ISBN 978-82-471-2214-3 (printed ver.)
ISBN 978-82-471-2215-0 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2010:123

Kjetil Bergh Ånonsen

Advances in Terrain Aided Navigation for Underwater Vehicles

Kjetil Bergh Ånonsen

Doctoral Thesis

NTNU
Norwegian University of
Science and Technology
Thesis for the degree of
philosophiae doctor
Faculty of Information Technology, Mathematics and
Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2010:123

NTNU

NTNU
Norwegian University of
Science and Technology

Kjetil Bergh Ånonsen

Advances in Terrain Aided Navigation for Underwater Vehicles

Thesis for the degree of philosophiae doctor

Trondheim, June 2010

Norwegian University of
Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of philosophiae doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-471-2214-3 (printed ver.)
ISBN 978-82-471-2215-0 (electronic ver.)
ISSN 1503-8181

Doctoral Theses at NTNU, 2010:123

Printed by Tapir Uttrykk

With gratitude to my parents and my grandmother Åse Marie

Summary

This thesis deals with the subject of terrain aided underwater navigation for underwater vehicles. The term "navigation" is throughout the thesis understood as the task of estimating the position and attitude of a vehicle, together with the corresponding uncertainties. The main focus is on autonomous underwater vehicles (AUVs), but the methods discussed can also be readily used by other underwater vehicles.

Terrain aided navigation is an attractive concept for obtaining submerged position fixes for the main navigation system, in most cases an inertial navigation system, for which the deterioration of the position accuracy necessitates external aiding methods. Terrain navigation has been used for decades in land and air applications, e.g. in aircraft and cruise missiles. In recent years, the technique has been applied also in underwater vehicles. The thesis concentrates on what is known as "bathymetric terrain navigation", in which bathymetric measurements are matched directly with a map, as opposed to the related area of "feature-based navigation", in which features extracted from the bottom measurements are used for position updates. Throughout the thesis, the terrain navigation system is treated as an external module, providing measurement updates for the main navigation system in a loosely coupled manner. This approach makes the terrain navigation module more portable and the overall system more robust to errors in the terrain navigation updates, although it is more difficult to exploit the internal states of the main navigation system in the terrain navigation algorithm.

The thesis starts with a recapitulation of existing methods for terrain navigation (Chapter 2), with focus on Bayesian estimation methods. The problem is formulated as a recursive state-space estimation problem, which is highly nonlinear, due to the nonlinear nature of the terrain measurement function. As a consequence, nonlinear estimation methods like point mass filters (PMFs), particle filters (PFs) and sigma point Kalman filters (SPKFs) must be used. Special emphasis is put on the fact that, due to the computational complexity of the estimation methods, one must often use low-dimensional state-space models, leading to discrepancies between the true system and the filter model. Such discrepancies sometimes lead to inaccuracies and overconfidence in the estimators.

Chapter 3 continues with a discussion on the special difficulties that arise when using terrain navigation techniques underwater, e.g. different sensor characteristics, the effect of tide compensation etc. It is also shown how unknown depth biases can be handled, either through estimation of the bias in the filter, or by using relative depth information only. A novel formulation of the filter process model is also developed, in an attempt to model the drift in the main navigation system more accurately. Chapter 4 gives an overview of map databases, which are essential to the success of terrain navigation.

The main contributions of the thesis are related to the computational results presented in Chapter 5. The behavior in different terrain types of the TERCOM (Terrain Contour Matching) algorithm, the PMF algorithm, various particle filters and the SPKF are compared, using sea data from an AUV equipped with a multibeam echosounder. All

the tested methods are able to estimate the position of the AUV with an accuracy within the horizontal resolution of the terrain map, with the PMF as the most accurate and robust, though also the most computationally demanding, method. A clear positive effect on accuracy and robustness from the inclusion of the depth bias is observed. The main problem with the methods discussed in the thesis is their tendency of overconfidence, i.e. the estimated uncertainty is too low compared to the true uncertainty. This can be partially solved by sub-sampling the terrain measurements, minimizing the effect of unmodeled correlations. Results from computations using the novel process model derived in Chapter 3 show that this approach does not solve the inconsistency problem, though the accuracy and stability are slightly improved on the tested data. Chapter 5 closes with some simulations using a map database based on real MBE data from an area with pockmarks, i.e. small craters on the sea floor. The sea floor in the area is otherwise flat. The simulations indicate that the pockmarks contain enough terrain information to facilitate the use of terrain navigation in areas previously thought of as unsuited.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree "Philospiae Doctor" at the Norwegian University of Science and Technology (NTNU), the Department of Engineering Cybernetics. The work has been part of the UNaMap (Underwater Navigation and Mapping) project, funded by the Norwegian Research Council (Norges forskningsråd), Kongsberg Defence and Aerospace and Kongsberg Maritime, and was conducted at the University Graduate Center at Kjeller (UniK), in close cooperation with the HUGIN AUV groups at the Norwegian Defence Research Establishment (FFI) and at Kongsberg Maritime (KM), Horten.

The work that eventually resulted in this thesis started back in August 2003, when I, armed with a fresh Sivilingeniør (M.Sc.) degree in Industrial Mathematics from the Department of Mathematical Sciences at NTNU, knowing next to nothing about the extremely interesting research area of underwater navigation, commenced my project as a doctoral candidate at UniK. After four years at UniK, including a valuable year as a visiting grad student at MIT, and nearly three years as a scientist in the HUGIN group at FFI, with my thesis as a "side project", I can look back at a very interesting period of my life.

Many people have contributed to the thesis as it appears today. First, I would like to thank my advisor, professor Oddvar Hallingstad at UniK. His wide knowledge on estimation theory and strong opinions on notation and terminology have led to many interesting discussions. Also thanks to my great fellow PhD students at UniK, both for valuable professional discussions and for countless coffee breaks, dinners and parties. Next, I have to thank Ove Kent Hagen at FFI, whose contribution to my work has been enormous. His experience and knowledge on underwater vehicles in general and terrain navigation in particular have been an essential resource to me. I also thank Magne Mandt, who gave me an excellent introduction to terrain navigation, and the rest of my colleagues in the HUGIN group at FFI. Thanks also to Bjørn Jalving of Kongsberg Maritime, who during his time at FFI initiated the UNaMap project and whose insightful comments during project meetings have been of great value. I am also grateful to my superiors at FFI, Per Espen Hagen (now KM) and Roy Hansen, for allowing me to use some of my workdays for writing up my thesis. I thank Professor John Leonard at the MIT Center of Ocean Engineering/MIT CSAIL for giving me the opportunity to learn from and work with a number of talented and great people.

I also take the opportunity to thank my family; my parents Ingunn and Gunnar, whose continuous love and support over 32 years have contributed enormously to making me who I am today, my sisters Hilde and Kjersti, and my brother Magne, for all the great times we have had over the years. Last, but definitely first, I thank my wonderful girlfriend Ragnhild. Without your unlimited love, support and encouragement, this thesis would still have been half finished.

Oslo, March 2010 Kjetil Bergh Ånonsen

Contents

1

Introduction

OCEANS cover around 70 % of the Earth's surface, yet most of the deep oceans remain relatively unexplored, especially at large depths. Deep oceans are hostile environments, with high pressure and difficult light conditions, making the exploration of these extremely challenging. The advent of advanced underwater vehicles has, however, made detailed exploration of the oceans feasible, both in shallow and deep waters. The applications of such vehicles are numerous, spanning from oceanographic research, seabed mapping and environmental monitoring to military applications, e.g mine hunting and reconnaissance. Essential to all these applications is the ability to accurately determine the location of the vehicle. A measurement is of little value unless one knows the location at which the measurement was taken. This thesis deals with important aspects within the field of underwater vehicle navigation, particularly those related to the use of terrain properties for navigation, known as terrain-based navigation. The focus is mainly on autonomous underwater vehicles (AUVs), though the methods presented can also be used by other underwater vehicles, like submarines and remotely operated vehicles (ROVs).

1.1 Motivation

1.1.1 Autonomous Underwater Vehicles

Over the last 10–20 years, a number of different AUVs have been developed, both commercially and academically. As AUV technology has matured, an increasing commercial demand has emerged, especially in the offshore sector (Wernli, 2000). Among the leading commercial AUVs today are the REMUS vehicles from Kongsberg Hydroid, the HUGIN vehicles from Kongsberg Maritime (Kongsberg Maritime AS, 2009) and the

Bluefin Robotics AUVs (Bluefin Robotics, 2010). In addition, a number of universities worldwide have also developed their own AUVs for academic purposes.

1.1.2 Underwater Navigation in General

The term 'navigation' has been used with several different meanings throughout history. Traditionally the word was used both for determining the whereabouts of ship a or vehicle and for the task of maneuvering a vehicle from a place A to a place B. In this thesis, the term has a narrower meaning, as is common in many research areas today. 'Navigation' is throughout the thesis understood as the process of estimating the position, velocity and attitude (i.e. the roll, pitch and heading angles) of a vehicle.

The principal problem in underwater navigation is the lack of GPS signals. As GPS signals are blocked by water, they are only available to the vehicle as long as its GPS antenna is above the water surface. Because of this, most advanced AUVs today are equipped with navigation systems based on inertial navigation. Even high-end inertial navigation systems will have an unacceptable drift in their position accuracy when no aiding sensors are used. For extensive underwater operations, aiding sensors are needed, providing the inertial navigation system with additional measurements. The most important aiding sensor in modern AUVs is the Doppler velocity log (DVL), an active acoustic sensor, which uses the Doppler shift of sound signals refracted from the sea floor to obtain bottom referenced measurements of the vehicle's velocity. Even with DVL aiding the inaccuracy in the inertial navigation system will become unacceptably high for extended submerged operations. As an example, modern HUGIN AUVs, which are equipped with a HG9900 Inertial Measurement Unit (IMU), typically has a drift of 0.1% of distance traveled when DVL aiding is used and the vehicle travels along a straight line. If the vehicle travels in a lawn-mower pattern, some of the DVL biases become observable and can be filtered out, resulting in a drift of 0.025% of the distance traveled. When the vehicle travels with its typical speed of 2 m/s second, this drift corresponds to an error drift of 2 m/h (Hagen et al., 2009). For operations longer than a few hours, the navigation system will therefore need additional aiding in order to maintain an acceptable accuracy.

The IMU provides the navigation system with inertial measurements, i.e. specific forces (acceleration pr mass unit) measured by the accelerometers, and angular rates, measured by the gyroscopes. These measurements are integrated to obtain position, velocity and attitude and fused with the aiding sensor measurements in a Kalman filter (see Section 2.4.3 of this thesis and the references therein). Figure 1.1 gives an overview of how this is done in the HUGIN navigation system. The figure is taken from Hagen et al. (2009). In this system, en error-state Kalman Filter is used for integration of the inertial measurements and the aiding sensors. The system is described in more detail in Jalving et al. (2003).

In addition to the DVL velocity updates, crucial aiding methods for underwater vehicle navigation include:

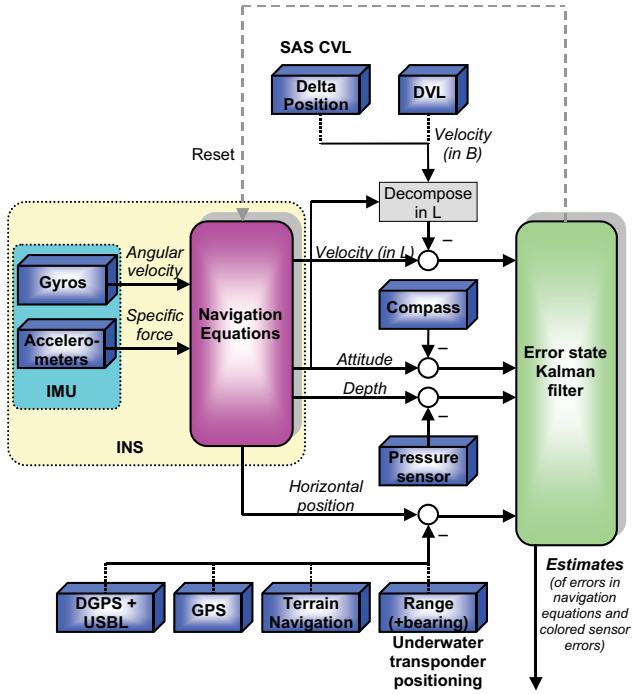


Figure 1.1: *Block diagram of the HUGIN integrated navigation system. Figure reprinted with permission from Hagen et al. (2009).*

- **GPS updates**: The vehicle is equipped with a GPS receiver for use when surfaced.
- **Acoustic aiding from mother ship**: A combination of ship GPS/Differential GPS (DGPS) or even RTK GPS (Real Time Kinematic GPS) and ultra-short baseline positioning (USBL) of the vehicle with respect to the ship. This requires that the mother-ship follows the vehicle during the operation.
- **Long-Baseline (LBL) positioning**: A network of transponders with known positions are placed on the sea floor, and the position of the vehicle is calculated using triangulation from acoustic measurements (Milne, 1983).
- **Hydrodynamic model aiding**: The dynamics of the vehicle as a function of fin deflections and propeller motion are calculated and used as measurements to the Kalman filter. This requires estimation of the sea current (Hegrenæs et al., 2008).
- **Underwater Transponder Positioning (UTP) / Synthetic Long Baseline**: A related method to LBL, in which range-only measurements are tightly integrated with the inertial system and fewer transponders are needed (Larsen, 2000; Hegrenæs et al., 2009).

Terrain navigation is an additional option for obtaining submerged position fixes, and the rationale behind using terrain navigation will be explained briefly in the next subsection. Reviews of the research area of underwater navigation, with more details on the various techniques described above, can be found in Kinsey et al. (2006) and Hagen et al. (2009).

1.1.3 Why Terrain Navigation?

All underwater vehicles carry sensors that enable them to explore the underwater environment. The quality of such sensors vary, from low-end acoustic altimeters to advanced sonar and camera systems. Modern AUVs are often used for terrain mapping, and they are consequently often equipped with advanced multibeam mapping sonars. Using terrain measurements for navigation purposes has been a well-known technique for years in air and land applications and the application of such methods also in the underwater environment has martured over the last decade or so, as will be shown later in Chapter 2.

The use and meaning of the term ‘terrain aided navigation’, in this thesis often shortened to ‘terrain navigation’, vary between authors. It includes at least two related, but different classes of methods:

- **Bathymetric terrain navigation**: The bathymetric measurements are used directly, and compared with a bathymetric map database to obtain a position estimate.
- **Feature-based terrain navigation**: Features on the sea floor, e.g. rocks, topographic features, wrecks etc. are extracted from the data and compared with a feature database to obtain a position estimate.

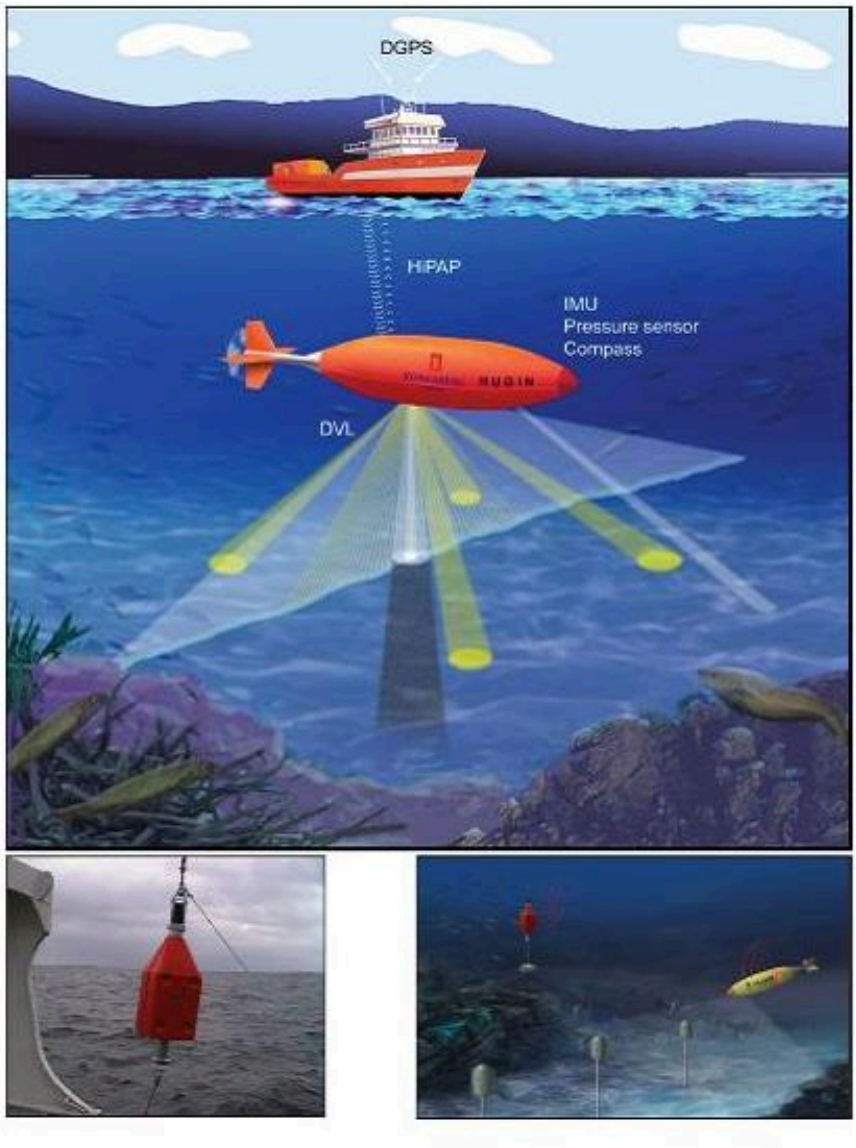


Figure 1.2: *Illustration of different aiding techniques used by the HUGIN AUV. Top: The vehicle is using acoustic aiding from a mother ship. Bottom left: UTP transponder. Bottom right: UTP aiding. Courtesy Kongsberg Maritime.*

This thesis focuses on bathymetric terrain navigation, and the term 'terrain navigation' will be used for this class only, unless otherwise stated explicitly. Other terms for the same class of methods are found in the literature, e.g. 'map based navigation', 'terrain relative navigation' or 'terrain correlation'.

The terrain navigation methods treated in this thesis require the existence of a prior terrain map database. Depending on the operation, such a map may or may not exist. The need for a prior map may seem more restrictive than it is. When a vehicle is equipped with a mapping sensor, it is possible for the vehicle to construct its own local map, which can be used for relative navigation when returning to the area later in the mission, using the methods described herein. Using the methods in this manner borders the vast research area of simultaneous localization and mapping (SLAM), which will be briefly reviewed in Section 2.9.

Terrain navigation, both bathymetric and feature-based, can also be said to belong to a wider class of navigation methods, together with related methods like gravitation based or geomagnetic based navigation. This class of navigation methods may be termed *geophysical navigation methods*.

1.2 Contributions

The contributions of this thesis to the research area of underwater navigation are the following:

- The comparison of the Terrain Contour Matching (TERCOM) and the Point Mass Filter (PMF) algorithms for underwater terrain aided navigation, using real AUV measurement data. The results from this work were first presented in Ånonsen et al. (2005) and appear here in Section 5.2.1.

- The comparison between different types of point mass and particle filters in 2D and 3D, with focus on depth bias estimation. Real AUV data were used, and the work was first presented in Ånonsen and Hallingstad (2006). The results appear here in Section 5.2.2, while the theoretical foundations of the depth bias estimation is discussed in Section 3.3.1.

- The development and real data testing of an extended drift model for terrain navigation. The development of the model is done in 3.2.1, while the results are given in Section 5.2.3. This work was first presented in Ånonsen et al. (2007).

- The application of the Sigma Point Kalman Filter for terrain navigation using real AUV data. This work was first presented in Ånonsen and Hallingstad (2007) and appears here in Section 5.2.4

- The simulations and discussions on the use of pockmarks for underwater terrain navigation in Section 5.2.5. This work was first published in Ånonsen and Hagen (2009).

1.3 Thesis Scope

In the early stages of this doctoral project, it was decided to keep the terrain navigation system as an external module, providing position updates for the inertial navigation system. In other words, the terrain navigation position estimates are treated as a position aiding sensor, in much the same manner as a GPS position update. Thus, the terrain navigation updates are loosely coupled with the inertial navigation system. There are several advantages with such an approach. First and foremost, it makes the system more portable, as it is not dependent on a specific implementation of a navigation system. The terrain navigation system can therefore easily be used on other platforms with different navigation systems. The navigation system need not be based on inertial navigation. Many low-cost systems are for example based on dead reckoning of velocity and compass measurements. The terrain navigation system outlined in this can easily be ported to such a system. In other systems, the user may not have full access to the raw measurements of the inertial system, prohibiting a tightly-coupled approach. Also, sticking to the loosely-coupled approach makes integrity and quality control of the terrain navigation updates easier. It is also safer from a system point of view, as fewer modifications of the main navigation system are needed in order to integrate the terrain navigation measurements.

However, as will be evident in later chapters, there are certain limitations to the loosely-coupled approach. Most importantly, it is difficult to model the drift of accuracy in the inertial sensors properly without access to the internal biases of the inertial system.

The scope of this work has been to make a best possible terrain navigation system based on the limitations above. The focus is on different estimation algorithms and their performance in real underwater scenarios. Topics related to integration of the terrain navigation estimates in the main navigation system, e.g. integrity control, convergence definitions etc. have not been given high priority, though they are briefly discussed in Section 3.4.

1.4 Outline of the Thesis

The outline of the thesis is as follows. Chapter 2 gives an overview of the basics of terrain aided navigation in general, i.e. the common problems of the concept in all possible environments. The main focus in this chapter is on the various Bayesian estimation methods that will be used later in the thesis. Most of the material here is a review of earlier work within the area, though the notation and presentation have been adopted to fit into this thesis.

In Chapter 3 the focus is on the particular problems that arise when terrain navigation is used in an underwater environment. First, the different sensors that can be used for underwater terrain navigation are described, before presenting the system and measurement models that will be used in the results part of the thesis.

Chapter 4 presents the principles of underwater mapping and the use of map databases

in the terrain navigation algorithms, while Chapter 5 contains all the computational results of the thesis. The focus is mainly on application of the various algorithms on real AUV data.

The thesis concludes with Chapter 6, in which the results are summarized and some suggestions for future research within the area are made.

Lists of acronyms, abbreviations and mathematical notation can be found in Appendix A.

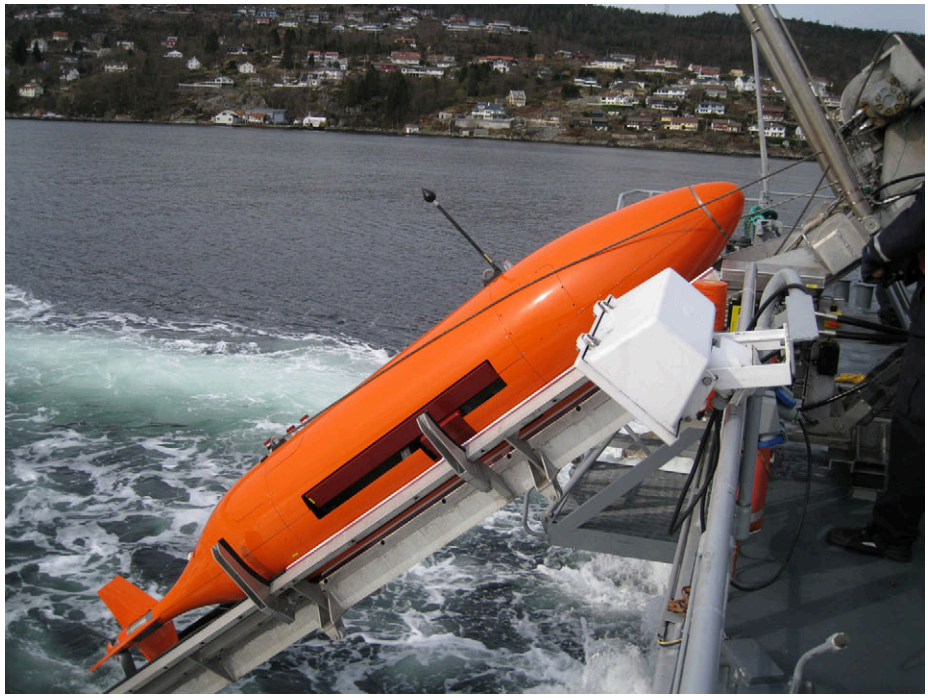


Figure 1.3: *Military version of HUGIN 1000, owned and operated by the Royal Norwegian Navy. The vehicle is equipped with a synthetic aperture sonar ideal for mine countermeasure operations. Picture courtesy of FFI.*

2

Terrain Navigation

TERRAIN aided navigation, or just terrain navigation for short, has been used for decades in air and land application, e.g. in aircraft and cruise missiles. This chapter gives an overview of the techniques and algorithms that have previously been used for terrain navigation.

2.1 Introduction

The idea behind terrain navigation is to use a set of terrain measurements to build up a terrain profile, and then find an optimal position estimate by comparing it to a prior map database. What is meant by 'optimal' varies from method to method. Terrain navigation can be carried out both in batch and recursively. In batch processing, all the measurements in the profile are processed at the same time, whereas in a recursive algorithm the estimate is being updated recursively as each measurement arrives. Many algorithms may be formulated both in batch and recursive form. The choice between batch and recursive algorithms depends on the situation and the sensors being used. For sensors giving multiple measurements at each measurement instant, like a multibeam echosounder (MBE) in an underwater application, batch processing is often natural.

The following overview of terrain navigation algorithms is similar to those given in Bergman (1999) and Mandt (2001), but with a few additional methods and considerations. All of the algorithms described here need an initial position estimate with some uncertainty measure. The initial position and its uncertainty are fed into the algorithms before running the terrain matching. The accuracy needed in the initial position is dependent on the method used and the terrain characteristics in the area. All terrain navigation algorithms require a certain degree of terrain variation to work. The degree of terrain variation necessary varies with the resolution of the terrain map and the accuracy of the

sensors.

2.2 Approaches to Terrain Navigation

Terrain navigation methods can generally be divided into *search area methods* and *gradient search methods*. Although this thesis mainly focuses on the search area methods, both groups are described shortly in the following sections.

2.2.1 Search Area Methods

The characteristic of the search area methods is that they make a search through the map, or parts of the map, to find the profile that best matches the measured terrain profile, in some sense. The first and simplest terrain navigation algorithm, the TERCOM algorithm, developed at the Wright-Patterson US air force base in the 1970s (Baker and Clem, 1977; Golden, 1980), belongs to this class of methods. The Bayesian terrain navigation methods in Bergman (1999), Bergman et al. (1999) and Nordlund (2002) also belong to the search area methods, in the sense that a search through the map is performed in order to find the optimal solution in the Bayesian sense. This thesis mainly focuses on this class of methods.

The concept of convergence is important for the search area methods and a convergence criterion of some kind must be defined. In a recursive terrain navigation algorithm, the estimates will typically have some fluctuations in the beginning of the matching procedure, before stabilizing around a position as more information is added through the measurement updates. A simple convergence criterion may be based on changes in the position estimate as new measurements are added; when no change has been observed over a predefined measurement interval, the method is said to have converged. The Bayesian methods, described in Section 2.4, have an intrinsic uncertainty measure in the form of an estimate covariance matrix, which also can be used as a convergence criterion. Notice that the converged estimate is not necessarily the correct position. Especially in self-similar terrain, multiple possible solutions may occur, as discussed in Henley (1990), Mandt (2001) and Nygren (2005).

2.2.2 Gradient-Based Methods

In the gradient-based methods, the local changes in the terrain near the initial position are modeled in some sense, often linearly. When a measurement is made, the estimated position is moved along the gradient in the direction indicated by the measurement, as shown in Figure 2.1. If the measured height/depth is lower than that in the map, the estimated position is moved iteratively towards larger depths, and vice versa. The gradient-based methods are often implemented in a Kalman filter, often tightly integrated with the INS system.

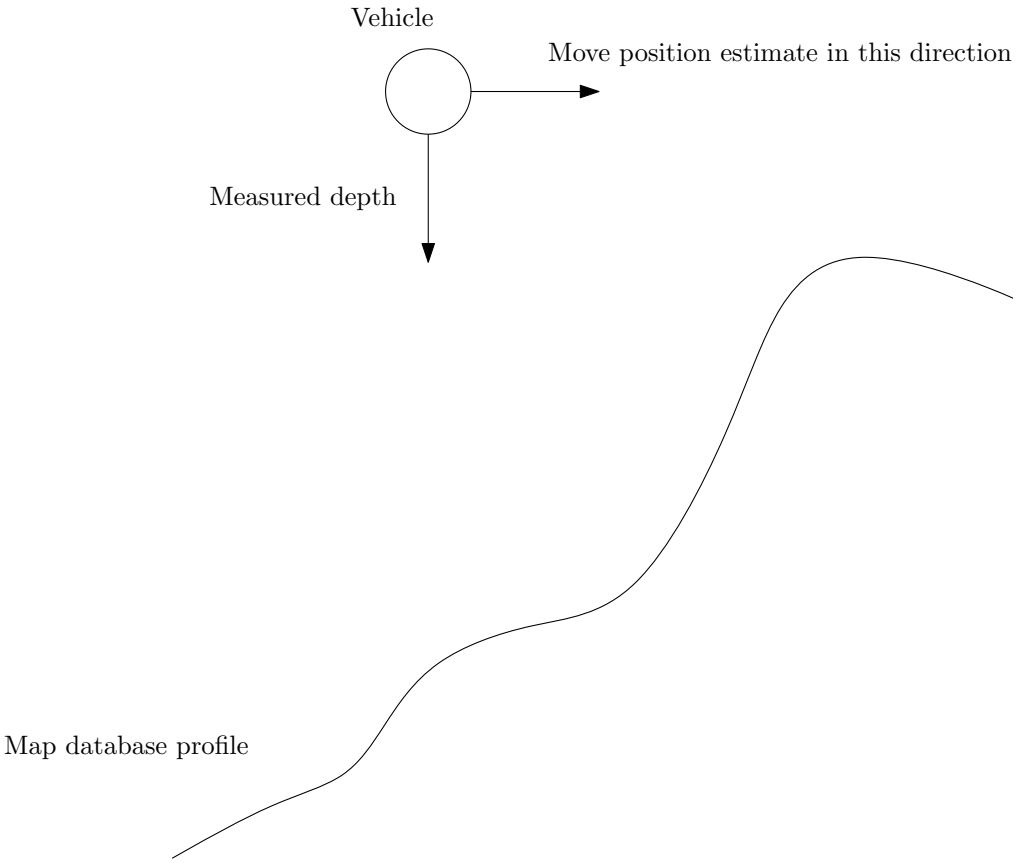


Figure 2.1: *Illustration of the gradient-based principle.*

An example of a gradient-based methods is the SITAN (Sandia Inertial Terrain Aided Navigation) algorithm (Hostetler, 1978). SITAN was the first recursive terrain navigation algorithm, and it is based on a modified Kalman filter that uses an adaptive stochastic linearization technique in an effort to overcome the effect on nonlinearities in the terrain. Several developments of the SITAN algorithms have been made, for example in Boozer and Fellerhoff (1988) and Hollowell (1990), where multiple Kalman filters arranged in a horizontal grid are used.

The main disadvantage with the gradient-based methods is that they require a much better initial position estimate than the search methods. If the initial uncertainty is too high, the gradient methods will diverge. An alternative would be to use some kind of hybrid method, where a search area method is used initially to limit the uncertainty. As the gradient methods do not perform any search operations, the computational load may be held quite low, in contrast to what is the case for the search area methods. Also, since the gradient methods use the Kalman filter of the navigation system, a tighter integration of the TerrNav algorithm and the inertial system is possible, e.g. when it comes to modeling of velocity errors, depth errors and so on. This is usually not possible in the search area methods, as will be discussed later.

This thesis focuses on the search area methods, and the gradient-based methods will not be pursued any further here.

2.2.3 Other Methods

Some other terrain navigation methods, which do not fit into the two main groups described above have also been developed. These may be combinations of the classes above or completely different methods. One example is the TERPROM system (British Aerospace Ltd., 1995). The TERPROM system has been used among others by US Air Force F-16 aircraft. It operates in two modes, an acquisition mode and a tracking mode, in concert with the INS system. In acquisition mode, the system builds up a series of terrain measurements and use them to establish a position estimate in a TERCOM-like manner. When an accurate position estimate has been established, the system enters tracking mode, in which the terrain is tracked in order to monitor the quality of the position estimate. If the quality of the estimate is poor, the system reenters acquisition mode in order to establish a better position fix. The TERPROM system also contains features for forward obstacle avoidance. Because of the military use of the system, it is not as well documented as other terrain navigation systems.

The VATAN algorithm (Viterbi Algorithm Terrain Aided Navigation) (Enns and Morrell, 1995) uses a version of the Viterbi algorithm (Viterbi, 1967; Forney, Jr., 1973). The VATAN algorithm uses a hidden Markov model (HMM) approach, see Rabiner (1989) for an introduction. Hidden Markov models and the Viterbi algorithm have been used in numerous applications within signal processing and pattern recognition. In a hidden Markov model, the estimated quantity, i.e. the vehicle position in the TerrNav case, is modeled as a discrete Markov chain. The transitions between the different states are modeled in a Markov transition matrix, which gives the probability of transition from

one state to another. The states are only accessible through measurements of a different quantity, i.e. the depth in our application, and they are thus 'hidden' from direct observation. Simulations in Enns and Morrell (1995) show that the VATAN method has good convergence properties, but it also has a tendency of bias errors in the estimates. Furthermore, implementation of the algorithm is quite complicated and it is computationally very demanding. As the VATAN method uses a Markov model, it is in some sense similar to the Bayesian methods described in Section 2.4, the difference being that the Bayesian methods utilize a spatially continuous state-space description of the problem.

2.3 The TERCOM Algorithm

The Terrain Contour Matching algorithm was the first terrain navigation algorithm that was described in the literature. It was developed at Wright-Patterson US Air Force base in the 1960s and 70s. Descriptions of the original TERCOM algorithm can be found in Baker and Clem (1977) and Golden (1980). TERCOM has been used mainly in cruise missiles, like in the Tomahawk missiles (Tsai, 1996), but also in aircraft (Uijt de Haag and Vadlamani, 2006).

As mentioned above, the TERCOM algorithm belongs to the class of search area methods. Upon initialization, a search area around the initial position estimate from the INS must be defined. It is crucial to the performance of any search area method that the true position be within this search area. The search area can for example be $\pm 3\sigma$ around the INS estimate. The algorithm has access to prior terrain data stored in a regular grid.

The original TERCOM algorithm is batch oriented, but it can also be formulated as a recursive algorithm. The algorithm implicitly assumes no position drift between the terrain measurements, and in its original form, no maneuvers are allowed during the matching process. The reason for this is that the original algorithm stored the prior terrain data in matrices that were aligned with the planned orientation of the missile or vehicle. Also, the grid spacing had to be consistent with the measurement frequency of the vehicle. By allowing interpolation of the map database, the terrain properties at an arbitrary position can be estimated, and these restrictions can be relaxed.

Several minimization criteria can be used to determine the 'best' position estimate in the TERCOM algorithm. The classic TERCOM criterion is the MAD (mean absolute difference). The search area is divided into a candidate grid $\{\boldsymbol{x}_l(i,j)\}_{i=1\ldots M, j=1\ldots N}$, where $\boldsymbol{x} \in \mathbb{R}^2$ represents a horizontal position candidate at time instant l. As the vehicle is moving while the measurements are being taken, the grid itself will also be moving in accordance with the vehicle displacements as measured by the INS. Let $\{z_l\}_{l=1\ldots k}$ represent a series of scalar terrain measurements. The MAD at position candidate (i,j) is then given as

$$\mathrm{MAD}_k(i,j) = \frac{1}{k}\sum_{l=1}^{k} |z_l - h(\boldsymbol{x}_l(i,j))|, \tag{2.1}$$

where $h(\cdot)$ denotes the terrain height given by the map data base. The correlation sum (2.1) is calculated for every candidate position $\boldsymbol{x}_k(i,j)$, and the TERCOM MAD estimate is the candidate that minimizes the correlation sum:

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= \boldsymbol{x}_k(i^*,j^*) : (i^*,j^*) = \arg\min_{i,j} \mathrm{MAD}_k(i,j) \\ &= \arg\min_{i,j} \frac{1}{k}\sum_{l=1}^{k} |z_l - h(\boldsymbol{x}_l(i,j))|. \end{aligned} \tag{2.2}$$

When plotting the correlation sum (2.1) for every candidate position, the so-called correlation surface appears. It is customary to plot the correlation surface with a negative sign, such that the highest point on the correlation surface corresponds to the position that gives the best match. One can also plot the inverse of the correlation sum, in order to have the peaks stand out more sharply. This is shown in Figure 2.2, which shows correlation surfaces from real AUV data, using a Doppler velocity log as a terrain navigation sensor. It is clearly seen how the main peak of the correlation surface gets as more information is incorporated into the correlation surface. After one measurement update, there are several peaks of almost the same height, whereas after 10 measurements three to four candidates seem to dominate. Finally, after 35 measurement updates one peak is clearly the highest, and this is indeed the correct vehicle position, though there is still one other peak with a high correlation sum.

Several modifications of the TERCOM method are possible. One example is to use a weighting factor for each measurement,

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= \boldsymbol{x}_k(i^*,j^*) : (i^*,j^*) = \arg\min_{i,j} \mathrm{MAD}_k(i,j) \\ &= \arg\min_{i,j} \frac{1}{k}\sum_{l=1}^{k} \beta_l |z_l - h(\boldsymbol{x}_l(i,j))|, \end{aligned} \tag{2.3}$$

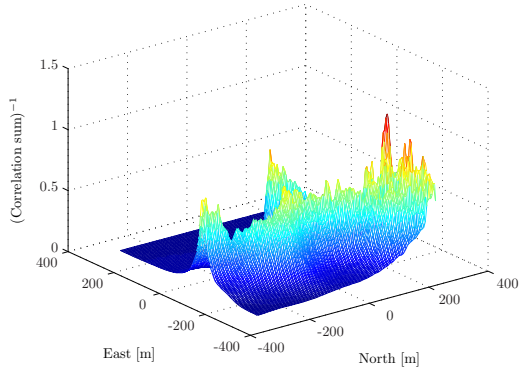
which might me useful if some of the measurements are known to be of better quality than others. It is also possible to use squared differences instead of the MAD criterion. The estimate is then given by

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_k(i^*,j^*) : (i^*,j^*) = \arg\min_{i,j} \frac{1}{k}\sum_{l=1}^{k} \big(z_l - h(\boldsymbol{x}_l(i,j))\big)^2. \tag{2.4}$$

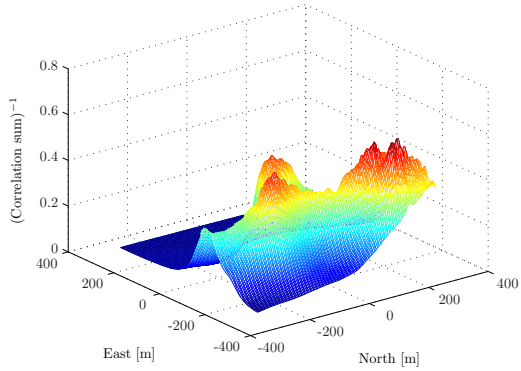
By using the squared differences as in (2.4), the method becomes more analytically tractable, at a slightly higher computational cost. The squared difference criterion is also more sensitive to measurement wild points than the MAD criterion.

Because the vehicle is moving as the measurement profile is taken, it is more convenient to formulate the position estimation problem in such a manner that the position offset from the INS system is estimated, instead of the position itself. If the INS position at time step k is denoted $\tilde{\boldsymbol{x}}_k$, the vehicle movement is given by

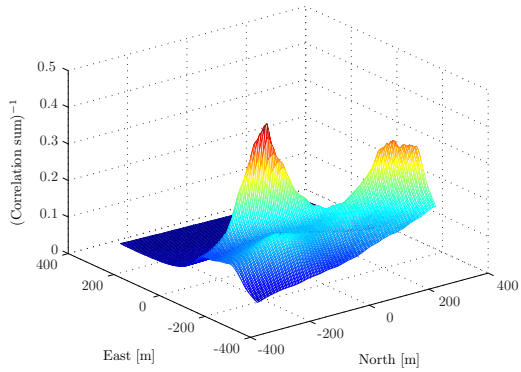
$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{x}}_k + \boldsymbol{u}_k, \tag{2.5}$$



(a) *After 1 measurement update.*



(b) *After 10 measurement updates.*



(c) *After 35 measurement updates.*

Figure 2.2: *TERCOM correlation surfaces.*

where $\boldsymbol{u}_k$ denotes the position change from time step k to $k+1$. Letting $\boldsymbol{x}_k$ be the true position at time step k, the position offset $\delta\boldsymbol{x}_k$ from the INS position is given by $\delta\boldsymbol{x}_k := \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k$. On the other hand, as the algorithm assumes no drift in the INS position, the true position at $k+1$ is also given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{u}_k. \tag{2.6}$$

Consequently, the motion equation (2.5) becomes

$$\delta\boldsymbol{x}_{k+1} := \boldsymbol{x}_{k+1} - \tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_k + \boldsymbol{u}_k - \tilde{\boldsymbol{x}}_k - \boldsymbol{u}_k = \delta\boldsymbol{x}_k, \tag{2.7}$$

so a fixed candidate grid $\{\delta\boldsymbol{x}(i,j)\}_{i=1\ldots M, j=1\ldots N}$ can be used. With this new grid, the TERCOM MAD criterion becomes

$$\mathrm{MAD}_k(i,j) = \frac{1}{k}\sum_{l=1}^{k} |z_l - h(\tilde{\boldsymbol{x}}_l + \delta\boldsymbol{x}_l(i,j))|, \tag{2.8}$$

with the correspoinding MAD estimate

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_k(i^*,j^*) : (i^*,j^*) = \arg\min_{i,j} \mathrm{MAD}_k(i,j) = \arg\min_{i,j} \frac{1}{k}\sum_{l=1}^{k} |z_l - h(\tilde{\boldsymbol{x}}_l + \delta\boldsymbol{x}_l(i,j))|. \tag{2.9}$$

The delta formulation used above is practical in all search area methods, since the grid does not have to be moved and re-centered at every time step. Mathematically the two formulations are identical, but conceptually and from an implementational point of view, the delta formulation is favorable, and similar formulations will be used for all the methods throughout this thesis.

Despite its simplicity, the TERCOM algorithm has proven quite useful and accurate in a number of applications, see for example Jalving et al. (2004b), Ånonsen et al. (2005), and Section 5 of this thesis. Its main disadvantage is that it has no intrinsic uncertainty measure, at least not in its original form, and the estimate is known to possess an oscillatory behavior in some situations, even after a relatively large number of measurements have been included.

When a terrain navigation algorithm is to be integrated with an INS, a covariance matrix for the position estimate is needed. As mentioned above, the TERCOM algorithm in its original form does not have such an uncertainty measure. One solution may be to use a fixed covariance matrix; once the algorithm has converged, it is assumed that the estimated covariance is at a certain predefined level, related to the map resolution, terrain type or other factors. The main difficulty with this approach would be to determine the fixed covariance properly. If the covariance is set to low, the integration Kalman filter treats the TERCOM position 'measurement' as very accurate, and a false TERCOM estimate may potentially damage the final estimate severely. On the other hand, if the

covariance is set too high, the information contained in the terrain navigation algorithm will not be utilized properly. Another, more sophisticated approach, outlined in Jalving et al. (2004b) would be to try to develop a covariance matrix based on the characteristics of the correlation surface. If the correlation surface has multiple high peaks, the covariance is high and vice versa. This approach can at the same time work as a convergence measure for the algorithm. However, one should bear in mind that the TERCOM correlation surface is not a probability density, and there is consequently not necessarily a direct correspondence between the surface and the estimate covariance matrix.

