PF4 is larger, both in x_k and y_k direction. Although we cannot draw conclusions from this phenomenon, we can look for the particle clouds (the results of one run) at those particular time steps. First, we take a look at a situation where the bias of the PF4 is larger ($t = 55$): figures B.21 and B.22. We see that the target trajectory is still covered by PF4 but not by PF2. This indicates a better performance of PF4. In the situation where the bias of PF4 is smaller than the bias of PF2 ($t = 73$), we are looking at a target located about 3σ inland. See figures B.23 and B.24. Again the trajectory is covered by PF4, but not by PF2.

Chapter 7

Optimization

As we have seen in section 6.4.1, the two methods developed to deal with computing the value of $\Phi_k(\hat{\mathbf{s}}_k^{(i)})$ perform similar. However, both the shortest distance method and the grid method have their drawbacks. Due to these drawbacks the methods cannot be applied in practice, not as accurate as we want them to perform that is.

In this chapter the drawbacks of the shortest distance method and the grid method are reviewed. We extend the shortest distance method and try to make it perform faster. The first steps are made towards improving the performance of this method.

7.1 Drawbacks of the methods

As we saw in section 6.4.3, the grid method is fast enough. After an initialisation step where the grid is made, the method ensures that the factorization function Φ_k can be approximately computed fast enough. In section 6.2, we explained that the the grid method is an approximation in the sense that the state space is discretized. We divide the state space in equal cells and assign values of Φ_k to every cell. When we want to make the approximation better, our only option is to make the cells smaller, hence more accurate. This directly invokes another problem: a larger number of cells means storing a larger matrix in the computer memory. Consider for instance the map we used in the simulations in chapter 6: this map covers an area of $90,000,000\ m^2$. If we want to cover this map with cells of size 1 by 1 metres (remember, this is only 0.2σ in our case), we need to store a $10,000 \times 9,000$-dimensional matrix. For larger maps, this problem grows rapidly and makes the method unusable in its current form. So for this method, accuracy is directly related to the memory capacity of the computer.

The shortest distance method, on the other hand, is (given the assumptions in section 6.1.1) not an approximation. Also, the method does not require a large memory. The only problem with the shortest distance method lies in its speed. For example, computing the simulation results for a single run, as shown in figures B.3 and B.4, required a computional time of about 6 hours. This is due to the fact that the shortest distance from every particle to every polygon has to be computed for every time step. Polygons range from having 3 edges to over 2000 edges in the example map in figure 6.1. So one can imagine that this process can take a lot of time.

We look for a method that is, in contradiction to the current two methods, performable in practice. In fact, we want to combine the speed of the grid method with the precision and low memory cost of the shortest distance method. It seems impossible to reduce the memory cost of the grid method, since this method always makes use of storing a grid. The shortest distance method seems a better starting point for optimization. It appears to be possible to reduce computational time when smaller areas are taken into account. The exact method is explained in the next section.

7.2 Optimization of the shortest distance method

Like we explained in the previous section, the problem with the shortest distance method is that we have to compute the shortest distance from every particle to every polygon, for every time step. The particle filter requires us to keep the number of particles high enough, so we assume that we cannot change anything here. This means that we have to focus our attention to the number of polygons.

When a target violates the constraints, we say that the corresponding polygon is active. The probability of a polygon, with a large shortest distance to the particle, of being active is commonly very low. Consider for instance a map with imprecision, being distributed according to a Gaussian with a standard deviation of σ. Then every polygon with a shortest distance of greater than 3σ has a probability of at most 0.1% of being active.

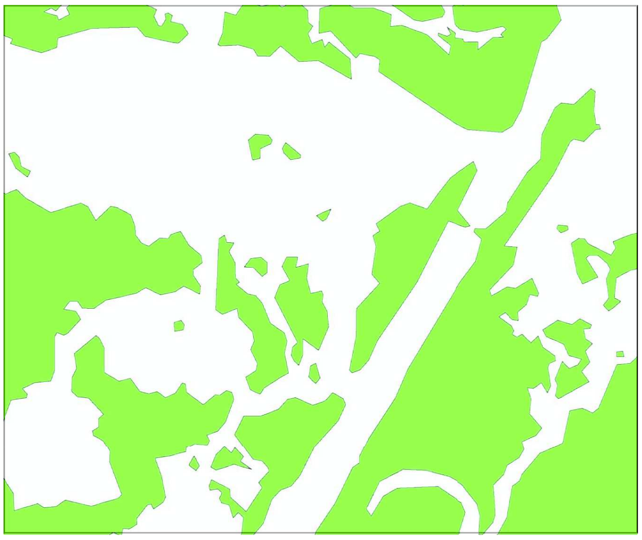


Figure 7.1: Map without partitioning

Consider the map given in figure 7.1. Inspired by the grid method, we make a partition of the map in large cells. This way, we only have to consider the direct surroundings of the particle and compute only the shortest distance to the nearest polygons. An example of a partition is depicted in figure 7.2. The problem with this partition is that particles at the edge of the cell lose all information of adjacent cells. This can be problematic since in theory there might be nearby

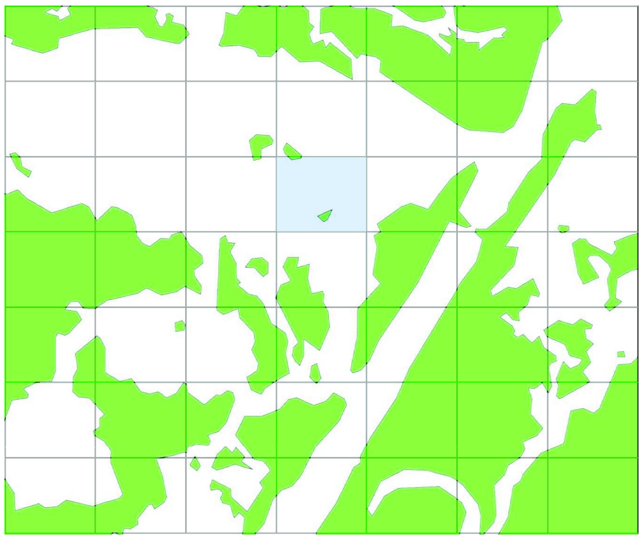


Figure 7.2: Map with simple partitioning

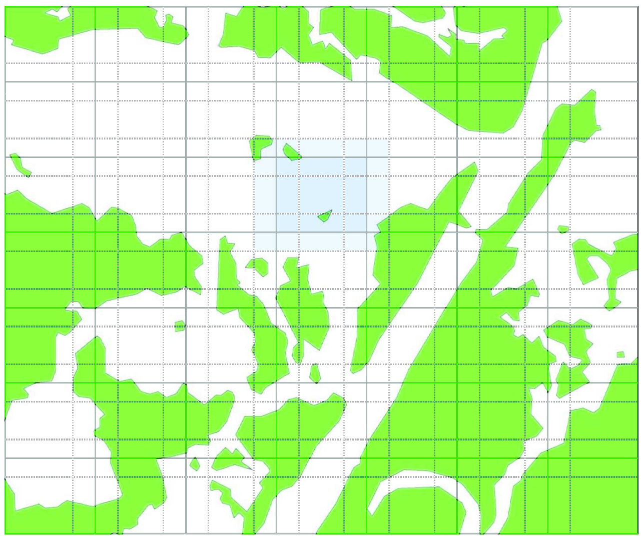


Figure 7.3: Map with intelligent partitioning

polygons which we do not consider in our computations, e.g. the polygons to the northwest or southeast of the shaded area in figure 7.2. A better way of making a partition of the map is by giving each cell a 'boundary area', we call this method *intelligent partitioning*. An example of this is shown in figure 7.3, the boundary area is shown by the dotted line, the inner cell by the solid line. If we make the width of the boundary area 3σ we have a probability of less than 0.1% that particles on the edge of the inner cell violate constraints which we do not include in our computations. For particles further away from the edge of the inner cell, this probability is even smaller. The polygons that were not considered in the case of simple partitioning (polygons to the northwest or southeast of the shaded area in figure 7.2), are taken into account with intelligent partitioning, see figure 7.3.

In practice, the optimization method described in this section is applied as follows. First we make a partition of the map into cells. The amount of cells has some influence on the performance of the filter: a small amount of large cells creates a slow filter and a limited use of memory, a large amount of small cells does the opposite. For every cell (including the boundary area), all active polygons are stored. While performing the tracking, a check is performed to see in what cell the particle lives. Only for the polygons which are active in that cell, the shortest distance is computed.

It appears that the computational time is greatly reduced. Remember that without using intelligent partitioning, the computational time for the shortest distance method was large. When we include intelligent partitioning in the tracking filter, the computional time is reduced considerably, with an order of magnitude of 10. This specific computational time was acquired with a partitioning into 5×5 cells. More research has to be done to find an optimal partitioning and conditions under which it is optimal. Another decrease in computational time could be gained through cutting the active polygons along the cell boundary. This way, the total amount of edges taken into account when computing the smallest distance is reduced. Additional research has to be carried out to show the influence of this technique.

Chapter 8

Conclusion and discussion

In the introduction of this thesis we wrote in the problem description the following: We are familiar with models derived to deal with perfect knowledge in target tracking. However, in practice the knowledge we can use to improve target tracking is imperfect. This arises the question how to deal with imperfect knowledge in target tracking. Imperfections, in general, are a very broad class of errors applied to data. One of the research goals of this thesis is to categorize imperfections and make models for the different categories. Furthermore, we are interested in using these models to derive a way of dealing with imperfect knowledge in target tracking. Finally, we need practical algorithms to deal with imperfect knowledge in realistic situations.

8.1 Summary of the results

In chapter 2, we reviewed the subject of imperfect knowledge. Measurements appeared to be subject to imperfections according to

$$y = \mu + \Delta + e.$$

The systematic error (Δ) and the random error (e), were the cause of bias and imprecision, respectively. The systematic error and random error together were called the accuracy of the measurement. Furthermore, we distinguished the imperfections known as incompleteness and inconsistency.

Dealing with imprecise knowledge was considered in chapter 4. We modeled imprecise knowledge as constraints, and did this as follows,

$$\begin{aligned}
&\mathbf{a}_k \leq \mathbf{C}_k(\mathbf{s}_k) \leq \mathbf{b}_k \\
&\mathbf{a}_k = \overline{\mathbf{a}}_k + \boldsymbol{\alpha}_k, \quad \boldsymbol{\alpha}_k \sim \phi_k \\
&\mathbf{b}_k = \overline{\mathbf{b}}_k + \boldsymbol{\beta}_k, \quad \boldsymbol{\beta}_k \sim \phi_k \\
&\phi_k = \phi_k(\mathbf{s}_k).
\end{aligned}$$

The constraints $\mathbf{a}_k$ and $\mathbf{b}_k$ were distributed according to distribution function ϕ_k. Constraints were incorporated into a Bayesian filter by multiplying the likelihood function by the factorization function Φ_k. The factorization function was computed as

$$\begin{aligned}\Phi_k(\mathbf{s}_k) &= \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{a}}_k} \phi_k(\boldsymbol{\alpha}_k)\mathrm{d}\boldsymbol{\alpha}_k \left[1 - \int_{-\infty}^{\mathbf{C}_k(\mathbf{s}_k)-\overline{\mathbf{b}}_k} \phi_k(\boldsymbol{\beta}_k)\mathrm{d}\boldsymbol{\beta}_k\right], \\ &= \Phi_{\boldsymbol{\alpha},k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{a}}_k)[1 - \Phi_{\boldsymbol{\beta},k}(\mathbf{C}_k(\mathbf{s}_k) - \overline{\mathbf{b}}_k)].\end{aligned}$$

The model for imprecise hard constraints applied to position is a little different from the general model. We set $\overline{\mathbf{a}}_k = \overline{\mathbf{a}}$, $\overline{\mathbf{b}}_k = \mathbf{b}_k$ and $\phi_k = \phi$.

In chapter 5, we performed simulations with the models developed in chapter 4. We showed that in the case of perfect information, the filter with hard constraints performed better than the filter without constraints. However, when information is not perfect (which is a more realistic assumption), this filter throws away important information. In this case the filter with imprecise hard constraints performs better. Also, we conluded that point estimators give a distorted view of the actual results.

Chapter 6 addresses dealing with imprecise knowledge in a realistic scenario. Two methods for computing $\Phi_k(\hat{\mathbf{s}}_k^{(i)})$ were developed: the shortest distance method and the grid method. The methods seemed to perform similarly. The shortest distance method is slow but precise and with a low memory cost. The grid method needs a lot of memory and approximates $\Phi_k(\hat{\mathbf{s}}_k^{(i)})$, at the same time the method is fast. From the results in chapter 6, we could draw the same conclusions: in the case of perfect knowledge the filter incorporated with hard constraints performs best, in case of imprecise knowledge the filter with imprecise hard constraints appeared to perform better.

Finally, in chapter 7 we made the first steps towards optimizing the shortest distance method. The speed of the method was increased using intelligent partitioning.

8.2 Discussion

In chapter 4, we proposed a model for imprecise hard constraints. This model has its limitations though. In principle its use is for constraints which change over time. We use the general model, which changes every time step, also for cases where the constraints are constant. We can make this way of using the general model reasonable. The model is valid if we track a target which is moving fast enough. Consider a ship moving along a coastline. If the ship moves fast enough parallel to the coastline, it will encounter a 'new' piece of coastline every time a measurement is done. If the new position of the ship $\mathbf{s}_k$ is such that the correlation between the point on the coastline closest to the ship and the point on the coastline closest to the previous position of the ship is negligible, then we can safely draw new values for $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ from ϕ. This reasoning does not hold for stationary or barely moving targets. Using imprecise hard constraints with these type of targets requires a different model.

In chapter 6 we developed two methods to deal with imprecise hard constraints in a realistic scenario. Both methods have their drawbacks however. We explained that the the grid method is an approximation in the sense that the state space is discretized. When we want to make the approximation better, we need to store a larger matrix. The problem with the shortest distance method lies in the fact that its speed is very low. This is due to the fact that the shortest distance from every particle to every polygon has to be computed for every time step.

