Terrain Aided Underwater Navigation Using Point Mass and Particle Filters

Kjetil Bergh Ånonsen, Oddvar Hallingstad

University Graduate Center (UniK), Kjeller, Norway
Norwegian University of Science and Technology, Department of Engineering Cybernetics, Trondheim, Norway

***Abstract*— This paper focuses on obtaining submerged position fixes for underwater vehicles from comparing bathymetric measurements with a bathymetric map. Our algorithms are tested on real data, collected by a HUGIN AUV equipped with a multibeam echo sounder (MBE). Due to our strongly non-linear and non-Gaussian problem, local linearization methods such as the extended Kalman filter (EKF), has proven unsuitable in many terrain types. We therefore focus on two different recursive Bayesian methods, namely the point mass filter (PMF) and a Bayesian bootstrap particle filter (PF). The PMF is a grid-based approximation method, whereas the PF is a sequential Monte Carlo method, based on sampling from the underlying distributions. The two methods are first compared using a 2-dimensional state-space model, only estimating the horizontal position of the vehicle. A problem with this approach is that uncertainty in the estimation of tidal levels may lead to large errors in the depth measurements. To counter this, one may extend the state vector to three dimensions, and this is investigated in the paper. Characteristics such as accuracy, convergence time, terrain dependency, and computational demands for the two methods are compared. In suited terrain both methods yield a horizontal accuracy comparable to the resolution of the map. The results from the PMF are slightly better, but the PMF is also more computationally demanding than the PF. Extending the methods from 2D to 3D makes them significally more robust to depth errors. The computational demands for the PMF increases dramatically in 3D, whereas they are virtually unchanged for the PF. Previous problems of overconfidence in these methods are effectively reduced through sub-sampling of the MBE data.**

I. Introduction

Navigation of most autonomous underwater vehicles (AUVs) and submarines is based on inertial navigation systems (INS). To limit the drift in these systems, different aiding techniques are used, e.g. Doppler velocity logs. Even with Doppler aiding these systems will drift off with time, and to allow the vehicle to stay submerged for longer operations, position fixes are needed. Since GPS signals are blocked by water, GPS fixes are only available when an antenna is above the water surface. It is therefore desirable to develop methods that obtain position fixes while the vehicle is submerged. One solution is to use acoustic ranging to underwater transponders [1]. We focus on another possibility, namely that of obtaining fixes derived from terrain profile measurements, often referred to as terrain aided navigation or simply terrain navigation.

Terrain navigation has been used for decades in air and land navigation systems [2], [3], [4]. However, few results from underwater applications have been published. Some of the issues of underwater terrain navigation are addressed in [5], [6], [7]. In [8] the behavior of the 2-dimensional point mass filter (PMF) and terrain contour matching (TERCOM) algorithms for terrain based underwater navigation using multibeam echo sounders (MBEs) was discussed. The PMF approximates the the integrals in the optimal non-linear Bayes estimator by using a grid of point masses, converting the integrals to finite sums over this grid. Another approach is to approximate the probability density by a set of samples, or particles, drawn from the underlying probability distributions. This leads to a family of recursive Bayesian methods known as particle filters (PF) or sequential Monte Carlo methods [9], [10]. We will consider one particular particle filter, namely the Bayesian bootstrap algorithm [11], [4]. The methods will first be tested in a 2-dimensional setting, estimating northerly and easterly position only. To be able to estimate depth uncertainties, for example due to erroneous tide estimates, 3-dimensional models will be developed for the PMF and the PF and tested on real data. While we focus on AUV navigation, the methods presented may also be used by other types of underwater vehicles, like submarines.

II. The Estimation Methods

In our application, the measurements represent the total water depth at the MBE footprint, i.e. the sum of the AUV depth, calculated by a pressure sensor, and the altitude measurements from the MBE. To be able to compare these measurements to the depth values in the map data base, they must be converted to the same horizontal datum as the map database, usually mean sea level. Due to errors in the pressure-to-depth conversion and uncertain tidal estimates, this may lead to large errors, which are strongly correlated both between MBE beams and between consecutive time steps. When 2-dimensional filter models are used, these errors will in turn lead to larger estimat uncertainties and in some cases failure of the terrain navigation algorithms. There are several ways to counteract such errors. The simplest is to still work in a 2-dimensional setting, using relative terrain profiles only, i.e. subtracting the mean or median from the profiles before comparing them. However, this may lead to false results in self-similar terrain, and should therefore be used with care. Another approach is to extend our models to a 3-dimensional setting, estimating depth error together with the 2-dimensional position offset. In this section, we will first present the truth model of our system, before developing both 2-dimensional

and 3-dimensional filter models for the problems and show how these can be estimated using point mass and particle filters.