2.4 Bayesian Terrain Navigation

There are traditionally two approaches to estimation theory, the Fisher approach and the Bayesian approach (van Trees, 1968; Bar-Shalom et al., 2001). In the Fisher approach the quantity to be estimated is viewed as an unknown parameter, with no available prior information about the value of the parameter. The parameter value is estimated based solely on measurements of the quantity of interest. These measurements may be direct or indirect, and the statistics of the measurement noises are assumed known. The most common Fisher estimate is the Maximum likelihood (ML) estimate, which is dicussed in any introductory book on estimation, e.g. van Trees (1968).

Bayesian estimation differs from the Fisher approach in the sense that the quantity to be estimated is considered a random variable, with a known statistical distribution, known as the prior distribution. As measurements are taken, the goal is to fuse the prior information with the information from the measurements, utilizing the statistics of the measurement errors, to obtain what is known as the posterior density. This is done through the celebrated Bayes' formula for random variables (Bar-Shalom et al., 2001), which in the scalar case reads

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}, \tag{2.10}$$

where $p(x|z)$ is the posterior probability density function of the estimated stochastic variable x, conditioned on the measurement z. The likelihood $p(z|x)$ is taken from the measurement model, whereas $p(x)$ is the prior probability density of x. The denominator is the total probability of the measurement z, and this can be calculated through conditioning on x, in which case Bayes' formula (2.10) becomes

$$p(x|z) = \frac{p(z|x)p(x)}{\int_{\mathbb{R}} p(z|x)p(x)\,\mathrm{d}x}. \tag{2.11}$$

Note that the denominator in (2.11) is merely a constant once the measurement z is known. It is therefore sufficient to calculate the product $p(z|x)p(x)$ and do a simple normalization in order to get a proper posterior probability density.

When using Bayesian estimation in systems that change with time, a series of measurements is typically taken at different time instants, and the quantity to be estimated

also changes between these time instants. A natural framework for such a problem, is the state-space approach (Gelb, 1974; Maybeck, 1979), in which the process is modeled in a process equation, describing the dynamics of the problem, and a measurement equation, describing the measurements. Bayesian estimators in state-space systems can in many cases be formulated recursively and are therefore often known as recursive Bayesian estimators. In the case of linear process and measurement models and Gaussian error distributions, the celebrated Kalman filter (Kalman, 1960; Kalman and Bucy, 1961; Jazwinski, 1970) is the optimal solution in a minimum mean square error sense. The terrain navigation application, however, is an example of a highly non-linear estimation problem, to which the linear Kalman filter does not provide a satisfactory solution. In the following, the general optimal nonlinear recursive Bayesian filter will be presented. Though the optimal solution can be written in closed form, the integrals in the equations can generally not be solved analytically, and numerical approximation methods are needed. A number of different approximation methods will be described here, with the terrain navigation application in mind.

2.4.1 The Optimal Bayesian Filter Equations

Consider the general non-linear discrete time state-space model

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \qquad k = 0, 1, \ldots \tag{2.12}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{w}_k), \tag{2.13}$$

with states $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, known input $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$, measurement $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ initial distribution $\boldsymbol{x}_0 \sim p_{x_0}(\boldsymbol{x}_0)$ and error distributions $\boldsymbol{v}_k \sim p_{v_k}(\boldsymbol{v}_k)$, and $\boldsymbol{w}_k \sim p_{w_k}(\boldsymbol{w}_k)$. These initial state and noise vectors are also considered mutually statistically independent, both within an between time steps, such that the noise sequences are white, i.e. $E[\boldsymbol{x}_k \boldsymbol{x}_l^T]$ is a zero matrix whenever $k \neq l$. Note that nothing is assumed about the distributions; they may be of arbitrary type. However, it is often practical and sufficient in many situations to assume additive noise, such that the model becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k, \tag{2.14}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k. \tag{2.15}$$

The models (2.12) –(2.13) and (2.14) –(2.15) possess the important Markov property, see e.g. van Trees (1968) or Bar-Shalom et al. (2001), which means that entire state history is contained in the probability density function of the state $\boldsymbol{x}_k$ at time step k. This property can be formulated as

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots \boldsymbol{x}_0) = p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k), \tag{2.16}$$

i.e if the state at some time step k is known, the probability distribution and density function of the state at time step $k+1$ are independent of all the states prior to k. The

Markov property is a consequence of the function $\boldsymbol{f}_k$ being dependent on the state $\boldsymbol{x}_k$ and the input $\boldsymbol{u}_k$ only, and of the whiteness of the noises.

The *filtering problem* in Bayesian estimation is defined as the task of estimating the marginal density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$, where

$$\boldsymbol{Z}_k = \{\boldsymbol{z_0}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$$

denotes an aggregated measurement vector containing all the measurements up to and including time step k. Thanks to the Markov property of the problem, the filtering problem can, provided the prior distribution $p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{k-1})$ be known (analytically or approximately), be estimated using the process model and the most recent measurement only. The filtering problem has its counterpart in the *full estimation problem* or *smoothing problem* which estimates the density $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$, i.e. each time a measurement comes in, the full history

$$\boldsymbol{X}_k = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$$

of the system is reestimated. The smoothing problem is much more complex than the filtering problem. In real-time situations, e.g. during the operation of an underwater vehicle, the filtering density is normally of more interest, as one cannot remake the decisions already made in the past. However, when the position accuracy is critical, e.g. in underwater mapping, all available information should be utilized in order to make the map as accurate as possible, and consequently one is more interested in the full estimation problem. In this thesis, terrain navigation is mostly viewed as a real-time aiding to the INS system, and consequently most effort is put into the filtering problem.

Assume that the posterior density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ at time step k is known. The time update of the system can be expressed by using that

$$\begin{aligned} p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k|\boldsymbol{Z}_k) &= p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{Z}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k) \\ &= p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k), \end{aligned} \tag{2.17}$$

which follows from the Markovian property of the system. By marginalizing out the state $\boldsymbol{x}_k$, the expression for the Bayesian time update becomes:

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k. \tag{2.18}$$

Equation (2.18) is known as the Chapman-Kolmogorov equation (Jazwinski, 1970), and it is the discrete counterpart of the Fokker-Planck equation, which will be discussed later in Section 2.7.

The measurement update is found by applying Bayes' formula (2.10) to the last measurement $\boldsymbol{z}_{k+1}$ in the measurement series $\boldsymbol{Z}_{k+1}$,

$$
\begin{aligned}
p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_{k+1}) &= \frac{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}, \boldsymbol{Z}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)}{p(\boldsymbol{z}_{k+1}|\boldsymbol{Z}_k)} \\
&= \frac{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)}{p(\boldsymbol{z}_{k+1}|\boldsymbol{Z}_k)} \\
&= \frac{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)}{\int_{\mathbb{R}^{n_x}} p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_{k+1}},
\end{aligned} \tag{2.19}
$$

where the second equality follows from the Markovian property, and the third equality uses the law of total probability in the denominator, as in (2.11).

The two equations (2.18) and (2.19) constitute the optimal recursive Bayesian filter. Given the initial probability density function $p(\boldsymbol{x}_0)$, the likelihood function $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$, the time update kernel $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$, and the measurement sequence $\boldsymbol{Z}_k$, the posterior density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ can in principle be calculated, using (2.18) and (2.19) recursively. However, the integrals in these equation can only in simple cases be calculated analytically, and in the general case approximation methods are needed. There are generally two kinds of such methods, local and global approximation methods. In the local methods, a local approximation around a point estimate of the states is done, in order to obtain a closed form expression for the estimate. The most common example is the extended Kalman filter (EKF), which makes a local linearization of the model around the current estimate. If the problem is highly nonlinear, which is often the case for the terrain navigation problem, due to the highly unstructured an nonlinear nature of the terrain database, the EKF has proven not to work well in many cases, see e.g. Mandt (2001). An alternative to local approximation methods are the global approximation methods, which try to estimate the complete posterior probability density. The point mass filter (PMF) (Bucy and Senne, 1971; Bergman, 1999) uses a grid approach for the approximation of the posterior. Particle filters (sequential Monte Carlo filters), see e.g. Gordon et al. (1993), Andrieu et al. (2001) and Crisan and Doucet (2002), are another approach, in which the posterior is estimated using a set of particles, sampled from the underlying distributions.

When the posterior density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ or an approximate density is known, an estimate of the states can readily be found. In principle, the choice of estimate is based on a cost function $C(\boldsymbol{x}_k, \hat{\boldsymbol{x}}_k)$, as described in van Trees (1968). The cost function reflects the cost of making an error in the state estimation, and the most common is the square error cost function

$$
C_{\mathrm{SE}}(\boldsymbol{x}_k, \hat{\boldsymbol{x}}_k) = \|\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k\|^2, \tag{2.20}
$$

where the norm is usually the common *Euclidian norm* or *2-norm*,

$$
\|\boldsymbol{v}\|^2 = \sum_i^n v_i^2, \tag{2.21}
$$

for $\boldsymbol{v} \in \mathbb{R}^n$. Minimization of the expectation of the cost function leads to the minimum

mean square error (MMSE) estimate, which can be shown (see e.g.van Trees (1968)) to coincide with the mean of the posterior density:

$$\hat{\boldsymbol{x}}_k^{\mathrm{MMSE}} = E[\boldsymbol{x}_k|\boldsymbol{Z}_k] = \int_{\mathbb{R}^{n_x}} \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k, \tag{2.22}$$

with the covariance matrix of the estimate given by

$$\begin{aligned}\hat{\boldsymbol{P}}_k^{\mathrm{MMSE}} &= E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{\mathrm{MMSE}})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{\mathrm{MMSE}})^T|\boldsymbol{Z}_k] \\ &= \int_{\mathbb{R}^{n_x}} (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{\mathrm{MMSE}})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{\mathrm{MMSE}})^T p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k. \end{aligned} \tag{2.23}$$

The MMSE estimate is often a useful estimate when dealing with unimodal posterior densities. In the terrain navigation case, however, the posterior is often multimodal, and in such cases the MMSE may not be a good estimate. For example, if the posterior has two distinct peaks, the MMSE estimate will be located somewhere between the peaks, in an area where the probability of the vehicle position estimate being correct is low, or even zero.

An alternative to the MMSE estimate, which may be a better choice when the posterior is multimodal, is the maximum a posteriori (MAP) estimate, which is simply the value of the state vector that maximizes the posterior

$$\hat{\boldsymbol{x}}_k^{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}_k} p(\boldsymbol{x}_k|\boldsymbol{Z}_k). \tag{2.24}$$

The MAP estimate will always be located at one of the grid nodes if a grid method is used, or it will coincide with one of the particles in a PF.

The choice of estimate type may be a difficult task in a real terrain navigation system. What estimate that is the most favorable may vary with sensors and terrain types. In many cases, however, the MMSE and MAP estimates will coincide, especially in terrain suited for terrain navigation. Other estimates are also used, based on other types of cost functions (van Trees, 1968). One example is the median estimate.

2.4.2 The State-Space Model for Terrain Navigation

The state-space model for terrain navigation will here be presented. In a real problem, there will usually be a discrepancy between the system to be estimated and the model that is used in the filters. This is discussed e.g. in Gelb (1974) and Bar-Shalom et al. (2001). The real system is usually so complex that is impossible to model all its properties in a finite state filter. In the terrain navigation application this is especially true, due to the computational complexity of the point mass and particle filters. In order to keep the computational load at a reasonable level, the number of states in these filters must be kept very low, usually as low as 2-3 states in the point mass filters, whereas the particle filters usually can handle a few more states.

The discrepancy between the true system and the filter model is a major source of error in the terrain navigation estimates. In the following presentation, the distinction between the true system and the filter model will be made explicit. Generally one speaks of three possibly different systems; the *physical system*, the *system model* and the filter model. The *physical system* is generally infinite-dimensional. The physical system is modeled in the *truth model*, or *system model*, and this is the most accurate model of the real system that is available. Finally, the *filter model* is the model that is implemented in the estimator, and this model can in some cases be significantly different from the truth model.

Truth Model

Consider the following generic truth model for the motion of the vehicle:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k + \boldsymbol{u}_k + \boldsymbol{v}'_k, \tag{2.25}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{g}(\boldsymbol{v}_k) + \boldsymbol{v}''_k, \tag{2.26}$$

where $\boldsymbol{x}_k = (x_{N,k}, x_{E,k})^T$ is the horizontal AUV position vector (north, east) decomposed in the $\{e\}$-frame, $\boldsymbol{u}_k$ is the position change from time step k to $k+1$, calculated from the inertial navigation system, and $\boldsymbol{v}'_k$ and $\boldsymbol{v}''_k$ are white noise sequences. Equation (2.26) models the strongly correlated error propagation of the inertial navigation system. Note that neither the dimension of $\boldsymbol{v}_k$ nor the function $\boldsymbol{g}(\cdot)$ are specified, so this generic model can be made arbitrarily complex.

The system measurement equation reads

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{2.27}$$

where the function $\boldsymbol{h}(\boldsymbol{x}_k)$ denotes the true total sea depth (or altitude, for an aircraft or missile) at vehicle position $\boldsymbol{x}_k$ and $\boldsymbol{w}_k$ is the corresponding noise. For a single-measurement sensor, like a radar altimeter on an aircraft or a single beam echo sounder on an underwater vehicle, the depth measurement is scalar. In most cases that will be treated in this thesis, the measurements are vector measurements, for example from a multibeam echo sounder (MBE). The MBE measures the depth at a number of points arranged in a fan below the vehicle. In this case, the function $\boldsymbol{h}(\boldsymbol{x}_k)$ also has to take care of the positions of the measurement points on the sea floor relative to the vehicle position, often referred to as the sensor footprint. The function is thus a mapping from $\mathbb{R}^2$ to $\mathbb{R}^{N_b}$, where N_b again denotes the number of measurement beams. In the MBE case, the function $\boldsymbol{h}(\boldsymbol{x}_k)$ also depends on the attitude of the vehicle, in order to determine the right orientation of the footprint on the sea floor. With a slight abuse of notation, this dependency will not be explicitly included in the argument list of the $\boldsymbol{h}$ function, as the attitude will not be part of the state vector in this thesis.

As mentioned previously, it is convenient to formulate the terrain navigation problem as an estimation of the position offset $\delta\boldsymbol{x}_k$ from the INS calculated position $\tilde{\boldsymbol{x}}_k$. By

defining

$$\delta \boldsymbol{x}_k = \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k, \tag{2.28}$$

the system model can be written in the form

$$\begin{aligned}
\delta \boldsymbol{x}_{k+1} &= \boldsymbol{x}_{k+1} - \tilde{\boldsymbol{x}}_{k+1} \\
&= \boldsymbol{x}_k + \boldsymbol{u}_k + \boldsymbol{v}'_k + \boldsymbol{v}_k - \tilde{\boldsymbol{x}}_k - \boldsymbol{u}_k \\
&= \delta \boldsymbol{x}_k + \boldsymbol{v}'_k + \boldsymbol{v}_k, && (2.29) \\
\boldsymbol{v}_{k+1} &= \boldsymbol{g}(\boldsymbol{v}_k) + \boldsymbol{v}''_k, && (2.30) \\
\boldsymbol{z}_k &= \boldsymbol{h}(\tilde{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k) + \boldsymbol{w}_k. && (2.31)
\end{aligned}$$

Filter Model

As mentioned above, the system model (2.25)–(2.27) must be simplified before the Bayesian methods can be used. In most earlier work on terrain navigation, like in Bergman (1999) and Nygren (2005), a two-dimensional filter model is used. Using asterisks for variables and functions in the filter model, in order to distinguish it from the system model, the conventional filter model reads

$$\begin{aligned}
\boldsymbol{x}^*_{k+1} &= \boldsymbol{x}^*_k + \boldsymbol{u}_k + \boldsymbol{v}^*_k && (2.32) \\
\boldsymbol{z}_k &= \boldsymbol{h}^*(\boldsymbol{x}^*_k) + \boldsymbol{w}^*_k, && (2.33)
\end{aligned}$$

where the noise sequences are assumed zero mean and white, i.e.

$$\begin{aligned}
E[\boldsymbol{v}^*_k \boldsymbol{v}^{*T}_l] &= \boldsymbol{Q}^*_k \delta_{kl}, && (2.34) \\
E[\boldsymbol{w}^*_k \boldsymbol{w}^{*T}_l] &= \boldsymbol{R}^*_k \delta_{kl}, && (2.35)
\end{aligned}$$

where $\boldsymbol{Q}^*_k$ and $\boldsymbol{R}^*_k$ are the process and measurement noise covariance matrices, which may differ from the corresponding truth model covariance matrices $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$, and δ_{kl} is the Kronecker delta,

$$\delta_{kl} = \begin{cases} 1, & \text{if } k = l, \\ 0, & \text{if } k \neq l. \end{cases} \tag{2.36}$$

This implies that the process and measurement noises are uncorrelated between time steps, i.e. they are white noise sequences. It is also assumed that the noise sequences and the initial state $\boldsymbol{x}_0$ are mutually uncorrelated. A convenient, but not necessary, assumption is to assume Gaussian noises and initial distributions, i.e.

$$p_{x_0^*}(\boldsymbol{x}_0) = \mathcal{N}(\tilde{\boldsymbol{x}}_0, \boldsymbol{P}_0^*) \tag{2.37}$$

$$p_{v_k^*}(\boldsymbol{v}_k^*) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k^*), \tag{2.38}$$

$$p_{w_k^*}(\boldsymbol{w}_k^*) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k^*). \tag{2.39}$$

Since the noises are the results of averaging over a number of different, unknown stochastic processes, the Gaussian assumption is justifiable and will be used in most cases in this thesis. However, since the non-linear Bayesian methods that will be used for the estimation of these systems are able to handle non-Gaussian cases, this assumption is not necessary. The non-linear Bayesian estimation techniques were in fact developed to be able to handle non-Gaussian distributions.

The filter model (2.32)–(2.33) has been simplified in several ways, compared to the system model (2.25)–(2.27). Most importantly, the colored noise sequence $\boldsymbol{v}_k$ has been replaced by a white noise sequence $\boldsymbol{v}_k^*$. This is an oversimplification of the dynamics of the INS system, which may lead to unfavorable results, like filter divergence. This will be further discussed in Chapter 3.

The function $\boldsymbol{h}^*(\cdot)$ represents the digital terrain map. In this thesis, gridded terrain maps will be used, using some kind of interpolation between the grid nodes. Consequently, the measurement error $\boldsymbol{w}_k^*$ must contain contributions both from map errors, including interpolation errors, and from sensor errors. An error analysis of terrain maps is presented in Chapter 4. Terrain map interpolation is discussed in Section 4.3.

As previously discussed, it is often convenient to formulate the problem as an estimation of the offset from the INS position. By defining

$$\delta\boldsymbol{x}_k^* = \boldsymbol{x}_k^* - \tilde{\boldsymbol{x}}_k, \tag{2.40}$$

the filter model becomes

$$\delta\boldsymbol{x}_{k+1}^* = \delta\boldsymbol{x}_k^* + \boldsymbol{v}_k^* \tag{2.41}$$

$$\boldsymbol{z}_k = \boldsymbol{h}^*(\tilde{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k^*) + \boldsymbol{w}_k^*, \tag{2.42}$$

with the same assumptions as before. Using this filter model the Bayesian measurement update equation becomes

$$\begin{aligned} p(\delta\boldsymbol{x}_k^*|\boldsymbol{Z}_k) &= \frac{p(\boldsymbol{z}_k|\delta\boldsymbol{x}_k^*, \boldsymbol{Z}_{k-1})p(\delta\boldsymbol{x}_k^*|\boldsymbol{Z}_{k-1})}{p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})} \\ &= \alpha_k^{-1} p_{w_k^*}(\boldsymbol{z}_k - \boldsymbol{h}^*(\tilde{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k^*))p(\delta\boldsymbol{x}_k^*|\boldsymbol{Z}_{k-1}) \end{aligned} \tag{2.43}$$

where

$$\alpha_k = \int_{\mathbb{R}^2} \left[p_{w_k^*}(\boldsymbol{z}_k - \boldsymbol{h}^*(\tilde{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k^*))p(\delta\boldsymbol{x}_k^*|\boldsymbol{Z}_{k-1})\right] \mathrm{d}\delta\boldsymbol{x}_k^*.$$

Similarly, the time update equation now reads

$$\begin{aligned} p(\delta \boldsymbol{x}_{k+1}^* | \boldsymbol{Z}_k) &= \int_{\mathbb{R}^2} p(\delta \boldsymbol{x}_{k+1}^*, \delta \boldsymbol{x}_k^* | \boldsymbol{Z}_k) \, \mathrm{d}\delta \boldsymbol{x}_k^* \\ &= \int_{\mathbb{R}^2} p(\delta \boldsymbol{x}_{k+1}^* | \delta \boldsymbol{x}_k^*, \boldsymbol{Z}_k) p(\delta \boldsymbol{x}_k^* | \boldsymbol{Z}_k) \, \mathrm{d}\delta \boldsymbol{x}_k^* \\ &= \int_{\mathbb{R}^2} p_{v_k^*}(\delta \boldsymbol{x}_{k+1}^* - \delta \boldsymbol{x}_k^*) p(\delta \boldsymbol{x}_k^* | \boldsymbol{Z}_k) \, \mathrm{d}\delta \boldsymbol{x}_k^*. \end{aligned} \tag{2.44}$$

2.4.3 The Extended Kalman Filter

As mentioned above, the Kalman filter is the optimal minimum variance Bayesian estimator in the linear, Gaussian case. The Extended Kalman filter (EKF) is the most straightforward way to solve a nonlinear estimation problem. It is based on a local linearization of the nonlinearities in the process and measurement equations around the previously estimated states, whereas the Gaussian approximation of the noise distributions is kept, which makes the EKF less suitable for cases where the noises are far from Gaussian.

Like the linear KF, the EKF also represents the posterior solution by its two first moments, the mean and the covariance matrix. Gaussian random variables are completely characterized by their first two moments, since their probability density function can be written based on these parameters only. Furthermore, when Gaussian variables are passed through linear dynamic systems, the output is still a Gaussian, see e.g. van Trees (1968). However, this is not the case for non-linear systems; even if the noises are Gaussian, the posterior density will not be Gaussian after the states have been passed through the nonlinearities of the system. Consequently, the EKF is a suboptimal filter. Also, for strong nonlinear functions, the local linearization may not give a sufficiently accurate description of the local behavior of the system. For example, if the second order terms, which are assumed small in the EKF, can not be neglected, this will lead to inaccuracies in the EKF. One possible solution to this problem is to use a higher order filter, taking also the higher order terms into account. Iteration techniques also exist, in which the system is re-linearized several times at each update, to obtain a better linearization trajectory. Thorough discussions on the extended Kalman filter and its variations are given in e.g. Jazwinski (1970), Gelb (1974), and Bar-Shalom et al. (2001).

Despite its suboptimality, the EKF has been used successfully in a number of applications, e.g. in control systems, signal analysis and target tracking. The EFK is also the basis of any aided inertial navigation system (Savage, 1998a,b; Bar-Shalom et al., 2001). The EKF can, in principle, be used for estimation of the states in the terrain navigation filter model (2.32)– (2.33). However, EKF requires the evaluation of the measurement Jacobian matrix

$$\boldsymbol{H}^*(\boldsymbol{x}) = \frac{\partial \boldsymbol{h}^*(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial h_1^*(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial h_1^*(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m^*(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial h_m^*(\boldsymbol{x})}{\partial x_n} \end{bmatrix}, \tag{2.45}$$

which may be difficult to compute. Since $\boldsymbol{h}^*(\cdot)$ is not an analytical function, but a map database representing the real terrain, the partial derivatives have to be computed using some kind of numerical differentiation. Using the fact that the map database is already defined on a grid, finite differences (Iserles, 1996) is a natural choice for the numerical derivatives. The quality of these approximations may however be low, because of the relatively low resolution of the maps and the unstructured and nonlinear nature of most real terrain types.

Even though the measurement Jacobian can be calculated, the first order approximation of the measurement equation will often not be sufficiently accurate in many situations, and the EKF is therefore in many cases not well suited for terrain navigation. It should be noted, however, that several examples of EKF-based terrain navigation exist in the literature, in which different approaches are used in order to counter the linearization problems. One example is the SITAN approach mentioned in Section 2.2.2. Other examples are Bergem (1993) and Di Massa (1997), which both discuss underwater applications. Bergem (1993) defines a matching strength function that is computed for all positions within a validation gate, and the best match is used in a Kalman filter. Several simple process models in the KF are tested. Results from multibeam echo sounder data are presented. Di Massa (1997) uses a sophisticated coarse-to-fine search method in search areas of different resolution, in order to be able to handle large search areas with a large number of potential matches. Several matches are used simultaneously in a Kalman filter using a probabilistic data association (PDAF) approach (Bar-Shalom et al., 2001).

2.4.4 The Point Mass Filter

The idea of using grid based methods for approximating the probability densities in Bayesian estimation is quite old and was introduced by Bucy and Senne (1971). However, since the approach is extremely computationally demanding, it is not until recent years that modern computer technology has made it possible to implement grid based methods in practical situations. In Bergman (1997) and Bergman (1999), the point mass filter (PMF) was used for terrain navigation of aircraft.

Point Mass Approximation

Before defining the point mass grid a search area around the position estimate $\tilde{\boldsymbol{x}}_k$ from the INS has to be specified. The size of the search area can for example be determined by the uncertainty in the initial INS position, e.g. 3σ in each direction. The algorithm is started by discretization of the two-dimensional search area into a grid with M and N

grid points in each direction. At each grid point the density $p(\delta \boldsymbol{x}_k^* | \boldsymbol{Z}_k)$ is approximated by a probability mass weight

$$p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_k) \quad i = 1, \ldots, M,\ j = 1, \ldots, N. \tag{2.46}$$

Consider a uniform grid with grid resolution Δ in both directions. Let

$$p(\delta \boldsymbol{x}_0^*(i,j) | \boldsymbol{Z}_0) = p(\delta \boldsymbol{x}_0^*(i,j)) = \alpha_0^{-1} p(\delta \boldsymbol{x}_0^*)|_{\delta \boldsymbol{x}_0^* = \delta \boldsymbol{x}_0^*(i,j)}, \tag{2.47}$$

where

$$\alpha_0 = \sum_{m=1}^{M} \sum_{n=1}^{N} p(\delta \boldsymbol{x}_0^*)|_{\delta \boldsymbol{x}_0^* = \delta \boldsymbol{x}_0^*(i,j)} \Delta^2. \tag{2.48}$$

Notice the difference between $p(\delta \boldsymbol{x}_0^*(i,j))$, which is a point-wise probability mass approximation and $p(\delta \boldsymbol{x}_0^*)$, which is a continuous probability density function. Also notice that

$$\sum_{i=1}^{M} \sum_{j=1}^{N} p(\delta \boldsymbol{x}_0^*(i,j) | \boldsymbol{Z}_0) \Delta^2 = 1, \tag{2.49}$$

which is required for a true probability mass approximation. Then, for each time step, the measurement update is performed according to

$$p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_k) = \alpha_k^{-1} p_{w_k^*}(\boldsymbol{z}_k - \boldsymbol{h}^*(\tilde{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k^*(i,j))) p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_{k-1}), \tag{2.50}$$

$$\alpha_k = \sum_{i=1}^{M} \sum_{j=1}^{N} p_{w_k^*}(\boldsymbol{z}_k - \boldsymbol{h}^*(\tilde{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k^*(i,j))) p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_{k-1}) \Delta^2.$$

The MMSE estimate of the position offset can now be computed from

$$\widehat{\delta \boldsymbol{x}}_k^{\text{MMSE}} = \sum_{i=1}^{M} \sum_{j=1}^{N} \delta \boldsymbol{x}_k^*(i,j) p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_k) \Delta^2, \tag{2.51}$$

with covariance matrix given by

$$\hat{\boldsymbol{P}}_k^{\text{MMSE}} = \sum_{i=1}^{M} \sum_{j=1}^{N} (\delta \boldsymbol{x}_k^*(i,j) - \widehat{\delta \boldsymbol{x}}_k^{\text{MMSE}})(\delta \boldsymbol{x}_k^*(i,j) - \widehat{\delta \boldsymbol{x}}_k^{\text{MMSE}})^T \cdot p(\delta \boldsymbol{x}_k^*(i,j) | \boldsymbol{Z}_k) \Delta^2, \tag{2.52}$$

and the full vehicle position estimate is given by

$$\hat{\boldsymbol{x}}_k^{\text{MMSE}} = \tilde{\boldsymbol{x}}_k + \widehat{\delta \boldsymbol{x}}_k^{\text{MMSE}}. \tag{2.53}$$

Since the point mass approximation represents the full posterior probability density, any other type of estimate can also readily be calculated, for example the maximum a posterior (MAP) estimate, given by

$$\widehat{\delta \boldsymbol{x}}_k^{\mathrm{MAP}} = \arg\max_{i,j} p(\delta \boldsymbol{x}_k^*(i,j)|\boldsymbol{Z}_k). \tag{2.54}$$

The time update step in the PMF is calculated as

$$\begin{aligned} &p(\delta \boldsymbol{x}_{k+1}^*(i,j)|\boldsymbol{Z}_k) \\ &\quad = \sum_{m=1}^{M}\sum_{n=1}^{N} p_{\boldsymbol{v}_k^*}(\delta \boldsymbol{x}_{k+1}^*(i,j) - \delta \boldsymbol{x}_k^*(m,n)) p(\delta \boldsymbol{x}_k^*(m,n)|\boldsymbol{Z}_k)\Delta^2. \end{aligned} \tag{2.55}$$

Equation (2.55) can be viewed as a two-dimensional convolution, and it is the single most computationally demanding operation in the PMF. Bergman (1999) shows that, under the assumption that the expected position drift is equal in both grid directions and uncorrelated between the directions, this two-dimensional convolution can be substantially simplified. However, in the underwater case it is a known fact that the position drift of a Doppler velocity aided INS is different along and across the vehicle body coordinate system (Jalving et al., 2004a), so the validity of the equal drift assumption may be disputed. This is further pursued in Chapter 3 of this thesis.

Implementational Aspects

The two-dimensional point mass filter can be implemented quite easily in MATLAB™, storing the point masses in a matrix and using matrix operations for carrying out the time update convolution (2.55) and measurement update (2.50). However, as pointed out in Bergman (1999), many of the point masses will be close to zero and their contributions to the time and measurement updates will be negligible. Consequently it is possible to implement a version of the point mass filter with an adaptive grid, using sparse matrices in MATLAB™. By utilizing sparse matrix operations in MATLAB™, only the point masses with a nonnegilible contribution are used in the update equations. After every measurement update, grid points below a certain threshold value are removed from the grid. The threshold can be defined as a certain proportion of the average point mass value.

2.4.5 Particle Filters

In the PMF and other grid based methods, the integrals under consideration, i.e. the Bayesian time and measurement updates (2.19)–(2.18) and the calculation of the (MMSE) estimate (2.22) and its covariance (2.23), are calculated numerically using a deterministic grid. In particle filters (PF) the integrals are instead approximated using Monte Carlo integration, in which the integrand is calculated at a number of stochastically chosen grid

points. In the following, a brief description of Monte Carlo integration and particularly importance sampling is given, before moving on to a description of the class of recursive estimation algorithms known as particle filters or Sequential Monte Carlo Filters (SMCF).

Monte Carlo Integration

Numerical integration deals with the problem of numerically approximating a general integral

$$I = \int_{\Omega} g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{2.56}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$, $\Omega \subset \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is an arbitrary, scalar-valued function. The reason for evaluating the integral numerically may be that is either has no analytical solution or it is impractical to implement the exact solution. Monte Carlo integration deals with the special case when the integral can be written as

$$I = \int_{\Omega} f(\boldsymbol{x}) \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{2.57}$$

where the scalar-valued function $\pi(\boldsymbol{x}) \geq 0$ in Ω and integrates to unity:

$$\int_{\Omega} \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = 1.$$

Note that the familiar expectation and variance integrals for a scalar random variable are special cases of (2.57), with $\pi(\boldsymbol{x})$ taking the form of a probability density function. The same is true for the vector and matrix components of the mean vector and covariance matrix of a random vector, so the theory can be readily used for both scalar random variables as well as random vectors.

Monte Carlo approximation of (2.57) is based on the assumption that it is possible to generate $N >> 1$ samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ from the distribution $\pi(\boldsymbol{x})$. The integral is then approximated by taking the average over the set of samples

$$I_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i). \tag{2.58}$$

It can be shown (Andrieu et al., 2001) that as $N \rightarrow \infty$, the approximation I_N will converge almost surely to I by the strong law of large numbers, and that if all the samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ are independent, the approximation is unbiased. Also, provided the variance of $f(\boldsymbol{x})$ be finite,

$$\sigma^2 = \int_{\Omega} \left(f(\boldsymbol{x}) - I \right)^2 \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\Omega} f^2(\boldsymbol{x}) \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - I^2 < \infty, \tag{2.59}$$

the central limit theorem yields that the error $I_N - I$ converges in distribution to a Gaussian distribution,

$$\lim_{N\to\infty} \sqrt{N}(I_N - I) \sim N(0, \sigma^2). \tag{2.60}$$

Definitions of terms related to measure and probability theory, such as almost sure convergence and convergence in distribution can be found in Folland (1999).

An interesting aspect of (2.60) is the fact that the error in the numerical integration is of order $N^{-1/2}$, i.e.

$$I_N - I = \mathcal{O}(N^{-1/2}), \tag{2.61}$$

which means that the error is independent of the dimension, n_x, of the sample space. One should remember, however, that this is an asymptotic result, and for a practical situation one must of course resort to a finite N. For such cases, the constant factor hidden in (2.61) will generally be dependent on the dimension, and in practice more samples will be needed as the dimension of the problem increases. A thorough discussion on the efficiency of Monte Carlo integration is given in Bergman (1999).

Rejection Sampling and Importance Sampling

To be able to estimate integrals like (2.57) using Monte Carlo integration, it is necessary that a sampling algorithm for the distribution $\pi(\boldsymbol{x})$ exists. For a few common one-dimensional distributions, e.g. uniform distribution, Gaussian distribution etc., exact sampling algorithms exist. The uniform distribution is fairly easy to sample from, using pseudo randoms numbers (Ross, 1972), and if one has a random number ξ sampled from a uniform distribution on the interval $[0, 1]$, one can generate a new sample from an arbitrary distribution by feeding ξ through the inverse of the cumulative distribution function of interest. If $F_x(x)$ is the cumulative distribution of the desired stochastic variable x, i.e. $P[x \leq a] = \int_{-\infty}^{a} F_x(x)\,\mathrm{d}x$, a sample x_s having the desired distribution is found as

$$x_s = F_x^{-1}(\xi). \tag{2.62}$$

This method is sometimes referred to as the *reverse transformation method*. The proof is fairly straight-forward, based on $F_x(x)$ being a monotone function, and can be found in any standard text describing sampling of random variables, e.g. Ross (1972). Multi-dimensional distributions are typically sampled using mixtures or combinations of one-dimensional sampling algorithms.

For a typical Bayesian estimation problem the distribution of interest is the posterior (2.19) at each time step, and as time evolves, the posterior distributions are often highly irregular and may in most cases not be described by an analytic, closed-form expression. In terrain navigation multimodal posteriors are commonplace. Consequently, exact sampling algorithms are not useful in such problems.

Several algorithms for approximate sampling exist. The method of rejection sampling (Ross, 1972) is based upon the assumption that there exists a proposal distribution $q(\boldsymbol{x})$ which is somewhat similar to the target distribution $\pi(\boldsymbol{x})$ one wants to sample from. Moreover, the target distribution must be bounded by the proposal distribution, i.e. a known finite constant M exists, such that $\pi(\boldsymbol{x}) \leq Mq(\boldsymbol{x})$ for any $\boldsymbol{x} \in \Omega$. It is further assumed that the proposal distribution is easy to generate samples from and that $\pi(\boldsymbol{x})$ can be evaluated up to a normalizing constant at any point in Ω. The rejection sampling algorithm compares a sample $\boldsymbol{\xi}$ drawn from $q(\boldsymbol{x})$ with a sample u from a uniform distribution on $[0, 1]$. If $u < \frac{\pi(\xi)}{Mq(\xi)}$ the sample is accepted, otherwise it is rejected and the procedure is repeated. It can be readily shown (Ross, 1972) that the samples that are accepted by the rejection sampling algorithm are drawn from the distribution $\pi(\boldsymbol{x})$.

In many practical situations, the assumptions made in the rejection sampling algorithm do not hold. It may be difficult to evaluate $\pi(\boldsymbol{x})$ at any point in Ω or it may not be possible to find a constant M that bounds the proposal distribution in the entirety of the sampling space. If the constant M is too large, a large proportion of the samples generated in the algorithm are rejected, leading to an inefficient algorithm.

An alternative to the rejection sampling algorithm is the importance sampling algorithm (Ross, 1972; Andrieu et al., 2001). All the particle filter algorithms discussed in this thesis are based on importance sampling, and it is therefore worthwhile to describe it in somewhat more detail. Like rejection sampling, importance sampling is also based on the existence of a proposal distribution $q(\boldsymbol{x})$, also referred to as the importance distribution, that is easy to generate samples from. The support of the target distribution $\pi(\boldsymbol{x})$ must be included in the support of the proposal distribution, i.e.

$$\pi(\boldsymbol{x}) > 0 \Rightarrow q(\boldsymbol{x}) > 0, \quad \forall \boldsymbol{x} \in \Omega. \tag{2.63}$$

The sought integral (2.57) can now be rewritten as

$$I = \int_{\Omega} f(\boldsymbol{x})\pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\Omega} f(\boldsymbol{x}) \frac{\pi(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{2.64}$$

and the Monte Carlo approximation (2.58) becomes a weighted sum

$$I_N = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i) w(\boldsymbol{x}_i), \tag{2.65}$$

where the weight $w(\boldsymbol{x}_i) = \frac{\pi(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$ is the importance weight of the sample $\boldsymbol{x}_i$. The importance weight takes into account the likelihood of the sample x_i being from the distribution $\pi(\boldsymbol{x}_i)$. To use the importance sampling algorithm, one must be able to evaluate the importance weight $w(\boldsymbol{x}_i)$. In many cases, however, the importance weight can only evaluated up to a normalizing factor. As will be shown later, this is the case in Bayesian filtering problems. Fortunately, the Monte Carlo approximation (2.65) can still be computed, choosing weights that are proportional to the ratio $\frac{\pi(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$ and doing a normalization

of the final estimate, such that (2.65) becomes

$$I_N = \frac{\sum_{i=1}^{N} f(\boldsymbol{x}_i) w(\boldsymbol{x}_i)}{\sum_{i=1}^{N} w(\boldsymbol{x}_i)}, \tag{2.66}$$

where $w(\boldsymbol{x}_i) \propto \frac{\pi(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$. For notational convenience, the short-hand notation w_i for $w(\boldsymbol{x}_i)$ will often be used in this thesis, but one should bear in mind that the importance weight depends explicitly on $\boldsymbol{x}_i$.

The estimate (2.66) is sometimes referred to as the *Bayesian importance sampling* estimate. For finite N, this estimate is biased, but it can be shown (Geweke, 1989) that asympotically a law of large numbers as well as a central limit theorem hold.

As will be evident below, when the SMC methods based on importance sampling are introduced, in the case of recursive Bayesian estimation, the set of samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ with corresponding importance weights $\{w_i\}_{i=1}^{N}$ will usually be of more interest than the actual approximation to the integral (2.57). The Sampling Importance Resampling algorithm (Algorithm 2.1), due to Rubin (1988) generates a set of samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ which are an approximate independent draw from the target distribution $\pi(\boldsymbol{x})$ based on the importance sampling weighted approximation. Step 4 in the algorithm is an additional resampling step that is added in order to keep the diversity of the particle set.

Algorithm 2.1 Sampling Importance Resampling

1. Generate M independent samples $\{\boldsymbol{x}_i\}_{i=1}^{M}$ from the proposal distribution $q(\boldsymbol{x})$.
2. Compute the importance weights $\{\tilde{w}_i\}_{i=1}^{M}$ as $\tilde{w}_i \propto \frac{\pi(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$.
3. Normalize the importance weights, $w_i = \tilde{w}_i / \sum_{i=1}^{M} \tilde{w}_i$.
4. Resample with replacement N times from $\{\boldsymbol{x}_i\}_{i=1}^{M}$ with probability w_i of resampling particle x_i.

An important aspect of Algorithm 2.1 is that the M samples are resampled N times. To ensure an effective algorithm, M should be chosen greater than N. In Rubin (1988) a factor 10 is recommended, i.e. $M = 10N$. It should be noted, however, that it is common practice in particle filtering to choose $M = N$, often with good results. This is for example the case for many of the results presented in this thesis.

Sequential Monte Carlo Estimation

Recursive Bayesian estimation deals with the problem of estimating the posterior density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$, using the Bayesian update equations (2.18)–(2.19). Unfortunately, the importance sampling algorithm can not be used for recursive estimation in its original form. One must instead estimate the full posterior density $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$ at each time step $k = 0, 1, \ldots$, reusing the full measurement series $\boldsymbol{Z}_k$ every time. The dimension of the estimation problem will thus grow at every time step, making the method inefficient,

because a higher number of samples is required as the dimension of the estimated distribution is growing, leading to an ever increasing computational complexity. However, it is possible to utilize the Markovian property of the system (2.25)–(2.27) to slightly modify the importance sampling algorithm in such a manner that it can be used recursively (Gordon et al., 1993; Doucet et al., 2001).

First consider the problem of estimating the full density $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$, i.e the target distribution $\pi(\cdot)$ in the importance sampling algorithm is the full joint distribution $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$. The recursion formula for this problem can be easily derived using Bayes' formula,

$$p(\boldsymbol{X}_k|\boldsymbol{Z}_k) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}{p(\boldsymbol{z}_k|\boldsymbol{Z}_{k-1})} p(\boldsymbol{X}_{k-1}|\boldsymbol{Z}_{k-1}). \tag{2.67}$$

The importance function $q_k(\cdot)$, with the index k denoting time step k, can be chosen, in the general case, to be dependent on the measurements, i.e.

$$q_k(\cdot) = q(\boldsymbol{X}_k|\boldsymbol{Z}_k). \tag{2.68}$$

Using Bayes' formula, the importance function can be written as

$$q(\boldsymbol{X}_k|\boldsymbol{Z}_k) = q(\boldsymbol{X}_{k-1}|\boldsymbol{Z}_{k-1})q(\boldsymbol{x}_k|\boldsymbol{X}_{k-1}, \boldsymbol{Z}_k). \tag{2.69}$$

By iterating, the proposal of the full joint posterior distribution can be written as

$$q(\boldsymbol{X}_k|\boldsymbol{Z}_k) = q(\boldsymbol{x}_0)\prod_{i=1}^{k} q(\boldsymbol{x}_i|\boldsymbol{X}_{i-1}, \boldsymbol{Z}_i), \tag{2.70}$$

such that the importance weights $\{w_k^{(i)}\}_{i=1}^N$ can be computed recursively as

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p\left(\boldsymbol{z}_k|\boldsymbol{x}_k^{(i)}\right) p\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{x}_{k-1}^{(i)}\right)}{q\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k\right)}. \tag{2.71}$$

Notice that the proposal distribution in the denominator of (2.71) contains the full state history $\boldsymbol{X}_k^{(i)}$ of each sample. There are several possible choices for the form of the proposal distribution $q(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k)$. One convenient choice, which also simplifies the recursion procedure considerably, is to let the proposal be independent of the measurements, i.e. simply to let the proposal be equal to the prior distribution,

$$q\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k\right) = p\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}\right) = p(\boldsymbol{x}_0)\prod_{j=1}^{k} p\left(\boldsymbol{x}_j^{(i)}|\boldsymbol{x}_{j-1}^{(i)}\right). \tag{2.72}$$

The importance weight recursion in this case becomes,

$$w_k^{(i)} \propto w_{k-1}^{(i)} p\left(\boldsymbol{z}_k|\boldsymbol{x}_k^{(i)}\right), \tag{2.73}$$

i.e the importance weights are simply updated according to the measurement likelihood density $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$. This particular choice of importance function is the basis of the Bayesian bootstrap and the sequential importance resampling filters described in the following sections. Though these are widely used methods, and in many cases lead to satisfactory results, it must be remembered that they are based on a restriction of the far more general importance sampling framework.