8.3 Recommendations for further research

Up untill now, we have only looked at a model for imprecise knowledge. In chapter 2, we have seen that there are a lot more imperfections that can be present on measurements or measurement results. It is interesting to research the models for other types of imperfections such as bias, inconsistency or incompleteness.

In section 8.2 we discussed the use of our model for imprecise hard constraints. There were a few remarks that could be placed. We recommend looking into other ways of modelling imprecise hard constraints, such that also stationary targets can be tracked.

Finally, we recommend extending the optimization of the shortest distance method. Right now, the first steps towards a faster method are made, but there is still a lot to be accomplished. More research has to be done to find an optimal partitioning and conditions under which it is optimal. Another decrease in computational time could be gained through cutting the active polygons along the cell boundary. This way, the total amount of edges takes into account when computing the smallest distance is reduced. Additional research has to be carried out to show the influence of this technique.

Chapter 9

Working at Thales

Thales is a worldwide multinational with over 67, 000 employees in 56 countries. The company's name refers to the Greek mathematician Thales of Miletus. Thales is familiar among mathematicians for his theorem, alson known as the 'intercept theorem'. Worldwide, Thales is active in the markets of Defence, Aerospace & Space and Security. In 2011, the Thales group has an annual revenue of 13.03 billion euros.

This chapter will give an overview of the organization Thales as a company. We zoom in on Thales Nederland, the subdivision of Thales where I was active during my traineeship. The history of Thales is briefly reviewed followed by a discussion about the organisation of the company. The chapter is closed by my personal experience working at Thales during my period as a trainee.

9.1 History

Thales Nederland began in 1922 as 'Hazemeyer's Fabriek van Signaalapparaten'. Back then, the company was responsible for building the fire control systems for the Hr. Ms. Java and the Hr. Ms. Sumatra of the Dutch navy. The system had to be built according to the 'Duitsche Systeem', but this was problematic since the Treaty of Versailles prevented similar systems from being developped in Germany. An agreement was made between several companies, and the production of the systems could begin in the Netherlands, close to the German border.

In the second world war, Hazemeyer's Fabriek van Signaalapparaten was captured by nazi Germany and employees took refuge in Great Britain. When they returned after the war, all they found was a pillaged factory. The Dutch government soon realized that a good defense industry was vital and bought the installation. The company continued under the name 'Hollandse Signaalapparaten' or 'Signaal' for short. During the post-war reconstruction of the Netherlands, the company thrived and developed new techniques such as radar, fire control for the army, computers and air traffic control equipment.

Philips bought a large part of the shares and became the major shareholder in 1956. Due to the Cold War, governments had large budgets for defense which resulted in a period of growth for Hollandse Signaalapparaten. When the Cold War ended, the political theatre changed dramatically. Large cuts in defense budgets forced Signaal to reorganise, leading to a staff reduction. Meanwhile Philips decided that 'Defence and Control systems' were not part of its core-business and in 1990 Signaal was taken over by Thomson-CSF (now Thales). The name was changed in december 2000

Figure 9.1: The plant of Hollandse Signaalapparaten in Hengelo. Source: Signaal Museum

Figure 9.2: Signaal Engineers working on new technologies. Source: Signaal Museum

to Thales, to emphasize that the company was now internationally oriented. Thomson-CSF Signaal also changed its name into Thales Nederland.

9.2 Structure of the organisation

Thales group is a worldwide multinational with over $67,000$ employees in 56 countries. Thales headquarters is located in Neuilly-sur-Seine, a suburb of Paris. Worldwide, it is the eleventh largest defence contractor and 60% of its total sales are military sales.

Thales Nederland is the Dutch subdivision of Thales group. In the Netherlands, Thales has establishments in Hengelo, Huizen, Houten, Delft, Eindhoven and Enschede. The Dutch subdivision is responsible for a number of military and civil markets In Hengelo, marine radars and electro-optic sensors for defense purposes are developed. The department in Huizen is resposible for communication systems for command & control and battlefield management. In Eindhoven, cryocoolers and batteries for military goals are developed. In Houten the Thales transporation systems department is located: there they make for instance e-ticketing systems for the public transport. The management of Thales Nederland has its office in Hengelo, the main establishment.

Apart from the subdivision in establishments, Thales Nederland can also be subdivided into units: the units of 'Land Defence & C4I Systems/Transportation Systems', 'Naval Systems' and 'Sensors'. The department 'Sensors', where I was active during my traineeship, is responsible for the development and delivery of custom made radars and optical sensors, and accompanying knowledge and technology. Again, the unit Sensors is divided into different departments. During the period of my traineeship I was active in the department of Engineering. In the department of Engineering, software is developed for the processing of data and signals. Also, research is done to look for new, more accurate algorithms to improve performance of radar systems.

9.3 Personal experience

In november 2011 I got the opportunity to work on a project that would take me a year at Thales Nederland. This meant that I had to combine my traineeship and final project and spend the last year of my study in Hengelo. I could start in december 2011 and finish in december 2012. During the first month, I spent my days at the University of Twente because my supervisors at Thales had other occupations. This time was well spent studying the literature required for doing research on the topic of particle filters. I had my own office in the Citadel on the campus of the University of Twente. During those days I had meetings with my supervisors once in a while, such that I knew where to focus my study on.

After a month of studying, my workplace changed from the University of Twente to Thales Nederland, location Hengelo. There I actually started working on my graduation project.

9.3.1 Supervisors

During my traineeship and work on my final project I was supervised by three supervisors. From the University of Twente I was supervised by prof. dr. A.A. (Anton) Stoorvogel. At Thales Nederland, my supervisors were dr. Y. (Yvo) Boers and dr. M. (Martin) Podt. In may 2012, Yvo, my first supervisor, got seriously ill and was bound to stay home for at least a few months. This left me with Martin as my main supervisor for the rest of the period of my traineeship.

Every week I had meetings with Martin, and in the beginning, Yvo. We talked about the progress in my research and other work related topics. I met my supervisor at the University, Anton Stoorvogel, less often. During these meetings also the progress was reviewed and commented.

9.3.2 Daily routines

At Thales a regular working hours are from 8.00 untill 16.30, but in practice employees come and go during the day. This is due to flexible hours: employees are asked to work 40 hours a week with working hours between 7.00 and 19.00, from monday to friday. A half hour lunch is excluded from your working hours.

I worked in an office with three desks, accompanied by one other employee. He was not working in the same field as me, but that did not prevent us from having good conversations once in a while. When working at Thales, you are in the possesion of a computer. Due to strict security rules, laptops are not allowed on the terrain so I had to deal with the equipment available. This meant that installing new software is not as easy as usual (you had to ask the companies IT department), but this caused no major issues. Also, websites are not freely accessible: they can only be reached via a (slow) secure connection, unless they are blocked.

My daily activities consisted of doing research (through algebraic manipulation, simulation and literature research), discussing with supervisors and other colleagues and writing the thesis. Apart from the research, there were times I was involved into other activities; these are all viewed in sections 9.3.3, 9.3.4 and 9.3.5.

9.3.3 EKSPLORE

The project I was involved in was called EKSPLORE: Exploiting Knowledge in Sensor Processing for tracking Low-level Objects in Realistic Environments. EKSPLORE is a cooperative project between Thales en Saab, a company in Sweden focussed on defence and security. It is a European research project which is financed by the Dutch and Swedish ministries of defence, in the project they are called 'customers'.

Every half year a so called progress review meeting is held to inform the customers about the progress in the project. After the meeting, the customers decide if they find that there is enough progress to continue the project. During the second progress review meeting, which was held in Hengelo, I was able to present my work to the customer and other attendees. This was a formal meeting where I had to present my research to an international audience. In preparation for this presentation, a progress review report had to be written.

Altogether, this was a very nice experience, dealing with formal writing and presentation. Also, the formal and informal contact with international clients and researchers was a valuable experience. It was interesting to see how other researchers tackled their problems, which were similar to the problems I was dealing with.

9.3.4 Volonta

Thales takes good care for trainees working in Hengelo: they have set up a student association called Volonta. At Volonta, Thales trainees get to know eachother and participate in both exploratory and fun activities. Fun activities include bowling, karting, visiting the Grolsch brewery and having a barbecue. Exploratory activities are for example a tour in the Signaalmuseum or a visit to the Dutch marine base in Den Helder. In Den Helder, we got the opportunity to walk around inside a

Figure 9.3: The logo for EKSPLORE

Dutch frigate, this was really impresive. Also, we participated in a test of a Thales system at the Grolsch Veste. However, the most impressive event was the trip to Jouy-en-Josas, near Paris, where Thales University was located. Thales had organised an 'Interns Forum', this was an informative day where we could meet other interns and Thales recruiters from all over the world.

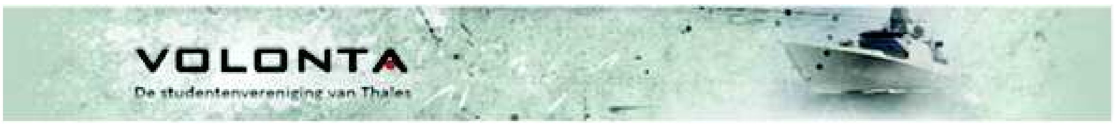


Figure 9.4: The logo for Volonta

9.3.5 Other noteworthy events

Apart from my regular business at Thales I was involved in a couple of side occupations. I already mentioned the work for EKSPLORE and the trips with Volonta.

Halfway my traineeship period at Thales, I held a presentation for direct colleagues who were also involved in sensor processing. This gave a good impression of the position of my research.

Once in a while, I was working at home or the University. This was mainly due to the fact that I needed software which was not available at Thales Nederland. You can think of Google Earth or for instance LaTeX-Word conversion software, needed for the EKSPLORE report.

In oktober the study association for mathematics from Enschede visited Thales. Together with two colleagues I was responsible for the guided tour through the facilities. After the tour, I held a short presenation about the activities of a mathematician at Thales Nederland.

Appendix A

Results for chapter 5

This appendix contains the simulation trajectories and simulations results from chapter 5.

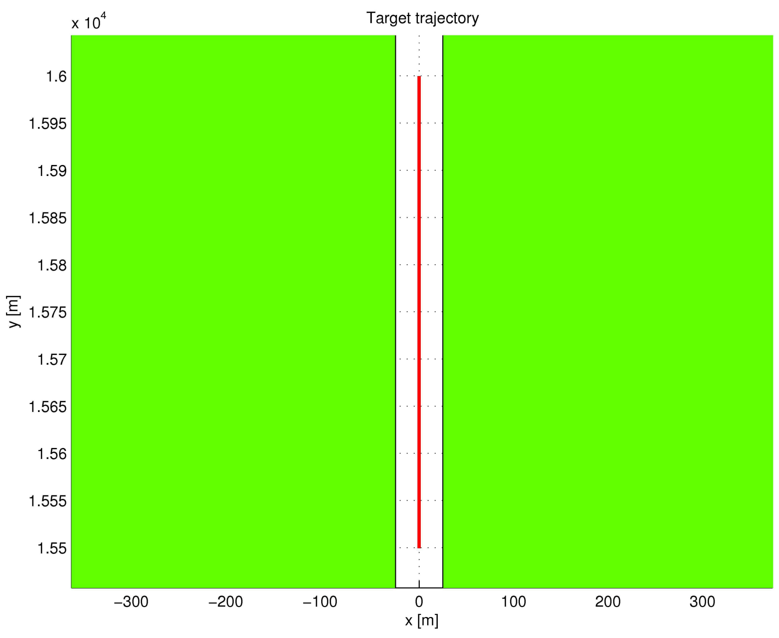


Figure A.1: Trajectory for perfect map

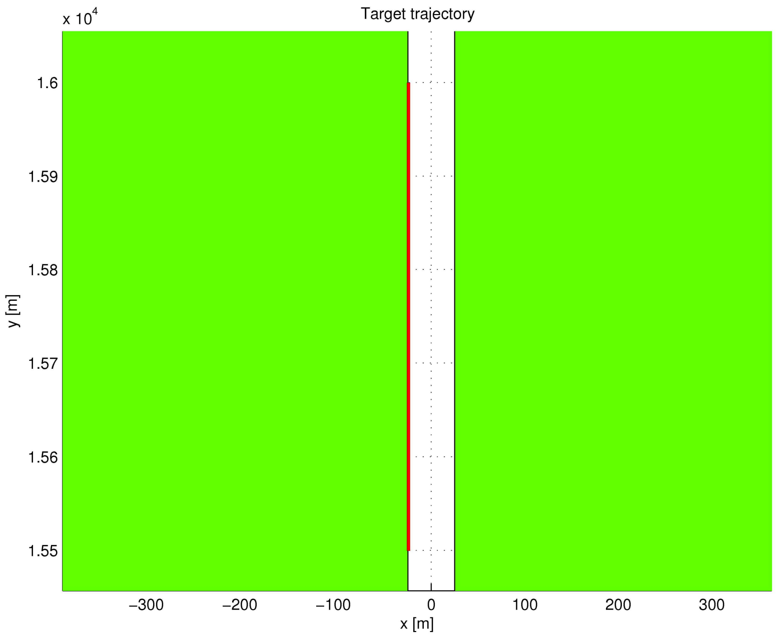


Figure A.2: Trajectory for perfect map

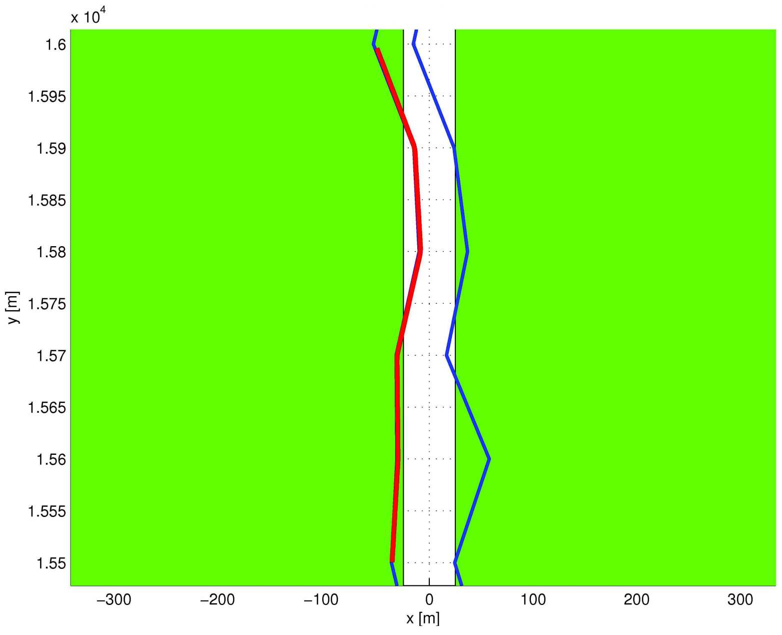


Figure A.3: Trajectory along the actual coast

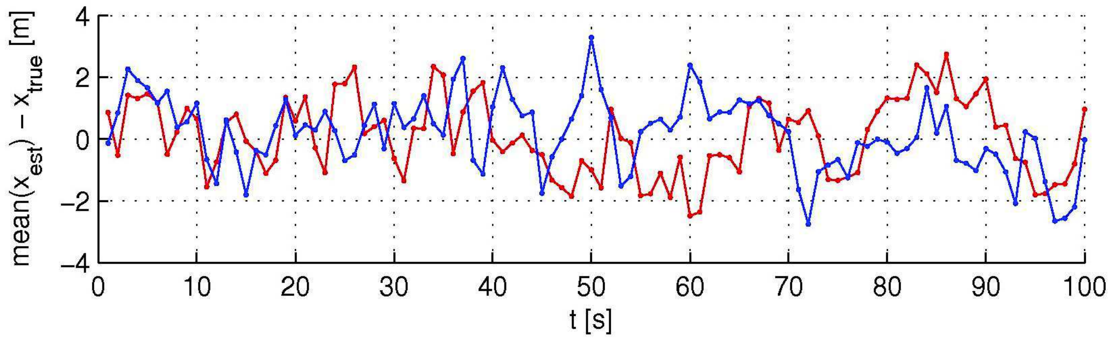


Figure A.4: Mean bias of PF1 (red) and PF2 (blue) for the trajectory through the middle of the canal