A. System Model

We first present the system (truth) model. We consider the following model for the motion of our AUV,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k + \mathbf{u}_k + \mathbf{v}'_k, \tag{1}$$
$$\mathbf{v}_{k+1} = \mathbf{g}(\mathbf{v}_k) + \mathbf{v}''_k, \tag{2}$$

where $\mathbf{x}_k = (x_{k,N}, x_{k,E}, x_{k,D})^T$ is the AUV position vector (north, east and depth), $\mathbf{u}_k$ is the position change from time step k to $k+1$, calculated from the inertial navigation system, and $\mathbf{v}'_k$ and $\mathbf{v}''_k$ are independent white noise sequences. Equation (2) models the strongly correlated error propagation of the inertial navigation system. The system measurement equation is given by

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{I} \cdot b_{k,D} + \mathbf{w}_k, \tag{3}$$

where $b_{k,D}$ is a depth bias term, related to tidal errors, and it is modeled as a constant bias term. $\mathbf{I}$ denotes the identity matrix.

Since we are dealing with multibeam echo sounders, the bottom depth measurement $\mathbf{z}_k$ is a vector measurement. The function $\mathbf{h}(\mathbf{x}_k)$ denotes the true mean sea depth at position $\mathbf{x}_k$, and has to be estimated by a digital terrain map. The term $\mathbf{w}_k$ denotes the sensor measurement noise, which is assumed to be white.

In order to simplify the mathematical description of our algorithms, we will express our true position as an offset, $\delta\mathbf{x}_k$, from the position estimate $\tilde{\mathbf{x}}_k$ of the INS. Our process then becomes, using that $\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \mathbf{u}_k$,

$$\delta\mathbf{x}_k = \mathbf{x}_k - \tilde{\mathbf{x}}_k, \tag{4}$$
$$\delta\mathbf{x}_{k+1} = \delta\mathbf{x}_k + \mathbf{v}_k + \mathbf{v}'_k, \tag{5}$$
$$\mathbf{v}_{k+1} = \mathbf{g}(\mathbf{v}_k) + \mathbf{v}''_k, \tag{6}$$
$$b_{k+1,D} = b_{k,D}. \tag{7}$$

with the measurement equation

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{I} \cdot b_{k,D} + \mathbf{w}_k. \tag{8}$$

B. 2-Dimensional Filter Model

Due to the computational requirements of the point mass filter, we first restrict ourselves to a 2-dimensional state vector, representing the horizontal position of the vehicle. The depth offset $\delta\mathbf{x}_{k,D}$ and the bias $b_{k,D}$ are therefore assumed to be zero. As there is no room for additional error states in our state vector, all our noises have to be modeled as white, and we have to consider a simpler system than that in (1)–(3). This will lead to a mismatch between the true system and the filter model, and this mismatch is in itself a major source of error. In order to distinguish between variables in our system and filter models, we attach asterisks to the variables in the filter model. Our filter model reads, using the same delta notation as in (4),

$$\delta\mathbf{x}^*_k = \mathbf{x}^*_k - \tilde{\mathbf{x}}_{k,P}, \tag{9}$$
$$\delta\mathbf{x}^*_{k+1} = \delta\mathbf{x}^*_k + \mathbf{v}^*_k, \tag{10}$$
$$\begin{aligned}\mathbf{z}_k &= \mathbf{h}^*(\mathbf{x}^*_k) + \mathbf{w}^*_k \\ &= \mathbf{h}^*(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}^*_k) + \mathbf{w}^*_k,\end{aligned} \tag{11}$$

where $\delta\mathbf{x}^*_k = (\delta x_{k,N}, \delta x_{k,E})^T$ and $\mathbf{x}_{k,P} = (x_{k,N}, x_{k,E})^T$ is the 2-dimensional horizontal subvector of $\mathbf{x}_k$. with the assumptions