An important fact related to choosing the prior as importance function, is that the posterior distributions estimated in the resulting particle filtering algorithms are still an approximation of the full joint density $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$. However, the state history $\boldsymbol{X}_{k-1}$ of each sample $x_k^{(i)}$ are not used in the updating procedure at time step k, so if one is interested in the filtering density $p(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ only, the state history for each particle can be safely thrown away to save memory requirements.

As mentioned above the choice of $p\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}\right)$ as importance function is only one of several possible choices. It can be shown (Doucet et al., 2000b) that, in terms of minimizing the variance of the importance weights $w_k^{(i)}$ conditioned upon the full simulated trajectory $\boldsymbol{X}_{k-1}^{(i)}$ and the measurements $\boldsymbol{Z}_k$, the optimal choice of importance function is to let

$$q\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k\right) = p\left(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{z}_k\right). \tag{2.74}$$

Using this choice of importance function, the recursion formula for the importance weights becomes $w_k^{(i)} = w_{k-1}^{(i)} p\left(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}^{(i)}\right)$. However, there are two problems with this optimal choice of importance function. First, in most cases it is difficult to sample from the distribution $p\left(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{z}_k\right)$ and second, the choice requires that one be able to evaluate the integral

$$p\left(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}^{(i)}\right) = \int_{\Omega} p\left(\boldsymbol{z}_k|\boldsymbol{x}_k\right) p\left(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(i)}\right) \mathrm{d}\boldsymbol{x}_k. \tag{2.75}$$

In the general case the integral in (2.75) has no closed form solution, though in the special case of a system with nonlinear process model, linear measurement model, and with additive Gaussian noise, the integral can be evaluated analytically (Doucet et al., 2000b). The terrain navigation problem, with its highly non-linear measurement model, does not fit into this class of problems, so these problems will not be discussed any further in this thesis.

The Bayesian Bootstrap Particle Filter

A generic sequential Monte Carlo filter can be derived by directly applying the recursive updating procedure in (2.73). This generic algorithms is sometimes referred to at the Sequential Importance Sampling (SIS) algorithm (Doucet et al., 2000b, 2001). The

problem with such a generic algorithm, is that as time evolves, the distribution of the importance weights becomes more and more skewed, until finally only one weight different from zero exists among the importance weights, a phenomenon known as degeneration of the particle set. Such a particle set will have a zero variance, and consequently not be a realistic approximation to the posterior density. Only a system with perfect measurements, i.e. zero measurement noise, can be estimated with no uncertainty, and perfect measurements never occur in a real-world situation. The solution to this problem is to introduce a resampling step into the algorithm, in the same manner as in the SIR algorithm (Algorithm 2.1). This is the idea behind the Bayesian Bootstrap Filter (Gordon et al., 1993), the first functional sequential Monte Carlo method that was presented.

Algorithm 2.2 The Bayesian Bootstrap Algorithm

1. $0 \rightarrow k$. Generate M samples $\{\boldsymbol{x}_k^{(i)}\}_{i=1}^M$ from the initial distribution $p(\boldsymbol{x}_0)$.
2. Calculate the importance weights $p(\boldsymbol{z}_k|\boldsymbol{x}_k^{(i)}) \rightarrow w_i$ for $i = 1, \ldots, M$.
3. Normalize the weights, $w_i \left(\sum_{i=1}^M w_i\right)^{-1} \rightarrow w_i$ for $i = 1, \ldots, M$.
4. Generate a new set $\{\boldsymbol{x}_k^{i^*}\}_{i^*=1}^N$ from resampling with replacement N times from $\{\boldsymbol{x}_k^{(i)}\}_{i=1}^M$ where the probability of resampling particle $\boldsymbol{x}_i$ is $P(\boldsymbol{x}_k^{i^*} = \boldsymbol{x}_k^{(i)}) = w_i$.
5. Predict each of the resampled particles independently r times, where $M = rN$. The new particle set $\{\boldsymbol{x}_{k+1}^{(i)}\}_{i=1}^M$ is given by

$$\boldsymbol{x}_k^{(i^*-1)r+k} \sim p(\boldsymbol{x}_{k+1}|x_k^i), \qquad \text{for } k = 1, \ldots, N \quad \text{and } i^* = 1, \ldots, M.$$

6. $k + 1 \rightarrow k$ and go to step 2.

The MMSE estimate and its covariance, or other estimates of interest can be calculated between step 3. and 4. in the Bayesian bootstrap algorithm, from

$$\hat{\boldsymbol{x}}_k^{\text{MMSE}} = \sum_{i=1}^M w_i \boldsymbol{x}_k^{(i)}, \tag{2.76}$$

$$\hat{P}_k^{\text{MMSE}} = \sum_{i=1}^M w_i (\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}_k}^{\text{MMSE}})(\boldsymbol{x}_k^{(i)} - \hat{\boldsymbol{x}_k}^{\text{MMSE}})^T. \tag{2.77}$$

The Bayesian bootstrap algorithm resamples the particles at every time step. Thus, after the resampling step, all the particles in the new particle set $\{\boldsymbol{x}_{k+1}^{(i)}\}_{i=1}^M$ have equal weights. The importance weights are therefore not used at the next time step and do not have to be stored.

The predicted estimate $\bar{\boldsymbol{x}}_k$ and the corresponding covariance $\bar{P}_k$, based on the prior filter density $p(\boldsymbol{x}_k|\boldsymbol{Z}_{k-1})$, are sometimes of interest. These can also be computed from the Bayesian bootstrap approximation, between step 1 and 2 in the algorithm. At this

stage, all the particles have equal weights, due to the resampling at the last time step, so the MMSE prediction and its covariance are found simply as

$$\bar{\boldsymbol{x}}_k^{\text{MMSE}} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{x}_k^{(i)}, \tag{2.78}$$

$$\bar{P}_k^{\text{MMSE}} = \frac{1}{M}\sum_{i=1}^{M} (\boldsymbol{x}_k^{(i)} - \bar{\boldsymbol{x}_k}^{\text{MMSE}})(\boldsymbol{x}_k^{(i)} - \bar{\boldsymbol{x}_k}^{\text{MMSE}})^T. \tag{2.79}$$

The resampling step is the computationally most demanding operation in the Bayesian bootstrap filter and in the other SMC methods described below. Several resampling algorithms exist, each with different properties related to the variability of the importance weights. The most effective resampling algorithm is the classical method of Ripley (1987), sometimes referred to as *multinomial resampling*. This algorithm implements the resampling the particle set $\{\boldsymbol{x}_k^{(i)}\}_{i=1}^N$ in $\mathcal{O}(N)$ time, utilizing the fact that it is possible to sample N ordered i.i.d. samples from a uniform distribution in $\mathcal{O}(N)$ operations. These N uniform weights are then compared to the cumulative distribution of the importance weights, in order to determine which particles are to be multiplied and which are to be thrown away. The process is described in detail in Carpenter et al. (1999) and Nordlund (2002). Examples of other resampling algorithms are *stratified resampling* (Carpenter et al., 1999) and *residual resampling* (Liu and Chen, 1998).

When the particle set $\{\boldsymbol{x}_k^{(i)}\}_{i=1}^N$ is resampled, the samples will no longer be statistically independent, as some of the samples are multiplied in the process. The theoretical asymptotic convergence properties of the algorithms are based on the particles being independent, so resampling will in some situations lead to poorer convergence properties, and in the worst case divergence, of the SMC algorithms. The frequency at which the resampling should be done therefore becomes a trade-off between degeneracy of the importance weights, and the convergence properties of the filter. Because of this, a degeneracy measure for the particle set, *the effective sample size* was presented in Kong et al. (1994) and Liu (1996),

$$\begin{aligned} N_{\text{eff}} &= \frac{N}{1 + \text{var}_{q(\cdot|\boldsymbol{Z}_k)}(w(\boldsymbol{X}_k))} \\ &= \frac{N}{E_{q(\cdot|\boldsymbol{Z}_k)}[w(\boldsymbol{X}_k)]}. \end{aligned} \tag{2.80}$$

The quantity N_{eff} in (2.80) can not be computed analytically, but a good estimate for the effective sample size is given by

$$\widehat{N_{\text{eff}}} = \frac{1}{\sum_{i=1}^{N} \left(w(\boldsymbol{X}_k^{(i)})\right)^2}, \tag{2.81}$$

or in the specical case where the proposal distribution is chosen as $q(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k) = p(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_k^{(i)})$, the expression for the estimated effective sample says becomes simply

$$\widehat{N_{\text{eff}}} = \frac{1}{\sum_{i=1}^{N}\left(w_k^{(i)}\right)^2}. \tag{2.82}$$

The Sequential Importance Resampling Filter

In the Bayesian bootstrap filter, resampling is done at every time step. As discussed above, this is not always beneficial, since too frequent resampling may lead to poorer convergence properties of the SMCF. The *Sequential Importance Sampling/Resampling* (SIS) framework is a general algorithm framework, which tries to minimize this effect by using the expressions for the effective sample size (2.81) and (2.82). At each time step, the effective sample size is compared to a threshold κN, where $0 < \kappa \leq 1$ is predefined by the user. If $N_{\text{eff}} < \kappa N$, a resampling is done, otherwise the weights are updated from the previous time step. The framework is given in Algorithm 2.3.

Algorithm 2.3 Sequential Importance Sampling/Resampling Framework

1. For $i = 1, \ldots, N$, let $0 \rightarrow k$, $\frac{1}{N} \rightarrow w_{-1}^{(i)}$. Generate N particles $\boldsymbol{x}_0^i \sim q(\boldsymbol{x}_0|z_0)$.
2. For $i = 1, \ldots, N$, evaluate the importance weights:
   - Compute the weights from
   $$w_{k-1}^{(i)} \frac{p\left(\boldsymbol{z}_k|\boldsymbol{x}_k^{(i)}\right) p\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{x}_{k-1}^{(i)}\right)}{q\left(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k\right)} \rightarrow w_k^{(i)}.$$
   - Normalize the weights, $w_k^{(i)}\left(\sum_{i=1}^{N} w_k^{(i)}\right)^{-1} \rightarrow w_k^{(i)}$.
3. $\frac{1}{\sum_{i=1}^{N}\left(w_k^{(i)}\right)^2} \rightarrow \widehat{N_{\text{eff}}}$. If $\widehat{N_{\text{eff}}} < \kappa$, go to step 5.
4. Generate a new set of samples $\{\boldsymbol{x}_k^{(i^*)}\}_{i^*=1}^{N}$ from resampling with replacement N times the set $\{\boldsymbol{x}_k^{(i)}\}_{i=1}^{N}$ where the probability of resampling particle $\boldsymbol{x}_k^{(i)}$ is $P(\boldsymbol{x}_k^{i^*} = \boldsymbol{x}_k^{(i)}) = w_i$. Reset the weights $\frac{1}{N} \rightarrow w_k^{(i)}$.
5. For $i = i^* = 1, \ldots, N$, predict the resampled particles $\boldsymbol{x}_{k+1}^{(i)} \sim q\left(\boldsymbol{x}_{k+1}^{(i)}|\boldsymbol{X}_k^{(i^*)}, \boldsymbol{Z}_{k+1}\right)$.
6. $k + 1 \rightarrow k$ and go to step 2.

The choice of the parameter κ in Algorithm 2.3 is crucial for the performance of the algorithm. If a value of κ close to 1 is chosen, resampling will be performed at nearly

every time step, whereas a lower κ-value will lead to less frequent resampling. The choice of a good κ-value is dependent on the problem in question. If the tendency of particle set degeneration is high, a large value should be chosen.

Algorithm 2.3 is very general, and it can be used with an arbitrary choice of importance function $q(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k)$. It should be noted that the Bayesian bootstrap filter (Algorithm 2.1) is actually a special case of the SIS framework, resulting from the choices $q(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_{k-1}^{(i)}, \boldsymbol{Z}_k) = p(\boldsymbol{x}_k^{(i)}|\boldsymbol{X}_k^{(i)})$ and $\kappa = 1$.

In the descriptions of the particle filters so far, the general discrete stochastic system (2.12)–(2.13) has been considered. As was explained in Section 2.4.2, in the terrain navigation problem a simplified filter model must often be used, due to the computational complexity of the estimation methods. This is also the case for particle filters, though they are generally not as computationally demanding as the PMF. The reformulation of the particle filter algorithms when the discrepancy between the truth model and the filter model is taken into account is trivial and mainly involves attaching asterisks to the state variables. The same is true for the inclusion of the delta formulation (2.28).

2.4.6 The Rao-Blackwellized Particle Filter

The number of particles needed in the particle filters in order to effectively sample the sample space increases with the dimension of the state-space. In particular, the mean squared error matrix of the estimate, $E[(\boldsymbol{x}_k - (\hat{\boldsymbol{x}}_k))(\boldsymbol{x}_k - (\hat{\boldsymbol{x}}_k))^T]$, where $\boldsymbol{x}_k$ is the true value of the stochastic vector to be estimated, increases in norm when the number of particles is fixed, while the dimension of the state-space is increasing.

The Rao-Blackwell theorem, sometimes also known as the Rao-Blackwell-Kolmogorov theorem (Blackwell, 1947; Rao, 1965), which in one of its forms states that, given an estimator $\hat{\theta}(z)$ of a parameter θ, observed through a measurement z, for the mean squared error of the *Rao-Blackwell estimator* $\hat{\theta}_{\mathrm{RB}}(z) = E[\hat{\theta}(z) \mid T(z)]$, where $T(z)$ is a *sufficient statistic* (van Trees, 1968) for θ, the following inequality holds:

$$E[(\theta - \hat{\theta}_{\mathrm{RB}}(z))^2] \leq E[(\theta - \hat{\theta}(z))^2]. \tag{2.83}$$

This theorem can easily be extended to the vector case, where the inequality (2.83) takes the form of an MSE matrix inequality.

In particle filtering, the Rao-Blackwell theorem is utilized in order to construct a more effective filter, in terms of the number of samples needed, by marginalizing out a subset of the state-space by defining

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{x}_k^n \\ \boldsymbol{x}_k^l \end{bmatrix}, \tag{2.84}$$

where $\boldsymbol{x}_k^n$ denotes the 'nonlinear' states, which are to be estimated using a particle filter, whereas $\boldsymbol{x}_k^l$ are the 'linear' states and are to be estimated using a conventional Kalman filter. The full posterior density $p(\boldsymbol{X}_k|\boldsymbol{Z}_k)$ can now be written as

$$p(\boldsymbol{X}_k|\boldsymbol{Z}_k) = p(\boldsymbol{X}_k^n, \boldsymbol{X}_k^l|\boldsymbol{Z}_k) = p(\boldsymbol{X}_k^n|\boldsymbol{X}_k^l, \boldsymbol{Z}_k)p(\boldsymbol{X}_k^l|\boldsymbol{Z}_k). \tag{2.85}$$

When this marginalization is used, and the sub-vector $\boldsymbol{x}_k^l$ can be described using a linear-Gaussian state model, the density $p(\boldsymbol{X}_k^l|\boldsymbol{Z}_k)$ can be updated analytically, and only the $\boldsymbol{x}_k^n$ states need to be sampled in the particle filter, estimating $p(\boldsymbol{X}_k^n|\boldsymbol{X}_k^l, \boldsymbol{Z}_k)$, using the linear states computed in the KF as a parameter. This is the key idea behind the Rao-Blackwellized particle filter (RBPF), sometimes also known as the *marginalized* particle filter. According to the Rao-Blackwell theorem, the variance of the RBPF estimate is at least as low as the variance of the full PF. As the particle filter target and proposal densities have a lower dimension in the RBPF, fewer particles are needed, and the computational complexity will thus be lower. The concept of using the Rao-Blackwell theorem for sampling schemes was discussed in Casella and Robert (1996), whereas the RBPF was presented in Doucet (1998) and Doucet et al. (2000a), and has later been used in a number of different applications, including terrain aided navigation (Schon et al., 2005; Nygren, 2008).

In this thesis, the terrain navigation system is viewed as an external position aiding system, providing the inertial navigation system (INS) with external position updates, as will be explained in Section 3.4. A high quality INS is normally based on a Kalman filter with a large number of states. One example is the HUGIN navigation system, described in Jalving et al. (2003). To use an RBPF in such a navigation system would require reimplementation and redesign of the system, modeling the INS states in the linear part of the RBPF. As this thesis was carried out in close cooperation with the HUGIN team at FFI and Kongsberg Maritime, it was decided at an early stage to keep the terrain navigation system as an external module, and consequently the RBPF has not been implemented in this thesis. This choice also makes the terrain navigation system more portable, as it can be easily integrated with other navigation systems. As an example, many vehicles have black box navigation systems which only permit conventional position updates and in which the user does not have access to raw inertial measurements. On the other hand, keeping the TerrNav system as an external module limits the possibility to accurately model the drift of the INS in the TerrNav algorithms, as will be thoroughly discussed in later sections.

2.4.7 The Sigma Point Kalman Filter

In the EKF the recursive non-linear estimation problem is solved using linearization of the nonlinear transformations. The Sigma Point Kalman Filter (SPKF), also called the Unscented Kalman Filter (UKF), was first presented by Julier and Uhlmann (1997), and is another approach for dealing with the nonlinearities of the problem. The SPKF uses the so-called *unscented transform*, in which the underlying probability densities are approximated by a set of deterministically chosen points, known as *sigma points*. The nature of the SPFK is very different from that of the SMC methods discussed in previous sections. Though both classes of methods represent probability densities using discrete points, the sigma points are chosen in a pre-defined deterministic manner, as opposed to the randomly chosen particles in the SMC methods. Typically, the number of sigma points needed in order to represent a given density is much smaller than the number of

particles needed in a corresponding SMC method.

The idea behind the unscented transform is that the set of sigma points, representing the prior probability distribution, is propagated through the nonlinearities of the problem, yielding a new set of sigma points. The sigma points are initially chosen in such a manner that the important aspects of the probability distribution are maintained before and after applying the nonlinear transformation, and therefore the new set of sigma points is a representation of the posterior probability distribution. For this idea to work, it is crucial that the set of sigma points be able to capture certain important characteristics of the underlying distribution. By tuning certain parameters in the computations of the sigma points, the algorithm can be tailor-made to fit a number of well-known distributions. Because the sigma points can be chosen in a number of different ways, the SPFK should be seen as a class of estimation methods, all based on various variants of the unscented transform, rather than one single algorithm. It should also be noted that although the SPFK framework is able to utilize more information about the underlying distribution than solely the first and second moments, which is what is used in the normal EKF, higher order information is in many cases unavailable. In such cases, using wrong assumptions for the higher order methods may seriously degrade the performance of the filter.

This section will proceed as follows. First, a general scheme for sigma point selection is presented and second, the actual filtering algorithm, in which the set of sigma points is used in recursive non-linear estimation, is outlined.

The Unscented Transform

Let $\boldsymbol{x} \in \mathbb{R}^{n_x}$ be a general random vector, having a probability density function $p_x(\boldsymbol{x})$ and $\boldsymbol{y} \in \mathbb{R}^{n_y}$ another random vector, given by a transformation of the variable x,

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}), \tag{2.86}$$

where the function $h : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ is nonlinear. Consider the task of computing the probability density function $p_y(\boldsymbol{y})$ using the unscented transform. The pdf $p_x(\boldsymbol{x})$ can be represented by a set of sigma points $\mathcal{S}$, which consists of $p + 1$ vectors and their associated weights, i.e.

$$\mathcal{S} = \{\boldsymbol{x}^{(i)}, W^{(i)}\}_{i=0}^{p}. \tag{2.87}$$

The weights $W^{(i)}$ can be positive or negative, as opposed to what is the case for the weights in an SMC method. For the resulting estimates to be unbiased, the weights must obey the unity sum condition, i.e.

$$\sum_{i=0}^{p} W^{(i)} = 1. \tag{2.88}$$

The new set of sigma points, representing the pdf $p_x(\boldsymbol{x})$ is now given by

$$\boldsymbol{y}^{(i)} = \boldsymbol{h}(\boldsymbol{x}^{(i)}), \tag{2.89}$$

and the mean of $\boldsymbol{y}$ is given by

$$\bar{\boldsymbol{y}} = \sum_{i=1}^{p} W^{(i)} \boldsymbol{y}^{(i)}. \tag{2.90}$$

The covariance is calculated, in a similar manner, as the outer product of the transformed sigma points, i.e.

$$P_{\boldsymbol{y}} = \sum_{i=1}^{p} W^{(i)} \left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right) \left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right)^T. \tag{2.91}$$

The estimate of any other statistic of the transformed random vector $\boldsymbol{y}$ can be computed in a similar manner. As mentioned above, the set of sigma points for a particular distribution can be chosen in such a manner that the errors in the estimated moments of the projected distributions are minimized. For a Gaussian, it is possible to construct a sigma point selection scheme that matches the first four moments exactly, using $2n_x^2 + 1$ sigma points (Julier and Uhlmann, 2004).

An example of a symmetric sigma point selection scheme, with $2n_x + 1$ points is given by

$$\begin{aligned}
\boldsymbol{x}^{(0)} &= \bar{\boldsymbol{x}}, \\
\boldsymbol{x}^{(i)} &= \bar{\boldsymbol{x}} + \left(\sqrt{\frac{n_x}{1 - W^{(0)}} P_x}\right)_i, \\
W^{(i)} &= \frac{1 - W^{(0)}}{2n_x}, \\
\boldsymbol{x}^{(i+n_x)} &= \bar{\boldsymbol{x}} - \left(\sqrt{\frac{n_x}{1 - W^{(0)}} P_x}\right)_i, \\
W^{(i+n_x)} &= \frac{1 - W^{(0)}}{2n_x},
\end{aligned} \tag{2.92}$$

where $i = 1, \ldots, N$, and the value of $W^{(0)}$ can be arbitrarily chosen. The expression $(\sqrt{\frac{n_x}{1-W^{(0)}} P_x})_i$ denotes the i-th row or column of the matrix square root. The sigma points in this set all lie on the $\sqrt{\frac{n_x}{1-W^{(0)}} P_x}$-th covariance contour of the pdf $p(\boldsymbol{x})$, except for $\boldsymbol{x}^{(0)}$, which coincides with the mean of the pdf. This set is accurate to the second order, i.e. the first and second order moments of the UT approximated match those of the true pdf exactly. The choice of $W^{(0)}$ determines the position of the non-mean sigma points. Lowering the value of $W^{(0)}$, e.g. to a negative value, moves the non-mean sigma points closer to the mean. A common choice is $W^{(0)} = 1 - \frac{n_x}{3}$.

Applying the UT to Recursive Estimation

The SPKF uses the unscented transform recursively to solve the general Bayesian state-space model with additive noise (2.14)–(2.15). The most general formulation of the SPKF starts with augmenting the state vector $\boldsymbol{x}_{k-1}$ with the process noise $\boldsymbol{v}_k$ and measurement noise $\boldsymbol{w}_k$ terms, to obtain the $(n_x + n_v + n_w)$-dimensional vector

$$\hat{\boldsymbol{x}}_{a,k-1} = \begin{bmatrix} \hat{\boldsymbol{x}}_{k-1} \\ \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix}. \tag{2.93}$$

The process and measurements equations can now be rewritten as

$$\bar{\boldsymbol{x}}_{a,k} = \boldsymbol{f}_a(\hat{\boldsymbol{x}}_{a,k-1}), \tag{2.94}$$

$$\bar{\boldsymbol{z}}_k = \boldsymbol{h}_a(\hat{\boldsymbol{x}}_{a,k}), \tag{2.95}$$

where the functions $\boldsymbol{f}_a : \mathbb{R}^{n_x+n_v+n_w} \rightarrow \mathbb{R}^{n_x+n_v+n_w}$ and $\boldsymbol{h}_a : \mathbb{R}^{n_x+n_v+n_w} \rightarrow \mathbb{R}^{n_z}$ are simple reformulations of the process and sensor equations to fit the dimensions of the augmented state vector. The overbar notation in $\bar{\boldsymbol{x}}_k^a$ and $\bar{\boldsymbol{z}}_k$ denotes the predicted state and measurements, respectively. Using the augmented state vector framework, a general recursive SPKF algorithm can be formulated as in Algorithm 2.4. In this general case, where the time and measurement update equations are both linear, two sets of sigma points are used. The augmented sigma point set $\{\boldsymbol{x}_{a,k}^{(i)}\}_{i=1}^p$ is used for the time update, whereas the set $\{\boldsymbol{z}_k^{(i)}\}_{i=1}^p$ is used for the measurement update. The measurement sigma points are propagated through the nonlinear measurement equation to obtain the predicted innovation $\bar{\boldsymbol{\nu}}_k$, the corresponding innovation covariance matrix $\bar{P}_{\nu\nu,k}$ and the predicted cross-covariance matrix describing the correlation between the predicted state and measurement. A special case of the innovation form of the standard linear Kalman filter measurement update equations (Gelb, 1974; Bar-Shalom et al., 2001) is used for the measurement update, with $H_k = I$, since the measurement function has already been taken care of by the unscented transformation.

When one of the process or measurement equations has a linear form the general SPKF framework given in Algorithm 2.4 can be simplified, using the unscented transform on the nonlinear transformation only and the conventional KF algorithms on the linear transformation. When the terrain navigation filter model of Section 2.4.2 is used, the process model is assumed linear, whereas the measurement equation, involving the terrain database inquiry, is highly nonlinear. Thus, when using the SPKF for this particular problem, the unscented transform is only used in the measurement update.

Algorithm 2.4 The Sigma Point Kalman Filter

1. Set $0 \rightarrow k$. Initialize the sigma points $\{\boldsymbol{x}_{a,0}^{(i)}\}_{i=1}^{p}$ and their corresponding weights $\{W^{(i)}\}_{i=1}^{p}$ using any sigma point selection scheme.
2. Calculate the predicted sigma point set, by propagating the sigma points through the process model equation,
$$\boldsymbol{x}_{a,k+1}^{(i)} = \boldsymbol{f}_a(\boldsymbol{x}_{a,k}^{(i)}).$$
3. If desired, calculate the predicted mean and its covariance from
$$\bar{\boldsymbol{x}}_{a,k+1} = \sum_{i=1}^{p} W^{(i)} \boldsymbol{x}_{a,k+1}^{(i)}, \tag{2.96}$$
$$\bar{P}_{a,k+1} = \sum_{i=1}^{p} W^{(i)} \left(\boldsymbol{x}_{a,k+1}^{(i)} - \bar{\boldsymbol{x}}_{a,k+1}\right)\left(\boldsymbol{x}_{a,k+1}^{(i)} - \bar{\boldsymbol{x}}_{a,k+1}\right)^T. \tag{2.97}$$
4. Set $k+1 \rightarrow k$.
5. Instantiate each of the predicted sigma points using the measurement model,
$$\bar{\boldsymbol{z}}_k^{(i)} = h_a(\boldsymbol{x}_{a,k}^{(i)}). \tag{2.98}$$
6. Calculate the predicted measurement,
$$\bar{\boldsymbol{z}}_k = \sum_{i=1}^{p} W^{(i)} \bar{\boldsymbol{z}}_k^{(i)}. \tag{2.99}$$
7. Calculate the innovation covariance matrix,
$$\bar{P}_{\nu\nu,k} = R_{k-1} + \sum_{i=1}^{p} W^{(i)} \left(\bar{\boldsymbol{z}}_k^{(i)} - \bar{\boldsymbol{z}}_k\right)\left(\bar{\boldsymbol{z}}_k^{(i)} - \bar{\boldsymbol{z}}_k\right)^T. \tag{2.100}$$
8. Calculate the cross-covariance matrix,
$$\bar{P}_{xz,k} = \sum_{i=1}^{p} W^{(i)} \left(\boldsymbol{x}_{a,k}^{(i)}\right)\left(\boldsymbol{x}_{a,k}^{(i)} - \bar{\boldsymbol{x}}_{a,k}\right)\left(\bar{\boldsymbol{z}}_k^{(i)} - \bar{\boldsymbol{z}}_k\right)^T. \tag{2.101}$$
9. Update the state estimate using the normal KF update equations,
$$\bar{\boldsymbol{\nu}}_k = \boldsymbol{z}_k - \bar{\boldsymbol{z}}_k, \tag{2.102}$$
$$K_k = \bar{P}_{xz,k}\bar{P}_{\nu\nu,k}^{-1}, \tag{2.103}$$
$$\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + K_k\bar{\boldsymbol{\nu}}_k, \tag{2.104}$$
$$\hat{P}_k = \bar{P}_k - K_k\bar{P}_{\nu\nu,k}K_k^T. \tag{2.105}$$
10. Go to Item 2.

2.5 Maximum Likelihood Terrain Navigation

All of the methods described in Section 2.4 use the Bayesian approach to the estimation problem, i.e. it is assumed that information on the prior distribution is available before incorporating the measurements to obtain the updated state estimate. The state of the system to be estimated is considered a stochastic process evolving in time, in accordance with the process model. The stochastic properties of the process model are assumed known and incorporated into the state estimate through the time update or prediction step. In the Fisher approach to estimation, on the other hand, no prior information is used. The quantity to be estimated is considered to be an unknown constant, or at least that its time variation is "slow" compared to that of the state variables of the system. The position of an underwater vehicle is obviously not constant in time, and taking the Fisher approach is therefore not possible if one wants to estimate the full position state of the vehicle. However, when using the delta approach introduced in (2.28), it is possible to formulate the terrain navigation problem in the Fisher framework. When doing so, the position offset $\delta\boldsymbol{x}$ from the real-time navigation system is implicitly assumed constant, and due to the drift in the INS, this assumption will be clearly be violated if the time it takes to collect the terrain measurements and perform the terrain navigation computations is too long. However, if one for example is able to make several measurements at the same time, as is the case for an AUV equipped with a multibeam echo sounder (MBE), the drift of the INS during the measurement process is negligible and a Fisher approach to the problem may be favorable. A Fisher estimator is normally not as mathematically and computationally complex as a Bayesian estimator, and especially if the information about the process model is unknown or has a high uncertainty, considering a Fisher estimator might be worthwhile. Due to the fact that the quantity under estimation is assumed constant or slowly varying, Fisher estimation is often also known as *parameter estimation*. However, this term may be misleading, as it is sometimes also used for the problem of estimating a parameter in the Bayesian framework. Some authors, e.g. Bar-Shalom et al. (2001), refer to this case as *random parameter estimation*, as opposed to *nonrandom parameter estimation*. To avoid confusion the terms Fisher estimation and Bayesian estimation will be used in this thesis.

In the general vector case Fisher estimation deals with the problem of estimating the unknown vector $\boldsymbol{x} \in \mathbb{R}^{N_x}$, which is observed through a set of measurements $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\}$, where each measurement vector $\boldsymbol{z}_j \in \mathbb{R}^{n_z}$ is assumed to come from a measurement model given by

$$\boldsymbol{z}_j = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w}_j), \qquad j = 1, \ldots, N, \tag{2.106}$$

where the $\boldsymbol{w}_j$'s are the disturbances or measurement noises. Like in the Bayesian case, a convenient and often sufficient assumption is that the noises be additive, i.e.

$$\boldsymbol{z}_j = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{w}_j, \qquad j = 1, \ldots, N. \tag{2.107}$$

The only statistical information used in Fisher estimation is the likelihood function of

the parameter vector, given by

$$L(\boldsymbol{x}) = p(\boldsymbol{Z}|\boldsymbol{x}). \tag{2.108}$$

In many situations, the measurements $\boldsymbol{z}_j$ are assumed statistically independent, in which case the likelihood takes the convenient form

$$L(\boldsymbol{x}) = \prod_{j=1}^{N} p(\boldsymbol{z}_j|\boldsymbol{x}). \tag{2.109}$$

Under the assumptions $\boldsymbol{w}_j \sim p_w(\boldsymbol{w}_j)$, where $p_w(\cdot)$ is an arbitrary pdf, and the w_j's are additive and independent, the likelihood function is given by

$$L(\boldsymbol{x}) = \prod_{j=1}^{N} p(\boldsymbol{z}_j|\boldsymbol{x}) = \prod_{j=1}^{N} p_w(\boldsymbol{z}_j - \boldsymbol{h}(\boldsymbol{x})). \tag{2.110}$$

The most common Fisher estimator is the maximum likelihood estimator, which simply maximizes the value of the likelihood function,

$$\hat{\boldsymbol{x}}^{\mathrm{ML}}(\boldsymbol{Z}) = \arg\max_{\boldsymbol{x}}(L(\boldsymbol{x})). \tag{2.111}$$

Despite its simplicity, the ML estimator can be shown to have a number of desirable properties (van Trees, 1968; Bar-Shalom et al., 2001). Among these are its unbiasedness and effiency. The efficiency of the ML estimator and its relation to the Cramér-Rao lower bound is further discussed in Section 2.7. The ML estimator is the Fisher counterpart to the Bayesian maximum a posteriori (MAP) estimate (2.24).

In the terrain navigation case, using the delta formulation, the measurement equation at a time step j takes the form of (2.31), repeated here for convenience,

$$\boldsymbol{z}_j = \boldsymbol{h}(\tilde{\boldsymbol{x}}_j + \delta\boldsymbol{x}_j) + \boldsymbol{w}_j. \tag{2.112}$$

The likelihood function of each individual depth measurement vector $\boldsymbol{z}_j$ is then given by

$$L_j(\delta\boldsymbol{x}_j) = p_{w_j}(\boldsymbol{z}_j - \boldsymbol{h}(\tilde{\boldsymbol{x}}_j + \delta\boldsymbol{x}_j)), \tag{2.113}$$

Utilizing the assumed independency and additivity of the measurement noise, the total likelihood at time step k is given by

$$L(\delta\boldsymbol{x}_k) = \prod_{j=0}^{N} L_j(\delta\boldsymbol{x}_j) \tag{2.114}$$

A recursive maximum likelihood terrain navigation estimator can now be formulated as

$$\widehat{\delta\boldsymbol{x}}_k^{\mathrm{ML}} = \arg\max_{\boldsymbol{x}_k} L(\delta\boldsymbol{x}_k). \tag{2.115}$$

Maximum likelihood terrain navigation can be seen as a limiting form of the optimal Bayesian filter as the uncertainty in the prior tends to infinity. This can be easily derived by introducing a so-called *diffuse prior*, i.e. a prior pdf over an interval which is large compared to the support of the likelihood function (Bar-Shalom et al., 2001).

2.6 The Cramér-Rao Lower Bound

The Cramér-Rao lower bound (CRLB) gives a theoretical lower bound for how precisely a quantity can be estimated. The CRLB was originally formulated for Fisher estimation, but it can be easily generalized to hold also in a Bayesian framework.

2.6.1 Fisher Case

For a Fisher estimator $\hat{\boldsymbol{\theta}}(\boldsymbol{Z})$, where $\boldsymbol{\theta}$ is an unknown, nonrandom vector, and $\boldsymbol{Z}$ is the collection of measurements used by the estimator, the CRLB gives a minimum theoretically attainable mean square error matrix

$$E[(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta})^T]$$

of the estimator. The bound is named after Harald Cramér and Calyampudi Radakrishna Rao, who were among the first to derive it (Rao, 1945; Cramér, 1946). The theory and proofs of the CRLB can be found in most modern books on estimation and statistical signal processing, e.g. van Trees (1968) and Kay (1993).

In the special case of an unbiased Fisher estimator, i.e. $E[\hat{\boldsymbol{\theta}}(\boldsymbol{Z})] = \boldsymbol{\theta}$ for all values of $\boldsymbol{\theta} \in \mathbb{R}^n$, the CRLB states that the covariance matrix of the unbiased estimator $\hat{\boldsymbol{\theta}}(\boldsymbol{Z})$ is bounded from below as

$$E[(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta}_0)(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta}_0)^T] \geq \boldsymbol{J}^{-1}, \tag{2.116}$$

where $\boldsymbol{\theta}_0$ is the true value of the unknown parameter vector and J is the Fisher information matrix (FIM), defined as

$$J := -E[\nabla_\theta \nabla_\theta^T \ln L(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}] = E[(\nabla_\theta \ln L(\boldsymbol{\theta}))(\nabla_\theta \ln L(\boldsymbol{\theta}))^T]|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}, \tag{2.117}$$

where $L(\boldsymbol{\theta}) = p(\boldsymbol{Z}|\theta)$ is the likelihood function, as defined in Section 2.5 and $\nabla_\theta(\boldsymbol{\theta})$ is the gradient vector of $\boldsymbol{\theta}$. The two formulations of the FIM are equivalent. The matrix inequality in (2.116) is to be understood in the sense that $\boldsymbol{A} \geq \boldsymbol{B}$ implies that the matrix $\boldsymbol{A}-\boldsymbol{B}$ is positive semi definite. It should be noted that the CRLB is dependent on the *true* value $\boldsymbol{\theta}_0$ of the unknown parameter vector and it can therefore only be calculated when the true value is known. Consequently, estimator performance can only be analyzed with respect to the CRLB in simulations in which the true value is known, or in real-time situations under the assumption that an 'almost perfect' ground truth is available.

The CRLB given in (2.116) is the theoretical lower bound attainable for any unbiased estimator of $\boldsymbol{\theta}$. If an estimator attains the CRLB, it is said to be an *efficient estimator*. A corresponding property to (2.116) can be defined for the more general case of biased estimators (van Trees, 1968).

2.6.2 Bayesian Case

An extension of the CRLB for Bayesian estimators can also be formulated in which case the matrix in question is the error correlation matrix of the Bayesian estimator $\boldsymbol{\theta}(\boldsymbol{Z})$:

$$E[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T], \tag{2.118}$$

where the expectation is to be taken with respect to both $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$. The CRLB for random parameters, sometimes referred to as the *posterior* CRLB, is a straight-forward generalization of the nonrandom case (van Trees, 1968). The $\boldsymbol{J}$ matrix now consists of two parts,

$$\boldsymbol{J}_T = \boldsymbol{J}_D + \boldsymbol{J}_P, \tag{2.119}$$

where $\boldsymbol{J}_D$ contains the information given in the data and is identical to the FIM defined in (2.117). The matrix $\boldsymbol{J}_P$ contains the information given in the prior data and is defined by

$$\boldsymbol{J}_P = -E[\nabla_\theta \nabla_\theta^T \ln P_0(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta_0}}] = E[(\nabla_\theta \ln P_0(\boldsymbol{\theta})(\nabla_\theta \ln P_0(\boldsymbol{\theta})^T]|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}], \tag{2.120}$$

where $P_0(\boldsymbol{\theta})$ is the prior probability density function of the random vector $\boldsymbol{\theta}$. The posterior CRLB can now be expressed by substituting $\boldsymbol{J}_T$ for $\boldsymbol{J}$ in (2.116) to obtain

$$E[(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta}_0)(\hat{\boldsymbol{\theta}}(\boldsymbol{Z}) - \boldsymbol{\theta}_0)^T] \geq \boldsymbol{J}_T^{-1}. \tag{2.121}$$

Bergman (1999) developed a recursive formulation of the posterior CRLB for the terrain navigation problem, and this was used to show that after convergence the PMF attained the CRLB asymptotically in Monte Carlo simulations, using a real terrain database. In other words, the PMF is an efficient estimator. The same was shown for a SIR particle filter by Karlsson et al. (2003). The CRLB for the terrain navigation problem was also thoroughly discussed in Nygren (2005).

Computing the CRLB in the terrain navigation problem involves computing the gradient of the terrain database, which may be a difficult task in real terrain databases, especially if the terrain data has a low sample resolution. To compute the CRLB, one also needs to know the true position of the vehicle, which is never the case in a real-time navigation system. Since this thesis is mostly concerned with applying terrain navigation algorithms to real vehicle data, the CRLB will not be pursued much further here. But it is a desirable property that the Bayesian estimators are efficient, at least asymptotically, under the assumption that there is no discrepancy between the true system and the filter model. As will be stressed later in the thesis, such a discrepancy often is a source of error in a real-time terrain navigation system, and when such a discrepancy exists, the estimators can not be expected to be efficient.

2.7 Relations to the Fokker-Planck Equation

So far in this thesis only discrete process models have been discussed. The discrete process equation for an arbitrary state-space model is described by the Chapman-Kolmogorov equation, given in (2.18), repeated here for convenience:

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) = \int_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k. \tag{2.122}$$

Equation (2.122) is a special case of the general Chapman-Kolmogorov equation (Jazwinski, 1970). Let $\{\boldsymbol{y}_i\}_{i=1}^n$ be a set of general random vectors, having joint probability density function

$$p_{y_1,\ldots,y_n}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n). \tag{2.123}$$

The general Chapman-Kolmogorov equation can then be derived by doing a simple marginalization over the random vector $\boldsymbol{y}_n$, i.e.

$$p_{y_1,\ldots,y_{n-1}}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{n-1}) = \int_{\mathbb{R}^{n_y}} p_{y_1,\ldots,y_n}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n)\,\mathrm{d}\boldsymbol{y}_n. \tag{2.124}$$

It should be noted that in the general Chapman-Kolmogorov equation (2.124), nothing is assumed about the ordering of the stochastic vectors. It holds in general, for marginalization over an arbitrary vector. Equation (2.122) can easily be obtained by applying the general equation to the vectors $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_k$, conditioned upon the measurements $\boldsymbol{Z}_k$, which yields

$$\begin{aligned} p(\boldsymbol{x}_{k+1}|\boldsymbol{Z}_k) &= \int_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_{k+1},\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k && (2.125)\\ &= \int_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k,\boldsymbol{Z}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k && (2.126)\\ &= \int_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{Z}_k)\,\mathrm{d}\boldsymbol{x}_k, && (2.127) \end{aligned}$$

where the last equality follows from the assumption that the state $\boldsymbol{x}_{k+1}$ at time step $k+1$ is independent of the measurements $\boldsymbol{Z}_k$ when the previous state $\boldsymbol{x}_k$ is known.

The continuous counterpart to the Chapman-Kolmogorov equation is the Fokker-Planck equation, a stochastic partial differential equation. This equation is also sometimes referred to as Kolmogorov's forward equation.

In continuous time, a general stochastic model can be described as,

$$\mathrm{d}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t),t)\,\mathrm{d}t + \boldsymbol{G}(\boldsymbol{x}(t),t)\,\mathrm{d}\boldsymbol{\beta}(t), \tag{2.128}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$, $f\colon \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x}$, $G\colon \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x \times n_x}$, and $\boldsymbol{\beta}(t)$ is an n_x-dimensional Brownian motion (Klebaner, 1998), with

$$E[\,\mathrm{d}\boldsymbol{\beta}(t)\,\mathrm{d}\boldsymbol{\beta}(\tau)^T] = \tilde{\boldsymbol{Q}}(t)\delta(t-\tau), \tag{2.129}$$

where $\delta(t-\tau)$ denotes the Dirac delta distribution. The tilda above $\tilde{\boldsymbol{Q}}(t)$ indicates that it is a spectral density matrix, contrary to the discrete case, where $\boldsymbol{Q}_k$ is a covariance

matrix. The first term on the right hand side of (2.128) known as the *drift term*, whereas the second is called the *diffusion term*.