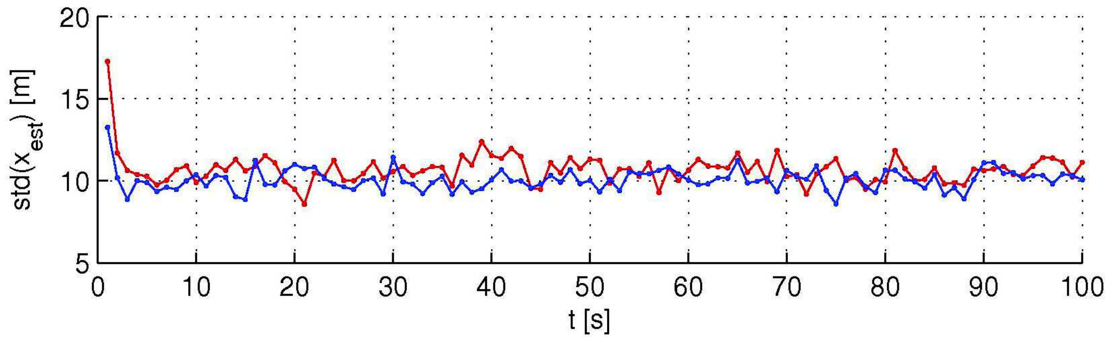


Figure A.5: Standard deviation of PF1 (red) and PF2 (blue) for the trajectory through the middle of the canal

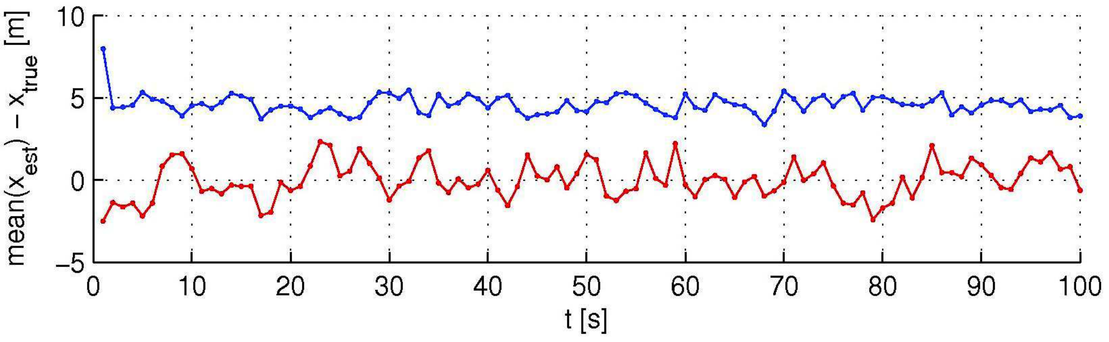


Figure A.6: Mean bias of PF1 (red) and PF2 (blue) for the trajectory along the coast of the canal

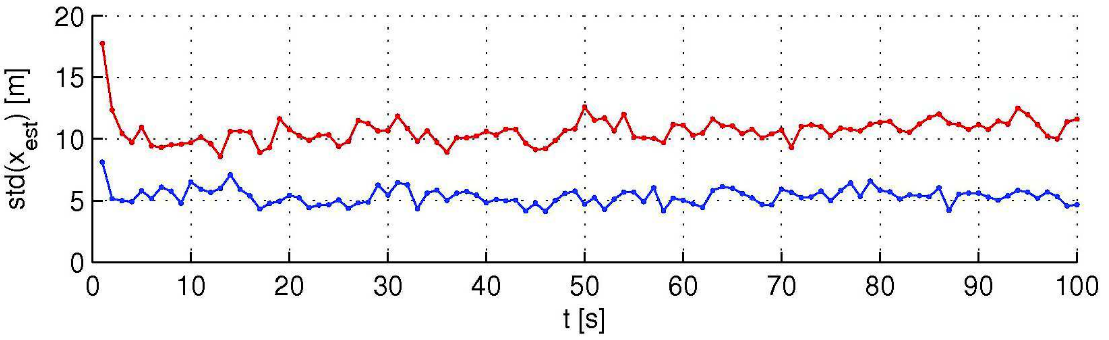


Figure A.7: Standard deviation of PF1 (red) and PF2 (blue) for the trajectory along the coast of the canal