$$E\{\mathbf{v}^*_k \mathbf{v}^{*T}_l\} = \mathbf{Q}^*_k \delta_{kl}, \tag{12}$$
$$E\{\mathbf{w}^*_k \mathbf{w}^{*T}_l\} = \mathbf{R}^*_k \delta_{kl}, \tag{13}$$

where δ_{kl} denotes the Kronecker delta, giving noise sequences that are uncorrelated from time step to time step. In (10) $\mathbf{v}_k$ and $\mathbf{v}'_k$ have been replaced by the white noise sequence $\mathbf{v}^*_k$. We also need to specify the densities of the noise sequences and the initial position offset, $\delta\mathbf{x}^*_0$. A convenient, but not necessary, assumption is to assume Gaussian densities,

$$p(\delta\mathbf{x}^*_0) = \mathcal{N}(\mathbf{0}, \mathbf{P}^*_0), \tag{14}$$
$$p(\mathbf{v}^*_k) = \mathcal{N}(\mathbf{0}, \mathbf{Q}^*_k), \tag{15}$$
$$p(\mathbf{w}^*_k) = \mathcal{N}(\mathbf{0}, \mathbf{R}^*_k). \tag{16}$$

Equation (14) indicates that the initial position has a Gaussian density centered around the estimated INS position $\tilde{\mathbf{x}}_0$. We further assume that the process noise, measurement noise and initial position are uncorrelated. The function $\mathbf{h}^*(\mathbf{x}^*_k)$ indicates the depth at position $\mathbf{x}^*_k$ and is given by the digital terrain map. We use terrain maps consisting of gridded nodes, and the depth values given by $\mathbf{h}^*$ are found by bilinear interpolation of the terrain database. The noise sequence $\mathbf{w}^*_k$ in (11) therefore models both map noise (including interpolation errors) and the sensor noise. The measurement $\mathbf{z}_k$ is the total sea depth at the current AUV position, and it is computed as the sum of the AUV depth, given by a pressure sensor, and the AUV altitude above the sea floor, given by the bathymetric sensor. The noise sequence $\mathbf{w}^*_k$ therefore contains contributions from map errors, pressure sensor noise and bathymetric sensor noise. For a detailed analysis of the depth accuracy of the HUGIN class AUVs, we refer to [12].

C. The Recursive Bayesian Filter Equations

Let $\mathbb{Z}_k$ be the augmented measurement vector consisting of all the measurements up to time step k. From Bayes formula (see e.g. [13]) and our filter model, (9)–(11), we have

$$\begin{aligned}p(\delta\mathbf{x}^*_k|\mathbb{Z}_k) &= \frac{p(\mathbf{z}_k|\delta\mathbf{x}^*_k, \mathbb{Z}_{k-1}) p(\delta\mathbf{x}^*_k|\mathbb{Z}_{k-1})}{p(\mathbf{z}_k|\mathbb{Z}_{k-1})} \\ &= \alpha_k^{-1} p_{\mathbf{w}^*_k}(\mathbf{z}_k - \mathbf{h}^*(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}^*_k)) \\ &\quad \cdot p(\delta\mathbf{x}^*_k|\mathbb{Z}_{k-1})\end{aligned} \tag{17}$$

where

$$\alpha_k = \int_{\mathbb{R}^2} \big[p_{\mathbf{w}_k^*}(\mathbf{z}_k - \mathbf{h}^*(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}_k^*)) \cdot p(\delta\mathbf{x}_k^*|\mathbb{Z}_{k-1})\big]\mathrm{d}\delta\mathbf{x}_k^*.$$

The minimum mean square error (MMSE) [13] estimate is then given by

$$\begin{aligned}\delta\hat{\mathbf{x}}_k &= E\{\delta\mathbf{x}_k^*|\mathbb{Z}_k\} \\ &= \int_{\mathbb{R}^2} \delta\mathbf{x}_k^* p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)\mathrm{d}\delta\mathbf{x}_k^*, \end{aligned} \tag{18}$$

with the covariance matrix

$$\hat{\mathbf{P}}_k = \int_{\mathbb{R}^2} \big[(\delta\mathbf{x}_k^* - \delta\hat{\mathbf{x}}_k^*)(\delta\mathbf{x}_k^* - \delta\hat{\mathbf{x}}_k^*)^T \cdot p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)\big]\mathrm{d}\delta\mathbf{x}_k^*. \tag{19}$$