Equation (2.128) is a stochastic differential equation, written on differential form, which is common for such equations. The differential form is just a short-hand notation for the stochastic integral equation

$$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \boldsymbol{f}(\boldsymbol{x}(\tau), \tau)\,\mathrm{d}\tau + \int_0^t \boldsymbol{G}(\boldsymbol{x}(\tau), \tau)\,\mathrm{d}\boldsymbol{\beta}(\tau). \tag{2.130}$$

The last term in Equation (2.130) is a stochastic integral with respect to the Brownian motion $\boldsymbol{\beta}(t)$. Such integrals need to be rigorously defined in order for (2.130) to make sense mathematically. In general, several definitions of the stochastic integral exist, including the *Ito integral* and the *Stratanovich integral*. It would lead too far to introduce this theory here, as the major concern in this thesis is discrete state-space models. Thorough introductions to the topics of stochastic calculus can be found e.g. in Øksendal (1985) and Klebaner (1998).

If the initial probability density function of $\boldsymbol{x}(t)$ in (2.128) is given by $p_x(\boldsymbol{x}, 0)$, it can be shown that the resulting pdf at time t is given by the n-dimensional Fokker-Planck equation,

$$\begin{aligned}\frac{\partial p_x(\boldsymbol{x}(t), t)}{\partial t} = &-\sum_{i=1}^{n_x} \frac{\partial}{\partial x_i}\big(f_i(\boldsymbol{x}(t), t) p_x(\boldsymbol{x}(t), t)\big) \\ &+ \sum_{i=1}^{n_x}\sum_{j=1}^{n_x} \frac{\partial^2}{\partial x_i \partial x_j}\big(G_{ij}(\boldsymbol{x}(t), t) p_x(\boldsymbol{x}(t), t)\big).\end{aligned} \tag{2.131}$$

The Fokker-Planck equation is a partial differential equation (PDE) of parabolic type (Renardy and Rogers, 1993), and it can be solved numerically using any kind of method suited for this class of PDEs. Nygren (2005) proposes to use finite element methods to solve the Fokker-Planck equation in terrain navigation applications. In the terrain navigation problem, the general Fokker-Planck equation (2.131) can be simplified.

2.8 Feature-Based Navigation

This thesis concentrates on bathymetric terrain navigation, i.e. only pure terrain height measurements are used. A related research area is that of feature-based navigation, in which *features* in the terrain are utilized in order to obtain a position update. Possible features include terrain features, such as characteristic land forms (e.g. valleys, rivers, mountains), objects (rocks, wrecks etc.) or man-made structures. In principle any feature that can be identified by the relevant sensor can be used for navigation, provided the position of the feature is known. Feature-based navigation can in general be performed

using a number of different sensors types, e.g. lasers, radars, cameras or sonars. In underwater applications one is often restricted to sonars, though camera-based navigation is also an alternative (Eustice, 2005). However, the short range of the camera requires the vehicle to operate within a few meters from the sea floor, which is in many cases infeasible.

Essential to the success of feature-based navigation is *feature extraction*, i.e. the process in which features are extracted from the sensor data. Feature extraction may involve complex signal and image processing techniques. In man-made environments, e.g. indoors or in urban areas, there are typically numerous, sharply defined features that can be used for navigation, for instance corners, doors, buildings etc. In more unstructured environments, like non-urban areas and especially underwater, there are typically fewer and less distinct features, making the task of feature extraction more difficult. Examples of feature extraction techniques are presented in Roman (2005), Eustice (2005) and the references therein.

After a feature has been extracted from the sensor data, it needs to be associated to the correct feature in the map. This task is normally referred to as *data association*. In some situations, the exact data association is known. This can for example be the case in an underwater application using long base line transponder navigation (Milne, 1983), where the transponders are able to identify themselves uniquely. In most cases, however, the data association is unknown and has to be estimated by the navigation system. Numerous algorithms for data association in different applications exist.

Having identified a feature in the sensor data and associated it to a feature in the map database, one is able to do a position update based on the location of the feature, its position error and the predicted position of the vehicle itself. The errors in this process typically come from

1. The error of the feature location in the map database.

2. Processing errors, resulting from coordinate transformations, sensor misalignment etc.

3. Measurement errors, including errors due to erroneous attitude estimates of the vehicle.

4. Data association errors.

Of the above error sources, the data association errors can be claimed to be the most critical. If a feature is not associated correctly, break-down of the navigation algorithm may occur.

The most straight-forward way to implement a feature-based navigation method is to use the feature updates as measurement to a Kalman filter of some type, e.g. an aided INS. Dependent on the sensor used, the measurement typically consist of range only, bearing only or both range and bearing, where the range and bearing are measured relative to the feature.

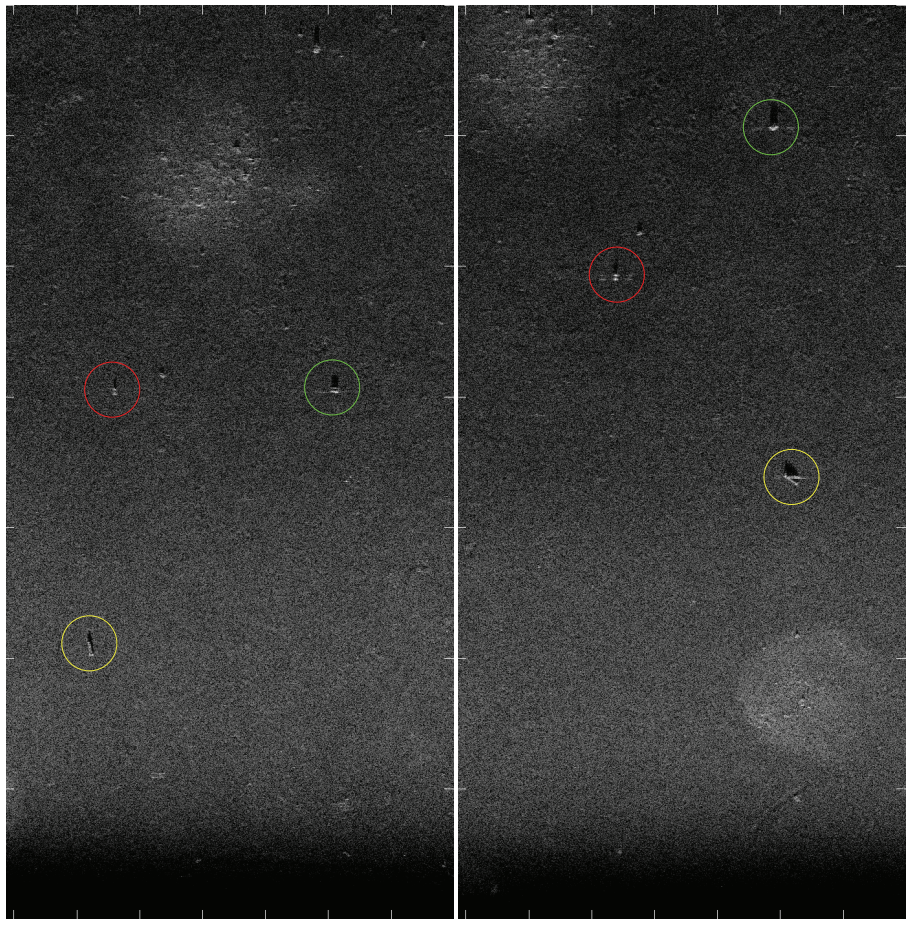


Figure 2.3: *Features on the sea floor, as observed from two different passes with an AUV equipped with a synthetic aperture sonar (SAS). Corresponding features are marked with corresponding colors. The task of feature association is crucial to the success of feature-based navigation. Figure courtesy of FFI.*

2.9 Simultaneous Localization and Mapping

In recent years, the topic of Simultaneous localization and mapping (SLAM) has been given a lot of attention within the robotics community. In essence SLAM deals with the problem of mapping an area with a robotic vehicle, and utilize the map for navigation simultaneously. The SLAM problem was formulated as a probabilistic estimation problem in the seminal paper by Smith et al. (1990), in which a feature-based approach is taken. The map is represented as a set of features, whose location are estimated in a extended Kalman filter. The state vector of the Kalman filter contains the position and attitude of the vehicle, together with the locations of the features or *landmarks*. When a new feature is discovered, the state vector is expanded, and the new feature is added to the state vector. As in all feature-based approaches, the problem of data association is crucial to the success of the algorithms. Each time a feature is sensed, the feature has to be associated to a previous one or declared as a new feature. Errors in the data association can be a source of failure and divergence of the SLAM algorithms (Neira and Tardos, 2001). In addition to the feature-based SLAM solution, starting with Smith et al. (1990) so-called *featureless* SLAM approaches have been developed, using e.g. laser range scan matching (Lu and Milios, 1997).

The computational burden of EKF-based SLAM scales as $\mathcal{O}(n^2)$, where n is the number of states. Consequently the EKF becomes unattractive when mapping areas with a large number of features. In addition, EKF is based on local linearization around the estimated vehicle trajectory, leading to potential divergence when the linearization is not accurate enough. Several alternatives to the EKF have been presented in order to overcome these problems. Examples are particle filtering, using a Rao-Blackwellization approach (Montemerlo et al., 2002), sparse information filters (Thrun et al., 2004; Eustice et al., 2004), junction tree filters (Paskin, 2003) and constraint networks (Lu and Milios, 1997). A different approach, using a set of local maps connected in a graph structure was presented by Bosse (2004). A thorough tutorial of both basic and state-of-the-art SLAM methods can be found in Durrant-Whyte and Bailey (2006) and Bailey and Durrant-Whyte (2006).

Traditionally, SLAM was developed and used primarily for indoors and land robots. However, a few results on underwater SLAM have been published during the recent years. Williams et al. (2000) use point features extracted from sonar measurements, whereas Newman and Leonard (2003) take advantage of pre-deployed acoustic beacons, like in conventional LBL navigation, but without tedious pre-localization of the beacons. Eustice et al. (2005) present a vision-based approach, using a camera for near-sea floor ROV navigation. Fewer attempts have been published using a featureless bathymetric approach. Roman (2005) uses small sub-maps built up using a multibeam imaging sonar. The sub-maps are registered pairwise, using a correlation or iterative closest point approach, to constrain the vehicle position estimates in accordance with the terrain.

In principle, the bathymetric terrain based navigation methods presented in this thesis can be used for SLAM. One possible solution is to do this in a sequential manner,

in which the vehicles is alternating between *map building mode*, *operation mode* and *navigation mode*. During map building mode the vehicle builds a map autonomously while exploring an unknown area, using some kind of map creation algorithm. After the map has been built, the vehicle switches to operation mode, conducting some operation with or within the mapped area. During operation, the real-time navigation accuracy will degrade, and a position fix is needed. This can be obtained by returning to the previously mapped area and enter navigation mode, in which bathymetric terrain navigation algorithm are used for navigation system aiding. Using this approach, the vehicle is able to bound its real-time navigation error. The global position uncertainty after having made a terrain navigation fix is bounded below by the vehicle position accuracy during the map building phase. It may be argued, however, that this simple approach is not SLAM in the sense defined in the aforementioned references, as the mapping an navigation is performed sequentially, rather than simultaneously. Nevertheless, such a strategy would be of great value when for instance performing submerged underwater operations in a previously unmapped area.

3

Underwater Terrain Navigation

In the previous chapter, the principles and algorithms for terrain-based navigation in general were described. In this chapter the special problems that arise when using terrain based navigation underwater are discussed. First, the different sensors used in underwater terrain navigation are described, before moving on to the stochastic models describing the problem.

3.1 Depth Measurements

Underwater terrain navigation deals with the task of measuring a part of the sea floor, and compare it to a terrain map of the area. Underwater maps give the depth at certain points on the sea floor, relative to a defined vertical datum. Numerous vertical datums exist, *mean sea level*(MSL) being the most widely used. Depth maps and their error sources will be discussed in Chapter 4. However, it is important to bear in mind that map databases are made from the same types of sensors that are being used for underwater terrain navigation, sharing a lot of error characteristics. It is therefore desirable that different sensor types are used for mapping and navigation, e.g. multibeam echo sounders with different frequencies, in order to minimize the statistical correlation between errors of the map database and errors of the depth measurements used for navigation.

The total sea depth at the location of an underwater vehicle is a combination of the vehicle depth below the surface, or more accurately, the depth below the vertical depth reference used, and the distance from the vehicle to the sea floor. This can be written as

$$z = z_v + h_v + a, \tag{3.1}$$

where z_v denotes the depth of the vehicle, h_v the height above the sear floor, and a denotes the vertical distance between the depth and height sensors. Note that the distance

a is dependent on the orientation of the vehicle.

The depth of the vehicle, z_v, is usually computed using a pressure sensor, through a pressure to depth conversion, whereas the height above the sea floor, h_v, is measured using some kind of bathymetric sensor, usually an acoustic sensor. Different bathymetric sensors and their characteristics are described in the subsequent subsections. Since the total sea depth measurement contains contribution from the pressure sensor as well as from the bathymetric sensor, the measurement error contains contribution from both of these sensors.

3.1.1 Bathymetric Measurements

With the exception of laser, all underwater bathymetric sensors are acoustic sensors. The principle of acoustic range measurements is simple. A sound pulse, with a known frequency and pulse-length is sent from the sensor transmitter towards the sea floor. The sound is reflected from the sea floor and back to the sensor. The time it takes from the pulse is transmitted until the echo is received at the sensor receiver, is known as the time-of-flight of the pulse. If the speed of sound c in the water is known, the range can be computed as

$$r = \frac{\tau c}{2}, \tag{3.2}$$

where τ is the two-way time-of-flight of the sonar pulse. In general the speed of sound will not be constant throughout the water-column, so the range calculation is generally more complex than in (3.2). In order to minimize the effects of sound speed errors, sound velocity profiles should be taken frequently.

A number of different sensors can be used for terrain navigation. In principle, any sensor measuring the depth below the vehicle can be used. However, the more information one can get about the sea floor, the faster the convergence and the better the robustness of the terrain navigation algorithms. It is therefore desirable to use sensors that give several depth measurements in each ping, e.g. a multibeam echo sounder.

Single Beam Echo Sounder

As the name indicates, a single beam echo sounder (SBE) measures the depth in one point only, usually directly below the vehicle. Before the advent of the multibeam echo sounder, SBEs were the primary sensor used for sea bed mapping. Therefore, many of the depth databases that exist today are made from single beam data.

For terrain navigation, one usually needs a profile consisting of a number of consecutive pings. Such a profile takes time to build up, and uncertainties in vehicle velocities will contribute to errors in the profile. However, many underwater vehicles not primarily intended for mapping, are equipped with single beam echo sounders e.g. for the purpose of safe maneuvering. This is for example the case for many submarines, facilitating the use of advanced terrain navigation.

Multibeam Echo Sounder

A multibeam echo sounder (MBE) measures the depth in a fan of points beneath the vehicle. Modern MBEs typically use several hundred beams, covering a large area beneath the vehicle in an effective way. Because of their high resolution and accuracy, MBEs are the primary sensor used in modern sea bed mapping.

An MBE consists of a transmitter array, aligned along the track of the vehicle, and a receiver array aligned across the track of the vehicle. The resulting footprint consists of a series of depth measurements arranged in a fan across the travel direction of the vehicle.

All MBE data used in this thesis are from a Kongsberg Maritime EM3000 MBE, which has a total of 128 beams and a frequency of 300 kHz. The beam width is 1.5×1.5 degrees, and the operating range is from 1 to 150 meters. It has a specified depth accuracy of 5 cm. The data are automatically roll and pitch compensated, using data from the real-time navigation system. The data are output in the AAD (along-across-depth) format, indicating the alongtrack distance, acrosstrack distance and depth measurement of each individual ping. Before the measurements can be used for terrain navigation, the MBE footprint has to be transformed into the coordinate system used in the map database. This transformation is based on the estimated heading of the vehicle. Theoretically, it would be possible to estimate the vehicle heading as an additional state in the terrain navigation algorithms. However, as INS equipped underwater vehicles usually have a very good heading estimate due to the implicit gyrocompassing capability of a high-end INS, heading estimation in the terrain navigation system is not addressed in this thesis.

By synthesizing consecutive pings from the MBE, a 3-dimensional profile can be built up from the individual MBE fans. Often, one MBE sample is not enough to obtain an accurate terrain navigation solution. This can for example be the case in flat areas. As more MBE fans are put together, more information about the sea floor can be utilized. However, it is important to be aware of the drift in the real-time navigation between consecutive pings. If this is large, the resulting 3-dimensional profile will contain errors leading to potential failure or inaccuracies in the terrain navigation solution. In such a scenario, there will typically also be correlations between the measurements from ping to ping, which should be incorporated in the algorithms. The difference in area coverage between SBEs and MBEs is depicted in Figure 3.1.

3D Sonar

A 3D sonar has 2-dimensional arrays, giving a 3-dimensional measurement profile from each ping. Thus the seafloor is sampled with an even higher efficiency than with an MBE. There is no need for synthesizing consecutive pings; an accurate terrain navigation estimate can be calculated as soon as the signal processing in the 3D sonar is done.

Nygren (2005) proposes the use of a 3D sonar for terrain navigation and derives some favorable asymptotic properties of the terrain navigation measurement model as

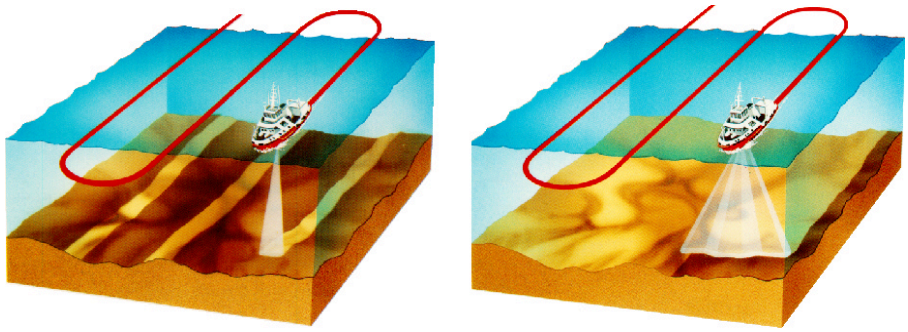


Figure 3.1: *Difference between the area covered by an SBE (left) and an MBE (right). Figure with permission from Jalving et al. (2004b).*

the number of measurement points increases.

Doppler Velocity Log

A Doppler velocity log measures the 3-dimensional velocity of an underwater vehicle in the vehicle body coordinate system, by measuring the Doppler shift of sound reflected from the sea floor or from the water surrounding the vehicle. It is possible to measure both the bottom relative and the water relative velocity, by pinging against the bottom or a water layer, respectively. To measure bottom relative velocity, it is necessary that the DVL has "bottom track", i.e. that the vehicle is so close to the bottom that the DVL is able to detect the reflected sound. The maximum range of a DVL is dependent on the frequency of the emitted sound; the lower frequency, the longer maximum range. However, lowering the frequency also lowers the resolution of the measurement, so choosing the DVL frequency is a trade-off between range and resolution, a well-known principle in underwater acoustics. Typical DVL frequencies used for underwater vehicles are 300, 600 and 1200 kHz.

Most commercially available DVLs today are so-called broadband DVLs, which measure velocity from the time dilation of a series of sound pulses (RD Instruments, 1996), instead of using a continuous-tone signal for the computation of the Doppler shift. A popular configuration, used i.e. in the RDI Workhorse Navigator DVLs, is the Janus configuration, in which four transducers are mounted facing downwards, each with an inclination of 30° to the vertical. Each of the beams measures the bottom referenced velocity projected onto the centerline of the beam axis, i.e.

$$\boldsymbol{v}_{\text{DVL,raw}}(t) = \begin{bmatrix} v_{b_1}(t) \\ v_{b_2}(t) \\ v_{b_3}(t) \\ v_{b_4}(t) \end{bmatrix}, \tag{3.3}$$

where v_{b_i} is a scalar measurement of the vehicle velocity $\boldsymbol{v}_{eb}^{b}$, decomposed along the

principal direction of the i-th beam. Since there are four measured components of a three-dimensional quantity, there is a redundancy in the measurement, which can be utilized to get an estimate of the error in the velocity measurement. In the DVL sensor coordinate system, the velocity measurement becomes

$$\boldsymbol{v}_{eb_{\text{DVL}}}^{b_{\text{DVL}}}(t) = M\boldsymbol{v}_{\text{DVL,raw}}(t), \tag{3.4}$$

where M is a transformation matrix. In order to use this measurement for navigation purposes, it must be compensated for the roll, pitch and heading of the vehicle, i.e. it must be transformed into a local, earth-referenced coordinate system.

As the DVL in bottom-tracking mode measures the velocity by analyzing the Doppler shift of sound reflected from the bottom, it also provides depth measurements. Each of the DVL beams gives rise to a depth measurement, so a DVL with four measurement beams, like an RDI Workhorse Navigator DVL, yields four depth measurement at each ping. The measured depth is the altitude above the sea-floor at each individual beam footprint, and to compute the location of the footprint in a geo-referenced frame, the attitude of the vehicle from the INS system should be used.

Most modern AUVs are equipped with a DVL to counter the drift in the INS system. Lower-end AUVs sometimes also have navigation systems based solely on a DVL, a pressure sensor and a compass, with no IMU. In either case, it is possible to utilize the depth measurements of the DVL for terrain navigation. It should be stressed, however, that both the range and the area of coverage are usually smaller than what is the case for an MBE, making the DVL-based terrain-navigation less robust and slowing down the convergence time. However, in the case where no more sophisticated depth sensors are available, the DVL is a good alternative.

Sidescan Sonars

A traditional sidescan sonar ensonifies the sea floor to each side of the vehicle, using fan-shaped pulses perpendicular to the path of the vehicle. The returned echoes are used to create sea floor images, based on the echo strength of the returned sound. Side-scan images are well-suited to identify features on the sea floor, like rocks or bathymetric structures. These images can therefore be used in feature-based navigation, after the images have gone through a feature extraction process. Traditional side-scan sonars do not produce bathymetry of the sea-floor. However, so-called *bathymetric sidescan sonars*, are equipped with two or more receiver arrays at each side of the vehicle or tow-fish, placed horizontally above each other with a known vertical displacement. Such systems are able to provide depth-measurements from interferometry techniques. The principles of a side-scan sonar are described in Lurton (2002).

Since sidescan sonars cover a much wider area than multibeam echo sounders, using sidescan bathymetry in bathymetric terrain navigation will make the algorithms more robust and effective. Combining sidescan and multibeam bathymetry, using the multibeam bathymetry as a gap-filler between the two sidescan areas, enables the vehicle to utilize

a large area for navigation, which is beneficial, especially in areas with flat topography where convergence using MBE only may be slow.

Synthetic Aperture Sonars

A synthetic aperture sonar (SAS) is able to image the sea floor much more accurately than a sidescan sonar. The principle is to illuminate each spot several times, and build up a synthetic aperture as the vehicle is moving along a straight line, similar to what is done in a synthetic aperture radar (SAR). By using sophisticated processing of the data, the SAS is able to create images with typically ten times better resolution than a conventional sidescan sonar. Interferometric SAS systems are equipped with two or more receivers at each side of the system, in the same manner as an interferometric sidescan. Interferometric SAS systems provide bathymetry with extremely high resolution. An introduction to SAS technology can be found in Lurton (2002) and review of current state-of-the-art is given in Hayes and Gough (2009).

SAS images are ideal for use in feature-based navigation, because of their high resolution. Likewise, the high resolution SAS bathymetry from an interferometric SAS would be ideal for use in bathymetric terrain navigation. However, most commercially available interferometric SAS systems available today do not provide real-time data processing, which would be necessary for a real-time TerrNav system to take fully advantage of the SAS results. Modern HUGIN AUVs can be delivered with HiSAS, a high resolution interferometric SAS that providing high quality bathymetry as well as high resolution sea floor images (Hagen et al., 2008). Figures 3.2 and 3.3 show examples of SAS imagery and bathymetry obtained from the HiSAS.

Vision-Based Terrain Navigation

As cameras do not produce bathymetric measurements, they can not be used for bathymetric terrain navigation. However, cameras are well suited for feature-based navigation which was explained in Section 2.8 and the references therein. Because of the short range of a camera, the vehicle will have to operate close to the sea floor, and the camera often needs to be equipped with its own light source.

3.1.2 Pressure-to-Depth Conversion

The total sea depth at the position of the vehicle is given as the sum of the vehicle altitude and the vehicle depth, as shown in Equation (3.1). The depth of an underwater vehicle is estimated using a pressure sensor. The pressure-to-depth conversion is an important source of error in the terrain navigation algorithms, and care should therefore be taken to minimize these error sources.

A map data base reports the total sea depth relative to some horizontal datum, e.g. mean sea level (MSL) or WGS-84. When performing terrain navigation, it is important that the measured depth is given in the same vertical datum as the map database. The

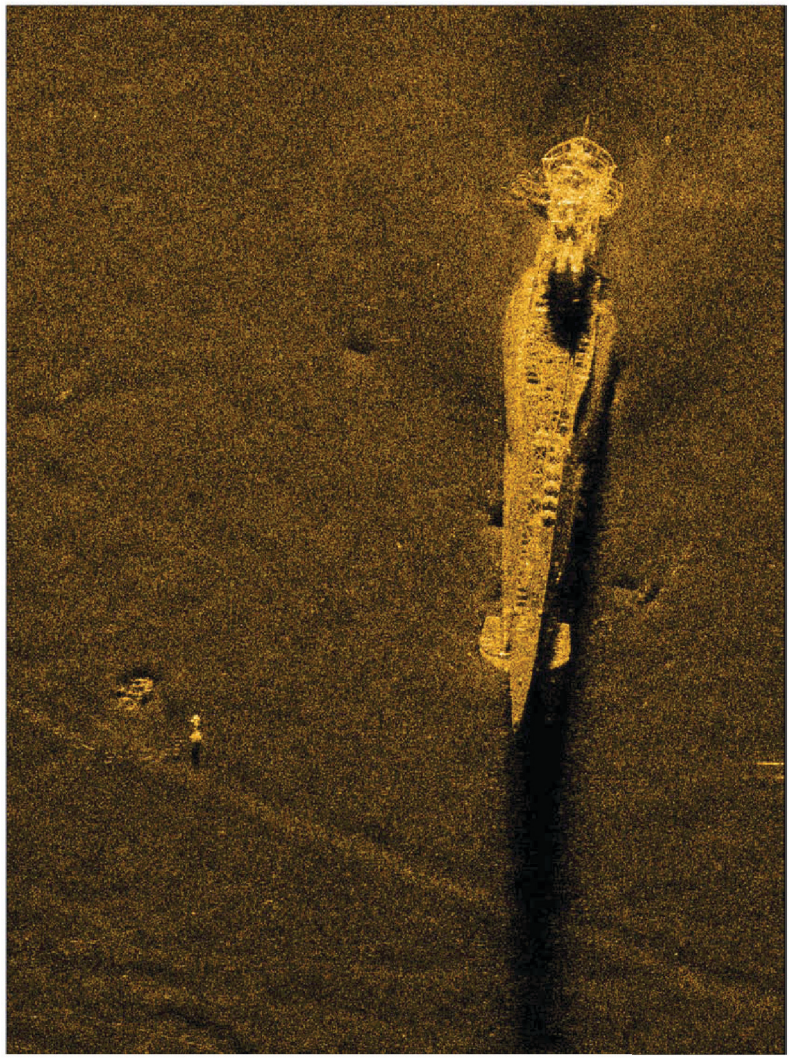


Figure 3.2: *Example of a SAS image of a ship wreck from the Kongsberg Maritime HiSAS. Courtesy of FFI.*

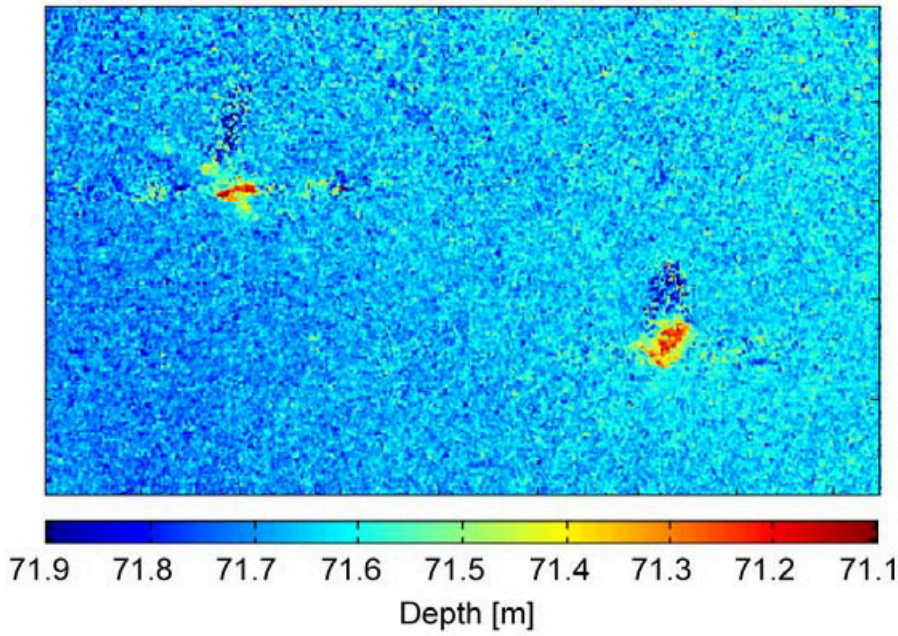


Figure 3.3: *Example of high resolution bathymetry obtained from the Kongsberg Maritime HiSAS. Courtesy of FFI.*

pressure measured by the pressure sensor is the absolute pressure at the depth of the vehicle, as shown in Figure 3.4. The absolute pressure measured at the vehicle can be written as

$$\tilde{p} = p_h + p_w + p_0 + e, \tag{3.5}$$

where p_h denotes the hydrostatic pressure, p_w is the dynamic pressure induced by surface waves, p_0 is the atmospheric pressure and e is the pressure sensor measurement error. The atmospheric pressure has a standard value of 101.325 kPa and varies with different weather conditions. The accurate atmospheric pressure should be known, either by letting the vehicle measure it itself before diving or by using some other kind of pressure forecast for the operation area. The tidal wave term p_w can be modeled as an oscillating error, with parameters according to the sea state in the area of operation. Modeling this term and filtering it in the INS Kalman filter, has proven useful and led to better accuracy in sea bed mapping in the presence of waves (Willumsen et al., 2007).

Let p denote the true absolute pressure at the vehicle, i.e. $p = p_h + p_w + p_0$, the depth z of the vehicle below the sea surface (in meters) can then be computed by the following formula, taken from Fofonoff and Millard, Jr. (1983):

$$z = \frac{\int_{p_0}^{p} V(35, 0, p)\, \mathrm{d}p}{g_0(\mu)\left(1 + \frac{1}{2}\gamma_p(p - p_0)\right)} + \frac{1}{9.8}\int_{p_0}^{p} \delta(S, T, p)\, \mathrm{d}p, \tag{3.6}$$

where, $V(S,T,p) = \frac{1}{\rho(S,T,p)}$ is the specific volume of sea water as a function of salinity S, temperature T and pressure, where $\rho(S,T,p)$ is taken from the International equation of state for sea water (EOS80), $g_0(\mu)$ is the sea surface gravity at latitude μ, γ_p is the mean vertical gradient of gravity with respect to pressure in the water column. The integral $\int_{p_0}^{p} \delta(S,T,p)\,\mathrm{d}p$ of the specific volume anomaly $\delta(S,T,p) = V(S,T,p) - V(35,0,p)$ gives the geopotential anomaly ΔD and represents corrections to the actual density profile of the water column compared to that of *standard ocean* (S=35 psu and T=0°C). This integral can be calculated from a CTD (conductivity, temperature and density) profile of the water column, e.g. using a standard numerical integration procedure. The first integral in (3.6) also needs to be approximated. In Fofonoff and Millard, Jr. (1983) a fourth order polynomial fit of this integral was presented, which has become the standard method of calculating pressure from depth. The conversion formula thus becomes

$$z = \frac{c_1 p + c_2 p^2 + c_3 p^3 + c_4 p^4}{g_0(\mu)\left(1 + \frac{1}{2}\gamma_p (p - p_0)\right)} + \frac{1}{9.8}\Delta D, \tag{3.7}$$

where the numerical values of the polynomial coefficients are $c_1 = 9.72659$, $c_2 = -2.25 \cdot 10^{-5}$, $c_3 = 2.279 \cdot 10^{-10}$ and $c_4 = -1.82 \cdot 10^{-5}$, provided the pressure is measured in decibars. A common model for the gravitation at sea surface is (in units of m/s^2)

$$g_0(\mu) = 9.780318(1 + 0.0052788 \sin^2 \mu + 0.0000236 \sin^4 \mu), \tag{3.8}$$

where the latitude μ is measured in radians.

After the depth below the sea surface has been approximated using (3.7), the depth has to be converted to the correct vertical datum, i.e. compensated for surface waves and tidal effects. As mentioned above, the effect of surface waves should be attempted filtered out by modeling it in the INS Kalman filter. The correct tide at the time of the operation should be recorded and compensated for before sending the depth measurements to the terrain navigation algorithms. However, as will be shown in later sections, a depth bias due to incorrect tidal compensation or uncertainties in the pressure to depth conversion, can be modeled and estimated in the terrain navigation algorithms, making the terrain navigation algorithms more robust to such errors.

3.2 Process Model

In Section 2.4.2 the state-space model for terrain navigation in general was introduced, together with the delta formulation of the estimation problem (2.28), in which the position is estimated as an offset from the position given by the real-time navigation system, which in the underwater case usually is an aided INS system. In the following, it is assumed that vehicle real-time navigation system is indeed an INS system.

The filter model process equation, in the conventional 2-dimensional case, estimating north and east position offset, was given in (2.41), repeated here for convenience,

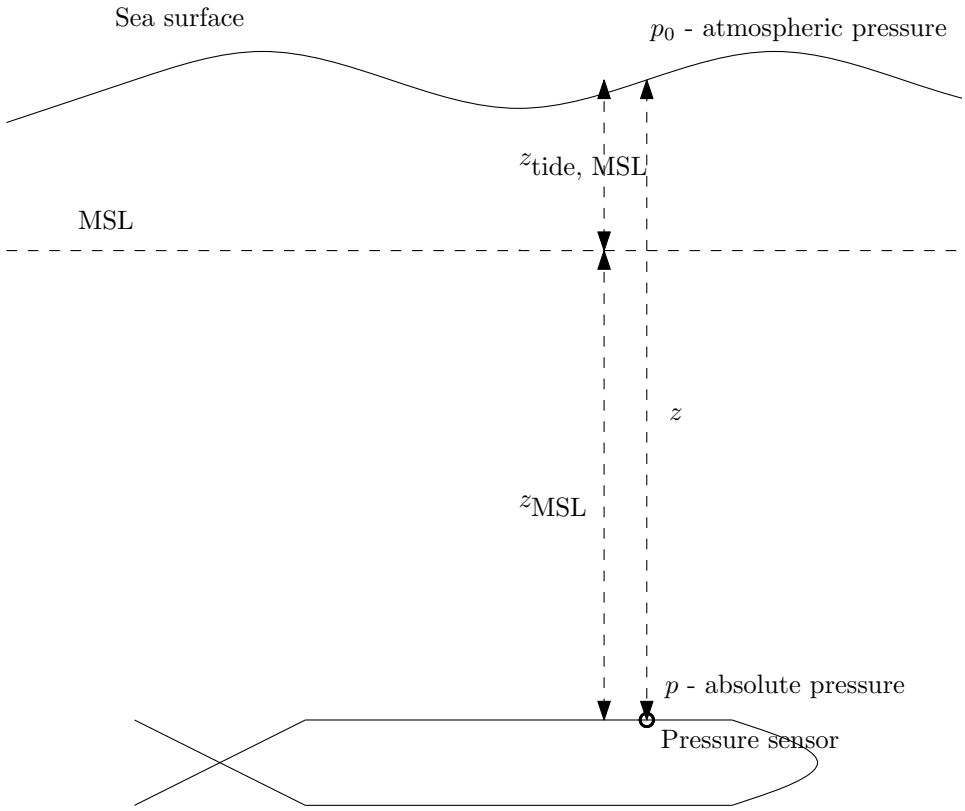


Figure 3.4: *Pressure to depth conversion.*

$$\delta \boldsymbol{x}_{k+1}^* = \delta \boldsymbol{x}_k^* + \boldsymbol{v}_k^*. \tag{3.9}$$

The process noise $\boldsymbol{v}_k^*$ was considered a discrete white noise sequence having a covariance matrix $\boldsymbol{Q}_k^*$, i.e. it is uncorrelated between time steps. The process noise represents the drift in the INS. Reiterating (3.9), one obtains

$$\begin{aligned} \delta \boldsymbol{x}_k^* &= \delta \boldsymbol{x}_{k-1}^* + \boldsymbol{v}_{k-1}^* \\ &= \delta \boldsymbol{x}_{k-2}^* + \boldsymbol{v}_{k-1}^* + \boldsymbol{v}_{k-2}^* \\ &\vdots \\ &= \delta \boldsymbol{x}_0^* + \sum_{i=0}^{k-1} \boldsymbol{v}_i^*, \end{aligned} \tag{3.10}$$

which shows that the total drift at time step k is given as the sum of a white noise sequence, i.e. the drift is a discrete-time random walk.

In an INS system, there are a number of different error contributions, both in the inertial sensors and aiding sensors, each contributing to the overall drift of the navigation accuracy. For example, the gyroscopes and accelerometers measurements, i.e. the angular rates and linear accelerations of the vehicle reference frame $\{b\}$ relative to the inertial reference frame $\{i\}$, include both biases and white noise. When white noises present in the measured angular rates and linear accelerations are integrated, it results in angular random walk (ARW) and velocity random walk (VRW), i.e. random walk in angle and velocity. When velocity is integrated to obtain position, the resulting position error is even more complex. An unaided inertial navigation system has a drift dependent on the quality of the inertial sensors used. High-end AUV navigation systems are typically based in so-called navigation grade IMUs, having an unaided drift of around 1 nautical mile per hour. To counter for this drift, aiding sensors are used, as explained in Section 1.1.2. A DVL aided INS will typically counter the position drift to be approximately proportional to the distance traveled. As an example, a HUGIN AUV traveling at 2 m/s ($\approx$ 4 knots) and equipped with a bottom-referenced 300 kHz DVL, is reported to have a typical along track error of 4 % of traveled distance i.e. about 30 meters per hour (Jalving et al., 2003). The across track drift will typically be smaller, around $1/3$ of the along-track drift. These errors can however be dramatically reduced by running certain regular mission patterns, making some of the internal biases in the navigation system observable. An example is the so-called lawn-mower pattern, which is widely used in sea floor mapping surveys.

To model the drift of an aided INS properly, a large number of states are needed. Most aided INS systems of today are based on an extended Kalman filter, estimating the errors in the position and attitude computed from the navigation equations, a set of differential equations from which the position, velocity and attitude of the vehicle can be computed from the initial states and the IMU measurements. The Kalman filter usually

includes 9 main states, 3 for position, 3 for velocity and 3 for attitude, plus a number of additional error states, modeling noises and biases in the sensors. A modern AINS for underwater vehicles, like the HUGIN navigation system described in Jalving et al. (2003) typically has around 20 states in the state vector. Hence, in a stand-alone terrain navigation algorithm, in which the number of states must be held low because of the computational constraints, it is impossible to model the correct behavior of the AINS drift.

The process model (3.9) was developed directly in the earth-referenced coordinate frame $\{e\}$. Hence, the drift in the INS position, described by the possibly time-variant covariance matrix $\boldsymbol{Q}_k^*$, must be specified directly in $\{e\}$. However, for a real vehicle the position drift is more easily characterized in the vehicle body-fixed coordinate system $\{b\}$. For instance, when Doppler velocity aiding is used, the drift is typically larger in the along track direction than in the across track direction. To be able to model this, the noise should be specified in the $\{b\}$ frame. This is further discussed in the following section.

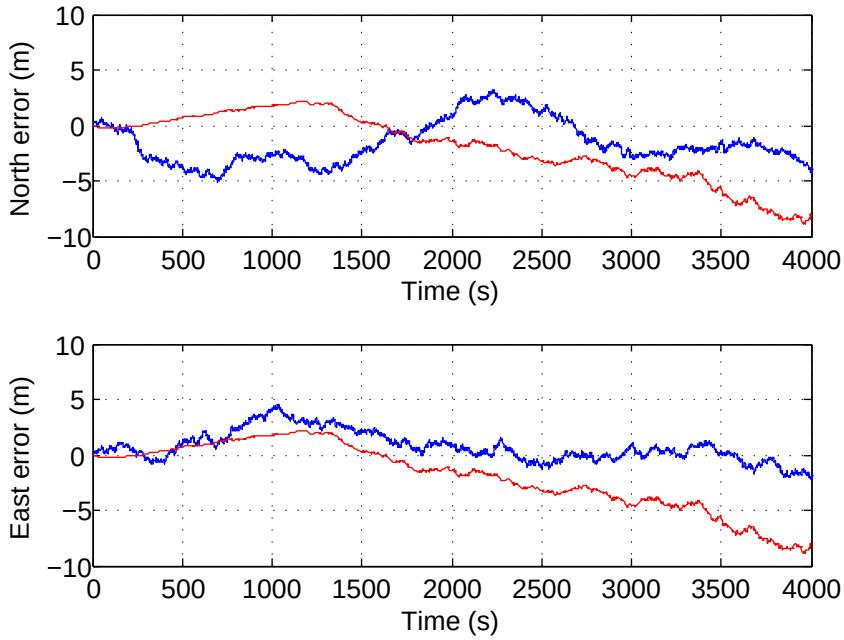


Figure 3.5: *Simulated position error in north and east direction using the simple discrete random walk model, Equation* (3.9)*, (blue line) and realistic AINS drift obtained from NavLab (red line).*

Figure 3.5 shows a realization of the north and east error obtained from the simplified process model (3.9) together with a corresponding realistic realization of the error

from an AINS. The process noise in the discrete random walk model has been chosen to be in accordance to the drift of the INS. The AINS realization was obtained using the simulator and estimator part of the NavLab simulation and post-processing tool (Gade, 2004). Simulated DVL, depth sensor and compass measurements were used as aiding sensors. The AINS realization is quite different from the discrete random walk realization. Especially at the start of the simulation the AINS realization has a much smoother trajectory. This is typical when using the simplified model; though the overall drift over a longer period is fairly realistic, the short-time behavior is very different from that of the full AINS model.

3.2.1 Extension of the State-Space Model

In this section an extension to the state-space model traditionally used in terrain navigation is developed. The focus is on the process model, and an alternative to the simple position error random walk model (3.9) is presented. The work presented here was first published in Ånonsen et al. (2007).

Coordinate Frames

A number of different coordinate frames are relevant in navigation problems. The coordinate frames used in this thesis are listed in Table A.3 in Appendix A. Typically one is interested in the position of the vehicle in a global coordinate frame. A common coordinate frame is the $\{e\}$ frame or ECEF (Earth-centered, Earth-fixed) system, which is a cartesian coordinate system with its origin in the mass center of the earth, the z-axis along the rotation axis of the earth pointing towards the north pole and the x-axis intersecting the surface of the Earth at 0° latitude and 0° longitude. The xy-plane coincides with the equatorial plane. As the name ECEF suggests, the $\{e\}$ frame rotates with the Earth. The position of the vehicle relative to the $\{e\}$ frame can be written as the position vector $\boldsymbol{p}_{eb}$, i.e. a position vector from the origin of the $\{e\}$ frame to the origin of the vehicle body frame $\{b\}$.

It is more common to specify the position of the vehicle relative to some reference ellipsoid that approximates the surface of the earth. The position is then given as the latitude and longitude degrees (in degrees or radians), as well as the height above/below the reference ellipsoid. The most common reference ellipsoid today is that of the WGS-84 (World Geodetic System) which is used e.g. in GPS systems. It should be noted that the surface of the reference ellipsoid does not coincide neither with the earth geoid nor with any of the common vertical references, like MSL. The vehicle depth must therefore be transformed when going from WGS-84 to MSL or any other vertical datum. The transformation from the ECEF system to WGS-84 is readily done using closed formulas or iteration techniques.