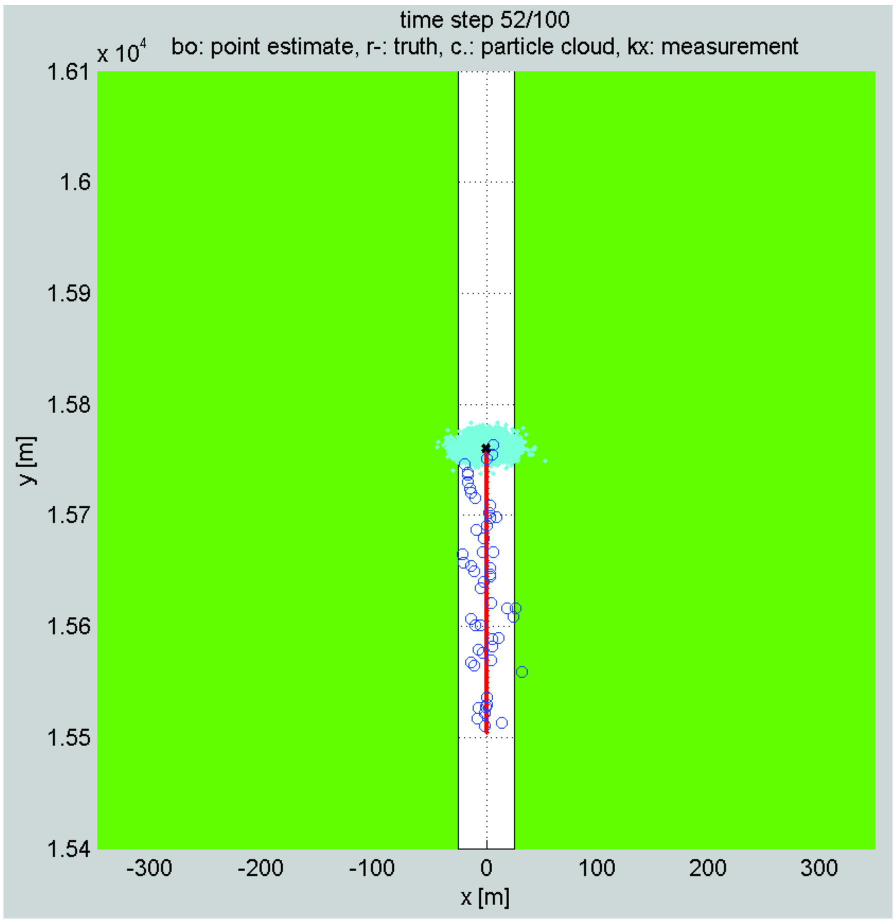


Figure A.8: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

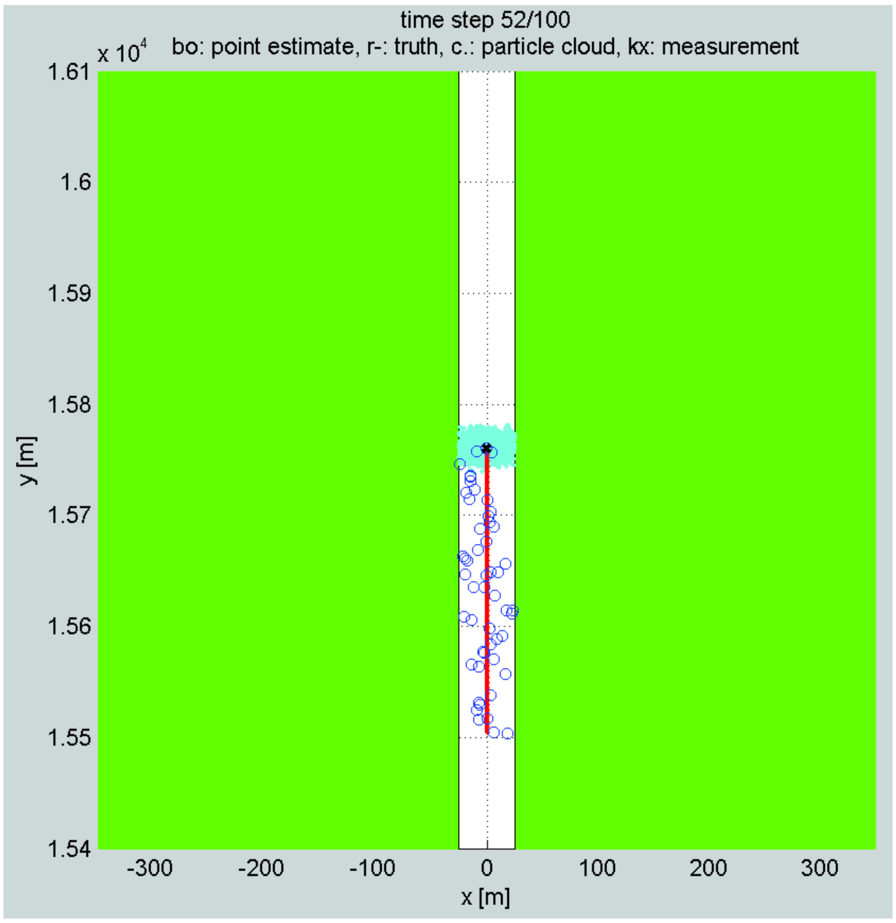


Figure A.9: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

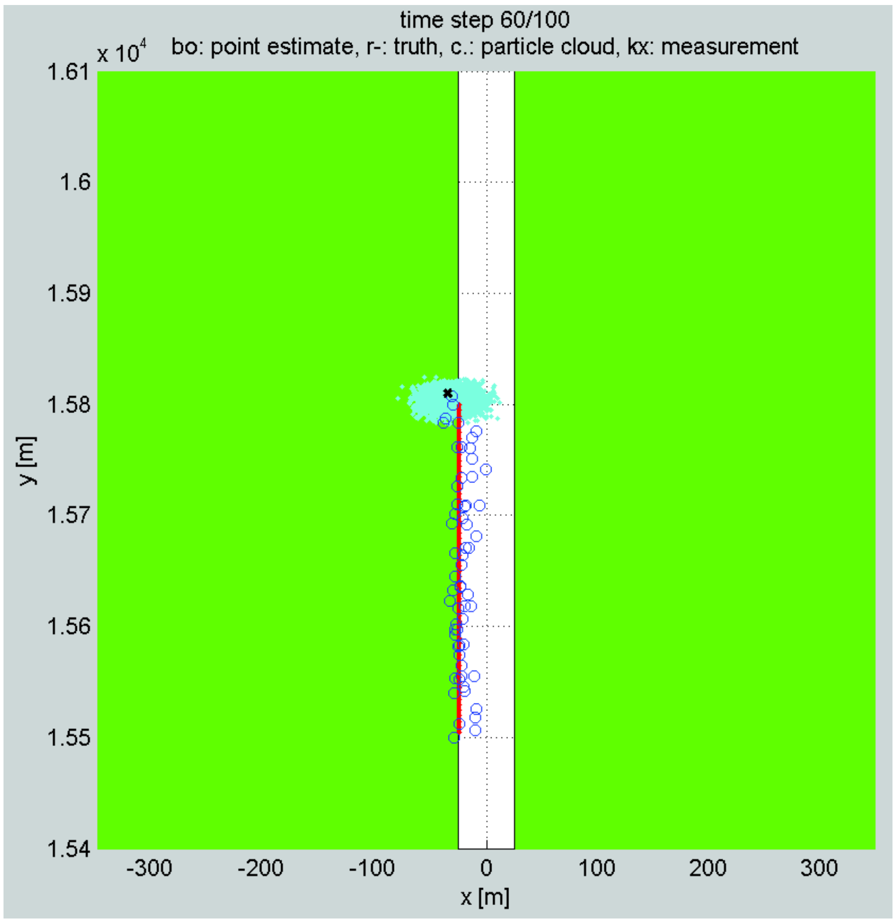


Figure A.10: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

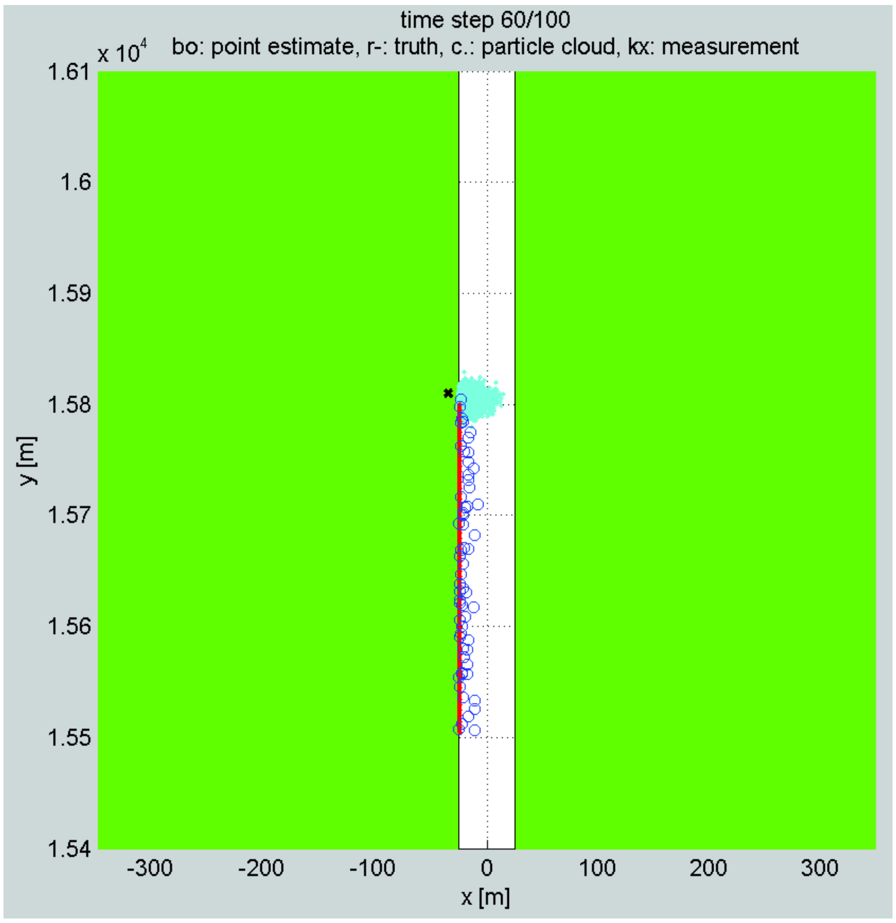


Figure A.11: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

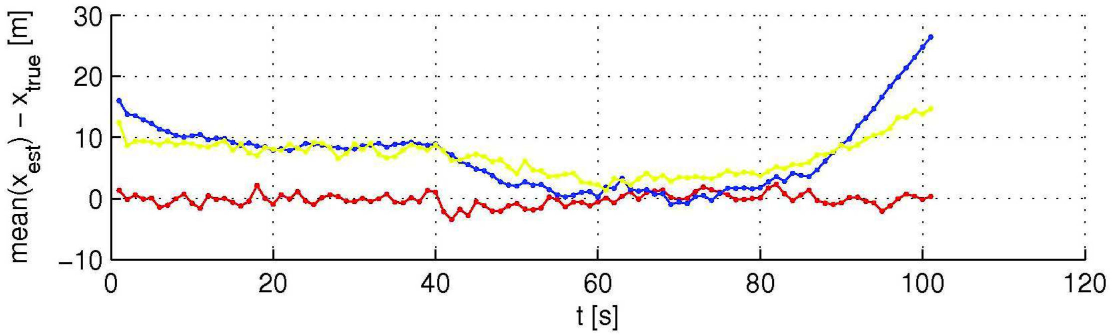


Figure A.12: Mean bias of PF1 (red), PF2 (blue) and PF3 (yellow) for the trajectory along the actual coast of the canal

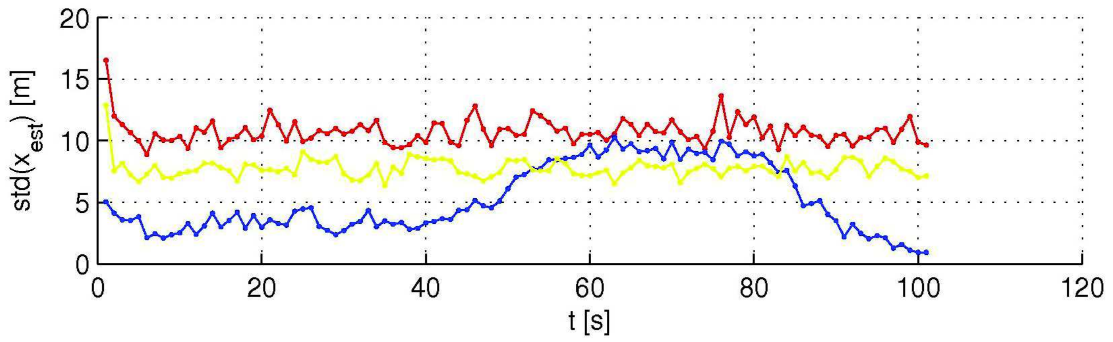


Figure A.13: Standard deviation of PF1 (red), PF2 (blue) and PF3 (yellow) for the trajectory along the actual coast of the canal