For the time update of our position density we have,

$$\begin{aligned} p(\delta\mathbf{x}_{k+1}^*|\mathbb{Z}_k) &= \int_{\mathbb{R}^2} p(\delta\mathbf{x}_{k+1}^*, \delta\mathbf{x}_k^*|\mathbb{Z}_k)\mathrm{d}\delta\mathbf{x}_k^* \\ &= \int_{\mathbb{R}^2} p(\delta\mathbf{x}_{k+1}^*|\delta\mathbf{x}_k^*, \mathbb{Z}_k) \cdot p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)\mathrm{d}\delta\mathbf{x}_k^* \\ &= \int_{\mathbb{R}^2} p_{\mathbf{v}_k^*}(\delta\mathbf{x}_{k+1}^* - \delta\mathbf{x}_k^*) \cdot p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)\mathrm{d}\delta\mathbf{x}_k^*. \end{aligned} \tag{20}$$

Given the density of the initial position, $p(\delta\mathbf{x}_0^*)$, (17) and (20) can now be used recursively to obtain the density of the position offsets for each time step. However, the integrals in the equations are not analytically solvable, and we therefore need to evaluate these integrals numerically. This can be done by means of the point mass filter, where the densities are approximated using a grid of point masses, or by a particle set, using a particle filter.

D. The Point Mass Filter

We here present the point mass filter, first presented in [14] and introduced for navigation purposes in [4].

1) Point Mass Approximation: First we have to choose a search area around our position estimate $\tilde{\mathbf{x}}_k$ from the INS. 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. We start the algorithm by discretization of our 2-dimensional search area into a grid with M and N grid points in each direction. At each grid point we approximate the density $p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)$ by a probability weight $p(\delta\mathbf{x}_k^*(i,j)|\mathbb{Z}_k)$, where $i = 1, \ldots, M$ and $j = 1, \ldots, N$. For simplicity we will here consider uniform grids only, with grid resolution Δ in both directions. We start by letting

$$\begin{aligned} p(\delta\mathbf{x}_0^*(i,j)|\mathbb{Z}_0) &= p(\delta\mathbf{x}_0^*(i,j)) \\ &= \alpha_0^{-1} p(\delta\mathbf{x}_0^*)|_{\delta\mathbf{x}_0^* = \delta\mathbf{x}_0^*(i,j)}, \end{aligned}$$

where

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

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

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

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

$$\begin{aligned} p(\delta\mathbf{x}_k^*(i,j)|\mathbb{Z}_k) &= \alpha_k^{-1} p_{w_k^*}(\mathbf{z}_k - \mathbf{h}^*(\tilde{\mathbf{x}}_{k,P} \\ &\quad + \delta\mathbf{x}_k^*(i,j))) \cdot p(\delta\mathbf{x}_k^*(i,j)|\mathbb{Z}_{k-1}), \end{aligned} \tag{21}$$

where

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

and we are now able to compute our estimated position and its covariance:

$$\delta\hat{\mathbf{x}}_k = \sum_{i=1}^{M}\sum_{j=1}^{N} \delta\mathbf{x}_k^*(i,j) p(\delta\mathbf{x}_k^*(i,j)|\mathbb{Z}_k)\Delta^2, \tag{22}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \delta\hat{\mathbf{x}}_k, \tag{23}$$

$$\hat{\mathbf{P}}_k = \sum_{i=1}^{M}\sum_{j=1}^{N} \big[(\delta\mathbf{x}_k^*(i,j) - \delta\hat{\mathbf{x}}_k)(\delta\mathbf{x}_k^*(i,j) - \delta\hat{\mathbf{x}}_k)^T \cdot p(\delta\mathbf{x}_k^*(i,j)|\mathbb{Z}_k)\big]\Delta^2. \tag{24}$$

We then proceed to the time update, which is computed by

$$\begin{aligned} p(\delta\mathbf{x}_{k+1}^*(i,j)|\mathbb{Z}_k) &= \sum_{m=1}^{M}\sum_{n=1}^{N} \big[p_{\mathbf{v}_k^*}(\delta\mathbf{x}_{k+1}^*(i,j) - \delta\mathbf{x}_k^*(m,n)) \\ &\quad \cdot p(\delta\mathbf{x}_k^*(m,n)|\mathbb{Z}_k)\big]\Delta^2. \end{aligned} \tag{25}$$

Equation (25) can be viewed as a 2-dimensional convolution, which in general is computationally burdensome. As suggested in [4], we here use the assumption that the process

noise is assumed to be independent and identically distributed in both directions, such that the 2-dimensional convolution may be computed from two 1-dimensional convolutions. For AUVs, however, this assumption is not necessarily true, and this model should be improved upon in future work.