The actual navigation computations are performed in a local coordinate frame, often labeled the *navigation frame* (the $\{n\}$ frame). Several variants exist, but in this thesis the NED (North-east-down) frame will be used. In this frame, the x-axis points towards

north, the y-axis towards east and the z-axis downwards. The frame is therefore moving with the vehicle, and the origin is usually chosen to coincide with the vehicle's center of navigation (most commonly the position of the IMU). The orientation of the frame will also change, in accordance with the curvature of the Earth. When approaching the poles, the orientation will be changing infinitely fast, which is a problem in inertial systems. In INS systems it is therefore common to use an alternative local frame, in which the x and y axes are allowed to "rotate freely" around the z-axis. This system is known as the *wander azimuth system* (Titterton and Weston, 2004) but is not used in this thesis.

Sensor errors, like velocity errors and inertial sensor errors, are naturally described in the $\{b\}$ frame, the *body frame* of the vehicle. For example, the DVL measures the velocity $\boldsymbol{v}_{eb}^{b}$, i.e. the velocity of the vehicle $\{b\}$ relative to the earth frame, decomposed in the $\{b\}$ frame.

The filter model (3.9) models the offset of the vehicle in the north and east directions. The model is therefore given in the $\{n\}$ frame. In order to model the drift more realistically, the filter model will now be reformulated in the $\{b\}$ frame and later transformed to the $\{n\}$ frame. Since the measurements from the MBE are often given in a roll and pitch compensated body system, it is actually easier to formulate the drift in this system, labeled the $\{b'\}$ system. Let ϕ, θ and ψ denote the roll, pitch and yaw angles of the vehicle. The coordinate transformation matrix from the $\{b\}$ frame to the $\{b'\}$ frame is then given by

$$R_b^{b'} = \begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}, \tag{3.11}$$

whereas the coordinate transformation matrix from $\{b'\}$ to $\{n\}$ is given by a simple heading compensation,

$$R_{b'}^{n} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.12}$$

The coordinate transformation matrix from $\{b\}$ to $\{n\}$ is given as

$$\begin{aligned} R_b^n &= R_{b'}^n R_b^{b'} \\ &= \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}. \end{aligned} \tag{3.13}$$

Body-Relative Drift

In the conventional filter model (3.9) a white noise model was used for the drift. A natural extension to this model is to use a first-order Markov model for the drift in the $\{b'\}$-frame. Then, in continuous time, the horizontal drift $\boldsymbol{v}^{b'} \in \mathbb{R}^2$ is given by the differential equation

$$\dot{\boldsymbol{v}}^{b'}(t) = \begin{bmatrix} -\frac{1}{\tau_1} & 0 \\ 0 & -\frac{1}{\tau_2} \end{bmatrix} \boldsymbol{v}^{b'}(t) + \boldsymbol{\zeta}(t), \tag{3.14}$$

where τ_1 and τ_2 are positive time constants and $\boldsymbol{\zeta}(t)$ is a 2-dimensional continuous-time white noise. Discretizing this equation yields

$$\boldsymbol{v}_{k+1}^{b'} = \boldsymbol{G}_v(t_{k+1}, t_k)\boldsymbol{v}_k^{b'} + \boldsymbol{\zeta}_k. \tag{3.15}$$

The matrix $\boldsymbol{G}_v(t_{k+1}, t_k)$ is given by

$$\boldsymbol{G}_v(t_{k+1}, t_k) = \begin{bmatrix} \exp(-\frac{t_{k+1}-t_k}{\tau_1}) & 0 \\ 0 & \exp(-\frac{t_{k+1}-t_k}{\tau_2}) \end{bmatrix}. \tag{3.16}$$

The discrete white noise sequence $\boldsymbol{\zeta}_k$ is related to the continuous white noise $\boldsymbol{\zeta}(t)$ as

$$\zeta_k = \int_{t_k}^{t_{k+1}} \boldsymbol{G}_v(t_{k+1}, \tau)\zeta(\tau)\,\mathrm{d}\tau, \tag{3.17}$$

with the covariance matrix (assuming that $E[\zeta_k] = 0$)

$$E[\zeta_k \zeta_l^T] = \boldsymbol{Q}_k \delta_{kl}, \tag{3.18}$$

where $\boldsymbol{Q}_k$ is related to the spectral density matrix $\tilde{\boldsymbol{Q}}(t)$ of the continuous white noise as

$$\boldsymbol{Q}_k = \int_{t_k}^{t_{k+1}} \boldsymbol{G}_v(t_{k+1}, \tau)\tilde{\boldsymbol{Q}}(t)\boldsymbol{G}_v^T(t_{k+1}, \tau)\,\mathrm{d}\tau. \tag{3.19}$$

A more general treatment of the discretization of a continuous-time stochastic system can be found in Gelb (1974) and Bar-Shalom et al. (2001).

The position offset, $\delta\boldsymbol{x}_k \in \mathbb{R}^2$ is still given in the $\{n\}$ system, so the drift $\boldsymbol{v}_k^{b'}$ must be transformed into the $\{n\}$ system using the 2-dimensional rotation matrix

$$R_{b'}^n = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{3.20}$$

i.e.

$$\boldsymbol{v}_k^n = R_{b'}^n v_k^{b'}. \tag{3.21}$$

Extended State-Space Model

Defining the new state vector $\begin{bmatrix} \delta\boldsymbol{x}_k^n \\ v_k^{b'} \end{bmatrix} \in \mathbb{R}^4$, the new state-space model becomes

$$\begin{bmatrix} \delta\boldsymbol{x}_{k+1}^n \\ \boldsymbol{v}_{k+1}^{b'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{R}_{b'}^n \\ \boldsymbol{0}_{2\times2} & \boldsymbol{G}_v \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_k^n \\ \boldsymbol{v}_k^{b'} \end{bmatrix} + \begin{bmatrix} \boldsymbol{R}_{b'}^n & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma}_k^{b'} \\ \boldsymbol{\zeta}_k^{b'} \end{bmatrix}, \tag{3.22}$$

where $\boldsymbol{\zeta}_k^{b'}$ and $\boldsymbol{\gamma}_k^{b'}$ are white noise sequences with

$$E[\boldsymbol{\zeta}_k^{b'}(\boldsymbol{\zeta}_k^{b'})^T] = \boldsymbol{Q}_k^{b'}\delta_{kl}, \tag{3.23}$$
$$\tag{3.24}$$

and

$$E[\boldsymbol{\gamma}_k^{b'}(\boldsymbol{\gamma}_k^{b'})^T] = \boldsymbol{\Gamma}_k^{b'}\delta_{kl}. \tag{3.25}$$

This concludes the development of the new state-space model. Letting $\boldsymbol{\xi}^* = \begin{bmatrix}(\delta\boldsymbol{x}_k^n)^* \\ (\boldsymbol{v}_k^{b'})^*\end{bmatrix}$, where the asterisks again indicate that this is a filter model, the model can be written on the form

$$\boldsymbol{\xi}_{k+1}^* = \boldsymbol{F}^*\boldsymbol{\xi}_k^* + \boldsymbol{\nu}_k^*, \tag{3.26}$$
$$\boldsymbol{z}_k^* = \boldsymbol{h}^*(\boldsymbol{\xi}_k^*, \tilde{\boldsymbol{x}}_k) + \boldsymbol{w}_k^*, \tag{3.27}$$

which is standard state-space form with white noise

$$\boldsymbol{\nu}_k^* = \begin{bmatrix}\boldsymbol{R}_{b'}^n & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2}\end{bmatrix}\begin{bmatrix}\boldsymbol{\gamma}_k^{b'} \\ \boldsymbol{\zeta}_k^{b'}\end{bmatrix}, \tag{3.28}$$

where

$$E[\nu_k^*(\nu_k^*)^T] = \begin{bmatrix}\boldsymbol{R}_{b'}^n & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Gamma}_k^* & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{Q}_k^*\end{bmatrix}\begin{bmatrix}\boldsymbol{R}_n^{b'} & \boldsymbol{0}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2}\end{bmatrix}. \tag{3.29}$$

Because of high computational requirements, the 4-state extended state-space model (3.22) is not well suited for point mass filter implementation. The model was therefore implemented in a particle filter framework. Results from real AUV data are presented in Chapter 5.

3.3 Measurement Model

In Section 2.4.2 the general truth model measurement equation was presented as

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k = \boldsymbol{h}(\tilde{\boldsymbol{x}}_k + \delta\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{3.30}$$

where in underwater applications the function $\boldsymbol{h}(\boldsymbol{x}_k)$ denotes the true sea depth at position $\boldsymbol{x}_k$ in some vertical datum. Since the true sea depth is not known, it has to be approximated by a map data base, such that the corresponding filter model measurement equation becomes

$$\boldsymbol{z}_k = \boldsymbol{h}^*(\boldsymbol{x}_k^*) + \boldsymbol{w}_k^*, \tag{3.31}$$

where the function $\boldsymbol{h}^*(\cdot)$ is the map depth at the horizontal position in question. As stated in (3.1), the measurements $\boldsymbol{z}_k$ are given as a combination of the vehicle depth and the vehicle altitude above the sensor footprint on the sea floor. The measurement noise $\boldsymbol{w}_k$ therefore includes contributions from

- The bathymetric sensor,
- Depth sensor noise, including pressure-to-depth conversion,
- Lever arm compensation between pressure sensor and bathymetric sensor,
- Map errors, including interpolation errors.

The following section will focus on the depth sensor noise. Since errors in the pressure-to-depth conversion or erroneous tide compensation leads to a constant or slowly varying depth error, the filter model measurement equation can be rewritten as

$$\boldsymbol{z}_k = \boldsymbol{h}^*(\boldsymbol{x}_k^*) + \boldsymbol{I} \cdot b_k^* + \boldsymbol{w}_k^*, \tag{3.32}$$

where $\boldsymbol{I}$ is an appropriately sized identity matrix, and b_k is a constant (or slowly varying) depth bias. For the case of a constant depth bias, one has

$$b_{k+1}^* = b_k^*. \tag{3.33}$$

3.3.1 Dealing With the Depth Bias

Essentially, the depth bias can be dealt with in two ways. The more straight-forward approach is to operate on relative depth profiles only, removing the mean from the measured depth profile as well as from the map profile. The second approach is to include the depth bias in the state vector, such that it is estimated along with horizontal position.

Relative Depth Profiles

When this approach is taken, the algorithms are run on relative measurement profiles only, i.e. the measurement vector $\boldsymbol{z}_k$ is replaced by

$$\boldsymbol{z}_{k,\text{rel}} = \boldsymbol{z}_k - \bar{\boldsymbol{z}}_k, \tag{3.34}$$

where $\bar{\boldsymbol{z}}_k$ denotes the mean of the measurement vector, given by $\bar{\boldsymbol{z}}_k = \frac{1}{N_b} \sum_{i=1}^{N_b} z_{i,k}$, where N_b is the number of measurement beams. The corresponding depth map profile is similarly replaced by

$$\boldsymbol{h}_{\text{rel}}^*(\cdot) = \boldsymbol{h}^*(\cdot) - \overline{\boldsymbol{h}^*(\cdot)}. \tag{3.35}$$

Using relative depth profiles is a simple solution to the depth bias problem, and in many cases it works well, as will be shown in Chapter 5. However, there are several potential problems related to it. Most important, the use of relative depth profiles will not

work in certain terrain types. One example is self-similar terrain, in which very similar terrain profiles can be found at different depths. In many cases, the only way to distinguish such similar profiles would be to compare their absolute depth. In other words, using relative depth profiles will increase the risk of multiple peaks in the posterior probability density. Another example of terrain not suited for relative terrain navigation is a linear beach. If the profile is taken along the beach, i.e. perpendicular to the direction of the slope, the relative profiles will be flat. On the other hand, if the profile is taken in the direction of the slope, all relative profiles will be sloped and equal for any distance to the shore. In either case, relative profile matching will not give any information about the position of the vehicle along the slope. It should be noted that if there is no terrain variations across the slope, it is not possible to determine the vehicle position in this direction, regardless whether relative or absolute terrain profiles are used.

Another disadvantage with the relative approach is that the depth bias itself is not estimated. The magnitude of the depth bias may be of interest in many cases. However, it would be possible to obtain a depth bias estimate by comparing the measured depth profile with the profile from the map after convergence of the terrain navigation algorithm.

Estimating the Depth Bias

The more general way of dealing with the depth bias is to estimate it as an additional state in the state vector. Starting with the conventional 2-dimensional filter model (2.32)–(2.33), the extended 3-dimensional state vector becomes

$$\boldsymbol{\xi}_k^* = \begin{bmatrix} \delta x_{N,k}^* \\ \delta x_{E,k}^* \\ b_k^* \end{bmatrix}. \tag{3.36}$$

Consequently, the new 3-dimensional process model becomes

$$\boldsymbol{\xi}_{k+1}^* = \boldsymbol{\xi}_k^* + \boldsymbol{v}_k^*, \tag{3.37}$$

where the process noise $\boldsymbol{v}_k \in \mathbb{R}^3$ is now a 3-dimensional white noise sequence with covariance matrix $\boldsymbol{Q}_k$. Note that for the special case of constant depth bias, like in (3.33), the third component of $\boldsymbol{v}_k$ equals zero for all k, and $(\boldsymbol{Q}_k)_{3,3} \equiv 0$.

The measurement equation in the new 3-dimensional filter model can be written as

$$\boldsymbol{z}_k = \tilde{\boldsymbol{h}}^*(\boldsymbol{\xi}_k^*; \tilde{\boldsymbol{x}}_k) + \boldsymbol{w}_k^*, \tag{3.38}$$

where the sea map function $\tilde{\boldsymbol{h}}^* : \mathbb{R}^3 \to \mathbb{R}_{N_b}$ is defined as

$$\tilde{\boldsymbol{h}}^*(\boldsymbol{\xi}_k; \tilde{\boldsymbol{x}}_k) = \boldsymbol{h}^* \left(\begin{bmatrix} \tilde{x}_{k,N} + \delta x_{k,N} \\ \tilde{x}_{k,E} + \delta x_{k,E} \end{bmatrix} \right) + \boldsymbol{I}_{N_b \times N_b} \cdot b_k, \tag{3.39}$$

where $\boldsymbol{h}^*(\cdot)$ is the same 2-dimensional map function that was used in the previous models.

Using the new filter model, the optimal Bayesian filter equations are now readily derived using the general Bayesian equations (2.18) and (2.19). The time update equations in the 3-dimensional model becomes

$$p(\delta\boldsymbol{\xi}_{k+1}^*|\boldsymbol{Z}_k) = \int_{\mathbb{R}^3} p_{v_k^*}(\delta\boldsymbol{\xi}_{k+1}^* - \delta\boldsymbol{\xi}_k^*)p(\delta\boldsymbol{\xi}_k^*|\boldsymbol{Z}_k)\,\mathrm{d}\delta\boldsymbol{\xi}_k^*, \tag{3.40}$$

which is identical to the corresponding 2-dimensional equation (2.44), except that the integral is now 3-dimensional, representing a 3-dimensional convolution.

The corresponding time update equation becomes

$$p(\delta\boldsymbol{\xi}_k^*|\boldsymbol{Z}_k) = \alpha_k^{-1} p_{w_k^*}(\boldsymbol{z}_k - \tilde{\boldsymbol{h}}^*(\delta\boldsymbol{\xi}_k^*; \tilde{\boldsymbol{x}}_k))p(\delta\boldsymbol{\xi}_k^*|\boldsymbol{Z}_{k-1}), \tag{3.41}$$

where

$$\alpha_k = \int_{\mathbb{R}^3} \left[p_{w_k^*}(\boldsymbol{z}_k - \tilde{\boldsymbol{h}}^*(\delta\boldsymbol{\xi}_k^*; \tilde{\boldsymbol{x}}_k))p(\delta\boldsymbol{\xi}_k^*|\boldsymbol{Z}_{k-1})\right]\mathrm{d}\delta\boldsymbol{\xi}_k^*.$$

The 3-Dimensional Point Mass Filter

Having extended the 2-dimensional filter model to three dimensions, it is natural to extend the 2-dimensional point mass filter of Section 2.4.4. The derivation is straightforward and starts with defining the 3-dimensional search grid, using M, N and Q grid points in each spatial direction. Let Δ be the resolution of the grid, which is assumed to be equal in all 3 directions. It should be noted that this assumption is not necessary, and relaxing this assumption would simply result in a substitution of Δ^3 by $\Delta_N\Delta_E\Delta_D$ in the equations below. Here the resolution is kept uniform simply for notational simplicity. As before, at each grid point the density $p(\boldsymbol{\xi}_k^*|\boldsymbol{Z}_k)$ is approximated by a probability mass weight

$$p(\boldsymbol{\xi}_k^*(i,j,l)|\boldsymbol{Z}_k) \quad i = 1,\ldots,M,\ j = 1,\ldots,N,\ l = 1,\ldots,Q. \tag{3.42}$$

The PMF measurement update equation now becomes

$$\begin{aligned} &p\left(\xi_k^*\left(i,j,l\right)|\boldsymbol{Z}_k\right) \\ &\quad = \alpha_k^{-1} p_{\mathbf{w}_k^*}\left(\mathbf{z}_k - \tilde{\mathbf{h}}^*\left(\delta\xi_k^*\left(i,j,l\right); \tilde{\mathbf{x}}_k\left(i,j,l\right)\right)\right) \cdot p\left(\xi_k^*\left(i,j,l\right)|\boldsymbol{Z}_{k-1}\right), \end{aligned} \tag{3.43}$$

where

$$\alpha_k = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{Q}\left[p_{\mathbf{w}_k^*}\left(\mathbf{z}_k - \tilde{\mathbf{h}}^*\left(\delta\xi_k^*\left(i,j,l\right); \tilde{\mathbf{x}}_k\left(i,j,l\right)\right)\right) \cdot p\left(\xi_k^*\left(i,j,l\right)|\boldsymbol{Z}_{k-1}\right)\right]\Delta^3.$$

Likewise, the PMF time update now becomes

$$p\left(\xi_{k+1}^{*}(i,j,l)\,|\boldsymbol{Z}_k\right) = \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=1}^{Q}\left[p_{\mathbf{v}_k^*}\left(\xi_{k+1}^{*}(i,j,l)-\xi_k^{*}(m,n,q)\right)\cdot p\left(\xi_k^{*}(m,n,q)\,|\boldsymbol{Z}_k\right)\right]\Delta^3, \quad (3.44)$$

which is a discrete 3-dimensional convolution. The (MMSE) position estimate is computed as

$$\hat{\xi}_k = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{Q}\xi_k^{*}(i,j,l)\cdot p\left(\xi_k^{*}(i,j,l)\,|\boldsymbol{Z}_k\right)\Delta^3, \quad (3.45)$$

and $\hat{\mathbf{x}}_{k,P} = \tilde{\mathbf{x}}_k + \delta\hat{\mathbf{x}}_{k,P}$. The corresponding error covariance matrix is given by

$$\hat{\mathbf{P}}_k = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{Q}\left[\left(\xi_k^{*}(i,j,l)-\hat{\xi}_k\right)\cdot\left(\xi_k^{*}(i,j,l)-\hat{\xi}_k\right)^T p\left(\xi_k^{*}(i,j,l)\,|\boldsymbol{Z}_k\right)\right]\Delta^3. \quad (3.46)$$

3-Dimensional Particle Filters and Sigma Point Kalman Filters

The equations for the particle filters in Section 2.4.5 and the Sigma Point Kalman Filter in Section 2.4.7 were presented for an arbitrary number of states in the state vector. These general algorithms are readily applied to the 3-dimensional state-space model. However, one should bear in mind that in the case of the particle filters, the number of particles needed in order to represent the probability distributions increases as the dimension of the state space model increases. The same is true in the SPKF; the number of sigma points required increases with the state vector dimension.

3.4 Integration of the Terrain Navigation System and the Main Navigation System

As stated in Section 1.3, the focus of this thesis is on the terrain navigation algorithms and how these can be utilized in a loosely-coupled terrain navigation system. However, for the terrain navigation results to be of use to the main navigation system, be it an INS system or another kind of navigation system, the terrain navigation updates have to be integrated into the main system. The principle is outlined in Figure 3.6. Navigation data from the main system are used as input to the TerrNav system and combined with the bathymetric data to obtain measurements in a geographic measurement frame. These measurements, together with the position from the main navigation system (what has been labeled $\tilde{\boldsymbol{x}}_k$ in previous sections) are used in the terrain navigation algorithms in the

box labeled 'Terrain Correlator'. Finally, when the terrain navigation algorithms have converged, the position update is sent back the navigation system, where it is treated in the same manner as another position update.

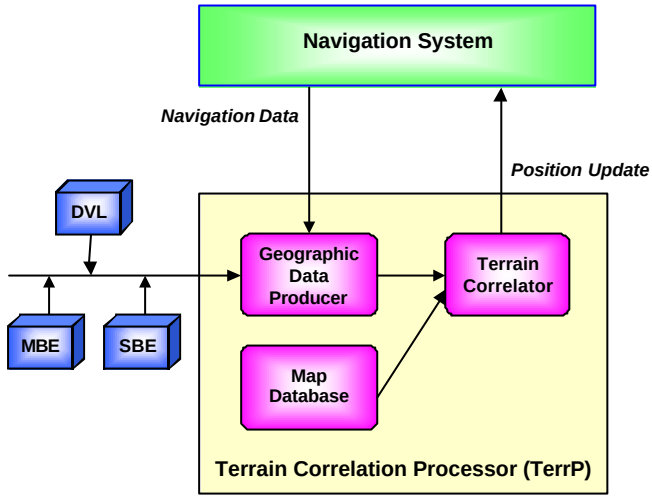


Figure 3.6: *Integration of the TerrNav and INS system. The blue boxes are the bathymetric measurement sensors, the yellow box is the TerrNav system and the green box is the main navigation system. Figure with permission from Jalving et al. (2004b)*

There are several possible problems with the integration scheme outlined above. As the main system is in most cases Kalman filter based, it is assumed that the measurements fed into it are uncorrelated with the errors in the main navigation system. However, as input from the main system is used in the terrain navigation algorithms, the TerrNav update will be correlated with the error states of the main system. Such a feedback loop may cause instabilities and inaccuracies in the main Kalman filter. Another problem is that only Gaussian measurement updates can be modeled in the Kalman filter. In the terrain navigation problem, non-Gaussian densities are commonplace. For example, multimodal posteriors sometimes occur, especially in less suited terrain.

A number of ad hoc measures can be taken in order to minimize the effects mentioned above. First of all, a proper convergence criterion must be defined, such that a TerrNav update is not sent to the main navigation system until the algorithms report that the uncertainty in the measurement is acceptable. The convergence criterion can be formulated such that only updates which have a "near Gaussian" posterior are used in the main system. In addition, integrity checks, based on the statistics of the differences between the measured map profile and the map profile at the converged TerrNav solution can be formulated. Also, in order to minimize effects of correlations between consecutive TerrNav fixes, the algorithms should be restarted whenever a fix is used in the main

navigation system.

The issues related to the integration between the two systems have not been the main focus of this doctoral work. However, these issues are extremely important when implementing a real-time terrain navigation system, especially due to the fact that the TerrNav algorithms sometimes are overconfident and yield false fixes, as will be shown in Chapter 5.

3.5 Terrain Dependency

An important prerequisite for TerrNav methods to work, is that the terrain has a sufficient amount of terrain variation. The degree of terrain variation needed varies with the quality of the sensors and with the resolution of the map. A terrain that looks flat with a crude terrain sensor may possess a lot of terrain information when a high resolution sensor is used. At the same time, a high sensor resolution is of little value if the map resolution is not comparable. In AUV terrain navigation scenarios using MBEs, the map is often the limiting factor, as sea maps are often created using measurements from surface vessels.

The task of assigning a measure for the suitability of a certain terrain for TerrNav is a difficult one. This topic is not among the main focus areas of this thesis, so it will only be briefly commented on here.

One obvious way to investigate the suitability of the terrain before planning a mission would be to do TerrNav simulations in different areas of the terrain and investigate the accuracy and precision of the solution. As this may be a time consuming approach, it would be desirable to develop a measure for the suitability, assigning to each point in the database a score for the terrain information. An obvious difficulty with such an approach is that the suitability varies with the nature of the trajectory of the vehicle. When crossing a valley, for example, the direction at which the vehicle approaches the valley is crucial to the quality of the TerrNav position fix that can be obtained. One single numerical value for the terrain suitability at each map cell is not able to reflect this trajectory dependency.

Several attempts of characterizing the terrain information exist in the literature. Both Bergman (1999) and Nygren (2005) propose to use an information measure derived from the CRLB (see Section 2.6). If the true sea depth at position $\boldsymbol{x}$ is denoted by $h(\boldsymbol{x})$, and this is measured by a single measurement

$$z = h(\boldsymbol{x}) + w, \tag{3.47}$$

where the meaurement noise is assumed to be Gaussian, $w \sim \mathcal{N}(0, q)$, the Fisher information matrix at the true position $\boldsymbol{x}_0$ can be written as (Bergman, 1999)

$$J(\boldsymbol{x}) = \frac{1}{q} E[\nabla_x h(\boldsymbol{x}_k) \nabla_x h(\boldsymbol{x}_k)^T], \tag{3.48}$$

which means that the average value of the norm of the gradient vector $\nabla_x h(\boldsymbol{x})$ can be taken as a measure for the terrain information in an area. There are, however, some

obvious disadvantages with this approach. The trajectory dependence is not accounted for. Also, in the extreme case of a constant, linear slope, the average gradient component in the direction of the slope, and hence also the average gradient norm, will be constant and possibly large, indicating good TerrNav properties. However, in such a terrain, the TerrNav accuracy in the direction perpendicular to the slope will be very low. The gradient norm is also unable to tell if the terrain is self-similar, i.e. if similar terrain repeats itself at different depths. As discussed in Section 3.3.1, such a terrain is not well suited when relative depth measurements are used.

Frequency content of the terrain is another natural approach when characterizing the terrain. Bar-Gill et al. (1994) define an entropy function for the terrain information, based on a Fast Fourier Transform (FFT) analysis of the terrain. This analysis is used for improving the performance of TerrNav in an aircraft/missile application.

In Mandt (2001), an additional simple terrain information measure is introduced, based on the uniqueness of the different depth values in the area. Some simulations using a real underwater map are reported, indicating that the uniqueness measure actually outperforms both the gradient and frequency approaches described above. A more thorough analysis on more depth data is however needed in order to draw any definitive conclusions.

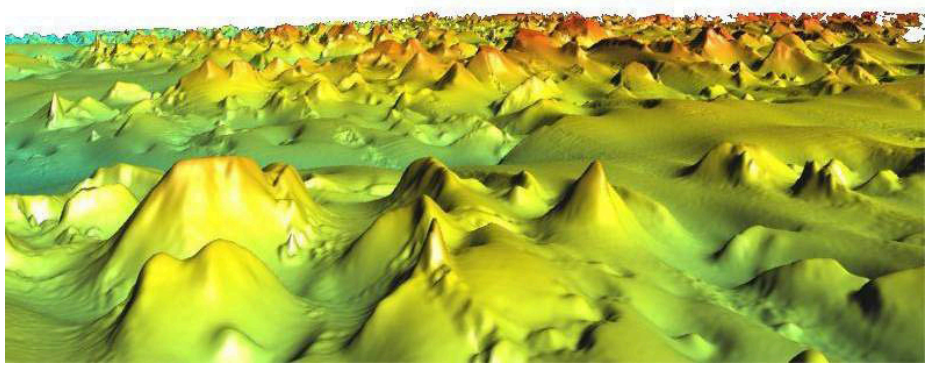


Figure 3.7: *Terrain from the Ormen Lange area off the west coast of Norway. This is an example of self-similar terrain. Courtesy Norsk Hydro ASA.*

4

Map Databases

MAP databases are essential to the success of terrain navigation. The accuracy and resolution of the map database clearly restrict the obtainable navigation accuracy in an area. As most terrain maps have been created using many of the same sensors as those used in terrain navigation, in modern map databases usually multibeam echo sounders, many of the database error contributions are of the same type as those of the bathymetric measurements of the vehicle itself. In this chapter, different types of map databases are first discussed. Then the different error sources in gridded map databases, the map representation used in this thesis, are identified and described.

Sea floor mapping and geographical information systems (GIS) constitute a research area of their own. This thesis does not primarily focus on mapping, and consequently only a high-level description of the various ideas and algorithms of mapping is provided, concentrating on the issues relevant to terrain navigation.

4.1 Map Representations

Modern sea maps are based on measurements from a wide range of different origins, from low-quality manual measurements to high-quality multibeam echo sounder data. It is important to distinguish between sea maps and digital terrain models (DTMs). Sea maps are conventional maps used for ship navigation and contain a wide variety of information relative to mariners, like the locations of lights and beacons, geographical names, depth contours, depth soundings etc. Digital sea maps are digitalized versions of conventional paper charts and are based on data from a number of different sources. The quality of the depth information in these maps may vary, and often it is simply a digitalization of depth information from old paper charts. The International Hydrographic Organization (IMO) has defined the S-57 data exchange standard (International Hydro-

graphic Organization, 1992), which is the governing data exchange standard for digital hydrographic data today.

DTMs, on the other hand, are digital models of the actual terrain. Although one often speaks of terrain maps when discussing terrain aided navigation, the algorithms actually use DTMs. In this thesis, the term 'map databases' refers to DTMs, or more specificly gridded DTMs in most cases. The term DTM was originally introduced by two American engineers at the Massachusetts Institute of Technology in the 1950s, and their definition of the term was: A "DTM is simply a statistical representation of selected points with known X, Y, Z coordinates in an arbitrary coordinate field" (Miller and LaFlamme, 1958). A thorough discussion on digital terrain modeling can be found in El-Sheimy et al. (2005).

A DTM can have several different data structures, the two most common being the grid data structure and the Triangular Irregular Networks (TINs). In the terrain navigation examples in this thesis only gridded maps are used, but the algorithms are not restricted to any particular type of data structure. However, a disadvantage with TINs is the time required when looking up the depth at an arbitrary point. In many cases, the large number of map look-ups needed in terrain navigation would take too long to process in a TIN-based data structure.

4.1.1 Gridded Maps

A gridded DTM simply consists of the sea depths at fixed horizontal grid points. The grid cells may be rectangular, square or curvilinear, which is the case when the grid points are defined by latitude and longitude coordinates. In most cases, however, the grid cells are squares.

The advantage of gridded maps is that the data structure is similar to the array storage structure of digital computers. Looking up a depth in the grid, storing and manipulating data can therefore be done quite efficiently. Each grid point is surrounded by a corresponding grid cell, and the grid point is defined to be at the center point of the cell. This is illustrated in Figure 4.1. Several interpretations concerning the depth values in the grid cells are possible. In this thesis, the depth at each grid node is interpreted to correspond to the sea depth at the grid point only. Depths between the grid points are estimated using interpolation, cf. Section 4.3. Another common interpretation is that of *category grids*, in which the depth is considered constant within each grid cell. The terrain model is in this case built up of a number of squares (or rectangles) with constant depth.

The accuracy of a gridded DTM representation of the terrain is highly dependent on the resolution of the grid. Because of this, certain terrain forms such as ridges, small peaks etc. may be between grid points and consequently poorly represented. The only way to increase the ability of representing such features is to refine the resolution of the grid. However, in flat areas this leads to a number of unnecessary grid points. Gridded DTMs may therefore be ineffective with regards to data storage. One possibility to get around this problem is to use subgrids with higher resolution in areas with large terrain

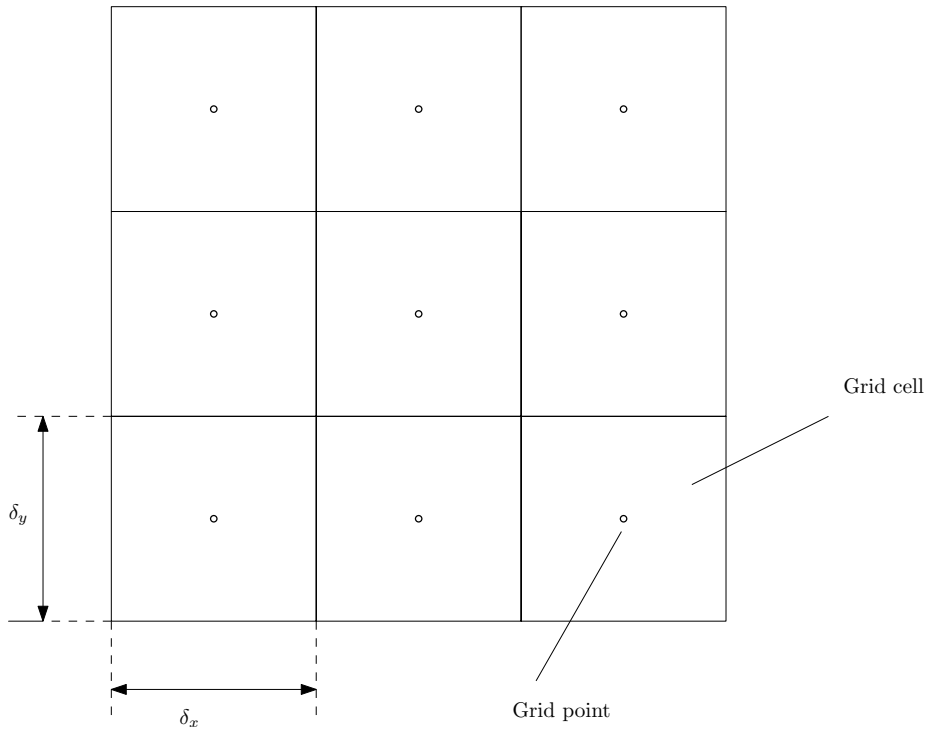


Figure 4.1: *Illustration of DTM grid cells and grid points.*

variations. However, this increases the complexity of the model, and the grid size is therefore normally held constant throughout large areas.

Gridding Algorithms

The raw measurements from a mapping surface consists of a number of more or less irregularly spaced depth measurements. Modern mapping surveys are usually conducted using multibeam echo sounders, either from a surface ship or from a towed fish or underwater vehicle. The survey is usually executed using a regular trajectory (e.g. a lawnmower pattern) in order to cover the survey area entirely. The process of data gridding involves making a regular grid based on these raw measurements. Usually some kind of a wild point filter is applied first. The resolution of the grid must be chosen such that each grid cell contains a sufficient number of raw grid points.

The most direct approach to data gridding, is the *moving average* approach, in which the depth value at the grid point is taken as a (possibly weighted) average of neighboring measurement points. The points to be included in the averaging are determined by some kind of search algorithm (El-Sheimy et al., 2005). Two examples of search algorithms are the n-nearest neighbor algorithm, which simply chooses the n measurement points closest to the grid point, and the radius approach, which include all measurements with

a distance below a certain value. A variation of this approach is to choose all grid points within the grid cell surrounding the grid point. The averaging is normally done such that points close to the grid point are assigned a higher weight than those farther away.

An alternative to the moving average approach is *linear projection gridding* (El-Sheimy et al., 2005), in which local trend surfaces at the locations of the measurement points are first made, using the surrounding measurement points. The depth at the grid point is then calculated as a weighted average of the trend surfaces from the n nearest measurement points, projected to the location of the grid point.

4.1.2 Triangulated Irregular Networks

The Triangulated Irregular Network model is an alternative to grid based DTMs. It was developed in the early 1970s (Peucker et al., 1978) as an attempt to facilitate more efficient storage of DTMs. The idea behind TIN is to build a surface based on irregularly sampled points. The points are connected by lines to form triangles and within each triangle the surface is usually represented using a plane. In this way, the model is built up using the actual measurement points, or a subset thereof. The triangulation can be found using a Delaunay triangulation (El-Sheimy et al., 2005, Sec. 3.7).

The advantage of TINs is that terrain can be sampled more efficiently, using a high number of nodes in areas with rough terrain and fewer nodes in flat areas. A lot of research has been done in order to choose which points to include in the TIN, as well as how to store the TIN efficiently, (Peucker et al., 1978; El-Sheimy et al., 2005).

Although TINs are able to store the terrain model efficiently, they are less suited for looking up arbitrary depths effectively. Algorithms for computing slopes and trends are also more complex for TINs than for grid models. Because of this, TINs are not very well suited for terrain navigation and will not be discussed further in this thesis.

4.1.3 Contour Based Map Representations

Contours, i.e. depth isocurves, are probably the most used terrain representation technique. Traditional sea charts are mostly contour based, possibly with some additional depth soundings. Contour maps are usually good for human visualization of the terrain, but their disadvantage is that they contain no information on the terrain between the contours. Methods for digitizing contour-based paper charts, as well as converting from digitized contour maps to grid models have been developed. Some of them are described in El-Sheimy et al. (2005).

In order to use the algorithms described in this thesis on contour based map data, the contours must first be converted to a gridded DTM. The problem with most commercially available contour maps, however, is that the resolution and quality is not good enough to be used for precise terrain navigation.

4.2 Error Sources in Map Databases

An underwater terrain database is the result of an advanced processing chain, from the data collection, via wild point filtering, to data gridding. It is natural to divide the error sources into two groups: those related to the raw data measurements and those related to the algorithms used for building the DTM.

4.2.1 Errors in Raw Measurement Data

The errors in the raw measurement data are often subdivided into *systematic errors*, *random errors*, and *gross errors* or *outliers* (Bjørke and Nilsen, 2007), according to their nature of origin. The following is a brief description of these error types.

Systematic Errors

Systematic errors are errors that influence all or parts of the measurements in the survey. Typical examples are errors due to incorrect roll and pitch compensation, mounting errors, incorrect sound velocity profile and incorrect tide compensation. Systematic errors are often possible to compensate for by using calibration. An example of a calibration method is given in Bjørke (2005).

Random Errors

Random errors are errors due to the measurement uncertainty of the bathymetric sensor and are assumed to be spatially uncorrelated. The random errors can be estimated if one has several different terrain models of the same area, provided other error sources have already been compensated for. A lot of research efforts have been put into estimating random errors in DTMs. Examples are Jakobsson et al. (2002) and Bjørke and Nilsen (2007).

Gross Errors

Gross errors or outliers are statistically unsound measurements. The source of these errors may for example be multipath reflection, i.e. sound pulses that are not direct returns from the sea floor. Gross errors should be removed from the data set before generating the terrain model. Traditionally this was done manually, but over the last two decades, several methods for automatic cleaning of the data sets have been proposed, e.g. Huang et al. (1999) and Debese (2001).

4.2.2 Algorithmic Error Sources

Algorithmic error sources stem from the algorithm that has been used for creating the DTM. In gridded DTMs the gridding algorithm generally smoothes the surface by weighting the grid cells within the cell. If the same measurement point is used to compute the

grid value in several grid nodes, as is the case for example in the n-nearest neighbor selection scheme, correlations between the grid nodes are also introduced.

4.3 Interpolation Techniques in Gridded Map Databases

In a gridded DTM, the depth between the grid points are estimated using some kind of interpolation technique. It should be noted that interpolation can also be used as part of the gridding algorithm, in order to compute the depths at the grid nodes from the raw data. In this case, errors resulting from this interpolation are part of what was here defined as algorithmic error sources in Section 4.2.2. When interpolation has been used during the gridding process, there will consequently be error contributions from two types of interpolation in the final terrain navigation solution; gridding interpolation and DTM look-up interpolation.

Interpolation is the task of estimating the depth at a point between the reference points by using the depth values at the reference points. In gridded DTMs, the term *reference points* refers to the grid points, whereas when interpolation is used directly on raw data, it refers to the raw measurements.

Interpolation methods are often classified according to the following characteristics:

- Exact vs. inexact interpolation,
- Global vs. local interpolation,
- Deterministic vs. stochastic interpolation.

Exact interpolation methods yield surfaces that match the depth at the reference points exactly, whereas inexact interpolation methods relax this requirement, allowing deviations from the depth references in order to obtain a better overall solution. Local interpolation methods use reference points close to the interpolation points only, whereas global interpolation uses the all the reference points available, in order to estimate the global trend of the terrain. Examples of global interpolation methods are trend surface analysis (TSA), Fourier analysis and KRIGING (El-Sheimy et al., 2005). KRIGING is also an example of a stochastic interpolation method in that it incorporates statistical information on the terrain into the interpolation.

The following will focus on interpolation methods used in gridded maps, i.e. the grid points are the reference points. In a gridded DTM the global trend has already been taken care of during the gridding phase, so the focus will be on local interpolation methods.

4.3.1 Nearest Neighbor Interpolation

Nearest neighbor interpolation is the simplest possible interpolation scheme. The depth at the interpolation point is simply found by choosing the depth value at the closest of the four surrounding reference points. The nearest neighbor algorithm is computationally simple, but in areas with large variations between neighboring grid points, the

interpolated depth value may be inaccurate. It is an example of an exact interpolation method. The resulting interpolation surface, i.e. the surface obtained by interpolation of the entire map region, is discontinuous and non-smooth.

4.3.2 Linear Interpolation

Linear interpolation uses the three nearest reference points to define a planar surface, and the depth value at the interpolation point is taken as the depth of the planar surface at this point. The principle is illustrated in Figure 4.2. If the interpolation point is above the diagonal from reference point 1 to 2 (the area marked $\delta = 1$ in the figure), the planar surface is constructed from reference points 1, 3 and 4, otherwise points 1, 2 and 4 are chosen. The interpolation value is then computed as

$$z(x,y) = \delta\big(z_1+(z_4-z_3)\bar{x}+(z_3-z_2)\bar{y}\big)+(1-\delta)\big(z_1+(z_2-z_1)\bar{x}+(z_4-z_2)\bar{y}\big), \quad (4.1)$$

where

$$\bar{x} = \frac{x-x_1}{\Delta x}, \quad (4.2)$$

$$\bar{y} = \frac{y-y_1}{\Delta y}, \quad (4.3)$$

and

$$\delta = \begin{cases} 1 & \text{if } \bar{x} \le \bar{y}, \\ 0 & \text{otherwise.} \end{cases} \quad (4.4)$$

Linear interpolation is an exact interpolation method and the resulting surface is continuous but not smooth.

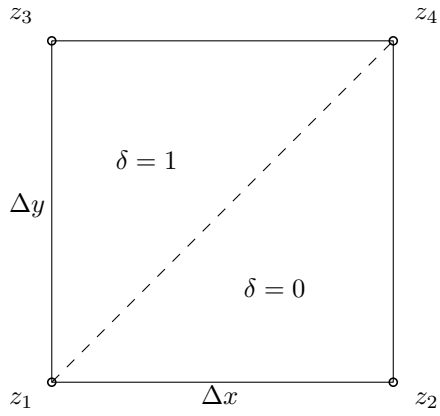


Figure 4.2: *Linear interpolation regions.*

4.3.3 Bilinear Interpolation

Bilinear interpolation constructs a bilinear polynomial of the form

$$z(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{4.5}$$

over the grid cell. The depth at the interpolation point is taken as the value of the polynomial at the point. The coefficients a_0, a_1, $a2$, and a_3 can be found using the values at the four reference points together with the fact that the interpolator is required to be exact. The resulting interpolation value becomes

$$z(x, y) = z_1 + (z_2 - z_1)\bar{x} + (z_3 - z_1)\bar{y} + (z_1 - z_3 + z_4 - z_2)\bar{x}\bar{y}, \tag{4.6}$$

where $\bar{x}$ and $\bar{y}$ are again given by (4.2)–(4.3). The bilinear interpolation method is exact by construction, and the resulting interpolation surface is continuous and non-smooth.

In the computational results in Chapter 5, bilinear interpolation has been chosen in as map database lookup interpolation method.

4.3.4 Higher Order Interpolation

Higher order interpolation methods also exist, e.g. *cubic interpolation* (El-Sheimy et al., 2005). Higher order interpolation methods are often computationally demanding and are not discussed further in this thesis.