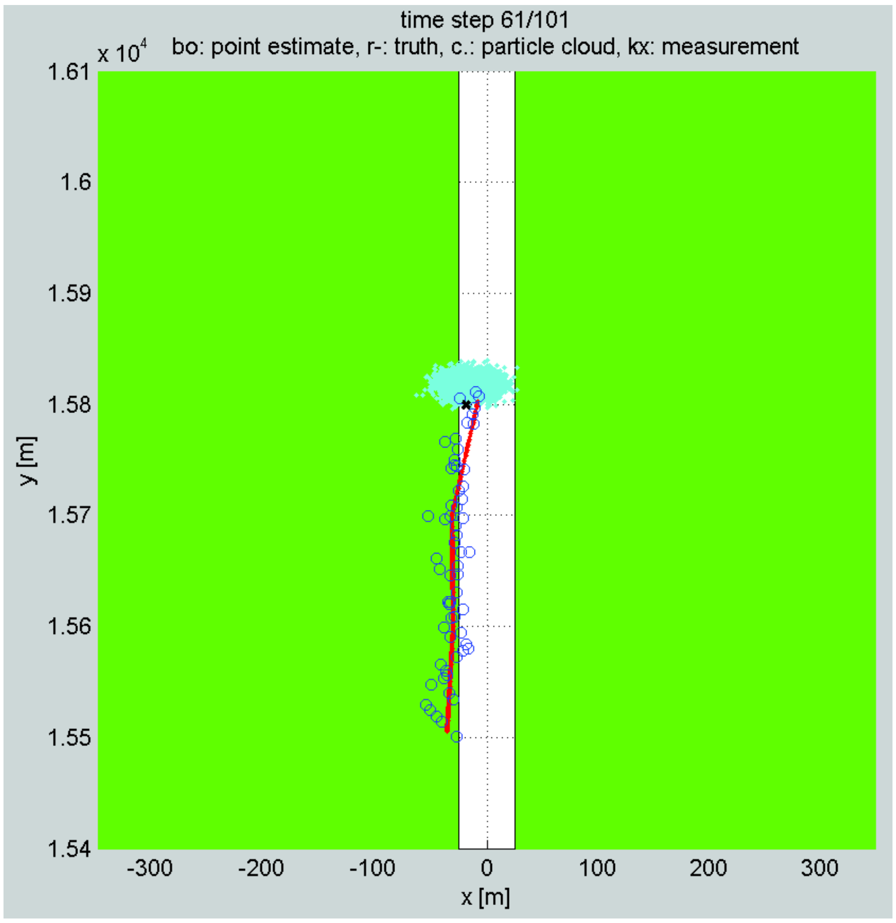


Figure A.14: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

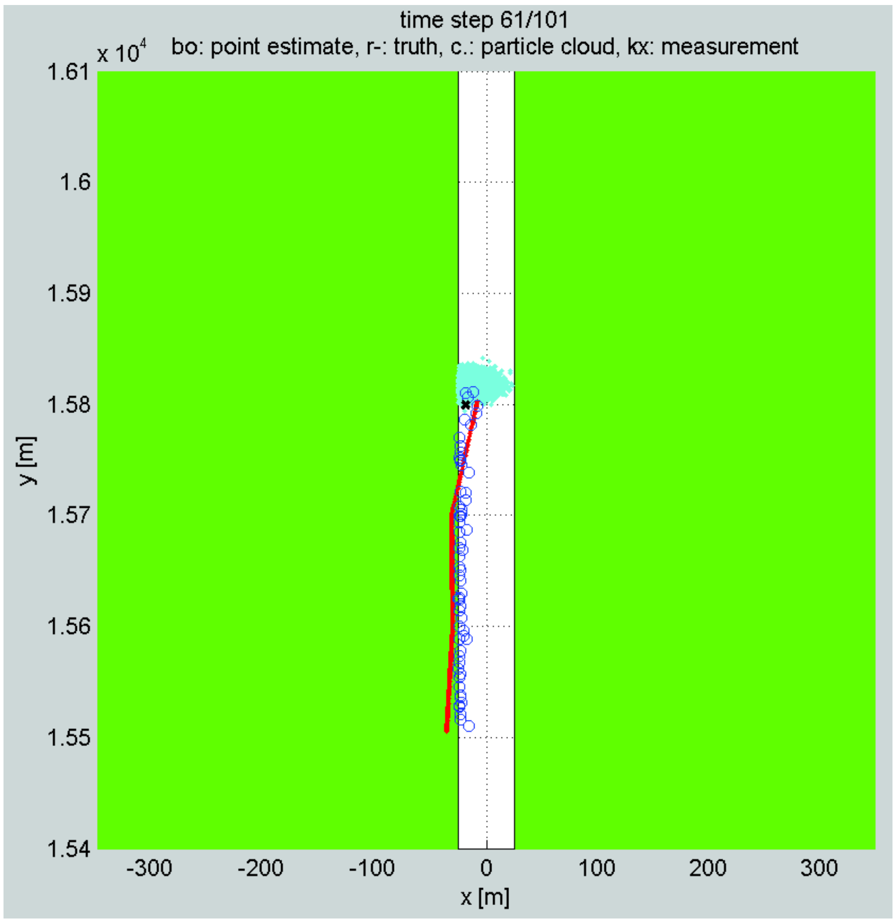


Figure A.15: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

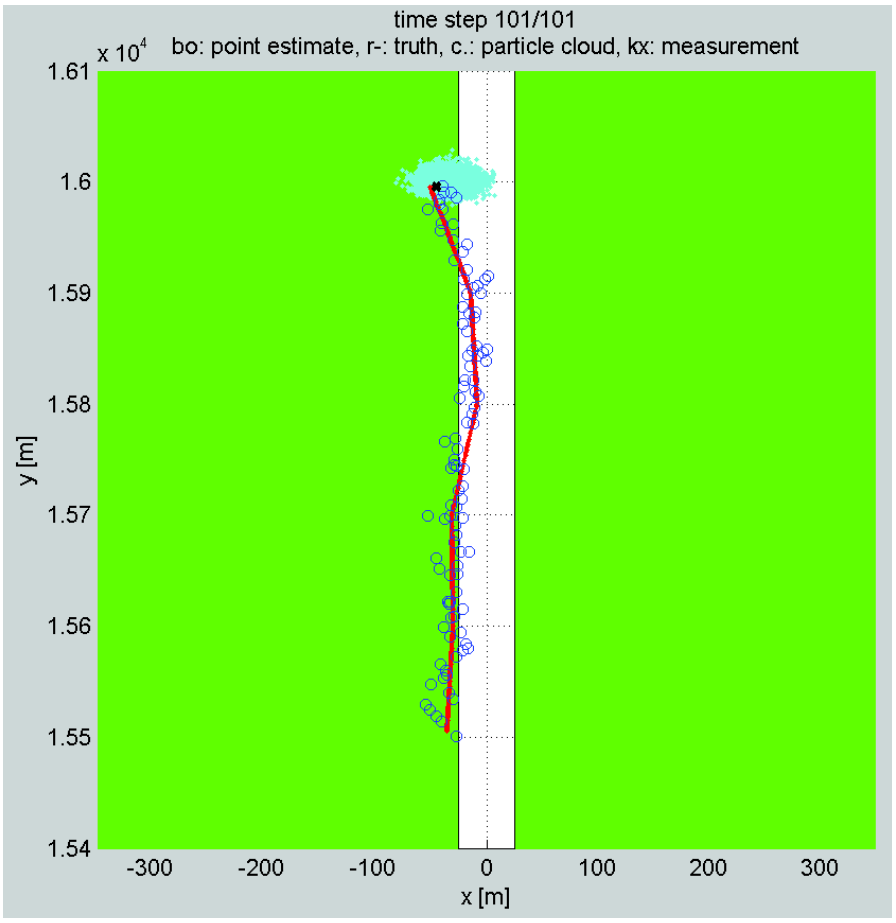


Figure A.16: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

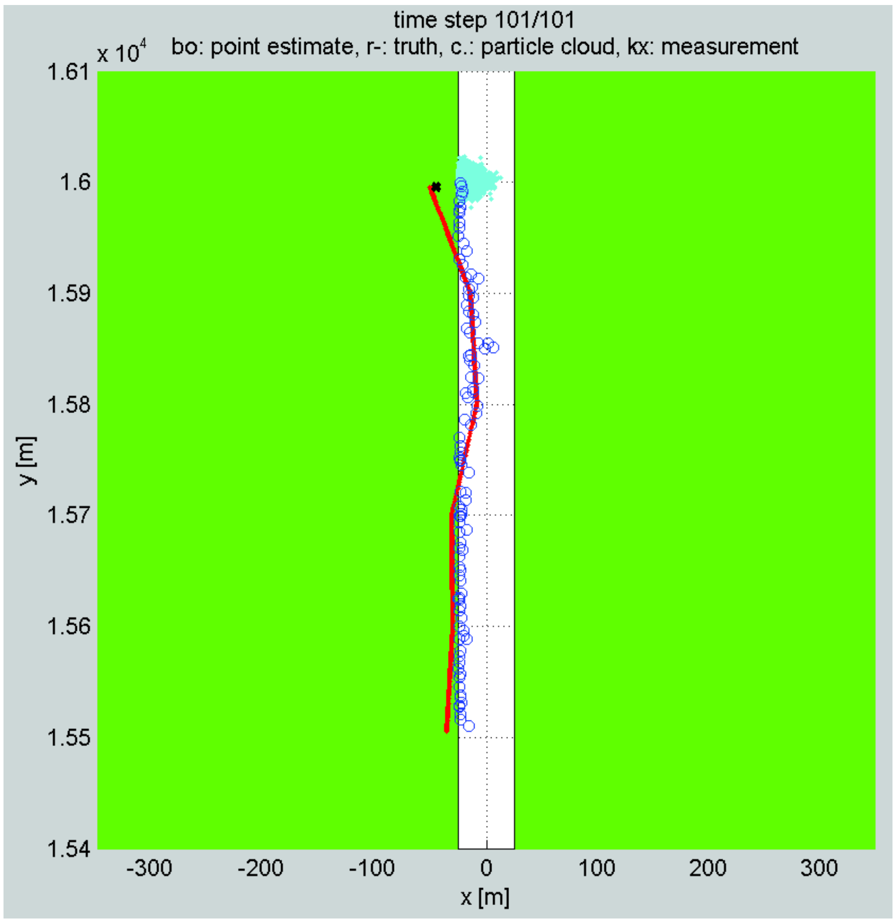


Figure A.17: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

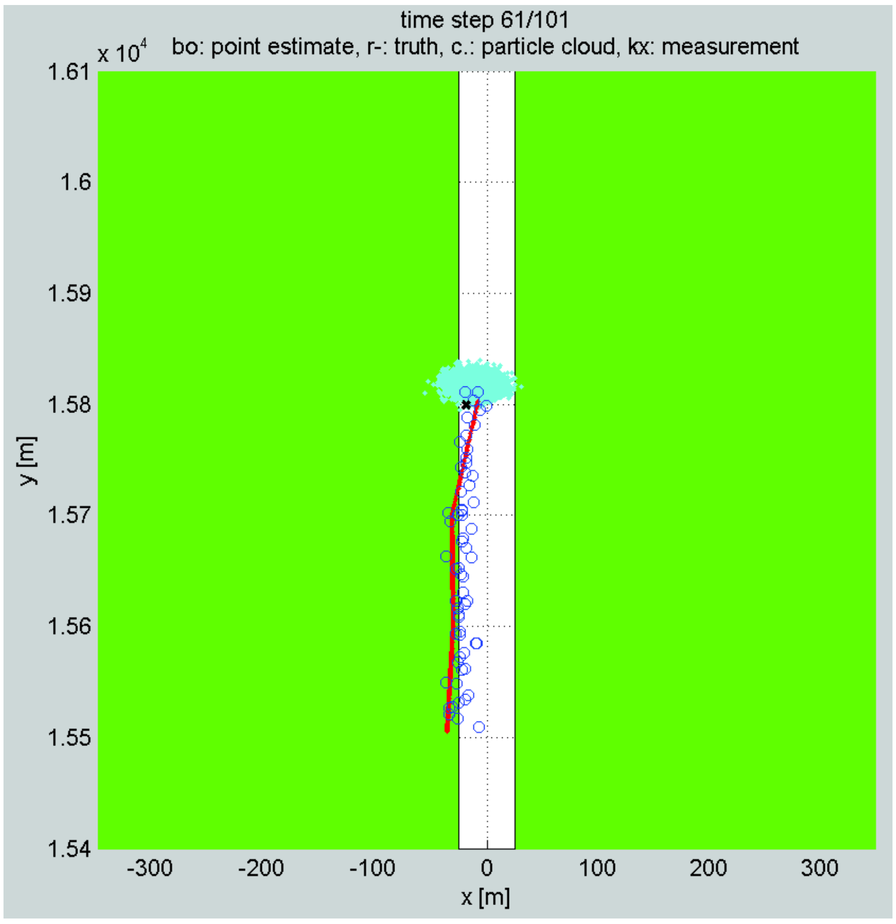


Figure A.18: Particle cloud for PF3. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

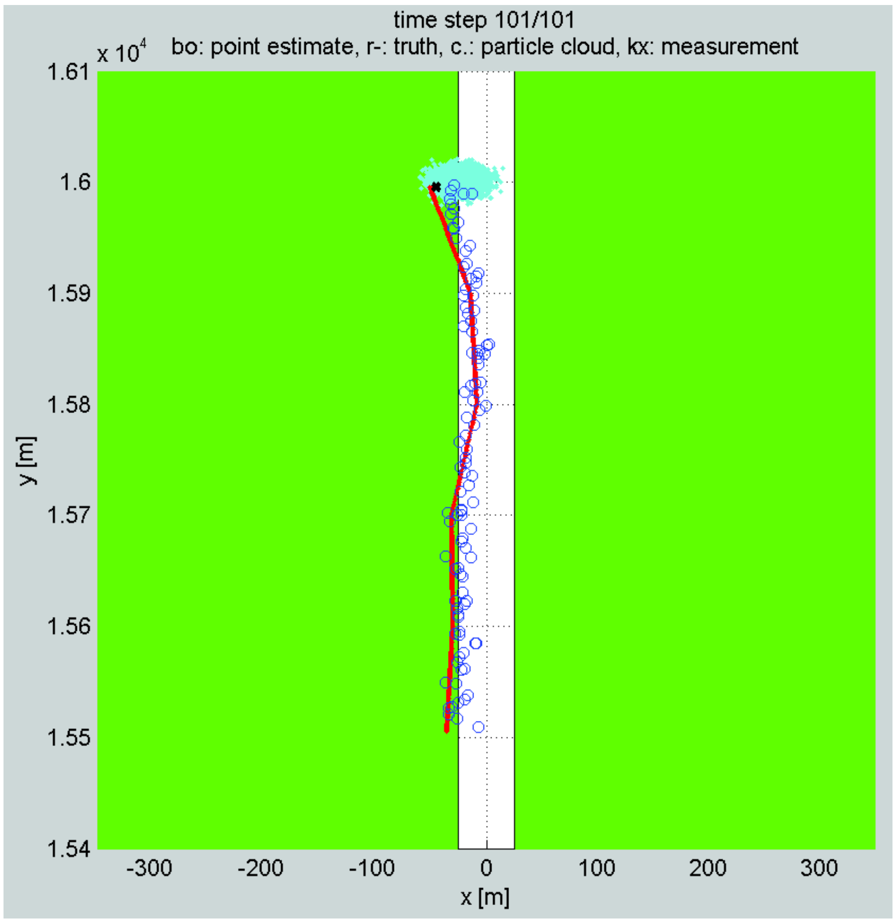


Figure A.19: Particle cloud for PF3. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

Appendix B

Results for chapter 6

This appendix contains the simulation trajectories and simulations results from chapter 6.