2) Implementational aspects: To ensure an efficient implementation of the PMF, it is useful to do some kind of adaptation of the grid while running the algorithm. In our MATLAB implementation of the PMF, we follow the guidelines given in [4, Sec 5.2.3 – 5.2.4]. Most of the time, many of the point masses $p(\mathbf{x}_k^*(i,j))$ will be very close to zero, and we therefore use sparse matrices for the point mass representation. After each measurement update the grid is truncated, by removing the point masses with a weight less than ε times the average weight of all the non-zero point masses. The truncation parameter ε is chosen a priori, and it is usually a small number, e.g. 0.05. This truncation will reduce the number of non-zero point masses, and in order to keep the number above a certain level, a minimum level N_0 is chosen. As soon as the number of point masses gets below N_0, the grid is refined, by inserting new grid points between every neighboring pair of grid points, using bilinear interpolation. Similarly, one also chooses a maximum number of non-zero point masses, N_1, and when this number is exceeded, the grid is decimated by removing every second row and column from the grid. Thus the PMF will be using a fine grid in areas where the variance of the estimate is low, and a coarser grid where the variance is high.

E. The Bayesian Bootstrap Particle Filter

Particle filtering has gained attention the recent years for estimation in nonlinear systems, in areas such as signal processing, tracking and navigation, to name a few. Introductions to the principles of particle filtering can be found in [9], [10]. We will here use the Bayesian Bootstrap particle filter, which is one of the simplest particle filter algorithms, first presented in [11]. The algorithm estimates the posterior density $p(\delta\mathbf{x}_k^*|\mathbb{Z}_k)$ at time step k by a set of particles, $\{\delta\mathbf{x}_k^i\}_{i=1,\dots,N}$, where N denotes the number of particles. The posterior density can be written as

$$p(\delta\mathbf{x}_k^*|\mathbb{Z}_k) \approx \frac{1}{N}\sum_{i=1}^{N}\delta\left(\delta\mathbf{x}_k^* - \delta\mathbf{x}_k^i\right), \tag{26}$$

where $\delta(\cdot)$ is the (2-dimensional) Dirac delta function. At each time step, starting with the particle set $\{\delta\mathbf{x}_k^i\}_{i=1,\dots,N}$, each of the particles are first predicted independently, by propagating them through the filter process equation (10). In our case this means adding a noise sampled from the process noise distribution, yielding the new particle set $\left\{\mathbf{x}_{k+1}'^i\right\}_{i=1,\dots,N}$. For each particle, a likelihood weight $w_{k+1}^i \propto p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}'^i)$ is computed, using the measurement $\mathbf{z}_{k+1}$. The weights are then normalized, such that $\sum_{i=1}^N w_{k+1}^i = 1$. The weighted particle set $\left\{\mathbf{x}_{k+1}'^i, w_{k+1}^i\right\}_{i=1,\dots,N}$ is now resampled with replacement N times, with the probability w_{k+1}^j of resampling particle $\mathbf{x}_{k+1}'^j$. We here use the systematic resampling procedure described in [9]. This completes the cycle, and we now have a particle set approximately distributed according to $p(\delta\mathbf{x}_{k+1}^*|\mathbb{Z}_{k+1})$.

The reason why we have chosen the Bayesian Bootstrap algorithm, is that it is one of the simplest particle filters to implement. Other algorithms, like the Sequential Importance Resampling (SIS) algorithm [10] have also been tested on our data, with very similar results to those from the Bayesian Bootstrap. We therefore concentrate on the Bayesian Bootstrap algorithm in this paper.