4.4 Using Error Models in TerrNav Algorithms

In Section 2.4.2 the traditional state space model for terrain navigation was presented, and in Section 3.3.1 it was shown how the model could be extended in order to take the effects of the depth bias into account. Although the map database also may include depth bias error sources, the depth bias discussed in Section 3.3.1 is a relative bias between the map and vehicle sensor vertical references. Consequently, for terrain navigation purposes, all of the depth bias can be associated with the vehicle bathymetric sensor. It is not possible for the terrain navigation algorithms to distinguish between constant or slowly varying depth biases in the map and corresponding errors in the depth sensor. In the following discussion, the map vs. sensor depth bias effect is therefore ignored, concentrating on the pure map error sources only. It should be stressed, however, that in applications in which one is interested in the absolute depth per se, bias error sources in the map database may be of crucial importance.

The conventional terrain navigation filter measurement equation was given in (2.33), repeated here for convenience:

$$\boldsymbol{z}_k = \boldsymbol{h}^*(\boldsymbol{x}_k^*) + \boldsymbol{w}_k^*. \tag{4.7}$$

The noise term $\boldsymbol{w}_k^*$ was assumed to be zero mean and white, with covariance matrix

$$E[\boldsymbol{w}_k^* \boldsymbol{w}_l^{*T}] = \boldsymbol{R}_k^* \delta_{kl}. \tag{4.8}$$

As the noise term $\boldsymbol{w}_k^*$ contains both map and sensor noise, it can be written

$$\boldsymbol{w}_k^* = \boldsymbol{w}_{\text{map},k}^* + \boldsymbol{w}_{\text{meas},k}^*, \tag{4.9}$$

and since map and measurement noise are assumed to be independent,

$$\boldsymbol{R}_k^* = \boldsymbol{R}_{\text{map},k}^* + \boldsymbol{R}_{\text{meas},k}^*. \tag{4.10}$$

All information about the map noise, i.e. all the error sources discussed in this chapter, has to be incorporated into the white noise term $\boldsymbol{w}_{\text{map},k}^*$. Due to the different nature of all the individual map error sources, it is difficult to model these realistically using a crude white noise model. However, it is possible to model some of the main effects in this simple error model. When using a multi-measurement sensor, i.e. an MBE, each row in $\boldsymbol{R}_{\text{map},k}^*$ corresponds to one measurement beam. The individual map error contributions are modeled in the diagonal elements of $\boldsymbol{R}_{\text{map},k}^*$, whereas the off-diagonal elements take care of the correlations between the beams within the same ping. Typically, neighboring beams will hit the sea floor within the same or in neighboring grid cells. Consequently the map database values for neighboring beams will be more correlated than for beams that are farther away from each other.

5

Computational Results

In the previous chapters a number of different terrain navigation methods have been presented. In this chapter results from the methods using real AUV data are shown. The chapter is a recapitulation, and to some extent an elaboration of the results presented in the papers Ånonsen et al. (2005), Ånonsen and Hallingstad (2006), Ånonsen et al. (2007), and Ånonsen and Hallingstad (2007). In addition, a section on terrain navigation using pockmarks is included, based on Ånonsen and Hagen (2009).

5.1 Presentation of the Data Sets

5.1.1 Breiangen Data Set

Most of the results presented in the aforementioned papers were obtained using a data set from the HUGIN 1 AUV, collected on November 29, 2001 in the Breiangen area in the Oslo fjord, close to Horten. The trajectory of the vehicle is shown in Figure 5.1, together with depth contours from the area. The data set has a duration of around 4 hours, and the vehicle travels through various terrain types during the run. It starts in an area with rather rough terrain, continues through a typical underwater valley (the westernmost part of the run), into a relatively flat area where it conducts a lawn-mower survey pattern, before returning to the rough area again. The total sea depth during the run, computed the pressure sensor and the DVL measurements is shown in Figure 5.2. The vehicle was equipped with a Kongsberg Maritime EM 3000 300 kHz multibeam echo sounder. The EM3000 uses up to 127 beams in each measurement ping, but in this particular data set a maximum of 92 beams were used. A map database of the area exists, constructed from depth measurements from an EM 1002 MBE mounted on a surface ship. The depth data were cleaned, wild point filtered and used to construct a gridded map database with 10

m horizontal resolution. The EM 1002 operates at a frequency of 95 kHz, which makes the map data statistically independent from the vehicle EM3000 bathymetric data.

During the run, the AUV was followed closely by a surface ship, taking continuous DGPS/USBL measurements. The vehicle navigation data were post-processed using the post-processing navigation package NavLab (Gade, 2004), yielding a reference solution with an expected position accuracy of 1 meter (1σ). This reference solution was used as ground truth for all the terrain navigation tests and is shown in Figure 5.1.

As noted in Section 4.3.3, in the results presented here, bilinear interpolation was used in the map lookups the terrain navigation algorithms. In the Bayesian methods, unless otherwise stated, the MMSE estimate is taken as the terrain navigation estimate. It should be noted, however, that this estimate in most cases coincides with the MAP estimate after convergence of the algorithms.

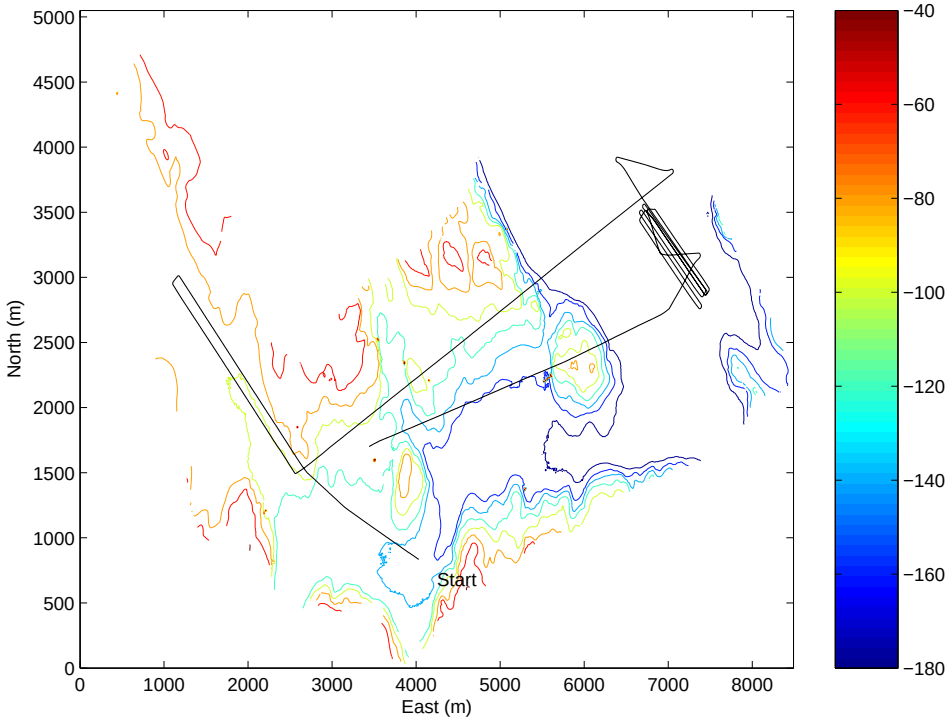


Figure 5.1: *AUV trajectory and depth contours from Breiangen data set.*

5.1.2 Pockmark Data Set

In the pockmark application to be presented in Section 5.2.5, a map constructed from data from a Kongsberg Maritime EM710 MBE was used for terrain navigation simulations. The data and the simulation process are further described in Section 5.2.5.

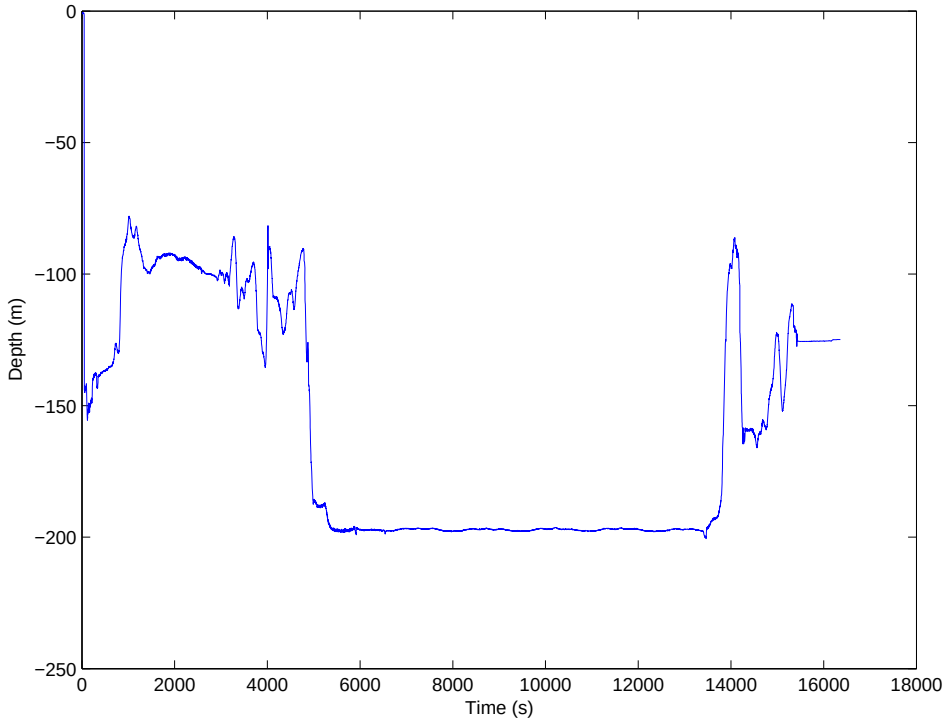


Figure 5.2: *Total sea depth during Breiangen run, computed from DVL and pressure sensor measurements.*

5.2 Results

5.2.1 Comparison of the TERCOM and PMF algorithms in 2D

The results in this section are based on the paper Ånonsen et al. (2005), in which a comparison between the TERCOM and PMF algorithms was presented. All the tests were done offline in MATLAB^{TM}, using the data set described in Section 5.1.1. The EM 3000 MBE was used as bathymetric sensor. Both algorithms were run using absolute depth profiles, ignoring potential depth offset problems.

In the implementation of the PMF, an adaptive grid approach was used (Bergman, 1999), utilizing the fact that many of the point masses will be very close to zero. By setting all point masses below a certain threshold to zero at each time step, the number of computations needed is effectively reduced, using sparse matrices in MATLAB^{TM}. Each time the number of non-zero point masses gets above or below certain limits, the PMF grid is decimated or refined, such that a fine grid is used in areas with low estimate uncertainty, whereas in areas where the uncertainty is higher, a coarser grid is used. However, as the map in this test had a resolution of 10 m, the PMF grid resolution was not allowed to get below a certain predefined limit. The adaptive grid approach is very useful when large search areas with high initial uncertainty are used, but for smaller areas, like the one treated here, it is not strictly necessary.

In real-time applications the algorithms are typically restarted frequently, and an estimate is computed from a series of MBE pings, cf. Section 3.4. The same approach was therefore followed in these tests. To test the algorithm sensitivity to errors in the a priori estimates, an artificial error was added to the initial INS estimate.

Table 5.1 shows a summary of the results obtained by the two algorithms in different areas of the run. A measurement update frequency of 0.5 Hz was used, and the algorithms were run on terrain profiles consisting of 100 pings each time they were restarted, corresponding to a profile length of around 400 m. Position estimates from the real-time navigation system, with an added error of 100 m in each horizontal direction were used as initial estimates. A 5 m grid resolution was used for TERCOM, whereas the minimum resolution of the adaptive PMF grid was set to 2 m. The final error after 100 time steps was computed as the total horizontal error of the estimate compared to ground truth. The errors listed in Table 5.1 are the minimum and maximum errors obtained among the different 400 m profiles in areas with different terrain characteristics. In most of the runs, except in the flat area, the errors were closer to the minimum values. There were, however, occasional false fixes (wild points), with low estimated PMF variance (respectively stable TERCOM estimates) but large errors, even in the well-suited areas.

Both algorithms performed quite well in the rough areas, with typical final errors around the map resolution (10 m), and they converged rather quickly to a stable estimate, typically after 30-40 time steps, sometimes after as few as 2-3 time steps. In the rough areas the estimates from each method after convergence were quite similar, though the nature of their estimates was very different. Figure 5.3 shows the errors from one of the tests in the rough terrain. Notice that the PMF estimate is much smoother than the

Time/s	Terrain properties	PMF error/m	TCM error/m
644-1644	Rough, valley	6-13	8-13
1644-2843	Valley	15-82	15-86
2843-5443	Rough	6-14	6-25
5443-13440	Flat	30-120	30-120
13440-1441	Rough	7-15	7-20

Table 5.1: *Summary of results from MBE data in different parts of the run.*

TERCOM estimate, which has sudden jumps even after 90 time steps.

As expected, both algorithms performed poorly in the flat area, see Figure 5.4. Some of the major disadvantages of the TERCOM algorithm were revealed here. The TERCOM estimate is very unstable, and it has no mechanism for indicating that quality of the estimate is poor. In contrast, PMF shows a much smoother behavior, and the covariance matrix reflects the uncertainty in the estimate. Some of the same TERCOM problems were also evident in the 'valley' area, where the error in the principal direction along the valley was large, a known phenomenon for terrain navigation in valleys or near beaches. Again, the PMF was able to detect this uncertainty through its covariance matrix.

A problem with the PMF throughout all these tests was that the estimate was overconfident. Though the estimated covariance matrix was able to reflect uncertainty in the estimate in different areas, as described above, the estimated variance in each direction was generally too low. For example, in the rough area, where the accuracy of the estimates were typically around 10 m, the estimated standard deviation in each direction was typically as low as 2 m. This is a serious problem if the PMF estimate is to be integrated in a Kalman filter based navigation system. Simulations, in which process noise and measurement noise were sampled from known distributions, suggested that this inconsistency stemmed from mismatch between the filter model and the true system.

As mentioned above, the adaptive grid approach was used in the PMF implementation. Tests done without grid adaptation show that the quality of the estimates does not suffer significantly from this approach, as long as the truncation threshold is not chosen too high. There is, however, a slightly higher risk of converging to a wrong estimate, and in real system this approach should be used with care.

Similar tests to the ones described here have been done with other data sets from the same test area, with very similar results. Both methods are robust to errors in the initial position given by the navigation system, as long as the correct position is within the search area.

Summary

In summary, the results described here section demonstrate that the 2-dimensional TERCOM and PMF algorithms both perform well in suited terrain and are able to obtain position fixes with a horizontal accuracy around that of the map resolution. Both algo-

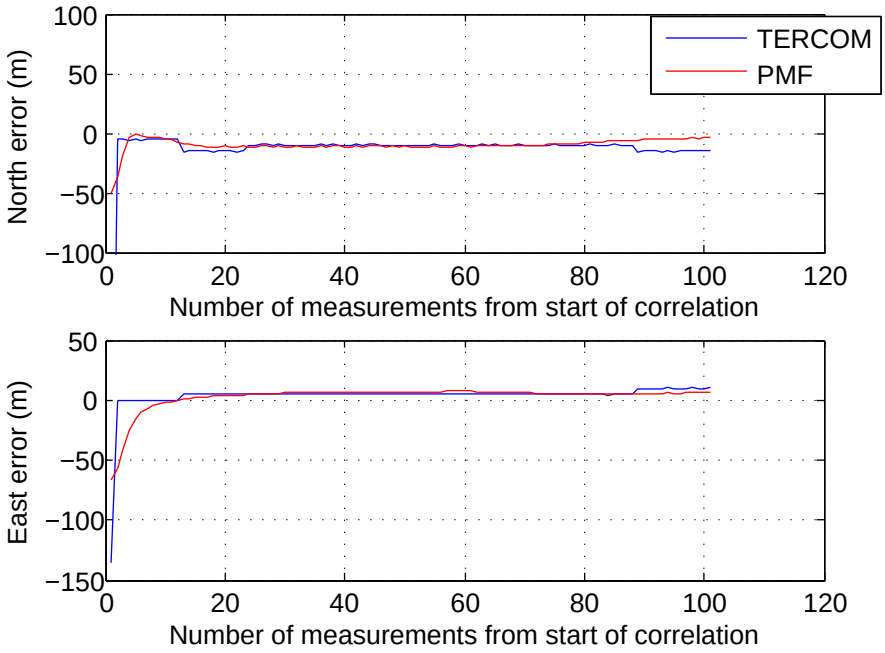


Figure 5.3: *TERCOM and PMF errors, 3797-3997 s from start of run.*

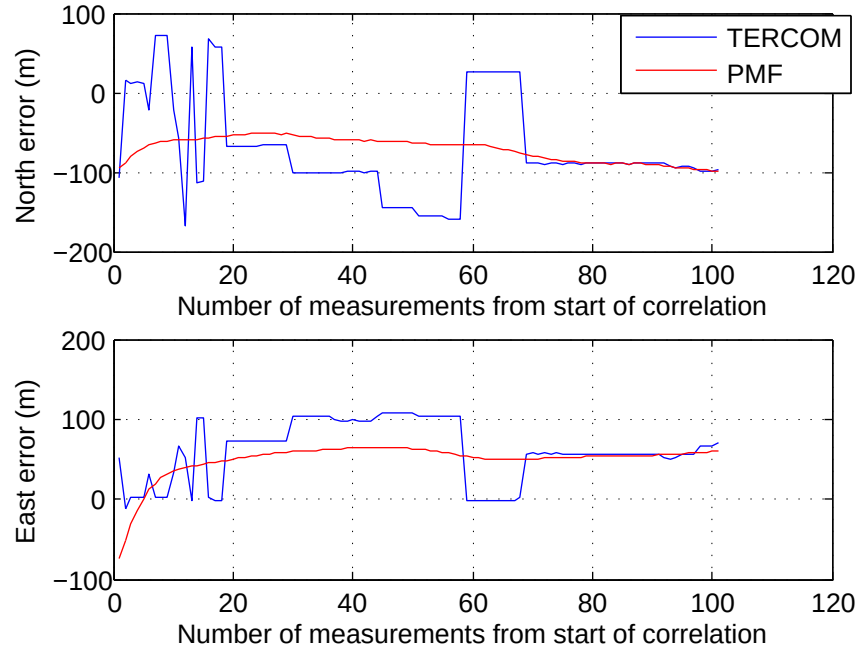


Figure 5.4: *TERCOM and PMF errors, 5397-5597 s from start of run.*

rithms show poor performance in flat area, as expected. The PMF estimate is generally smoother and more robust to errors than the TERCOM estimate. The PMF is also superior to TERCOM in that is able to estimate its own uncertainty to a certain extent. However, it is generally overconfident. Both algorithms have rare occurrences of false fixes, i.e. erroneous estimates with low estimated uncertainty. As such false fixes are damaging if fed back to the navigation system, a real-time system should include a wild point filter before accepting a terrain navigation position fix, as discussed in Section 3.4.

5.2.2 Point Mass and Particle Filters in 2D and 3D

The results in this section are based on the paper Ånonsen and Hallingstad (2006), in which point mass and particle filters in 2D and 3D are compared. The tests are based on the same HUGIN data set as in the previous section, again using the EM 3000 MBE as bathymetric sensor. The tests were conducted in a similar manner as described previously, comparing the results using the different algorithms on sequences containing 1000 seconds of data, which corresponds to a profile length of 2 km, i.e. considerably longer profiles than what was the case in Section 5.2.1. In order to counter for the overconfidence of the PMF, which was observed in earlier tests, a sub-sampling procedure was used, using only MBE beams with a footprint distance greater than 10 m. Since the horizontal resolution of the depth map database was 10 m, beams with footprint distance less than 10 m will frequently hit the sea floor within the same grid cell, leading to strong correlations between the corresponding map look-ups. As these correlations are not modeled in the sensor model, they may lead to overconfident terrain navigation results. Sub-sampling was also used between pings, to prevent the beams from subsequent pings to hit within the same grid cell.

The rest of this section will first consider the classic 2-dimensional version of the PMF and the Bayesian Bootstrap particle particle filter, before moving on to the corresponding 3-dimensional versions described in Section 3.3.1. Relative and absolute profile matching are also compared.

The terrain navigation performance in two different areas will be compared, the parts of the trajectory from 3000–4000 and from 9000–10000 seconds from the start of the run. The terrain in the former area has significant terrain variations, whereas the latter is relatively flat. The locations of the two trajectory parts are shown in Figure 5.5.

2-Dimensional Algorithms

In suited terrain, both the PMF and the PF converged quite fast to a stable estimate, sometimes after as few as 1-2 measurement updates. Since the Bayesian Bootstrap filter is a Monte Carlo method, subsequent runs using the same data yield different results. Because of this, Monte Carlo runs were conducted, and the mean estimates were compared to those of single-run PMF estimates. Generally, both methods worked well in the rough terrain, with typical errors after convergence of around 5-10 meters, i.e. around the horizontal resolution of the map grid. As expected, the results in the flat area were

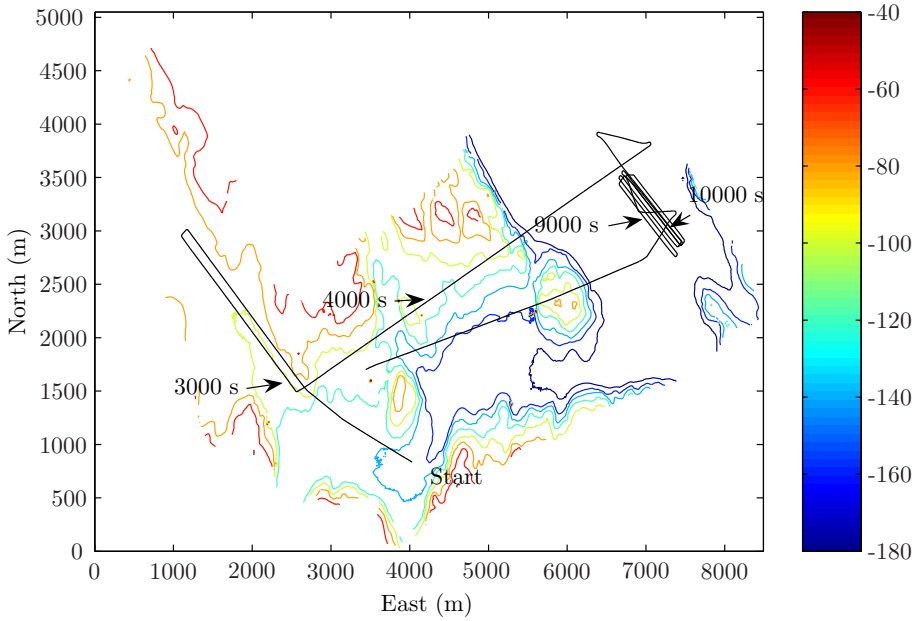


Figure 5.5: *Contour map and AUV trajectory. The parts of the trajectory from 3000–4000 s and 9000–10000 s are used as example areas in this section.*

inaccurate for both methods. In this terrain, the estimated posteriors from both methods typically stabilized rather slowly to distributions with standard deviations of about 50-100 meters in each horizontal direction. The accuracy of the MMSE estimates in this area were typically 20-100 meters, so the estimate covariance matrix adequately measured the uncertainty in the estimates for both methods. The overconfidence observed in the previous tests was still present to some extent, both for the PMF and the PF, but this problem was considerably reduced due to the sub-sampling of the measurements both within and between pings, to minimize the correlations between the different MBE beams.

Figure 5.6 and Figure 5.7 show typical horizontal errors for the two methods compared to ground truth, when they are run using 1000 seconds of data in the rough and flat area, respectively. A search area of ± 300 meters in each direction was used. The results shown for the PF are the mean errors from 50 Monte Carlo (MC) runs, using 1000 particles in each run. Larger particle clouds were also tested, without improving the estimates significantly. The PMF results are from single-run PMF, using an adaptive procedure with a maximum of 5000 non-zero point masses. These results are typical of the overall behavior of both methods. Generally, the estimates of the PMF algorithm are slightly more accurate than those of the PF, and the PMF also converges slightly faster to a stable solution. As can be seen in Figure 5.6, both methods yield estimates with an accuracy of around 10 meters in suited terrain. The PMF estimates are also smoother than the PF MC mean estimates, and this is also the case for the individual MC runs.

As noted in Section 5.2.1, false fixes may occur in terrain navigation. However, in these tests, no false fixes occured in the rough area. In the flat area occasional false fixes occurred in the PF results. Typically, of among 50 PF MC runs in the flat area, 1 or 2 diverged or resulted in false fixes. For the PMF, no false fixes occurred in either area. This indicates that the sub-sampling procedure also has a positive effect on the false fixes; in the tests described in Section 5.2.1 false fixes were observed also for the PMF. Both methods were very robust to errors in the initial position. In the experiments shown in Figure 5.6 and Figure 5.7, the methods were initialized with an error of 50 meters in each horizontal direction. As long as the true position was within the search area, the methods were able to effectively recover the correct position, provided the terrain was suitable for terrain navigation.

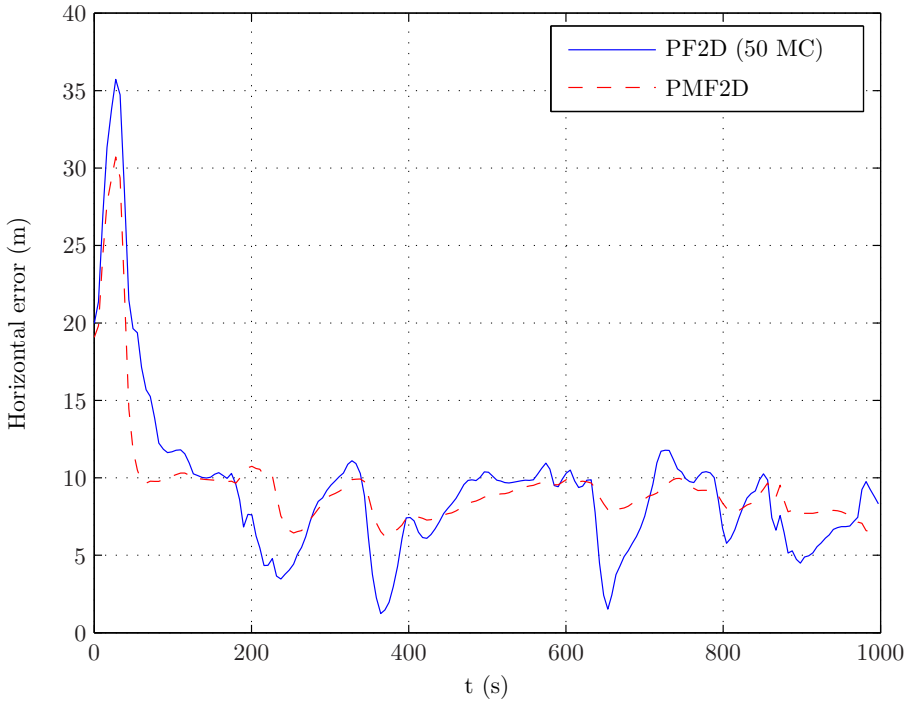


Figure 5.6: *Horizontal error from PMF and PF Monte Carlo mean error, 3000-4000 s from start of run (rough terrain).*

3-Dimensional Algorithms

Similar tests to the above, but with added depth biases, showed that the 2D methods were very sensitive to such depth biases. Even with small depth errors of around 0.5–1.0 meters, the quality of the estimates was significantly degraded, and the errors often grew from 5–10 meters to 40–50 meters. Depth biases of this magnitude are very realistic

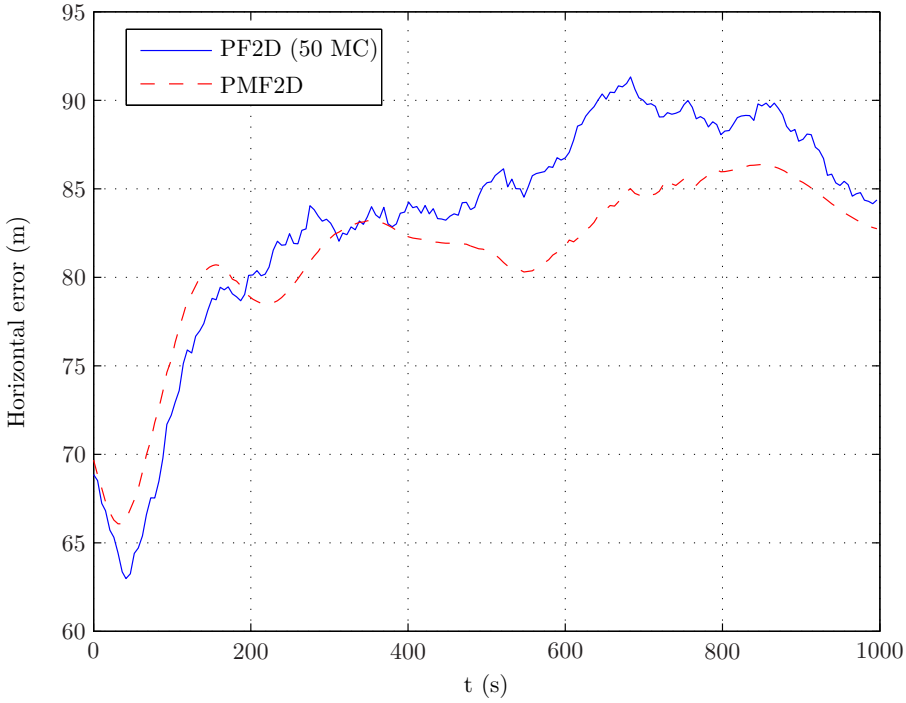


Figure 5.7: *Horizontal error from PMF and PF Monte Carlo mean error, 9000-10000 s from start of run (flat terrain).*

in real scenarios, for example due to wrong tidal estimates. Tests done with the 3D versions of the PMF and PF showed that these versions were able to resolve the depth error problem effectively. Since the depth biases are known to be relatively small, the vertical search area in the 3D methods was typically set to ± 10 meters. Even with no added artificial depth biases, the 3D methods were generally more accurate than the 2D methods. However, the frequency of false fixes appeared to be slightly higher for the 3D methods, and even the PMF gave a false fix on one occasion.

Figure 5.8 and Figure 5.9 show typical errors for both the 2D and 3D methods in the rough area, without and with an added depth bias of 1 meter, respectively. As before, the results from the PMF are single-run results, whereas those from the PF are the mean errors from 50 Monte Carlo runs. The accuracy of the 3D methods are superior in both cases, and their robustness to depth biases is clearly shown. Even with such small depth biases as 1 meter, the performance of the 2D methods is significantly degraded. This strongly advocates the use of 3D methods in real terrain navigation systems. The same conclusions could be drawn from all of the experiments in the rough area; the 3D methods were generally at least as accurate, and in most cases more accurate than the 2D methods. As in the 2D case, the results from PMF3D were also slightly more accurate than those from PF3D.

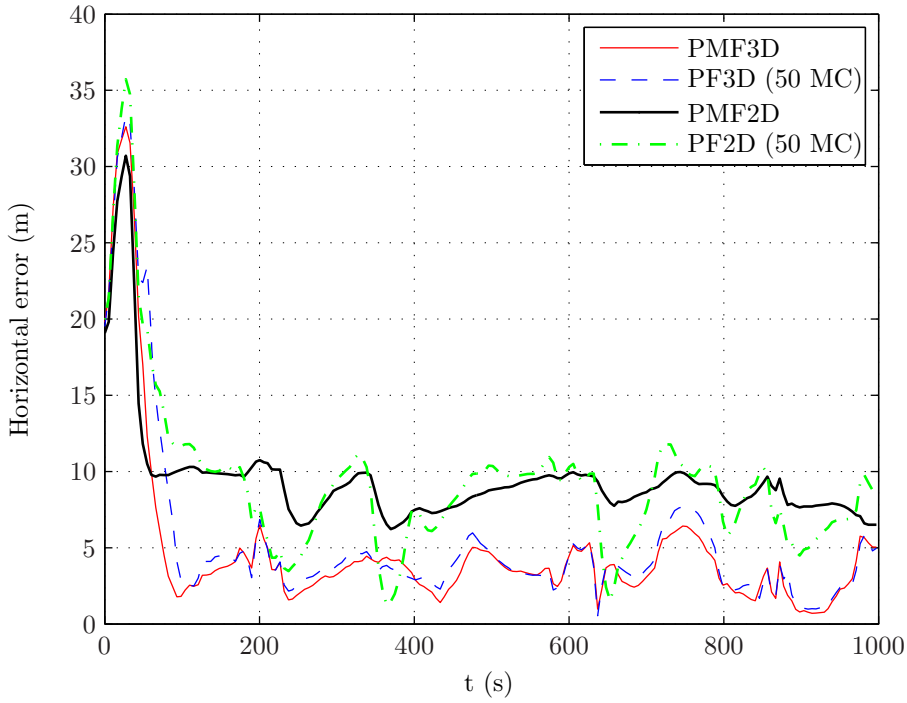


Figure 5.8: *Horizontal errors from 3D and 2D methods 3000–4000 s from start of run. No added depth bias.*

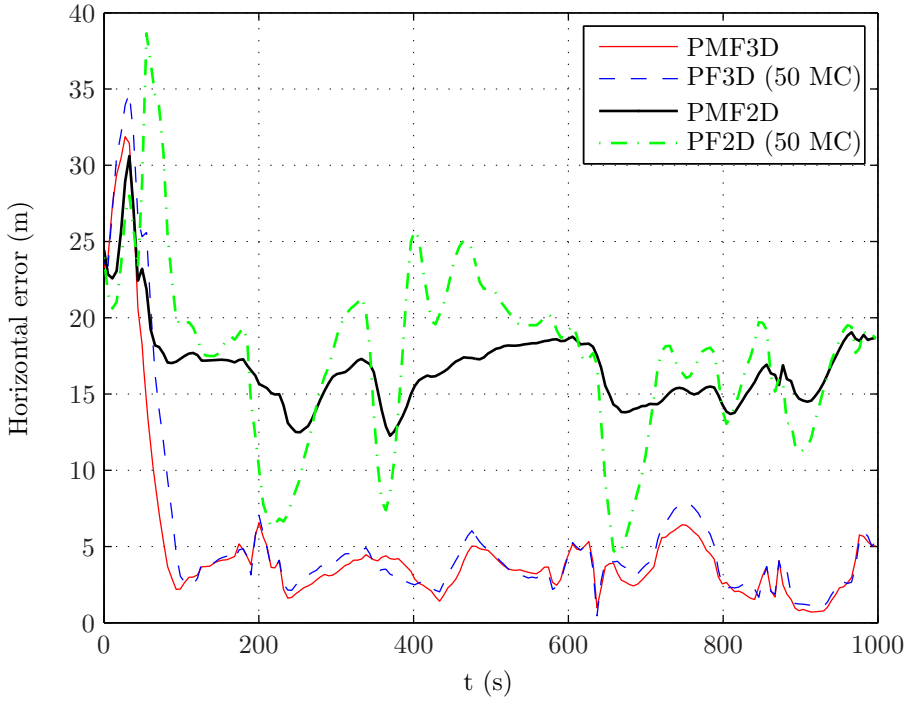


Figure 5.9: *Horizontal errors from 3D and 2D methods 3000–4000 s from start of run. 1 m added depth bias.*

Figure 5.10 and Figure 5.11 show corresponding results from the flat area. Though the errors are quite large for all methods, the 3D methods are again more accurate, both with and without added depth errors. The covariance matrices reflect the high uncertainty in the estimates, so the estimates are still useful to some extent. The 3D methods were in most cases more accurate also in the flat area, but both PMF3D and PF3D had a slightly higher rate of false fixes or divergence than the corresponding 2D methods.

The main disadvantage with the 3D methods is that they are extremely computationally expensive. This is particularly true for PMF3D. In a straight-forward MATLAB implementation of PMF3D, processing 1000 seconds of real data took around 6 hours on a 3.6 gigahertz computer with 2.0 gigabytes of RAM. This yields a real-time factor of around 20, when the real time factor is defined as

$$c_{\mathrm{RT}} = t_p/t_i, \tag{5.1}$$

where t_p is processing time and t_i is input time. In comparison, for the adaptive grid version of the PMF2D and the PF2D (with 1000 particles) the real-time factors were typically around 0.05. For PF3D, the computational demands did not increase significantly when extending them from 2D to 3D; the real-time factor was still around 0.05. Theoretically, one should need more particles to obtain the same level of accuracy in a 3D problem as in a 2D problem. However, 1000 particles turned out to be sufficient also in the 3D case. This is probably due to the relatively small vertical search area and due to the fact that the particles converge quickly to the correct vertical offset. Again, these results strongly advocate the use of 3D particle filters, as opposed to 2D. At virtually no extra computational cost, the accuracy is improved by extending the PF from 2D to 3D. It should be noted that not much work has been done in order to reduce the computational cost of PMF3D. An adaptive grid implementation of PMF3D is expected to significantly reduce the computational demand. However, computation times comparable to those of PF3D can not be expected for PMF3D. An adaptive grid implementation is also more difficult in 3D than in 2D, due to the 3D convolution that is involved.

Another way to resolve the depth bias problem without using a full 3D model is, as explained in Section 3.3.1, to use relative profiles, by subtracting the profile means and still work in a 2D setting. Similar tests to those above were conducted using this approach, and the results were generally very good. The quality of these estimates were very similar to those of the full 3D estimates. Figure 5.12 and Figure 5.13 show results from PMF3D and from PMF2D with and without subtracted mean. Both in the flat and the rough area, the performance of PMF2D with subtracted mean is comparable to that of PMF3D. This shows that using relative profiles is a good alternative to 3D methods, and this is also computationally less burdensome. However, one should bear in mind that in certain terrain types, the use of relative profiles may be dangerous. An example of this is so-called self-similar terrain, in which the terrain has similar local variations at different depths. This results in a high risk of convergence to erroneous position estimates in such terrain, and relative methods should therefore be used with care. Another disadvantage with relative methods is that they do not estimate the depth bias, which in itself can be

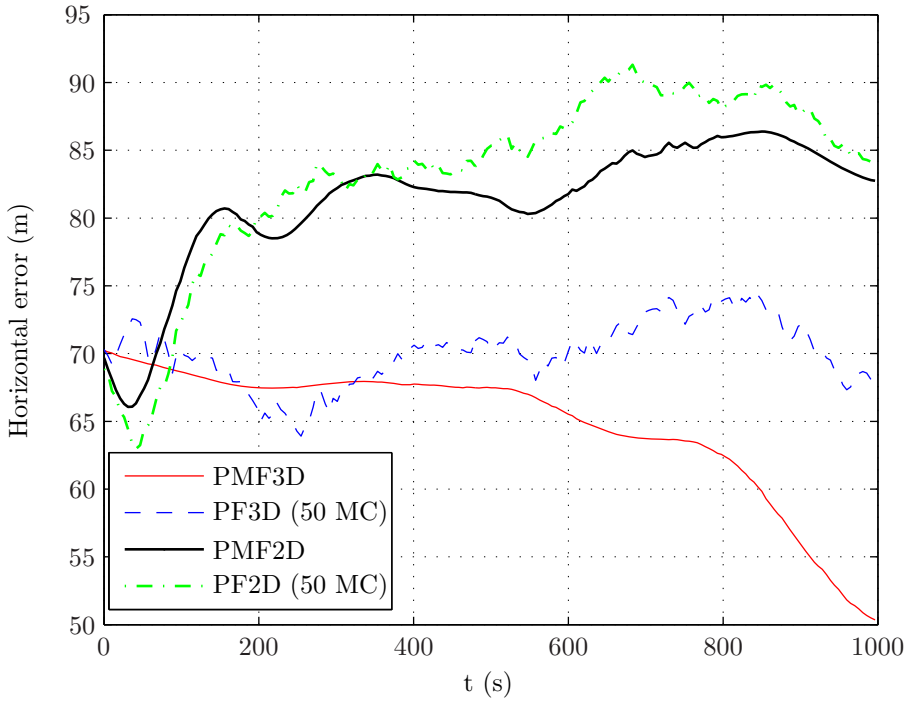


Figure 5.10: *Horizontal errors from 3D and 2D methods 9000–10000 s from start of run. No added depth bias.*

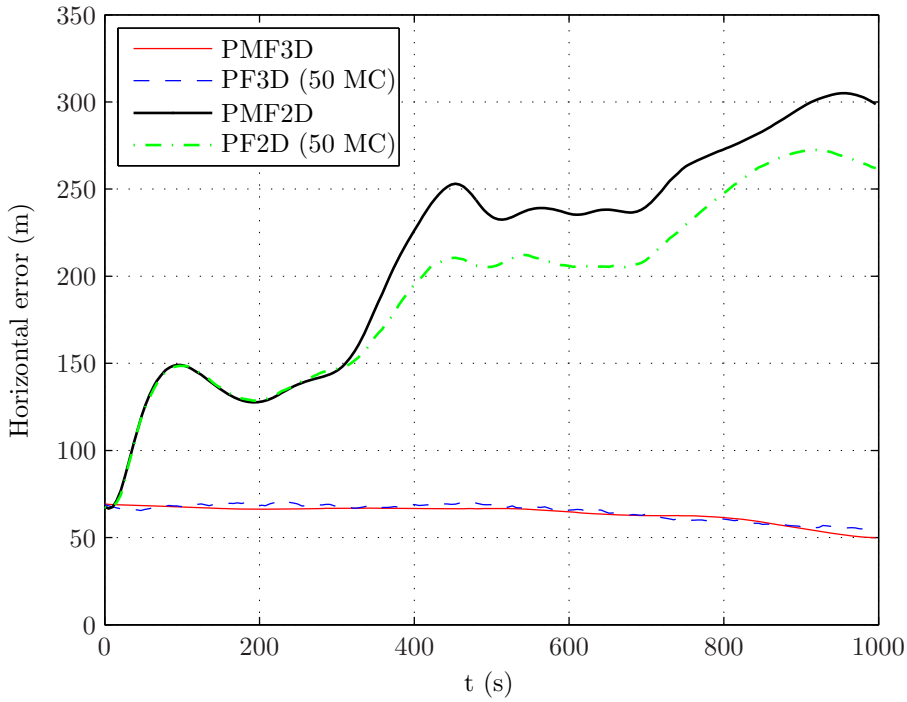


Figure 5.11: *Horizontal errors from 3D and 2D methods 9000–10000 s from start of run. 1 m depth bias.*

useful.

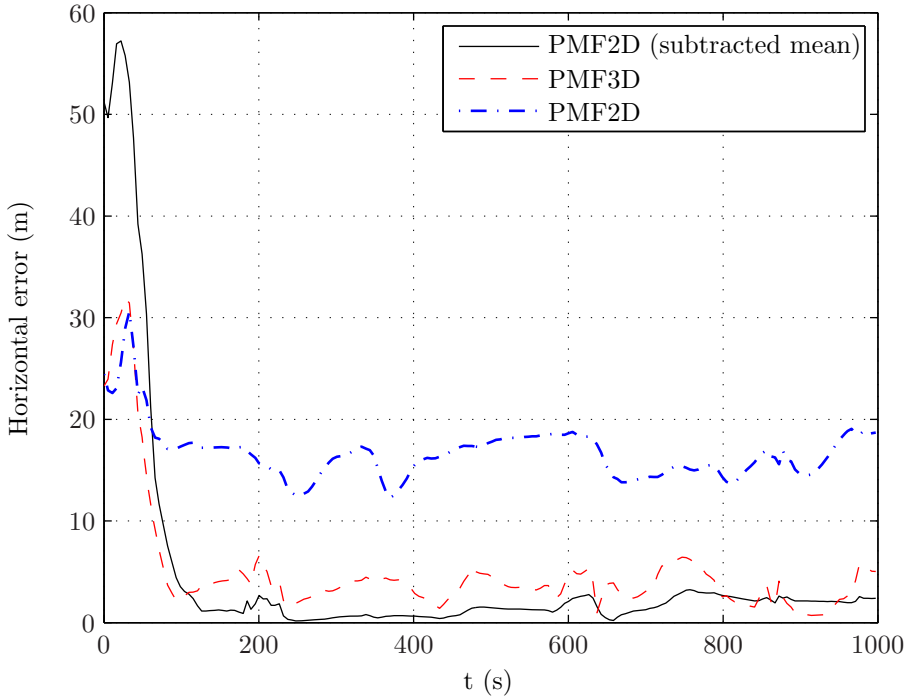


Figure 5.12: *Horizontal errors from PMF3D and PMF2D with and without subtracted mean, 3000–4000 s from start of run. 1 m added depth error.*

5.2.3 Extension of the Process Model

The results in Section 5.2.1 and 5.2.2 showed the behavior of the TERCOM, PMF and PF algorithms, and how the Bayesian methods were superior to the TERCOM algorithm. However, a problem with the Bayesian methods was that the reported variance of the estimates were too low. This problem was partially solved in Section 5.2.2 by using subsampling of the bathymetric data, to prevent unmodeled effects stemming from several measurement beam footprints within the same map grill cell. In order to try to reduce this effect further, particle filter implementations of the approach presented in 3.2.1 were developed, i.e. the process model was extended to use a first order Markov model for the drift, instead of the traditional plain white noise approach. The results presented in this section are based on the paper Ånonsen et al. (2007). As opposed to Section 5.2.2, in this section it is assumed that the depth bias has been properly compensated for, in order to limit the computational burden of the algorithms. The resulting state vector is therefore 4-dimensional. If needed, it is straightforward to implement 5-dimensional versions of the algorithms, estimating the depth bias also.