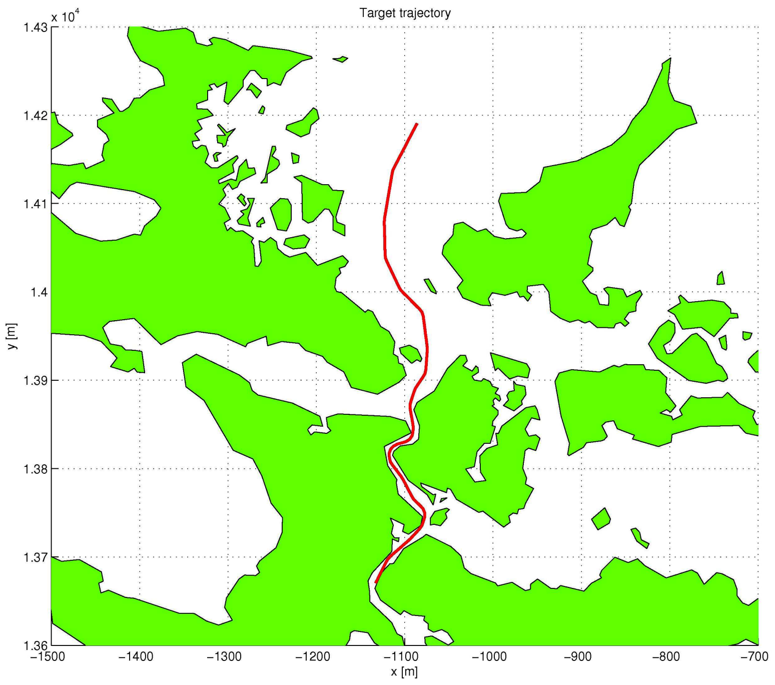


Figure B.1: Trajectory for perfect map

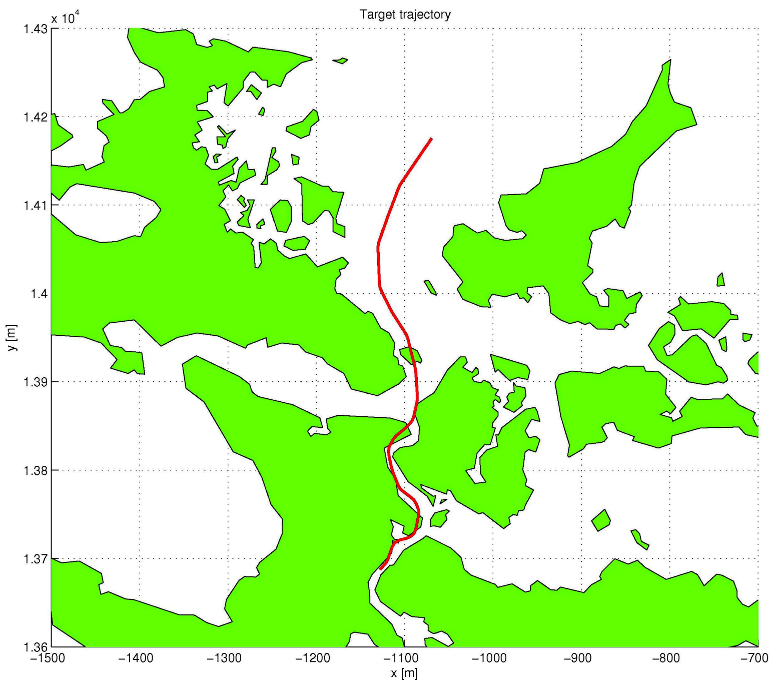


Figure B.2: Trajectory for imprecise map

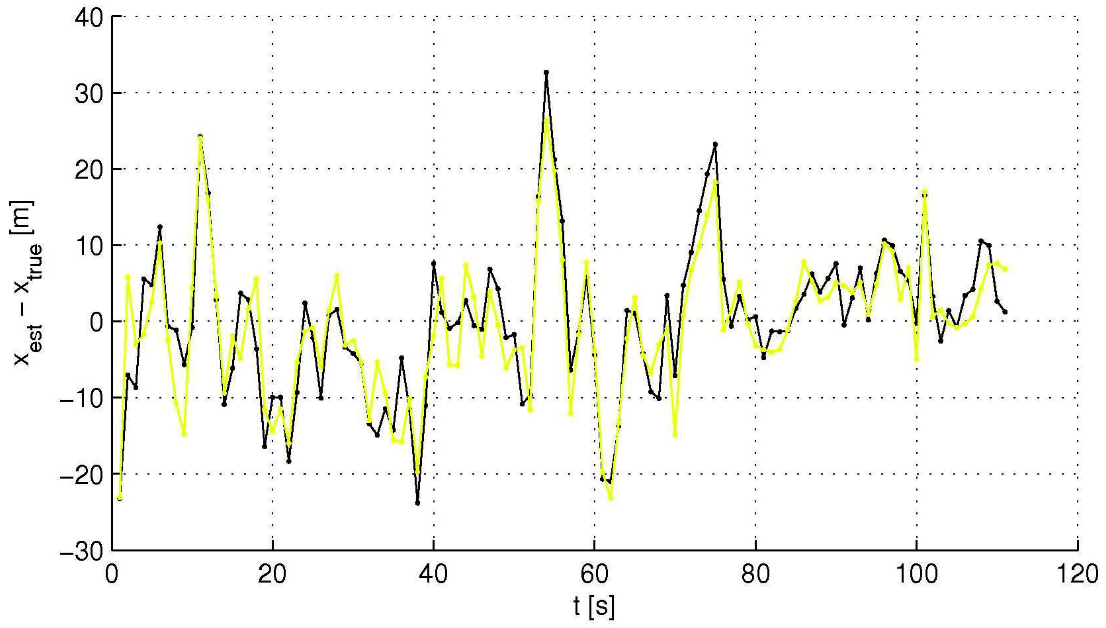


Figure B.3: Bias over horizontal position for two different methods

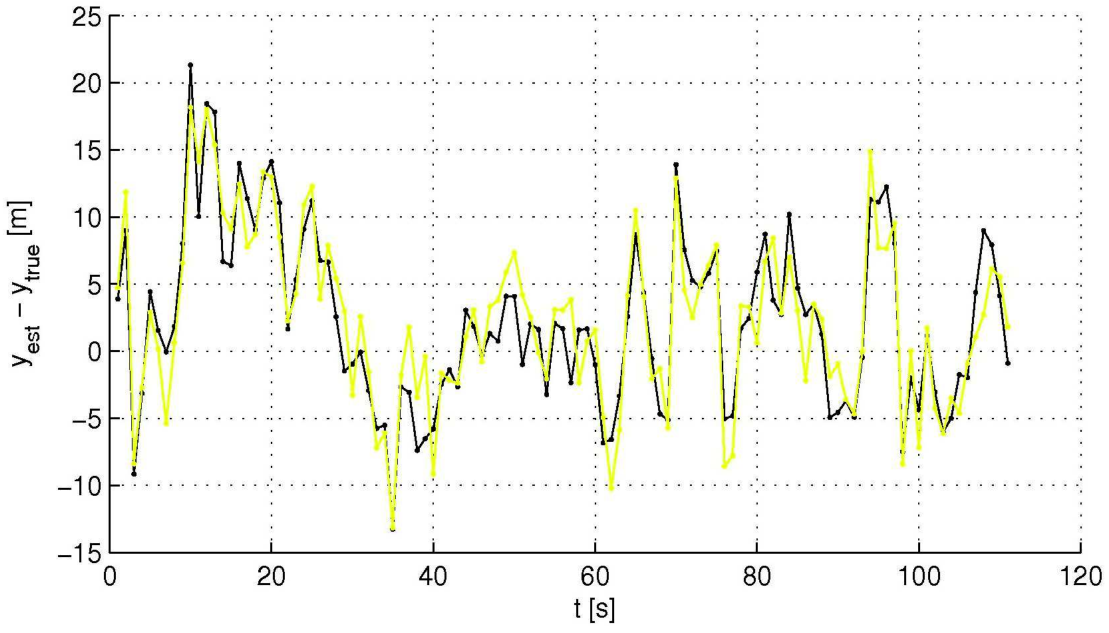


Figure B.4: Bias over vertical position for two different methods

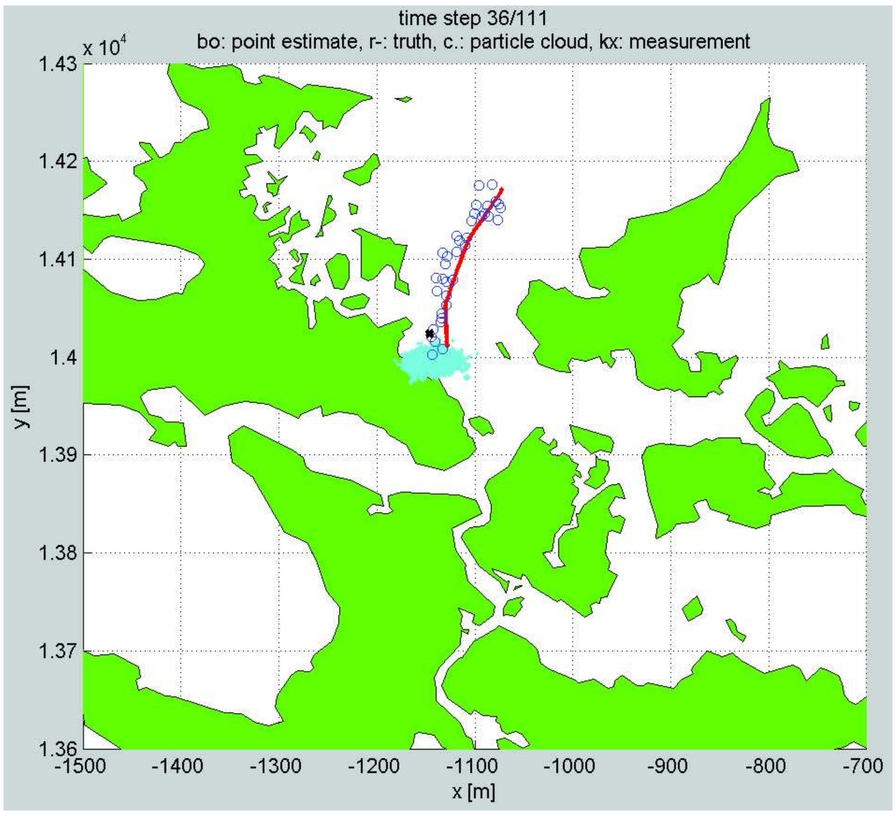


Figure B.5: Particle cloud for PF3. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

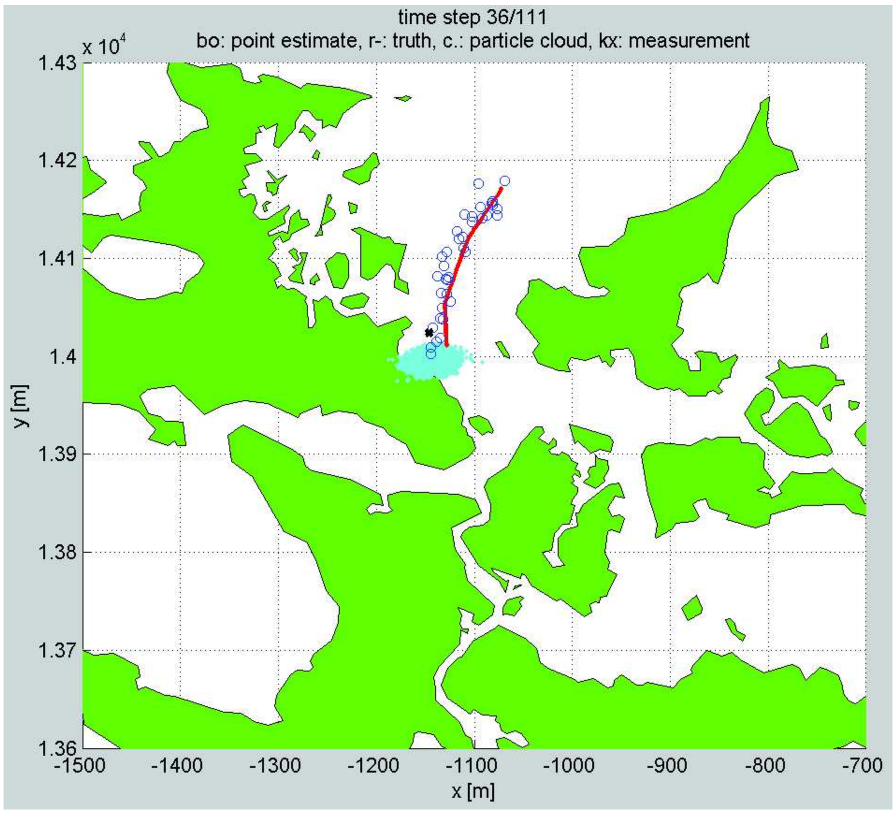


Figure B.6: Particle cloud for PF4. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

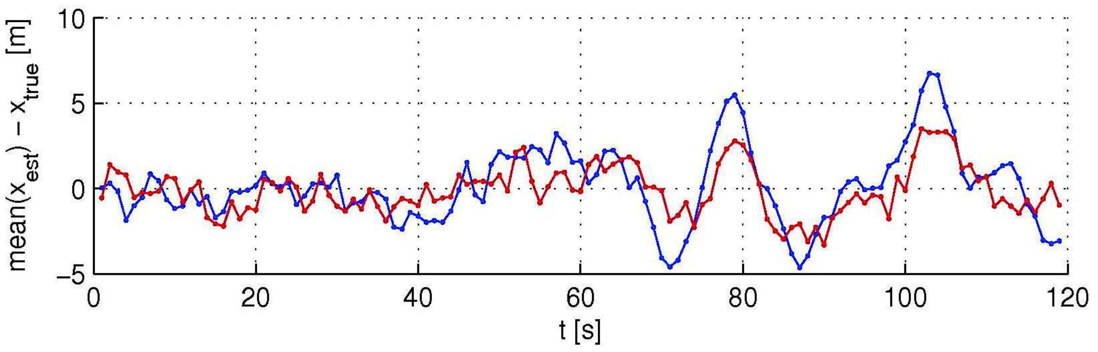


Figure B.7: Mean bias over horizontal position for simulations on perfect map

THALES

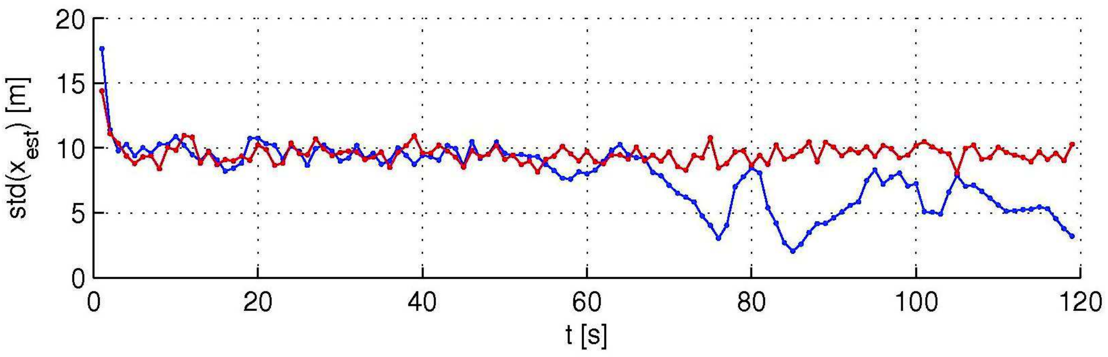


Figure B.8: Standard deviation over horizontal position for simulations on perfect map

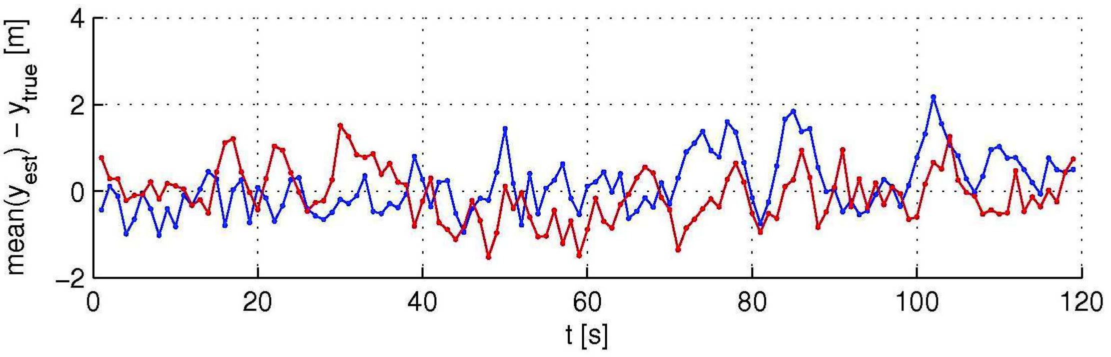


Figure B.9: Mean bias over vertical position for simulations on perfect map

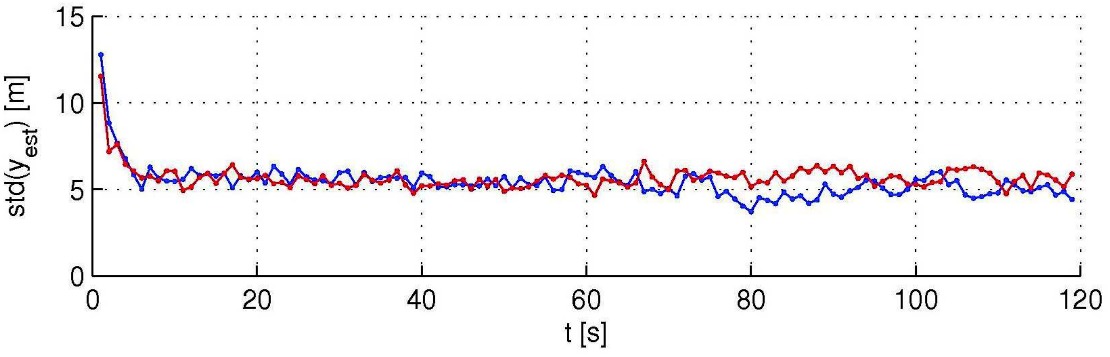


Figure B.10: Standard deviation over vertical position for simulations on perfect map