F. 3-Dimensional Filter Model

We now show how our filter model can be extended from two to three dimensions. As in the 2-dimensional case, we have to consider a simpler model in our filter, due to the computational demands of the point mass filter. Our filter model now becomes

$$\delta\mathbf{x}_{k+1}^* = \delta\mathbf{x}_k^* + \mathbf{v}_{k,P}^*, \tag{27}$$

$$b_{k+1,D}^* = b_{k+1,D}^* \tag{28}$$

$$\mathbf{z}_k = \mathbf{h}^*(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}_k^*) + \mathbf{I}\cdot b_{k,D} + \mathbf{w}_k^*, \tag{29}$$

where $\xi_k^* = (\delta x_{k,N}^*, \delta x_{k,E}^*, b_{k,D}^*)^T$ is our 3-dimensional filter state vector, $\mathbf{v}_k^* = (v_{k,N}^*, v_{k,E}^*, 0)^T$ is a 3-dimensional drift vector, and with the further assumptions

$$E\{\mathbf{v}_k^*\mathbf{v}_l^{*T}\} = \mathbf{Q}_k^*\delta_{kl}, \tag{30}$$

$$E\{\mathbf{w}_k^*\mathbf{w}_l^{*T}\} = \mathbf{R}_k^*\delta_{kl}. \tag{31}$$

As before, the process noise, measurement noise and initial position error from the inertial navigation system are assumed mutually independent. Furthermore, we still restrict ourselves to the case of Gaussian noise densities, though this is not necessary for our estimators to work. In the 3-dimensional model, the Bayesian filter equations now become

$$\begin{aligned} p\left(\xi_k^*|\mathbb{Z}_k\right) &= \frac{p(\mathbf{z}_k|\xi_k^*,\mathbb{Z}_{k-1})p\left(\xi_k^*|\mathbb{Z}_{k-1}\right)}{p(\mathbf{z}_k|\mathbb{Z}_{k-1})} \\ &= \alpha_k^{-1} p_{\mathbf{w}_k^*}(\mathbf{z}_k - \mathbf{h}^*\left(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}_k^*\right) \\ &\qquad - \mathbf{I}\cdot b_{k,D}^*)\cdot p\left(\xi_k^*|\mathbb{Z}_{k-1}\right), \end{aligned} \tag{32}$$

where

$$\begin{aligned} \alpha_k = \int_{\mathbb{R}^3} \big[p_{\mathbf{w}_k^*}\left(\mathbf{z}_k - \mathbf{h}^*\left(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}_k^*\right) - \mathbf{I}\cdot b_{k,D}^*\right) \\ \cdot p\left(\xi_k^*|\mathbb{Z}_{k-1}\right)\big]\mathrm{d}\xi_k^*. \end{aligned}$$

and

$$\begin{aligned} &p\left(\xi_{k+1}^*|\mathbb{Z}_{k+1}\right) \\ &\qquad = \int_{\mathbb{R}^3} p_{\mathbf{v}_k^*}\left(\xi_{k+1}^* - \xi_k^*\right) p\left(\xi_k^*|\mathbb{Z}_k\right)\mathrm{d}\xi_k^*. \end{aligned} \tag{33}$$

G. The 3-Dimensional Point Mass Filter

The equations for the 3-dimensional PMF are very similar to those of the 2-dimensional case, so we present them here without further derivations. The measurement update now becomes

$$p\left(\xi_k^*\left(i,j,l\right)|\mathbb{Z}_k\right) = \alpha_k^{-1} p_{\mathbf{w}_k}\big(\mathbf{z}_k - \mathbf{h}^*\left(\tilde{\mathbf{x}}_{k,P}\left(i,j,\cdot\right) + \delta\mathbf{x}_k^*\left(i,j,\cdot\right)\right) - \mathbf{I}\cdot b_{k,D}^*\left(\cdot,\cdot,l\right)\big)\cdot p\left(\xi_k^*\left(i,j,l\right)|\mathbb{Z}_{k-1}\right), \quad (34)$$

where

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

The prediction step becomes

$$p\left(\xi_{k+1}^*\left(i,j,l\right)|\mathbb{Z}_k\right) = \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{q=1}^{Q}\big[p_{\mathbf{v}_k^*}\left(\xi_{k+1}^*\left(i,j,l\right) - \xi_k^*\left(m,n,q\right)\right) \cdot p\left(\xi_k^*\left(m,n,q\right)|\mathbb{Z}_k\right)\big]\Delta^3. \quad (35)$$