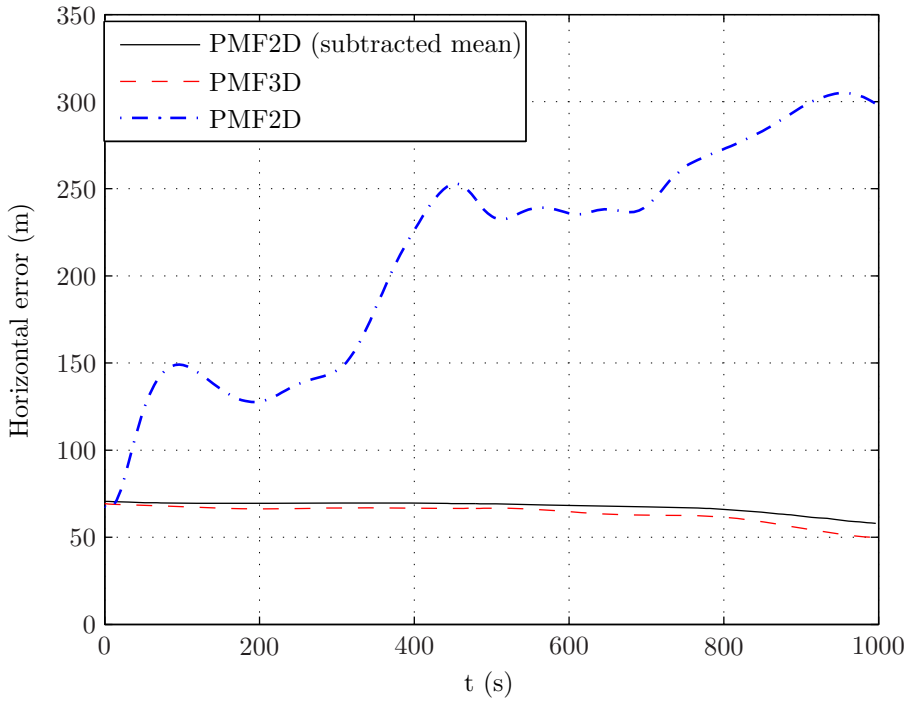


Figure 5.13: *Horizontal errors from PMF3D and PMF2D with and without subtracted mean, 9000–10000 s from start of run. 1 m added depth error.*

The algorithms were tested in the same manner as before, using profiles of approximately 2 km, each with 1000 seconds of AUV data. Again the EM 3000 was used as bathymetric sensor, and sub-sampling both within and between pings was used. Monte Carlo runs were conducted, using both the traditional 2D approach and the new 4D approach, and the mean results from the Monte Carlo runs were compared. Table 5.2 shows the drift noise parameters used in the filters in the various runs. Notice the low white noise parameters used in the extended model.

Table 5.2: *Filter drift noise parameters used in Monte Carlo runs.*

Method ID	τ_1[s]	τ_2[s]	White noise parameters $[\sqrt{q_{11}}\ \sqrt{q_{22}}]$ [m/s]
PF2D	–	–	[0.5 0.5] (north, east)
PF4	10	10	[0.001 0.0005] (surge, sway)
PF4	100	100	[0.001 0.0005] (surge, sway)

Figure 5.14 shows the average horizontal errors from 25 Monte Carlo runs 1000–2000 seconds after the start of the run, when the vehicle is in an area well suited for terrain navigation. The red line, labeled 'PF2D', shows the results using the 2-dimensional model, whereas the blue and black lines, labeled 'PF4', are the results obtained from the improved state-space model, with drift parameters $\tau_1 = \tau_2 = 10$ s and $\tau_1 = \tau_2 = 100$ s, respectively. As can be seen in Figure 5.14, the improved filter models seem to yield a more stable solution than the conventional one, which has a tendency of fluctuations. This is particularly evident in the interval between 200 and 400 seconds, where the estimate error from the conventional model suddenly increases for a while, before returning to approximately 5 meters after around 500 seconds. This behavior is also seen to some extent in the results from the extended model, but especially in the $\tau = 10$ s case the fluctuations are of much smaller magnitude, yielding smoother estimates. The reason for this fluctuating behavior is mainly that one has to use a much higher white noise parameter for the drift in the particle filter for the conventional model than what is the case in the extended model, which has some time correlation in the process noise. This effect was evident throughout all the tests in the suited areas; the improved filter model estimates are smoother and have a lower tendency of fluctuations and hence a lower probability of filter divergence.

Similar results can be seen in Figure 5.15, where corresponding results 3000–4000 seconds from the start of the run are shown. Here the tendency of fluctuations in the conventional results are even more evident. Though all of the variants yield estimates of high accuracy, within 10 meters, which is the horizontal map resolution, the results from the extended models are superior when it comes to smoothness of the estimates. The estimates from the extended model with drift parameter $\tau = 10$ s are the most accurate and stable.

One of the rationales behind the derivation of the improved state-space mode was to further decrease the overconfidence problems seen in the Bayesian terrain navigation

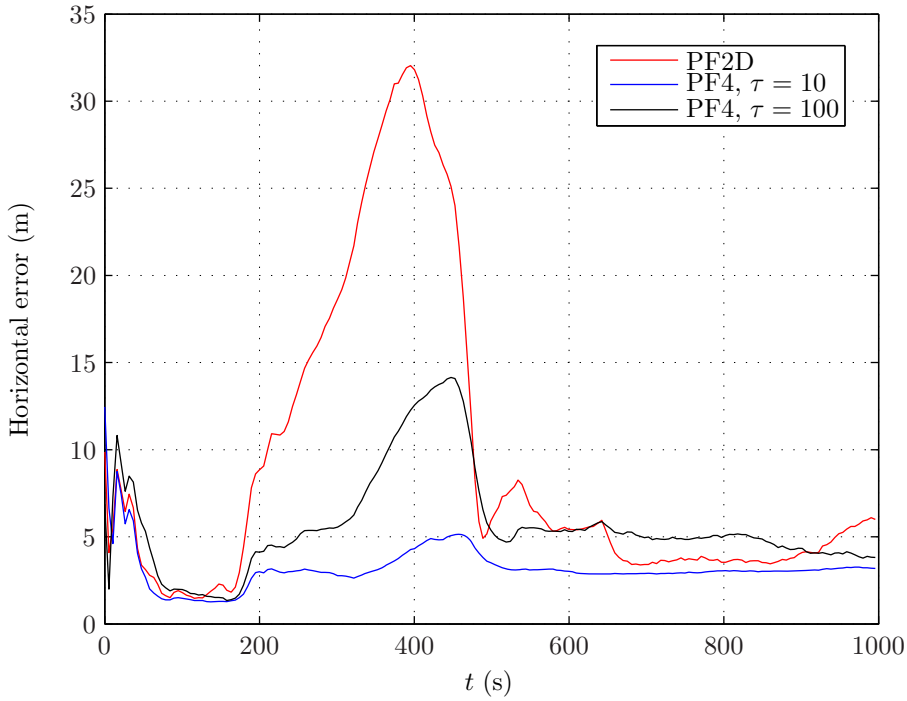


Figure 5.14: *Computational results from 25 MC runs, 1000–2000 seconds from start of run.*

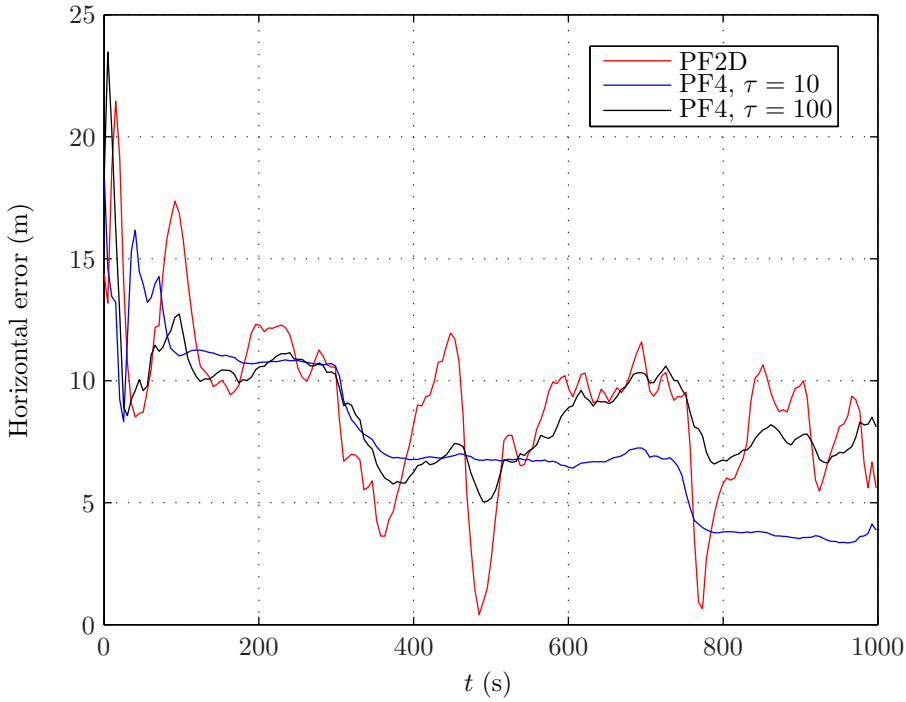


Figure 5.15: *Computational results from 25 MC runs, 3000-4000 seconds from start of run.*

algorithms. It turns out, however, that the covariance problem is not solved by the extended model. In fact, the estimated variances and covariances are much lower when the improved filter model is used. Figure 5.16 and 5.17 show the estimated north and east offsets and their standard deviations for the conventional and extended models, respectively. These results are from one single run with the particle filters. For a consistent estimate, the estimate should lie within the 1σ bounds around 68% of the time, if a Gaussian distribution is assumed. This is achieved for the conventional model, but for the extended model it is far from true. This means that there is still a discrepancy between the true system and the model. The improved smoothness of the estimates when using the extended model therefore comes at the cost of poorer covariance estimates. A higher covariance in the extended model can be attained by using a higher white noise parameter, but this severely degrades the stability and smoothness of the estimates.

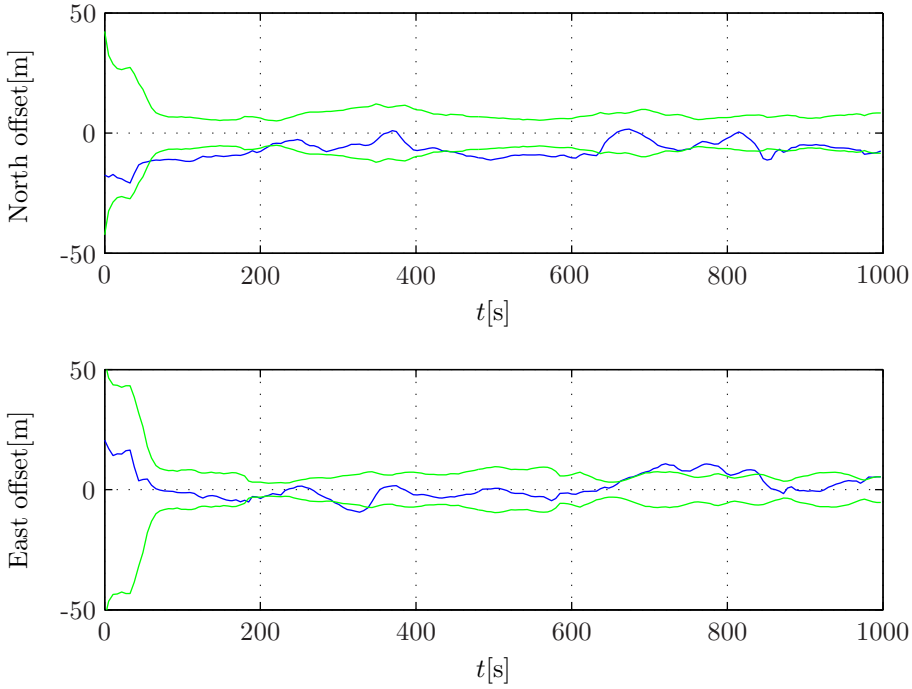


Figure 5.16: *Estimated north and east offsets (blue lines) and standard deviations (green lines), conventional filter model (single run).*

The behavior of the two models in flat terrain is shown in Figure 5.18. Here the vehicle is traversing the flat terrain in a lawn-mower pattern, as can be seen in Figure 5.1. Neither the conventional nor the extended model does well here, as the terrain does not have enough variation for the terrain navigation algorithms to obtain distinct fits. None of the results are accurate in this area, with typical horizontal uncertainties of 50–100 meters. It should be noted, however, that to some extent the covariance matrices esti-

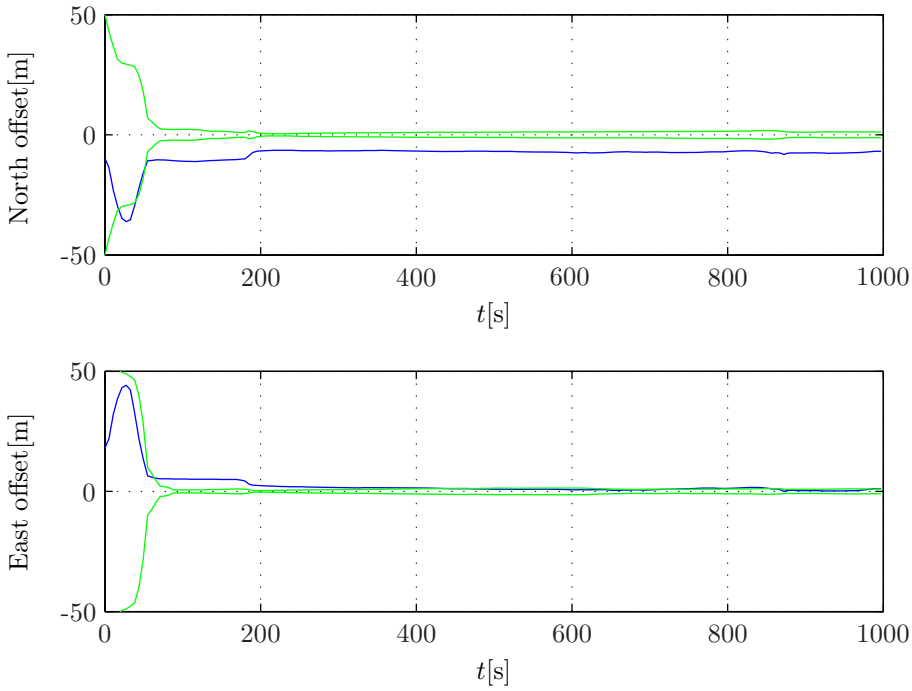


Figure 5.17: *Estimated north and east offsets (blue lines) and standard deviations (green lines), improved filter model with* $\tau_1 = \tau_2 = 10$ *s (single run).*

mated by the particle filter are able to reflect the uncertainties in the estimates both for the conventional and the extended model. However, there is a tendency of overconfidence in both cases. From these results it can be concluded that nothing is gained in the flat area by using the extended model. On the other hand, nothing is lost either. The terrain simply does not contain enough information within the current sensor and map accuracy for terrain navigation to work satisfactorily.

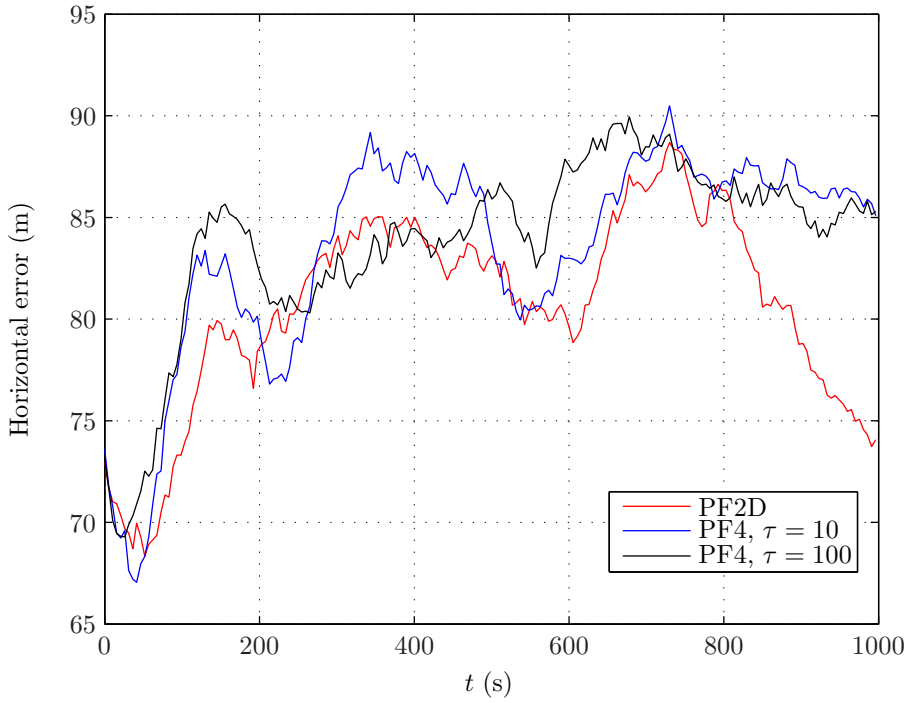


Figure 5.18: *Computational results from 25 MC runs, 9000-10000 seconds from start of run.*

5.2.4 Sigma-Point Kalman Filter Results

This section is based on Ånonsen and Hallingstad (2007), where offline terrain navigation results from a 3-dimensional Sigma-Point Kalman Filter (SPKF) were presented, using the same data set as in the previous sections. In Lang (2006), the SPFK was used for terrain navigation on simulated data, but to the best knowledge of the author, Ånonsen and Hallingstad (2007) was the first publication presenting results using the SPFK on real underwater vehicle data.

In Section 2.4.7 it was shown how the general SPFK framework is simplified using the usual linear process model for terrain navigation. The unscented transform only needs to be used in the measurement update. In the time update, the conven-

tional Kalman filter update equations are used. The prediction of the state vector and covariance matrix simply become

$$\bar{\boldsymbol{\xi}}_k = \hat{\boldsymbol{\xi}}_{k-1}, \tag{5.2}$$

$$\bar{\boldsymbol{P}}_k = \hat{\boldsymbol{P}}_{k-1} + \boldsymbol{Q}^*_{k-1}, \tag{5.3}$$

where the overbar denotes the one step ahead predicted state and covariance, and the hat denotes the filtered state and covariance. The matrix $\boldsymbol{Q}^*_{k-1}$ is the filter model covariance matrix of the process noise. In the implementation of the SPKF, the depth bias was included in the filter, resulting in a 3-dimensional state-vector. The results of the SPFK were compared to those of a 3-dimensional Bayesian Bootstrap particle filter, i.e. what was called PF3D in Section 5.2.2. The methods were tested in the same manner as in the previous sections, using 1000 seconds of AUV data in each profile.

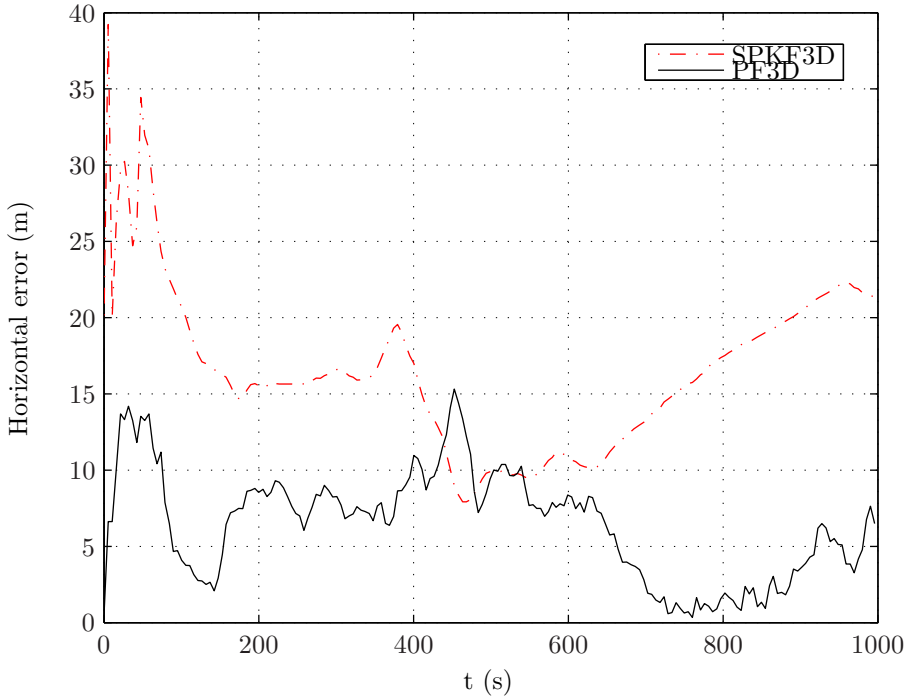


Figure 5.19: *Comparison of SPKF and PF 1000–2000 s from start. The time axis shows the time after restart of the algorithm*

.

Figure 5.19, 5.20 and 5.21 show horizontal errors from the SPKF and the PF (using 1000 particles) in three different areas of the test run. The results in Figure 5.19 and 5.20 are from rather rough areas, well suited for terrain navigation, whereas the results in Figure 5.21 are from a flat area. As can be seen in Figure 5.19 and 5.20, the results from

the SPKF are relatively accurate in the well suited terrain, even though in Figure 5.19 the SPKF is less accurate than the PF, with a final horizontal error of around 22 m, compared to around 6 m for the PF. However, the typical behavior is more like that in Figure 5.20, where the accuracy of the SPKF is comparable to that of the PF. In Figure 5.22 the corresponding horizontal trajectory can be seen. In the flat area, neither method works satisfactorily, as expected. The performance in a flat area is shown in Figure 5.21, where large errors are seen for both methods. The covariances obtained from the SPKF are very similar to those from the PF, both in the rough and flat terrain.

Both algorithms can easily be implemented in real time, the SPKF being slightly faster than the PF for this 3D model. The CPU times of the MATLAB implementations of the algorithms when running 1000 s of real-time data on a 2.66 GHz computer with 4.00 GB of RAM were typically around 30 s for the SPKF and 45 s for the PF. However, not much effort has been put into optimization of the algorithms, and they can probably easily be made more effective.

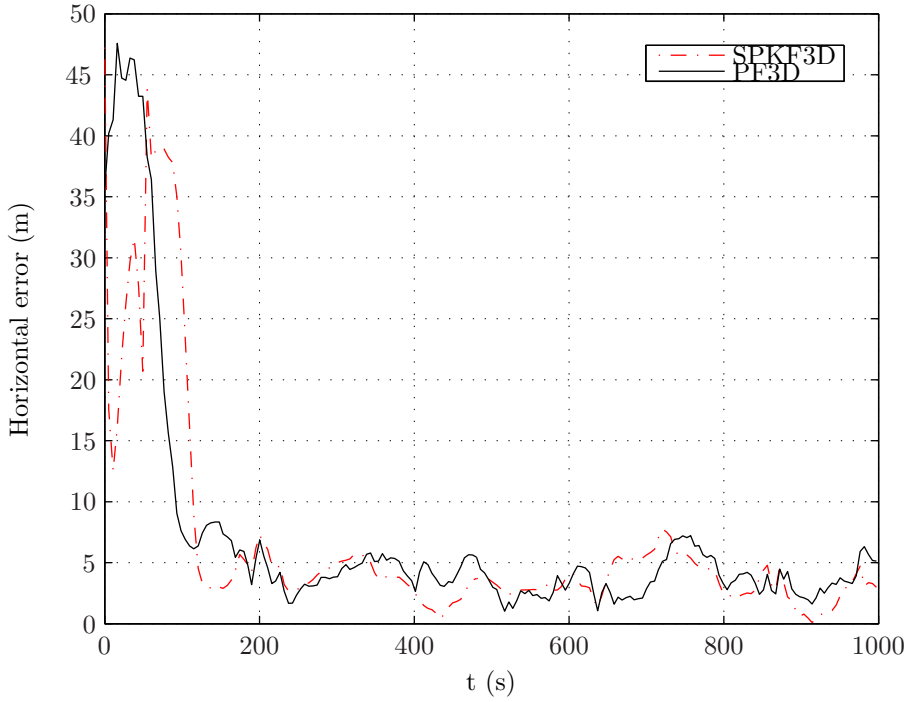


Figure 5.20: *Comparison of SPKF and PF 3000–4000 s from start.*

5.2.5 Pockmark Area Results

This section is based on the paper Ånonsen and Hagen (2009), in which terrain navigation performance in an area with pockmarks was investigated. Pockmarks are craters

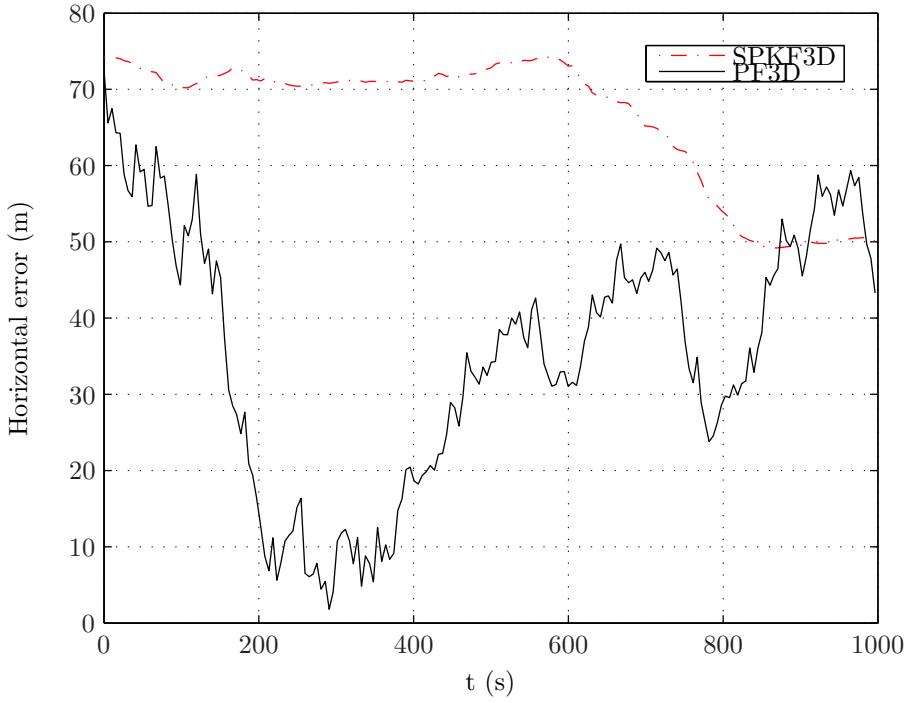


Figure 5.21: *Comparison of SPKF and PF 9000–10000 s from start.*

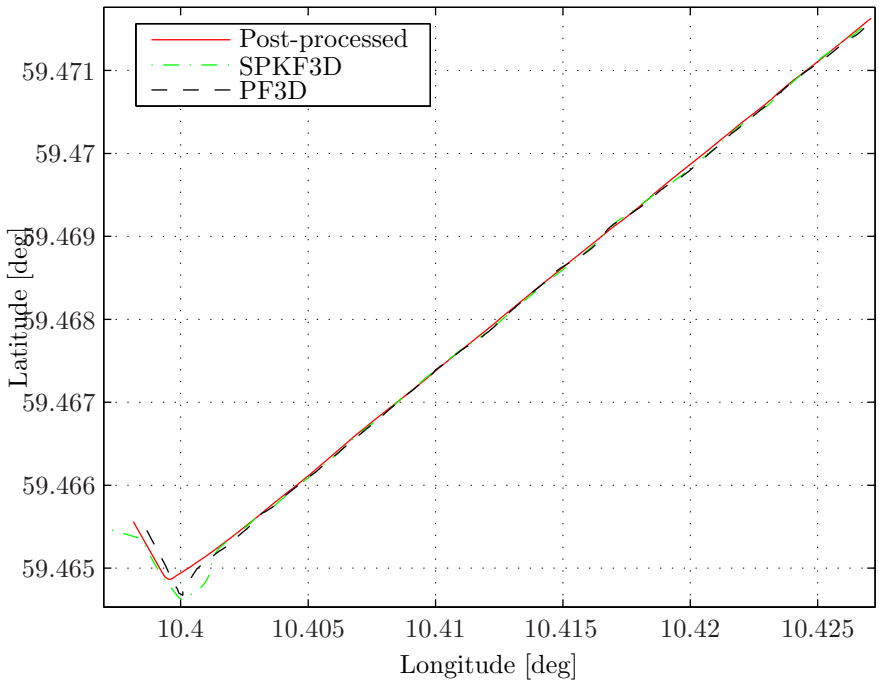


Figure 5.22: *SPKF, PF and post-processed trajectories (Latitude-Longitude) 3000–4000 s from start.*

on the sea floor resulting from the release of gas or liquid (Hovland and Dudd, 1988). Pockmarks of various sizes, ranging from a few meters to several hundred meters in diameter, occur worldwide. Since pockmarks often occur in areas where the sea floor is otherwise relatively flat, utilization of pockmarks for terrain-navigation purposes would facilitate terrain navigation in areas where it would otherwise not be suited.

To investigate the behavior of terrain navigation in a pockmark area bathymetric data from an area outside the Norwegian west coast were used. The data were collected by a surface vessel using a Kongsberg Maritime EM710 multibeam echo sounder, post-processed and converted to a grid format suitable for terrain navigation. A 3D plot of the terrain in the area is shown in Figure 5.23. The pockmarks in this area vary in size, with diameters from about 50 to 200 meters. The depth of the craters are typically between 3 and 10 meters. The total sea depth in the area is about 300 meters.

AUV navigation data and vehicle multibeam echo sounder data were simulated in a number of different scenarios, to investigate the performance of terrain navigation in the area. The post-processed EM710 data were chosen as the "true" terrain, and a new map, to be used as the terrain navigation map database, was created, adding white noise to the depth nodes in the true terrain map. This was done in order to emulate a true situation, in which the surveyed terrain map used by the terrain navigation methods is imperfect and includes noise from the survey sensor. The horizontal grid resolution of the depth map was 10 meters. AUVs are typically equipped with MBEs of different types than survey vessels. For example, the HUGIN AUV family (Kongsberg Maritime AS, 2009) is typically equipped with MBEs like the EM3000, EM3002 or EM2000. The simulated vehicle MBE data in this section use typical noise parameters from the EM3000 300 kHz MBE. The measurements consist of the vertical distance from the AUV to the MBE footprint, taken from the "true" terrain data, plus measurement noise. Navigation parameters used in the simulations are based on typical HUGIN 1000 parameters.

In the following the performance of terrain navigation in two different scenarios is presented. Both the PMF and several variants of the PF were used as terrain navigation algorithms. The terrain navigation solutions from both methods were very similar and nearly identical in most cases. The PMF showed a slightly more robust behavior, and the following subsections focus on the results from this algorithm. Unless otherwise stated, the results presented are based on the posterior mean, i.e. the MMSE estimate.

Scenario 1

In this scenario, the vehicle travels from a flat area and directly over a pockmark, with a speed of 2 m/s ($\approx$ 4 knots). The depth variation of the pockmark directly underneath the vehicle is about 3 meters. The total duration of the trajectory is approximately 210 seconds, corresponding to a traveled distance of about 400 m, and the AUV altitude is 30 meters. The trajectory in the map is shown in Figure 5.24.

The PMF was initialized with an initial error of 100 meters in the north and east direction, respectively. This was done in order to test the ability of the terrain navigation system to identify a large error in the navigation system. A standard deviation of 500

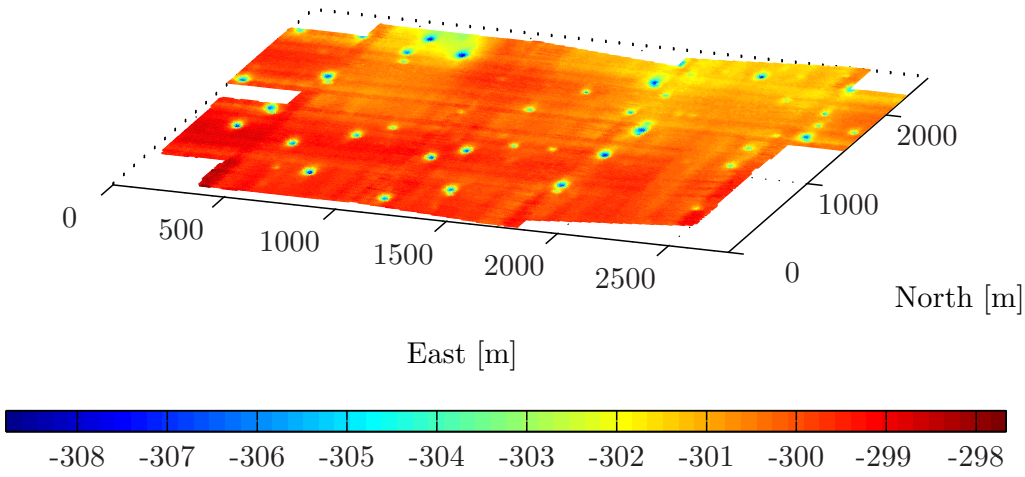


Figure 5.23: *Overview of the area with pockmarks.*

meters in each direction was used in the initial probability density function, together with a search area of $\pm 3\sigma$ (± 1500 m) around the initial position and a fixed grid resolution of 5 meters.

The terrain navigation results from the PMF in this scenario are shown in Figures 5.25 and 5.26. Figure 5.25 shows the estimated north and east position offset from the inertial navigation system, together with their estimated standard deviations. The standard deviations have been plotted around the true offset at $[-100 \text{ m}, -100 \text{ m}]$. Figure 5.26 shows the total horizontal error of the terrain navigation position estimate from the PMF. The total sea depth directly beneath the vehicle is shown in Figure 5.27. After 160 seconds, the pockmark is in the footprint of the vehicle MBE. The results clearly shows how the terrain variations around the pockmark can be utilized by the terrain navigation system. In the beginning of the scenario, where the terrain is flat, the terrain navigation estimate has a high uncertainty, with errors from 50 to 350 meters. This uncertainty is reflected in the estimated standard deviation. As can be seen in Figure 5.25, the estimated offsets stay within the 1σ band. In the flat area, the estimated probability density function is actually multimodal. As soon as the vehicle travels above the pockmark, this changes dramatically, and the terrain navigation system quickly finds the correct position. This is also reflected in the estimated uncertainty. It should be noted that even though there are several similar pockmarks within the search area, the algorithm finds the correct one.

Scenario 2

In this scenario, the vehicle passes two pockmarks. The trajectory in the map is shown in Figure 5.28. The first pockmark is rather deep, with a depth variation of 8–9 meters. The second pockmark is smaller; the depth variation directly underneath the vehicle is 1–2 meters. The vehicle does not pass directly over the deepest part of this pockmark,

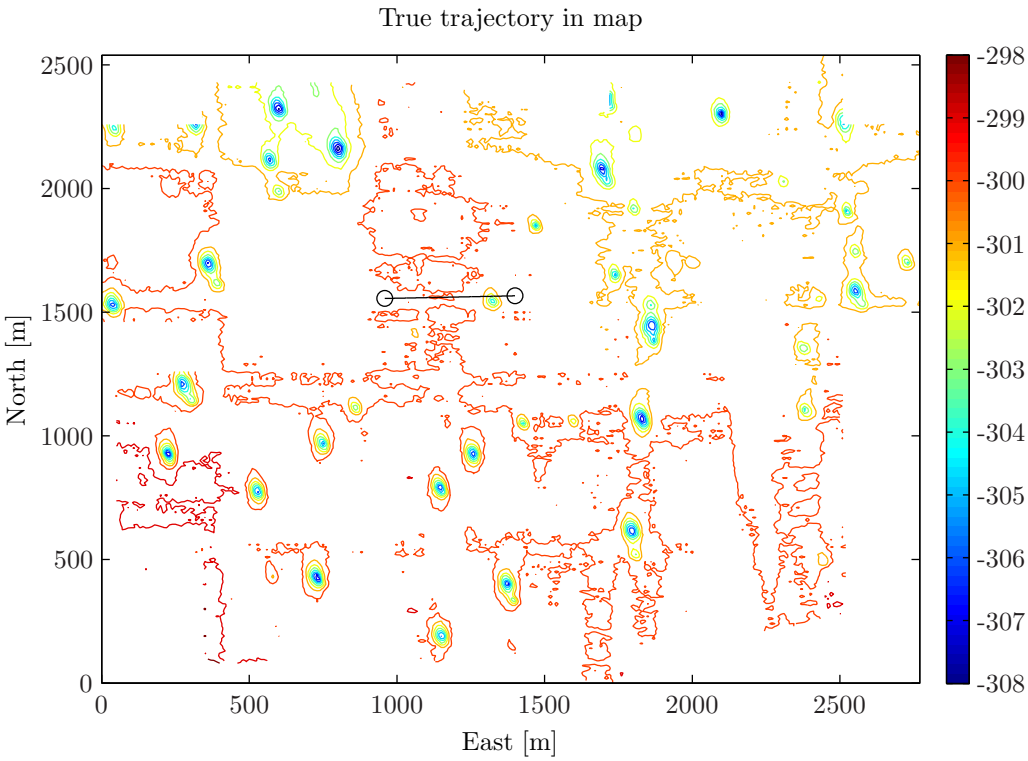


Figure 5.24: *Trajectory of Scenario 1. The direction of travel is from left to right in the figure.*

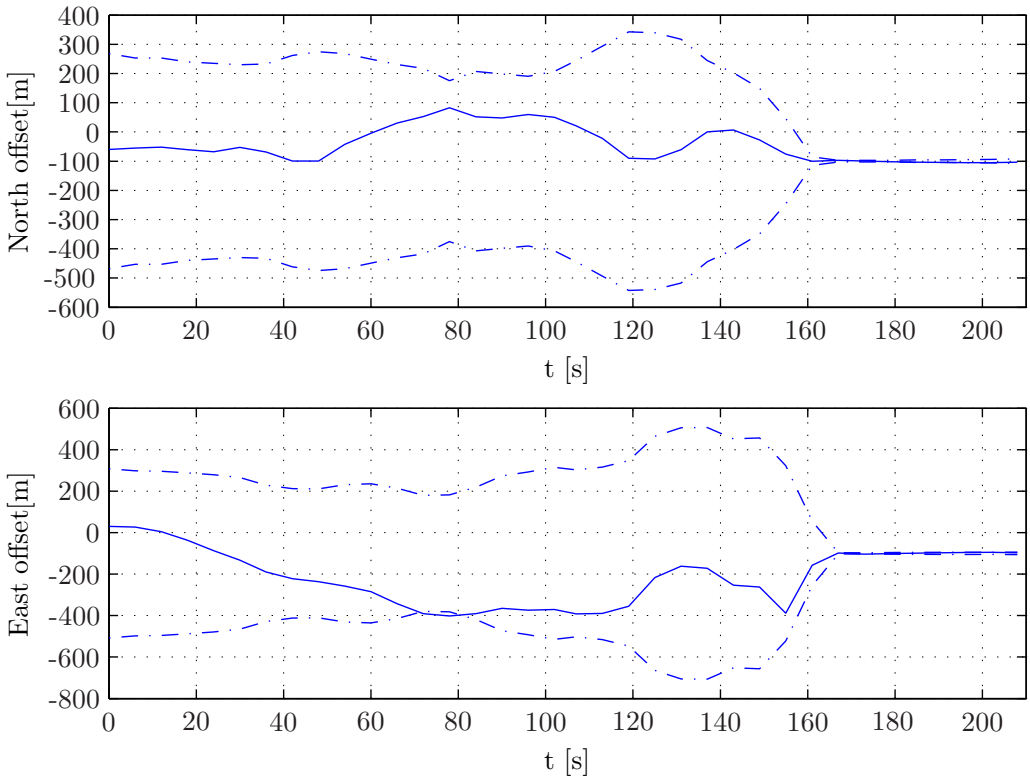


Figure 5.25: *Estimated north and east offset from PDF (solid line) and estimated uncertainty plotted around true offset (dash-dot), Scenario 1. The true offset is* $[-100\ m, -100\ m]$*. The PMF was initialized with an initial standard deviation of 500 meters in each direction.*

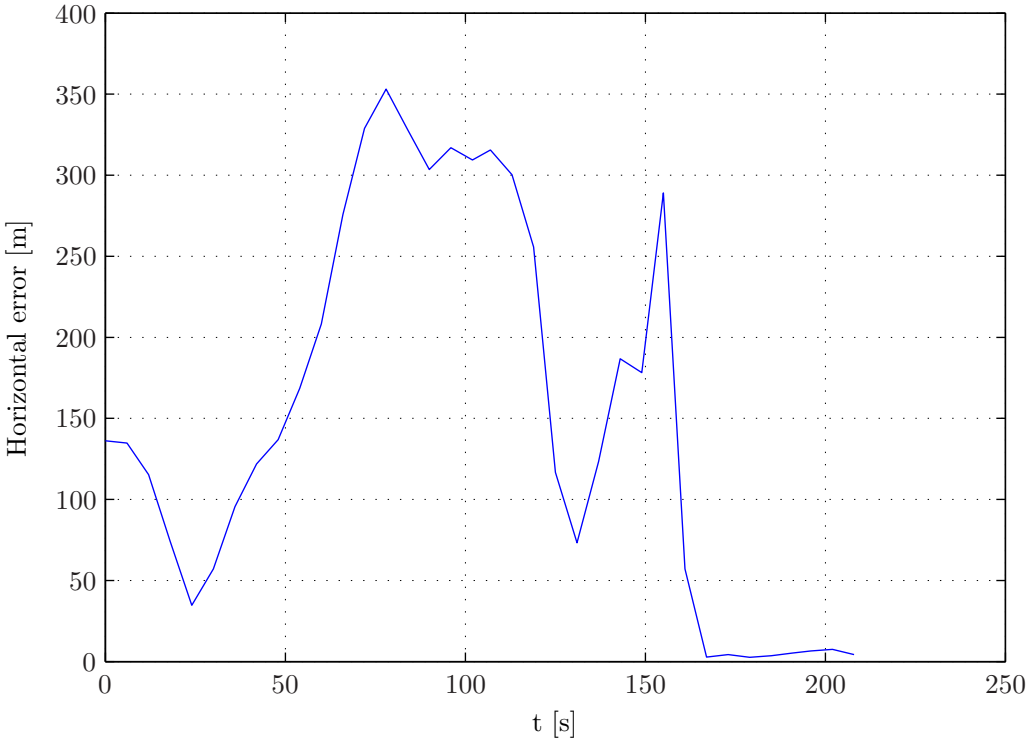


Figure 5.26: *Horizontal error in PMF estimate, Scenario 1.*

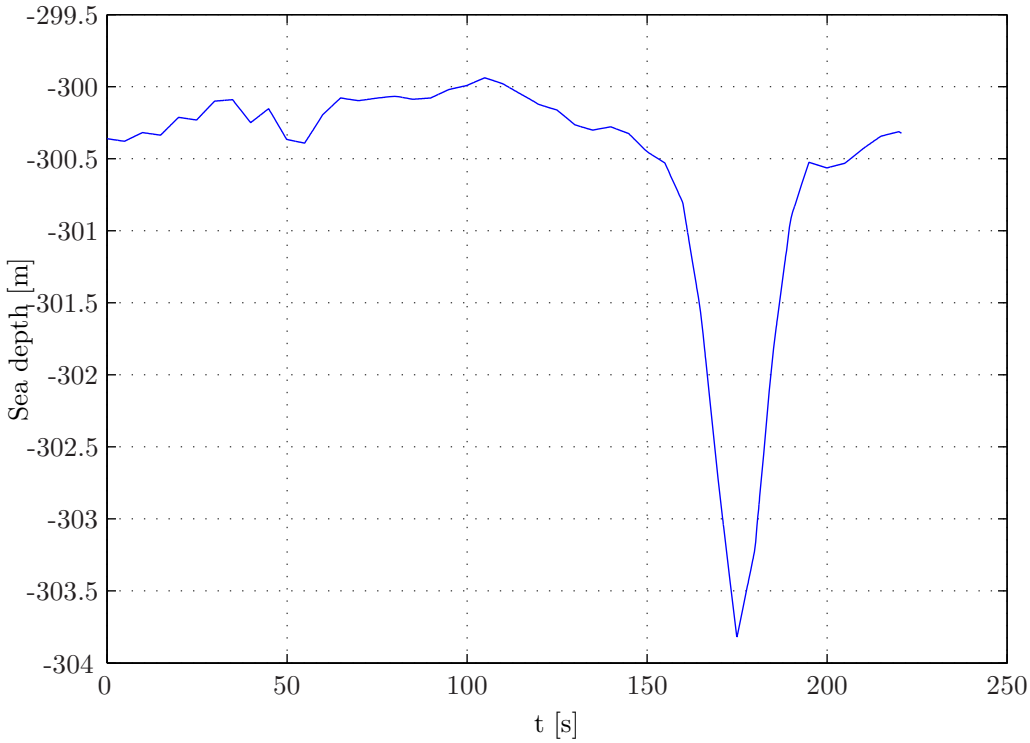


Figure 5.27: *Sea depth along vehicle trajectory, Scenario 1.*

but the MBE swath covers the entire pockmark. Between the two pockmarks, the sea floor is relatively flat, as can be seen in Figure 5.31. Like in the first scenario, an error of 100 meters in each direction was added to the position of the inertial navigation system. The same parameters as before were used for the initial probability density, search grid and PMF noise parameters.