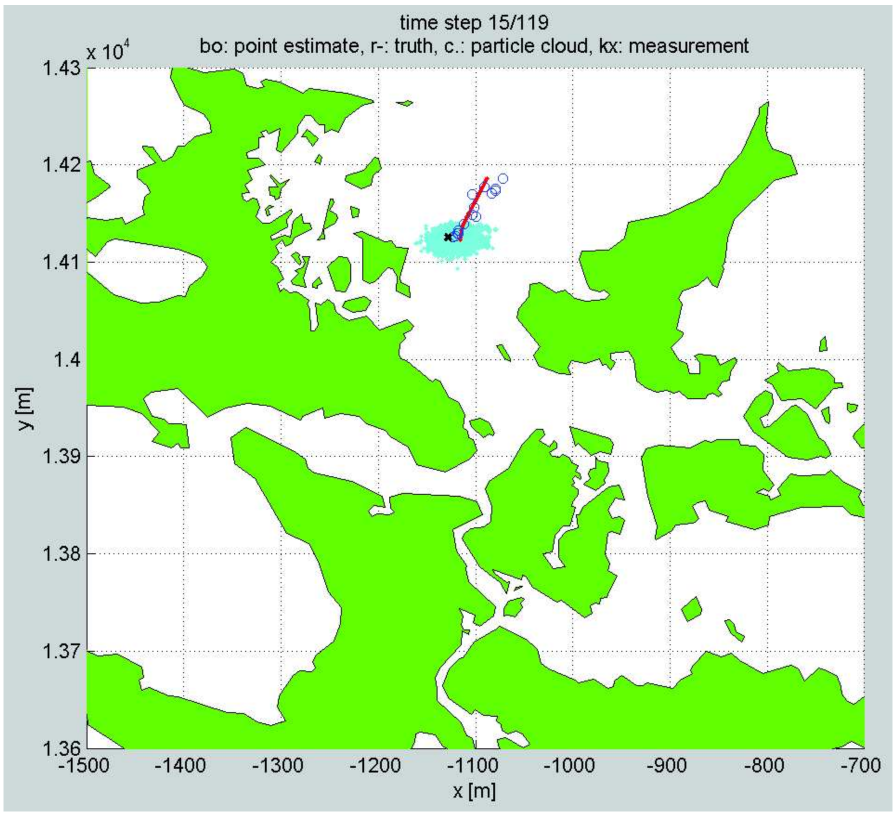


Figure B.11: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

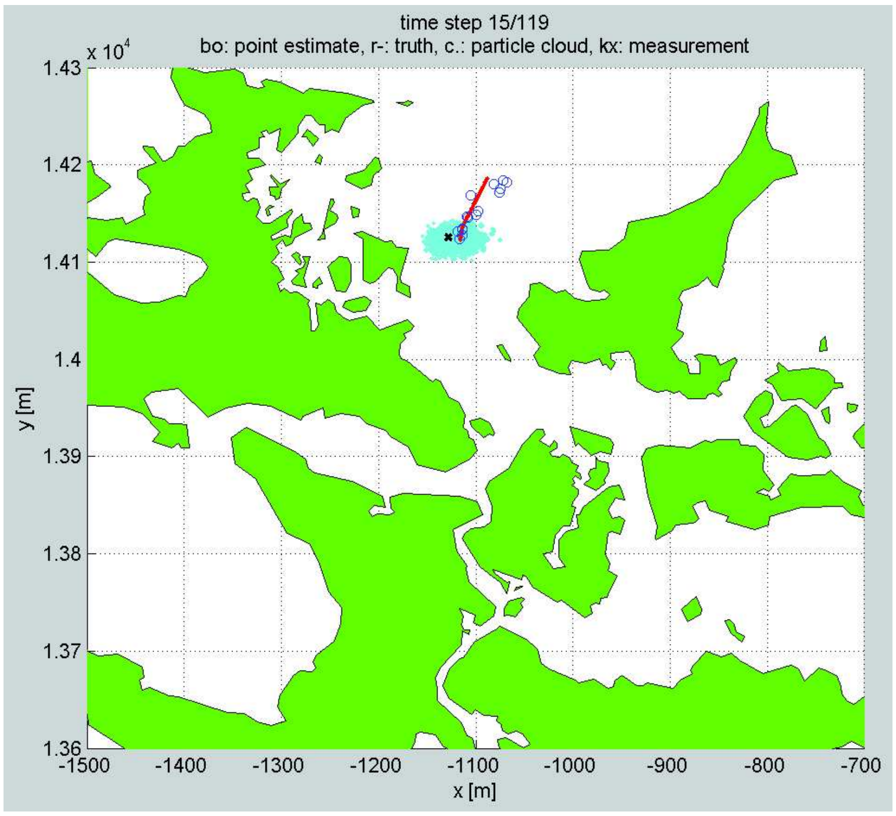


Figure B.12: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

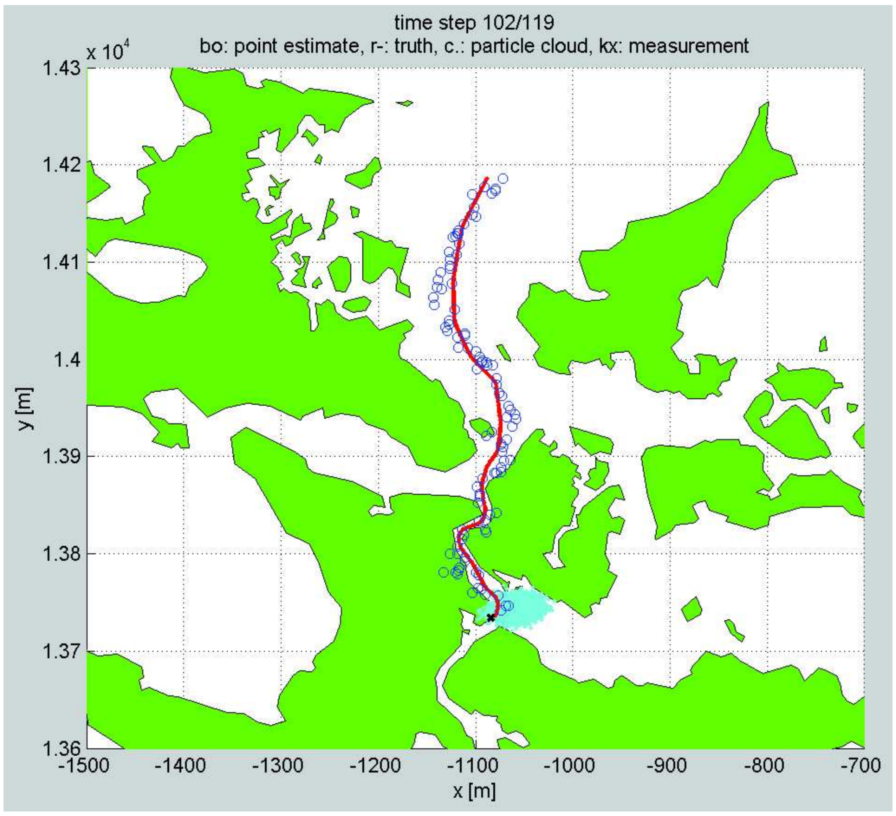


Figure B.13: Particle cloud for PF1. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

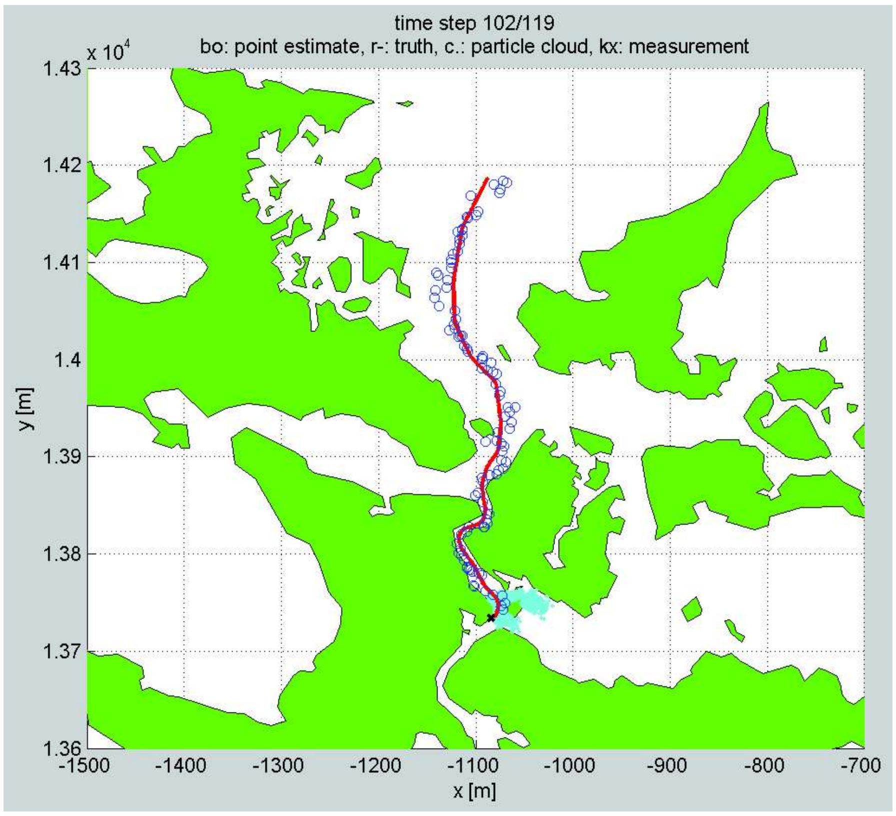


Figure B.14: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

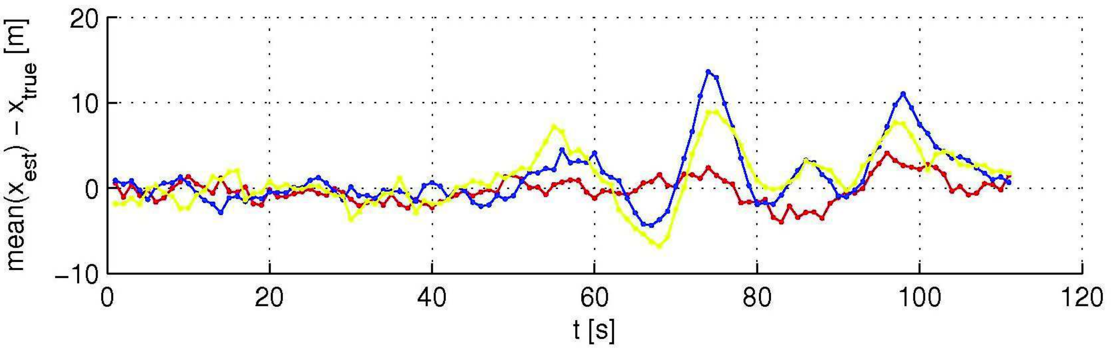


Figure B.15: Means bias over horizontal position for simulations on imprecise map

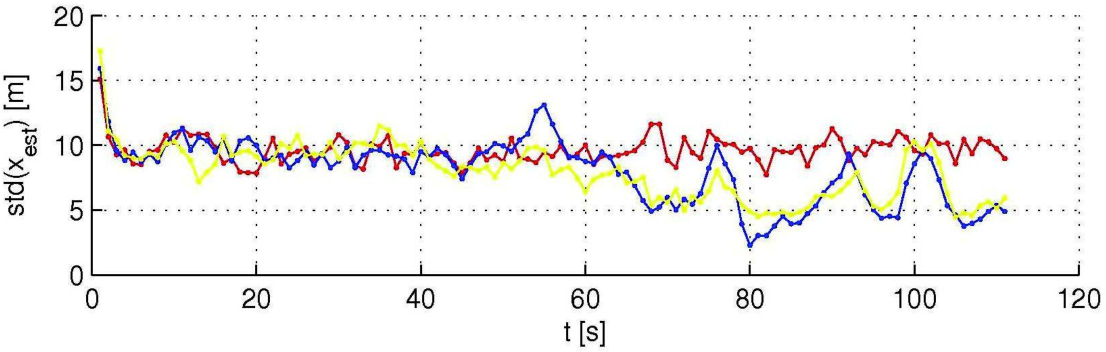


Figure B.16: Standard deviation over horizontal position for simulations on imprecise map

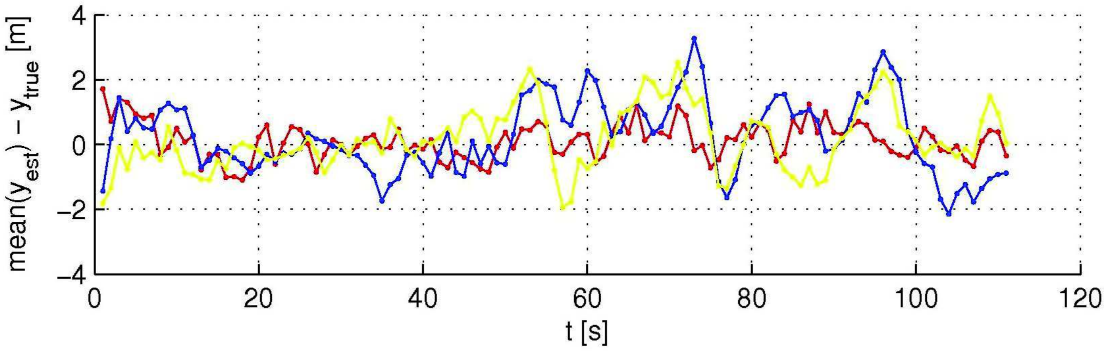


Figure B.17: Mean bias over vertical position for simulations on imprecise map

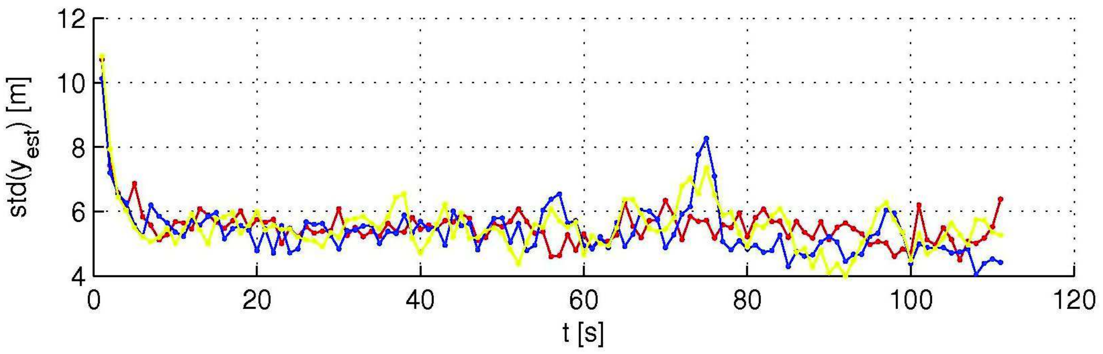


Figure B.18: Standard deviation over vertical position for simulations on imprecise map

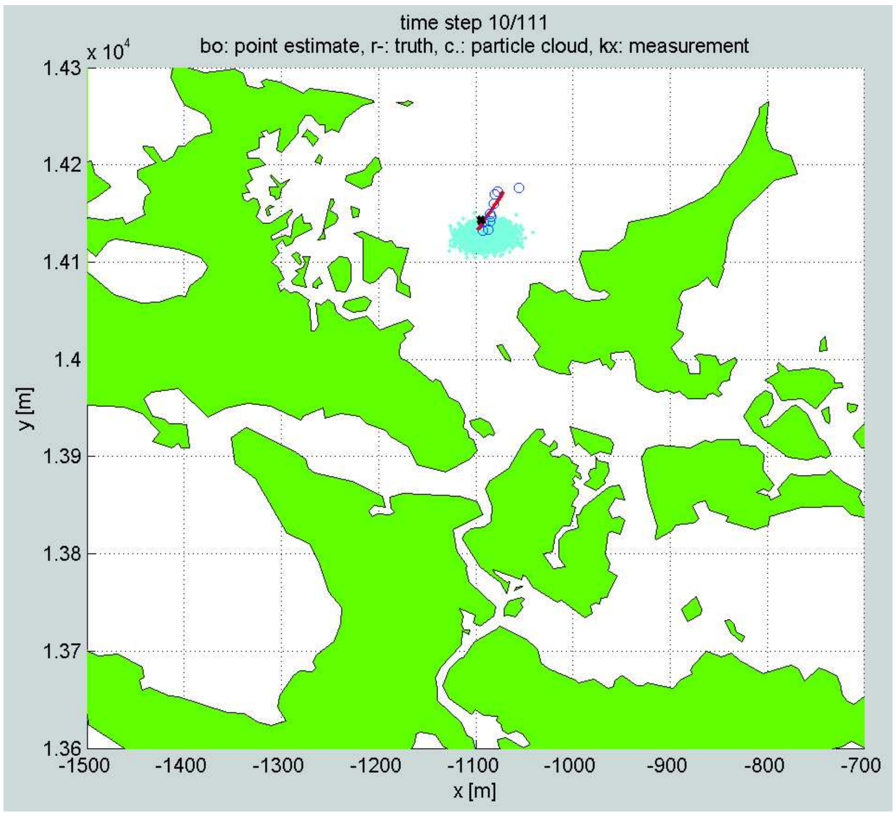


Figure B.19: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

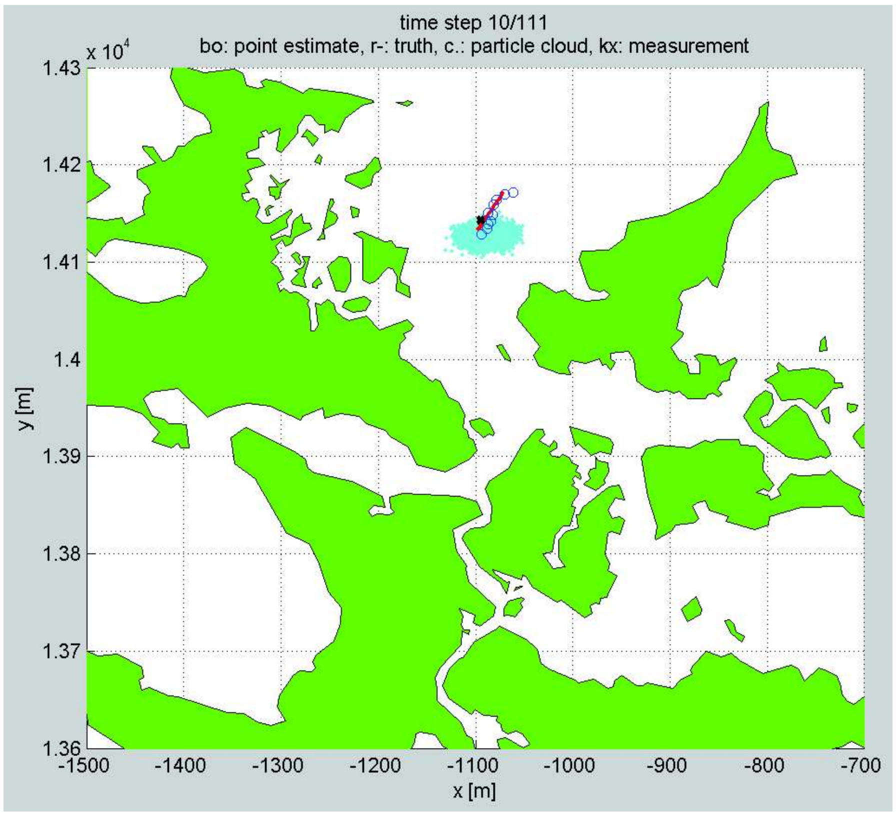


Figure B.20: Particle cloud for PF4. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

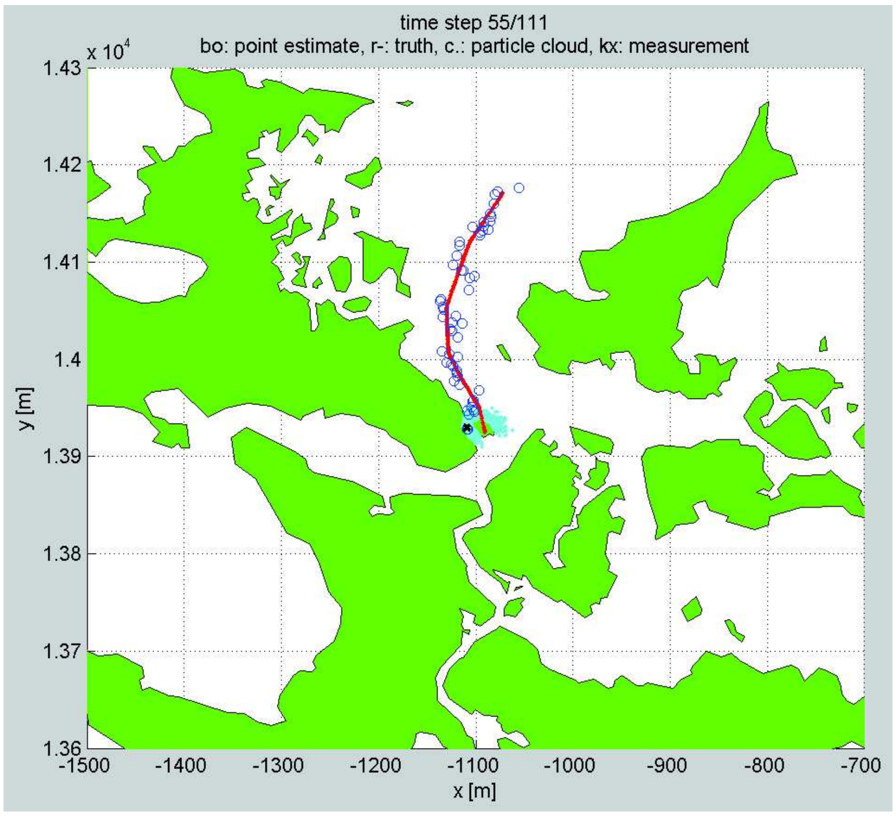


Figure B.21: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

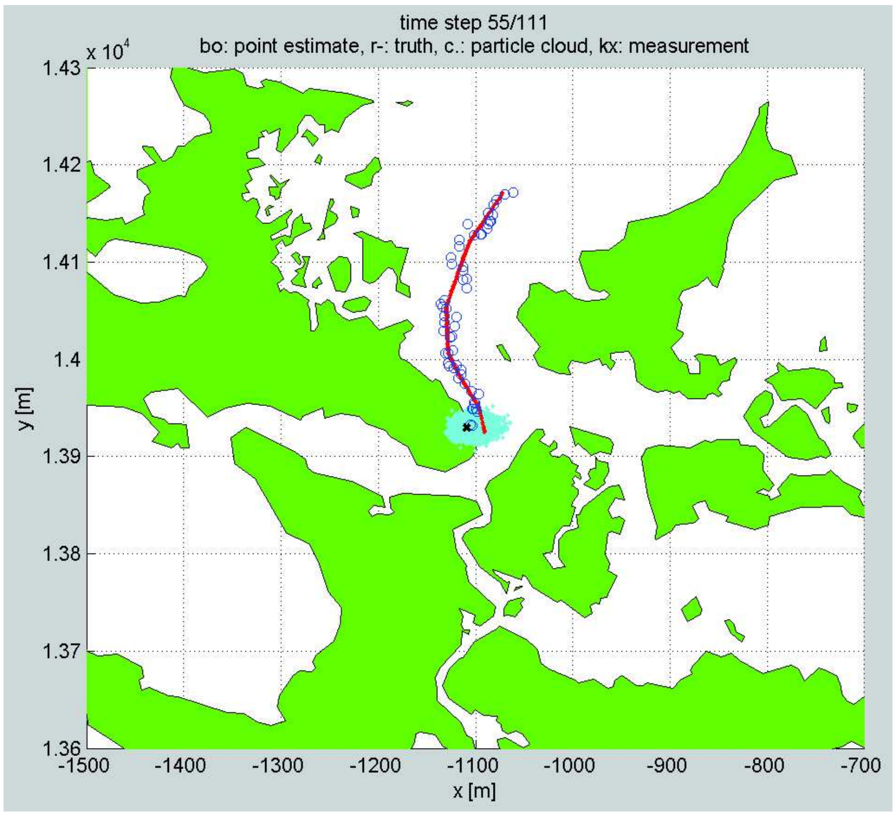


Figure B.22: Particle cloud for PF4. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

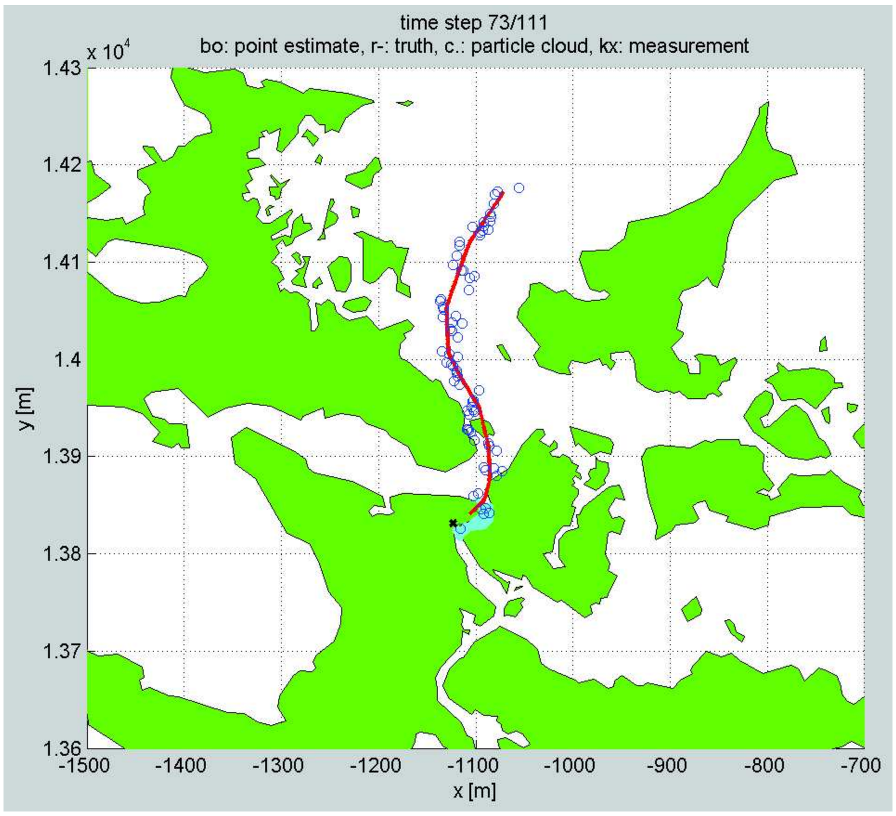


Figure B.23: Particle cloud for PF2. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

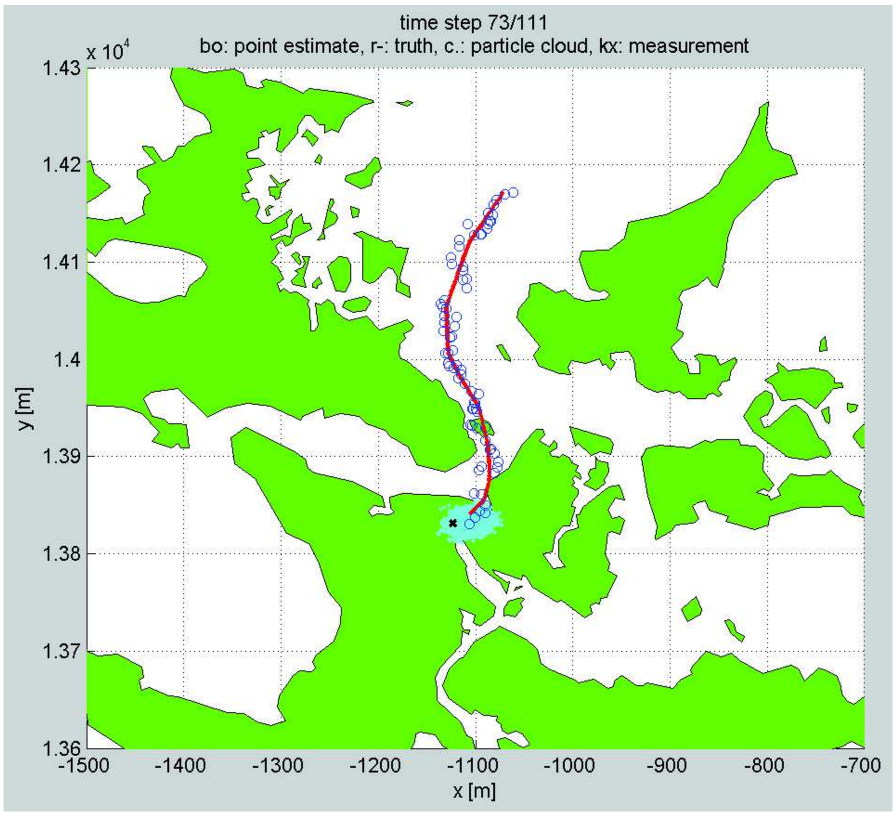


Figure B.24: Particle cloud for PF4. The blue circles represent the MAP estimates, the black 'x' is the current measurement and the cyan cloud the particle cloud.

Bibliography

[1] Oxford english dictionary, 2nd edition. Oxford University Press, 1989.

[2] Accuracy (trueness and precision) of measurement methods and results - part 1: General principles and definitions. Technical Report ISO 5725-1, International Organization for Standardization, Geneva, Switzerland, 1994.

[3] Accuracy (trueness and precision) of measurement methods and results - part 4: Basic methods for the determination of the trueness of a standard measurement method. Technical Report ISO 5725-4, International Organization for Standardization, Geneva, Switzerland, 1994.

[4] Statistics - vocabulary and symbols - part 2: Applied statistics. Technical Report ISO 3534-2, International Organization for Standardization, Geneva, Switzerland, 2006.

[5] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing (50,2)*, 2002.

[6] S. Arulampalam B. Ristic and N. Gordon. *Beyond the Kalman Filter: particle filters for tracking applications.* Artech House Inc., 2004.

[7] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Norwood, MA: Artech House, 1998.

[8] F. Papi et al. Bayes optimal knowledge exploitation for target tracking with hard constraints. *To be published*, 2012.

[9] F. Gustafsson. *Statistical Sensor Fusion.* Norwood, MA: Artech House, 1998.

[10] R.G. Mann. Inland electronic navigational charts in the u.s. army corps of engineers. *Oceans, 2001. MTS/IEEE conference and exhibition (vol. 4)*, pages 2342–2347, 2001.

[11] N. Perugini. Behind the accuracy of electronic charts. *Sea Technology - March 2001 issue*, 2001.

[12] N.E. Perugini. Noaa's electronic navigational chart program. *Oceans '02 MTS/IEEE (2)*, pages 1042–1045, 2003.

[13] S.M. Ross. *Introduction to Probability Models.* Academic Press, 2006.

[14] S. Saha, Y. Boers, H. Driessen, P.K. Mandal, and A. Bagchi. Particle based map state estimation: A comparison. In *12th International Conference on Information Fusion*, 2009.

[15] M.E. Skolnik. *Introduction to radar systems*. McGraw-Hill, 1962.