The terrain navigation results from the PMF in this scenario are shown in Figures 5.29 and 5.30. The terrain navigation position estimate has a large error in the start of the scenario, but as soon as the first pockmark is reached, the correct position is found. Between the two pockmarks, the terrain is less informative, and in this area the estimated offset does not change much. There is a slight increase in the estimated uncertainty of the terrain navigation solution in this area, due to the drift modeled in the process model of the PMF. When the vehicle reaches the second pockmark, a small decrease can be observed, both in the actual error and in the estimated uncertainty.

Again, the terrain navigation algorithm was able to locate the correct pockmark, even though several pockmarks of approximately the same size were present in the search area. This appeared to be typical for all of the simulations that were done in this particular terrain. Even though the shapes of the pockmarks are similar, they seem to contain enough information to provide a unique match in the terrain navigation methods.

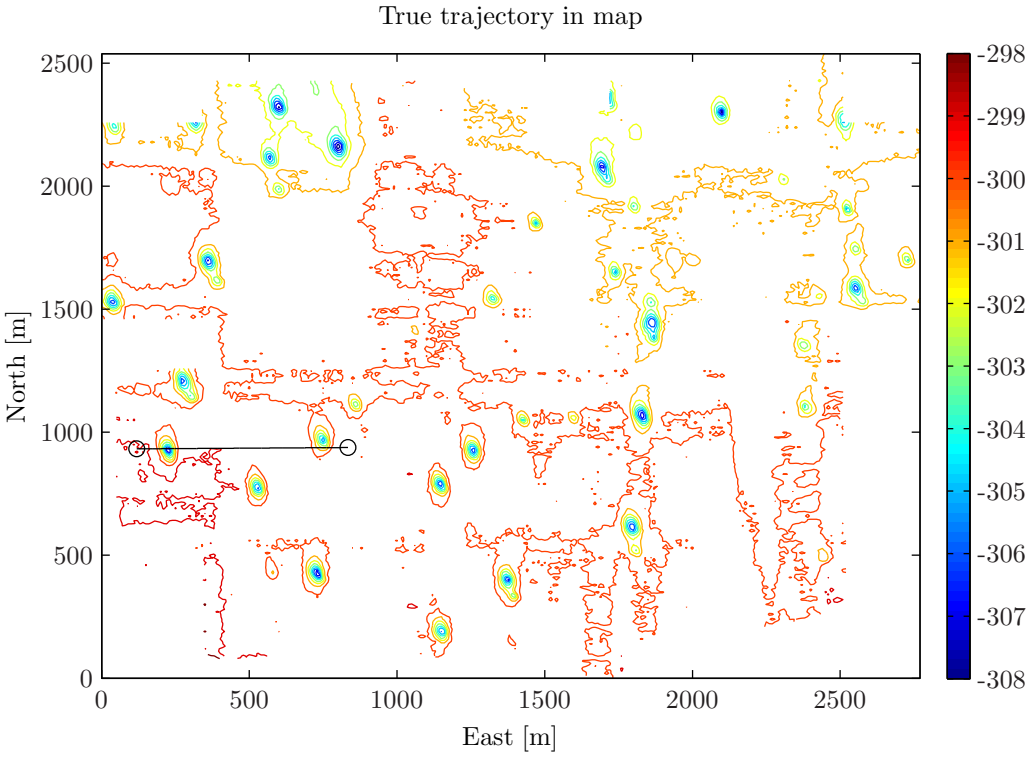


Figure 5.28: *Trajectory of Scenario 2. The direction of travel is from left to right in the figure.*

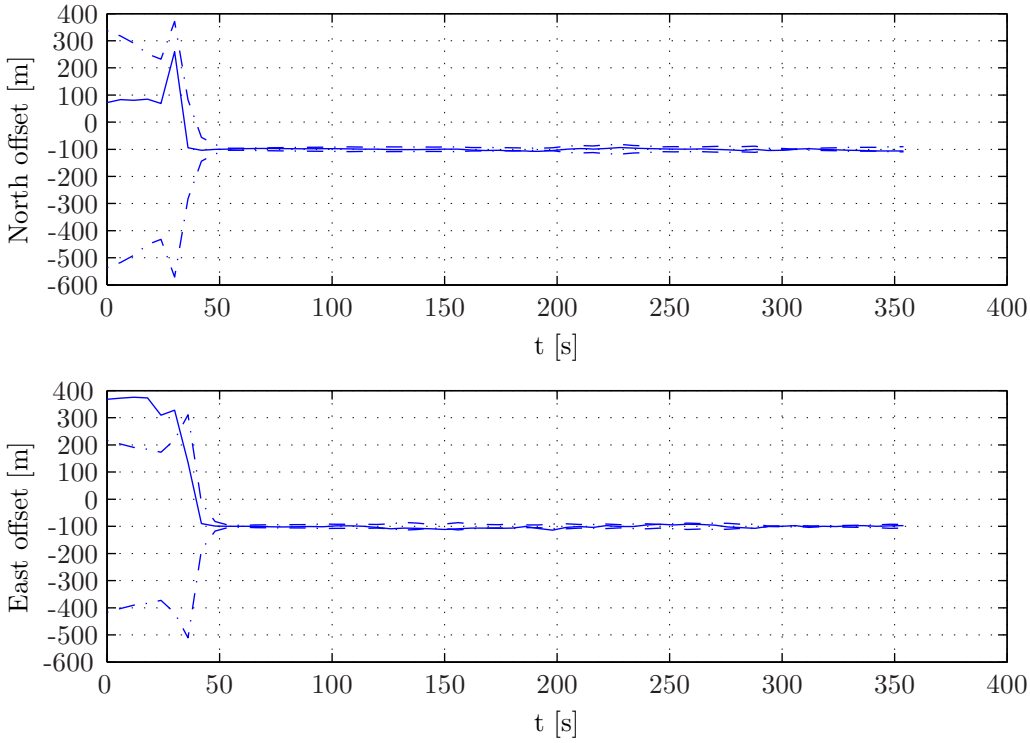


Figure 5.29: *Estimated north and east offset from PDF (solid line) and estimated uncertainty plotted around true offset (dash-dot), Scenario 2. The true offset is* $[-100\ m, -100\ m]$*. The PMF was initialized with an initial standard deviation of 500 meters in each direction.*

Summary

The simulations presented above indicate that terrain with pockmarks is suited for underwater terrain navigation using MBEs. In both scenarios presented, the terrain navigation algorithm was able to estimate the correct position of the vehicle within an accuracy comparable to that of the horizontal resolution of the depth map, 10 meters in this case. The presented results were from the PMF algorithm, but nearly identical results were obtained from particle filtering methods. In the terrain used in this paper, the pockmarks seem to be sufficiently different for the terrain navigation methods to obtain a unique match. However, when used in a real system, care should be taken in the case of multimodal estimated probability density functions. If possible, the terrain navigation solution should not be used until a unimodal density is obtained. If problems with multimodal densities occur, it would be advantageous to include several pockmarks in the measured terrain profile, such that the relative locations of the pockmarks are used, together with the shapes of the individual pockmarks.

The use of pockmarks for underwater terrain navigation facilitates the use of this

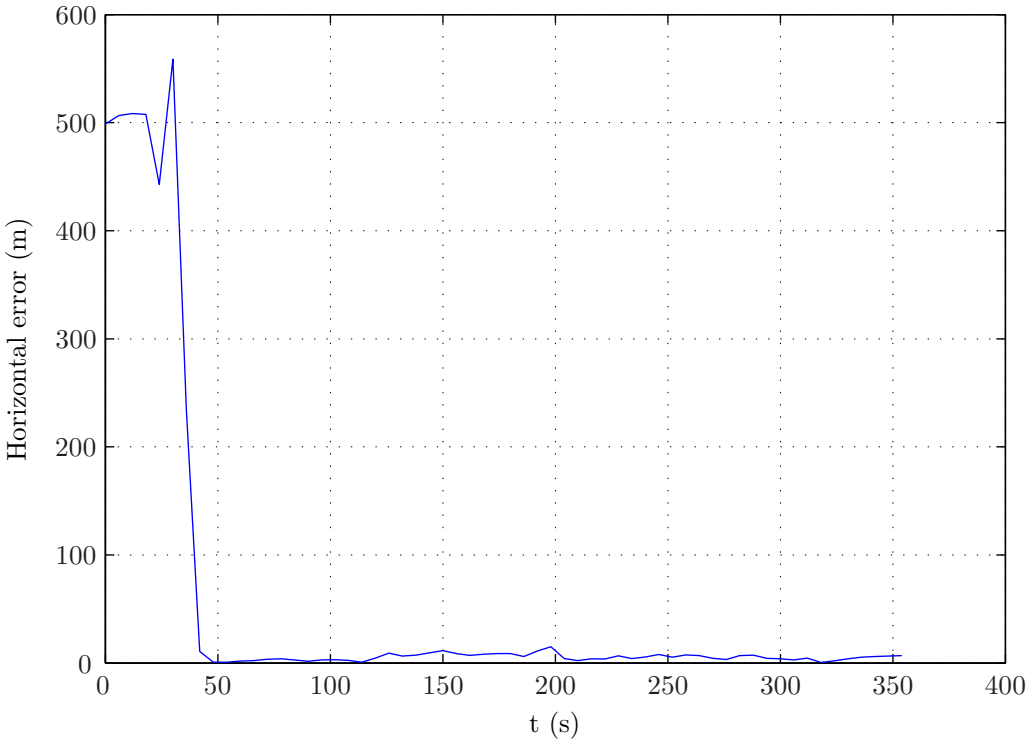


Figure 5.30: *Horizontal error in PMF estimate, Scenario 2.*

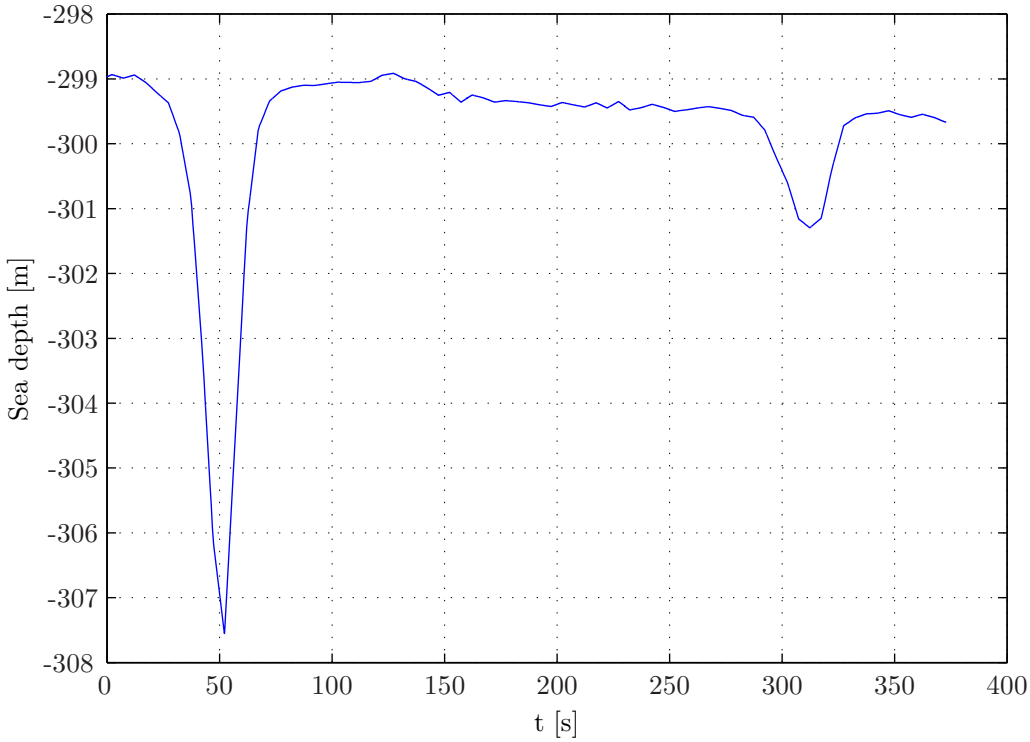


Figure 5.31: *Sea depth along vehicle trajectory, Scenario 2.*

navigation technique in new areas. As pockmarks often occur in areas with relatively flat terrain, where terrain navigation would otherwise be difficult, pockmark utilization extends the application possibilities of terrain navigation considerably. Pockmarks can also be used when planning an operation. If an area is known to contain pockmarks, it could be included in the path plan in order to obtain position fixes in this area. In areas with modest terrain variations but where distinct features are present, feature-based navigation methods have traditionally been thought of as the only option. Using traditional bathymetric terrain navigation instead may in some cases eliminate some of the problems related to feature-based navigation, e.g. the data association problem, the problem of correctly determining which of the observed features correspond to the same physical object.

5.3 Conclusions

The results in this chapter have demonstrated the performance of a suit of different terrain navigation algorithms on real AUV data. The performance in various terrain types, from rough areas, through valleys, pockmark areas and flat areas have demonstrated the terrain dependence of the algorithms. Through the estimated covariance matrix, all the methods, except for the TERCOM algorithm, are able to asses the quality of their own estimates.

The main problem throughout all the tests in this chapter is the tendency of overconfidence in the algorithms, due to the discrepancy between the true system and the filter model as a consequence of the limited number of states that are possible to estimate in the methods used. The discrepancy is present both in the process model and the measurement model. To some extent, the overconfidence has been succesfully reduced using sub-sampling of the measurements, to minimize the effect of unmodeled correlations in the methods. The attempt to improve the process model by extending the state space model yielded slightly more stable and accurate estimates, but the overconfidence turned out to be even stronger.

Of all the methods presented in this thesis, the 3-dimensional version of the PMF algorithm had the best performance on the experimental data, both in terms of accuracy and robustness to various error sources. Its main disadvantage is its computational complexity, though it is expected that the complexity can be significantly reduced using smart implementation techniques.

The occurrence of false fixes in the computational results demonstrates the need for integrity checks before the terrain navigation update is fed back to the main navigation system. Though the frequency of such false fixes is low, especially in the PMF method, they may be damaging to the overall performance of the navigation system.

6

Concluding Remarks

6.1 Main Conclusions

This thesis has dealt with the problem of utilizing terrain measurements for navigation of underwater vehicles. A number of different estimation algorithms have been presented, and their performance on real AUV data have been compared. Various aspects specific to the application on these methods in an underwater environment have been discussed. Throughout the thesis, the terrain navigation system has been treated as an external module, providing measurement updates for the main navigation system in a loosely coupled manner. This approach makes the terrain navigation module more portable and the overall system more robust to errors in the terrain navigation updates, although it is more difficult to exploit the internal states of the main navigation system in the terrain navigation algorithms.

The thesis started with a chapter on the general concepts and earlier work on terrain aided navigation (Chapter 2). Most of the effort was put on the class of Bayesian terrain navigation methods, in which the problem is formulated in a state space model. Due to the nonlinear nature of the problem, which stems from the terrain dependency of the measurement function, nonlinear estimation methods are needed. The Bayesian estimation methods point mass filters (PMFs), particle filters (PFs) and Sigma Point Kalman filters (SPKFs) were discussed in the framework of terrain aided navigation. Special attention was given to the fact that, due to the computational complexity of the estimation methods, discrepancies between the true system and the filter model arise and may be a source of inaccuracies in the terrain navigation results.

In Chapter 3, the special problems that arise when using terrain navigation underwater were discussed, including a presentation of various sensors used for terrain measurements. As most modern underwater vehicles carry a multibeam echo sounder (MBE),

and this sensor is well suited for terrain navigation, a lot of the issues discussed were formulated with the MBE in mind. Section 3.2.1 dealt with a possible extension of the state space model, in an attempt to minimize the effects of the discrepancies between the true system and the filter model process equation. It was also shown, in Section 3.3.1, how unknown depth biases can be dealt with in the algorithms, either by estimating the depth bias as an additional state or by using relative profiles, in which the mean is subtracted. The chapter concluded with a brief discussion on the terrain dependency of terrain navigation algorithms; in order for the methods to be of use, a degree of terrain variation that is observable both in the sensor data and in the map database must be present.

Chapter 4 gave an introduction to various forms of map databases, the process of converting raw bathymetric data to a gridded DTM (digital terrain model), as well as listing the various error sources present in terrain models. The effects of all these different error sources have to be modeled in the terrain navigation measurement equation, which can be challenging, especially when there is little room for extra states in the state vector.

In Chapter 5, results from all of the previously discussed algorithms, using data from sea trials with a HUGIN AUV, equipped with an MBE, were presented. First, in Section 5.2.1 the performance of the TERCOM (Terrain Contour Matching) and that of the PMF algorithm were compared. It was concluded that both algorithms yielded position estimates within the resolution of the map in suited terrain, but that the PMF was more stable and robust. However, the PMF suffered from overconfidence due to unmodeled correlations.

Section 5.2.2 showed the effects of estimating the depth bias, and results using the PMF and a PF with 2 and 3-dimensional state vectors were presented. The results revealed a considerable gain in robustness and accuracy from taking the depth bias into account. The 3-dimensional PMF algorithm yielded the best results, both in terms of accuracy and robustness, though some of the same improvements could be accomplished by using relative profiles. In addition, the problem of overconfidence was significantly reduced by sub-sampling of the MBE beam, minimizing the effects of correlated measurements.

In Section 5.2.3, results from a particle filtering implementation of the extended state-space model of Section 3.2.1 were presented. Though the accuracy and stability of the position estimates were slightly improved, it turned out that the overconfidence problem was not solved by the process model extension. On the contrary, the new algorithm was even more overconfident than the original one.

Section 5.2.4 showed the behavior of the SPKF algorithm on the same data set used in the previous tests. Though the results showed that the SPFK can indeed be used for underwater terrain navigation, the results were not as accurate and robust as those of the PF and PMF.

Chapter 5 concluded with some simulations done on real bathymetric data from an area with pockmarks, i.e. small craters on the sea floor. The simulations indicated that such pockmarks contain enough terrain information for terrain navigation to be feasible.

As pockmarks often occur in areas where the terrain is otherwise flat, these results show that terrain navigation can be used in terrain types previously thought of as unsuited.

Based on all the different algorithms tested on real AUV data in Chapter 5, it can be concluded that the 3-dimensional PMF showed the best behavior, both in terms of accuracy and robustness. The main problem with this algorithm is its computational complexity, which might prohibit its use in a real-time application. It should be noted, however, that it has not been the focus of this thesis to make the implementation of this algorithm more efficient. An adaptive grid version would make it more efficient. The PMF is also well-suited for parallelization. The constant development of computer technology will also reduce this problem in the years to come.

The single most important conclusion of this thesis is the problem of overconfidence in the terrain navigation algorithms, which remains partially unsolved. When using terrain navigation updates in real-time navigation system, the overconfidence must be kept in mind. On the other hand, this problem does not prevent the use of the terrain navigation algorithms discussed in this thesis. As long as one is aware of the problem, the system can be tuned conservatively, in such a way that the performance of the overall system is not degraded. Nevertheless, such ad hoc tuning prevents the system from utilizing the available terrain information in an optimal manner.

6.2 Suggestions for Future Work

A number of challenges that arise in underwater terrain navigation have been pointed out in this thesis, and due to the time limitations present in a doctoral project, not all of them have been addressed properly. Some problems that should be investigated further are listed below, though this list is by no means complete:

- Gain a better understanding of the overconfidence effects, possibly by integrating the terrain navigation updates more tightly in the inertial navigation system. One approach would be to implement a Rao-Blackwellized particle filter like in Schon et al. (2005) and Nygren (2008). None of these mention the overconfidence effect.
- Implement the algorithms outlined in this thesis in a real-time system. This is actually an on-going research effort at FFI.
- Investigate the different integrity tests and convergence criteria for the terrain navigation results. These are important issues when it comes to implementing a robust real-time terrain navigation system.
- Implement more effective versions of the 3-dimensional and higher-dimensional PMF, possibly through adaptive grid approaches or parallelization.
- Use the algorithms in a simultaneous localization and mapping (SLAM) setting.

Bibliography

C. Andrieu, A. Doucet, and E. Punskaya. *Sequential Monte Carlo Methods in Practice*, chapter Sequential Monte Carlo Methods for Optimal Filtering, pages 79–95. Springer-Verlag, New York, 2001.

K. B. Ånonsen and O. Hallingstad. Sigma point Kalman filter for underwater terrain-based navigation. In *Proceedings of IFAC Conference on Control Applications in Marine Vehicles*, Bol, Croatia, September 2007.

K. B. Ånonsen, O. Hallingstad, O.K. Hagen, and M. Mandt. Terrain aided AUV navigation - a comparison of the point mass filter and terrain contour matching algorithms. In *Proceedings of the Undersea Defence Technology Conference (UDT) Europe 2005*, Amsterdam, the Netherlands, June 2005.

K.B. Ånonsen and O.K. Hagen. Terrain aided underwater navigation using pockmarks. In *Proceedings of the MTS/IEEE Oceans Conference*, Biloxi, MS, October 2009.

K.B. Ånonsen and O. Hallingstad. Terrain aided underwater navigation using point mass and particle filters. In *Proceedings of the IEEE/ION Position Location and Navigation Symposium (PLANS) 2006*, San Diego, CA, April 2006.

K.B. Ånonsen, O. Hallingstad, and O.K. Hagen. Bayesian terrain-based underwater navigation using an improved state-space model. In *Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2007*, pages 499–505, Tokyo, Japan, April 2007. doi: 10.1109/UT.2007.370773.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II State of the art. *IEEE Robotics & Automation Magazine*, 13(3):108–117, September 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1678144.

W. Baker and R. Clem. Terrain contour matching (TERCOM) primer. Technical report, Wright Patterson Air Force Base Aeronautical Systems Division (1977), OH, USA, 1977.

A. Bar-Gill, P. Ben-Ezra, and I. Y. Bar-Itzhack. Improvement of terrain-aided navigation via trajectory optimization. *IEEE Transactions on Control Systems Technology*, 2(4): 336–342, December 1994. ISSN 1063-6536. doi: 10.1109/87.338654.

Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, New York, 2001.

O. Bergem. *Bathymetric Navigation of Autonomous Underwater Vehicles Using a Multibeam Sonar and a Kalman Filter with Relative Measurement Covariance Matrices.* PhD thesis, Department of Informatics and Computer Science, College of Arts and Science, University of Trondheim, Norway, 1993.

N. Bergman. Bayesian inference in terrain navigation. Licentiate thesis LiU-TECH-LIC-1997:50, Department of Electrical Engineering, Linköping University, Sweden, 1997.

N. Bergman. *Recursive Bayesian Estimation - Navigation and Tracking Applications.* PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 1999.

N. Bergman, L. Ljung, and F. Gustafsson. Terrain navigation using Bayesian statistics. *IEEE Control Systems Magazine*, 19(3):33–40, June 1999. ISSN 0272-1708. doi: 10.1109/37.768538.

J.T. Bjørke. Computation of calibration parameters for multibeam echo sounders using the least squares method. *IEEE Journal of Oceanic Engineering*, 30(4):818–831, October 2005. ISSN 0364-9059. doi: 10.1109/JOE.2005.862138.

J.T. Bjørke and S. Nilsen. Computation of random errors in digital terrain models. *Geoinformatica*, 11(3):359–382, 2007.

D. Blackwell. Conditional expectation and unbiased sequential estimation. *Annalls of Mathematical Statistics*, 18:105–110, 1947.

Bluefin Robotics. Bluefin Robotics web page, www.bluefinrobotics.com, 2010. Retrieved February 16, 2010.

D. Boozer and J. Fellerhoff. Terrain-aided navigation test results in the AFTI/F-16 aircraft. *Journal of The Institute of Navigation*, 35(2):161–175, 1988.

M. Bosse. *ATLAS, A Framework for Large Scale Automated Mapping and Localization.* PhD thesis, Massachusetts Institute of Technology, 2004.

British Aerospace Ltd. The F-16 digital terrain system. In *IEE Colloquium on Terrain Databases and Their Use in Navigation and Collision Avoidance*, 1995.

R.S. Bucy and K.D. Senne. Digital synthesis of non-linear filters. *Automatica*, 7(3): 315–322, May 1971.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. 146(1):2–7, February 1999. ISSN 1350-2395. doi: 10.1049/ip-rsn:19990255.

G. Casella and P. Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83 (1):81..94, 1996.

H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, NJ, 1946.

D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, March 2002. ISSN 1053-587X. doi: 10.1109/78.984773.

N. Debese. Use of a robust estimator for automatic detection of isolated errors appearing in bathymetry data. *International Hydrographic Review*, 2(2):32–44, September 2001.

D.E. Di Massa. *Terrain-Relative Navigation for Automous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology/Woods Hole Oceanographic Institution, 1997.

A. Doucet. On sequential simulation-based methods for Bayesian filtering. Technical report, Department of Engineering, Cambridge University, Cambridge, UK, 1998.

A. Doucet, N. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Uncertainty in Artificial Intelligence*, 2000a.

A. Doucet, S. Godsill, and Cristophe Andrieu. On sequential monte carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000b.

A. Doucet, J.F.G de Freitas, and N.J. Gordon. *Sequential Monte Carlo Methods in Practice*, chapter An Introduction to Sequential Monte Carlo Methods. New York: Springer Verlag, 2001.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I The essential algorithms. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

N. El-Sheimy, C. Valeo, and A Habib. *Digital Terrain Modeling - Acquisition, Manipulation, and Applications*. Artech House, Norwood, MA, 2005.

R. Enns and D. Morrell. Terrain-aided navigation using the Viterbi algorithm. *Journal of Guidance, Control and Dynamics*, 18(6):1444–1449, 1995.

R. Eustice. *Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology/Woods Hole Oceanographic Institution, June 2005.

R. Eustice, O. Pizarro, and H. Singh. Visually augmented navigation in an unstructured environment using a delayed state history. In *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the rms titanic with SLAM information filters. In *Robotics: Science and Systems, Conference Proceedings*, 2005.

N.P. Fofonoff and R.C. Millard, Jr. Algorithms for computation of fundamental properties of sea water. *UNESCO Technical Papers in Marine Science*, 44, 1983.

G.B. Folland. *Real Analysis*. John Wiley and Sons, New York, 2 edition, 1999.

G.D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973. ISSN 0018-9219.

K. Gade. NavLab, a generic simulation and post-processing tool for navigation. *European Journal of Navigation*, 2(4):51–59, November 2004. URL `www.navlab.net`.

A. Gelb, editor. *Applied Optimal Estimation*. The MIT Press, Cambridge, MA, 1974.

J. Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

J.P. Golden. Terrain contour matching (TERCOM): A cruise missile guidance aid. In T.F. Wiener, editor, *Image Processing for Missile Guidance*, volume 238, pages 10–18, San Diego, CA, 1980. The Society of Photo-Optical Instrumentation Engineers.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113, 1993.

P.E. Hagen, T.G. Fossum, and R.E. Hansen. Applications of AUVs with SAS. In *Proceedings of the MTS/IEEE Oceans Conference*, Quebec City, QC, Canada, September 2008.

P.E. Hagen, Ø. Hegrenæs, B. Jalving, Ø Midtgaard, M. Wiig, and O.K. Hagen. *Underwater Vehicles*, chapter Making AUVs Truly Autonomous, pages 129–152. In-Tech Education and Publishing, Vienna, Austria, 2009.

M.P. Hayes and P.T. Gough. Synthetic aperture sonar: A review of current status. *IEEE Journal of Oceanic Engineering*, 34(3):207–224, 2009.

Ø. Hegrenæs, E. Berglund, and O. Hallingstad. Model-aided inertial navigation for underwater vehicles. In *Proceedings of the IEEE Conference on Robotics and Automation 2008 (ICRA-08*, pages 91–119, Pasadena, CA, May 2008.

Ø. Hegrenæs, K. Gade, O.K. Hagen, and P.E. Hagen. Underwater transponder positioning and navigation. In *Proceedings of the MTS/IEEE Oceans Conference*, Biloxi, MS, October 2009.

A.J. Henley. Terrain aided navigation: Current status, techniques for flat terrain and reference data requirements. In *Proceedings of the IEEE Position Location and Navigation Symposium (PLANS)*, pages 608–615, Las Vegas, NV, March 1990. doi: 10.1109/PLANS.1990.66235.

J. Hollowell. Heli/SITAN: A terrain referenced navigation algorithm for helicopters. In *Proceedings of the IEEE Position Location and Navigation Symposium (PLANS)*, pages 616–625, Las Vegas, NV, March 1990. doi: 10.1109/PLANS.1990.66236.

L.D. Hostetler. Optimal terrain-aided navigation systems. In *AIAA Guidance an Control Conference*, Palo Alto, CA, 1978.

M. Hovland and A. Dudd. *Seabed Pockmarks and Seepages*. Graham and Trotman, London, 1988.

M.T. Huang, G.J. Zhai, Y.Z. Ouyang, and Z. Guan. Robust method for the detection of abnormal data in hydrography. *International Hydrographic Review*, 76(2):93–102, September 1999.

International Hydrographic Organization. IHO transfer standard for digital hydrographic data. Technical report, International Hydrographic Organization, 1992.

A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, Cambridge, UK, 1996.

M. Jakobsson, B. Calder, and L. Mayer. On the effect of random errors in gridded bathymetric compilations. *Journal of Geophysical Research*, 107(B12), 2002.

B. Jalving, K. Gade, O.K. Hagen, and K. Vestgård. A toolbox of aiding techniques for the HUGIN AUV integrated inertial navigation system. In *Proceedings of the IEEE Oceans 2003*, San Diego, CA, 2003.

B. Jalving, K. Gade, K. Svartveit, A. Willumsen, and R. Sorhagen. DVL velocity aiding in the HUGIN 1000 integrated inertial navigation system. *Modeling, Identification and Control*, 25(4):223–235, October 2004a. ISSN 0332-7353.

B. Jalving, M. Mandt, O.K. Hagen, and F. Pøhner. Terrain referenced navigation of AUVs and submarines using multibeam echo sounders. In *Proceedings of the UDT Europe 2004*, Nice, France, 2004b.

A. Jazwinski. *Stochastic Process and Filtering Theory*, volume 64 of *Mathematics in Science and Engineering*. Academic Press, New York, 1970.

S.J. Julier and J.K Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proc. of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, volume 92, pages 401–422, 2004. doi: 10.1109/JPROC.2003.823141.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, 82:35–45, 1960.

R.E. Kalman and R.S. Bucy. New results in linear filtering and prediction theory. *Transactions of the ASME – Journal of Basic Engineering*, 83:95–107, 1961.

R. Karlsson, F. Gustafsson, and T. Karlsson. Particle filtering and Cramér-Rao lower bound for underwater navigation. In *Proceedings IEEE Conference on Acoustics, Speech and Signal Processing*, Hong Kong, April 2003.

S.M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*, volume 1 of *Prentice-Hall Signal Processing Series*. Prentice Hall PTR, Upper Saddle River, NJ, 1993.

J.C. Kinsey, R.M. Eustice, and L.L. Whitcomb. A survey of underwater vehicle navigation: Recent advances and new challenges. In *Proceedings of the 7th IFAC Conference of Maneuvering and Control of Marine Craft (MCMC)*, Lisbon, Portugal, 2006.

F.C. Klebaner. *Introduction to Stochastic Calculus with Applications*. Imperial College Press, London, 1998.

A. Kong, J.S. Liu, and W.H. Kong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89:278–288, 1994.

Kongsberg Maritime AS. Autonomous Underwater Vehicle – AUV: The HUGIN Family. Available online: http://www.km.kongsberg.com (Retrieved August 4 2009), 2009.

A. Lang. Estimation methods for terrain navigation. Royal Institute of Technology, Stockholm, Sweden, October 2006. Available online from: http://www.ee.kth.se. Retrieved March 29, 2007.

M. B. Larsen. Synthetic long baseline navigation of underwater vehicles. In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 3, pages 2043–2050, Providence, RI, September 2000. doi: 10.1109/OCEANS.2000.882240.

J.S. Liu. Metropolized independent sampling with comparison to rejection sampling and importance sampling. *Statistics and Computing*, 6:113–119, 1996.

J.S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

X. Lurton. *An Introduction to Underwater Acoustics.* Springer Praxis Books, 2002.

M. Mandt. Terrengreferert posisjonering av undervannsfarkoster. Technical Report FFI/RAPPORT-2001/05900, Norwegian Defense Research Establishment, 2001. In Norwegian.

Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 141 of *Mathematics in Science and Engineering.* Academic Press, Inc., 1979.

C. Miller and R.A. LaFlamme. The digital terrain modeling-theory and applications. *Photogrammetric Engineering*, 24(3):433–443, 1958.

P.H. Milne. *Underwater Acoustic Positioning Systems.* Gulf Publishing Company, Houston, TX, 1983.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI-02*, 2002.

J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, September 2001.

P.M. Newman and J.J. Leonard. Pure range-only subsea SLAM. In *Proceedings of the IEEE International Conference of Robotics and Automation (ICRA)*, Tapei, Taiwan, September 2003.

P.J. Nordlund. Sequential Monte Carlo filters and integrated navigation. Licentiate Thesis LiU-TECH-LIC:2002:18, Department of Electrical Engineering, Linköping University, Sweden, 2002.

I. Nygren. *Terrain Navigation for Underwater Vehicles.* PhD thesis, Department of Signals, Sensors and Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, 2005.

I. Nygren. Robust and efficient terrain navigation of underwater vehicles. In *Proceedings of the IEEE/ION Position Location and Navigation Symposium*, pages 923–932, May 2008. doi: 10.1109/PLANS.2008.4570034.

B. Øksendal. *Stochastic Differential Equations.* Springer, 1985.

M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1157–1164, 2003.

T. Peucker, R. Fowler, J. Little, and D. Mark. The triangulated irregular network. In *Proceedings of the Digital Terrain Model Symposium*, St. Louis, MO, 1978.

L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

C.R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 1945.

C.R. Rao. *Linear Statistical Inference and its Applications*. John Wiley & Sons, New York, 1965.

RD Instruments. Acoustic Doppler current profiler: Principles of operation - a practical primer. Technical report, RD Instruments, San Diego, CA, USA, 1996.

M. Renardy and R. C. Rogers. *An Introduction to Partial Differential Equations*, volume 13 of *Texts in Applied Mathematics*. 1993. ISBN 0-387-97952-2.

B.D. Ripley. *Stochastic Simulation*. Wiley, New York, 1987.

C. Roman. *Self Consistent Bathymetric Mapping from Robotic Vehicles in the Deep Ocean*. PhD thesis, Massachusetts Institute of Technology/Woods Hole Oceanographic Institution, 2005.

S. M. Ross. *Introduction to Probability Models*. Academic Press, San Diego, CA, 1972.

D.B. Rubin. *Bayesian Statistics 3*, chapter Using the SIR Algorithm to Simulate Posterior Distributions, pages 395–492. Oxford University Press, 1988.

P. G. Savage. Strapdown inertial navigation integration algorithm design – part 1: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, 21(1), January–February 1998a.

P. G. Savage. Strapdown inertial navigation integration algorithm design – part 2: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, 21(2), March–April 1998b.

T. Schon, F. Gustafsson, and P.J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7): 2279–2289, July 2005. ISSN 1053-587X. doi: 10.1109/TSP.2005.849151.

C.M. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in computer vision and image understanding. *Autonomous Robot Vehicles, Springer Verlag*, pages 167–193, 1990.

S. Thrun, Y. Liu, D. Koller, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7–8):693–716, 2004. doi: 10.1177/0278364904045479.

D.H. Titterton and J.L. Weston. *Strapdown Inertial Navigation Technology*. Institution of Engineering and Technology, 2004.

S-X. Tsai. Introduction to the scene matching missile guidance technologies. Technical report, National Air Intelligence Center Wright Patterson Air Force Base, OH, USA, 1996.

M. Uijt de Haag and A. Vadlamani. Flight test evaluation of various terrain referenced navigation techniques for aircraft approach guidance. In *Proceedings of the IEEE/ION Position Location and Navigation Symposium 2006*, pages 440–449, April 2006.

H.L. van Trees. *Detection, Estimation and Modulation Theory*. Wiley, New York, 1968.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967. ISSN 0018-9448.

R. L. Wernli. AUV commercialization – Who's leading the pack? In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 1, pages 391–395, Providence, RI, USA, 2000. doi: 10.1109/OCEANS.2000.881290.

S.B. Williams, P. Newman, G. Dissanayake, and H.F. Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, April 2000.

A. B. Willumsen, O. K. Hagen, and P. N. Boge. Filtering depth measurements in underwater vehicles for improved seabed imaging. In *Proceedings of the IEEE Oceans Europe Conference*, Aberdeen, UK, June 2007. doi: 10.1109/OCEANSE.2007.4302249.

A

Notation

Abbreviations and Acronyms

Table A.1: *Abbreviations and acronyms*

Abbreviation	Explanation
AUV	Autonomous Underwater Vehicle
CRLB	Cramér-Rao Lower Bound
CTD	Conductivity, temperature and density
DGPS	Differential GPS
DTM	Digital Terrain Model
EKF	Extended Kalman Filter
FFI	Forsvarets Forskningsinstitutt (Norwegian Defence Research Establishment)
FIM	Fisher Information Matrix
GIS	Geographical Information System
GPS	Global Positioning System
i.i.d	Identically and independently distributed
IHO	International Hydrographic Organization
IMO	International Maritime Organization
IMU	Inertial Measurement Unit
INS	Inertial Navigation System

Continued on next page

Table A.1 – continued from previous page

Abbreviation	Explanation
KF	Kalman Filter
LBL	Long Base Line
MAP	Maximum A Posteriori
MBE	Multibeam Echo sounder
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
MSL	Mean Sea Level
PDE	Partial Differential Equation
PDF	Probability Density Function
PF	Particle Filter
PMF	Point Mass Filter
psu	Practical Salinity Unit
RBPF	Rao-Blackwellized Particle Filter
RMS	Root Mean Square
ROV	Remotely Operated Vehicle
RTK GPS	Real Time Kinematic GPS
SBE	Single Beam Echo sounder
SITAN	Sandia Terrain Aided Navigation
SMCF	Sequential Monte Carlo Filter
SPKF	Sigma Point Kalman Filter
TERCOM	Terrain Contour Matching
TerrNav	Terrain (aided) Navigation
TIN	Triangular Irregular Network
TRIN	Terrain Referenced Integrated Navigation
UKF	Unscented Kalman Filter
USBL	Ultra Short Base Line

Mathematical Notation

Table A.2: *Mathematical symbols used in the thesis*

Symbol	Explanation
$\boldsymbol{v}$	Vector quantity
s	Scalar quantity
$\boldsymbol{A}$	Matrix, $\boldsymbol{A} = [a_{ij}]$
$\boldsymbol{A}^T$	Matrix transpose
$\boldsymbol{f}(\boldsymbol{x})$	Vector valued function $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$
$\boldsymbol{v}^f$	Vector decomposed in frame f
$\boldsymbol{R}_a^b$	Coordinate transformation matrix, $\boldsymbol{v}^b = \boldsymbol{R}_a^b \boldsymbol{v}^a$
$\mathrm{d}\boldsymbol{x}$	Multidimensional differential, $\mathrm{d}\boldsymbol{x} = \mathrm{d}x_1\, \mathrm{d}x_2 \dots \mathrm{d}x_{n_x}$
$p_x(\boldsymbol{x})$	Probability density function of the stochastic vector $\boldsymbol{x}$. With a slight abuse of notation the subscript is often omitted, e.g. $p(x) \neq p(y)$
$E[\boldsymbol{x}]$	Expectation of the stochastic vector $\boldsymbol{x}$, $E[\boldsymbol{x}] = \int \boldsymbol{x} p_x(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$
$\hat{\boldsymbol{x}}$	Estimate of quantity $\boldsymbol{x}$
$\bar{\boldsymbol{x}}$	Mean or prediction of quantity $\boldsymbol{x}$
$\sigma_x(x)$	Variance of one-dimensional stochastic variable x, $\sigma_x(x) = E[x - \bar{x}]$
$\boldsymbol{P}_x(\boldsymbol{x})$	Covariance matrix of stochastic vector $\boldsymbol{x}$, $\boldsymbol{P}_x(\boldsymbol{x}) = E[(\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^T]$
$\boldsymbol{x}_k$	Quantity at time step k
$\boldsymbol{x}_k^*$	Quantity in filter model
$\boldsymbol{X}_k$	Collection (history) of quantities, $\boldsymbol{X}_k = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_k\}$
$\mathcal{N}(\bar{\boldsymbol{x}}, \boldsymbol{C})$	Multidimensional normal/Gaussian pdf with mean $\bar{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{C}$
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle

Table A.3: *Coordinate systems used in the thesis*

Symbol	Description
e	Earth-centered, Earth-fixed (ECEF) coordinate system
n	Local north, east down (NED) coordinate system
b	Body fixed coordinate system
b'	Body fixed, roll and pitch compensated coordinate system
m	Map coordinate system (earth fixed)