The position estimate and its error covariance are given by

$$\hat{\xi}_k = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{Q}\xi_k^*\left(i,j,l\right)\cdot p\left(\xi_k^*\left(i,j,l\right)|\mathbb{Z}_k\right)\Delta^3, \quad (36)$$

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

$$\hat{\mathbf{P}}_k = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{l=1}^{Q}\big[\left(\xi_k^*\left(i,j,l\right) - \hat{\xi}_k\right) \cdot \left(\xi_k^*\left(i,j,l\right) - \hat{\xi}_k\right)^T p\left(\xi_k^*\left(i,j,l\right)|\mathbb{Z}_k\right)\big]\Delta^3. \quad (37)$$

H. The 3-Dimensional Bayesian Bootstrap Filter

The extension of the Bayesian Bootstrap particle filter from two to three dimensions is trivial. The samples are now drawn from 3-dimensional distributions, and the weights are updated according to

$$w_{k+1}^i \propto p(\mathbf{z}_{k+1}|\xi_{k+1}'^i) = p_{\mathbf{w}_k^*}\left(\mathbf{z}_k - \mathbf{h}^*\left(\tilde{\mathbf{x}}_{k,P} + \delta\mathbf{x}_k^*\right) - \mathbf{I}\cdot b_{k,P}^*\right).$$

III. Experimental Results

In this section we present results from the aforementioned methods, using real data from a Hugin AUV. See [1] for a description of the Hugin class AUVs. We first investigate the 2-dimensional methods, before moving on to the 3-dimensional methods. We also compare the full 3D-methods with 2D-methods using relative depth profiles only. The data were collected by a Hugin 1 vehicle in November 2001, in an area in the Oslofjord. The vehicle was equipped with a Kongsberg Maritime EM3000 MBE (300 kHz), which together with a pressure sensor yields the total sea depth along a line across the track of the vehicle. The MBE has 127 beams, but here only up to 92 beams were used. As mentioned in [8], some of the MBE beams will hit the sea floor within the same grid cells, leading to strong correlations between the errors related to these measurements. There will be correlations both within and between time steps. This often leads to overconfidence in the estimators, and to counter for this effect, a sub-sampling procedure was implemented, using only some of the beams. We used our algorithms to estimate the position offset from the real-time navigation solution $\tilde{\mathbf{x}}_k$. To investigate the robustness of the estimators to errors in this initial position, the methods were initialized with errors in the magnitude of 50–100 meters. For ground truth we used a smoothed trajectory obtained using NavLab post-processing of the real-time DVL, IMU and DGPS/USBL data [15]. The accuracy of this solution is approximately 1 meter (1σ) [16]. A gridded terrain map, with 10 meter grid resolution, was used. This map was constructed using measurements from a Kongsberg Maritime EM1002 MBE (100 kHz) from a surface vessel, and the map data are therefore statistically independent from the EM3000 measurements. Fig. 1 shows a contour map of the area together with the AUV trajectory. The AUV travels from an area with a relatively rough terrain to a flat area, where it moves in a lawn mower pattern, before returning to the rough area again. As all terrain navigation algorithms require some terrain variation to work, we expect poor terrain navigation performance in the flat area.

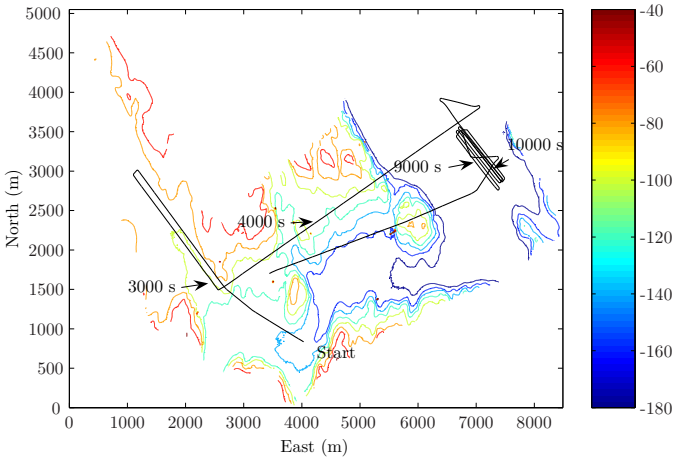

Fig. 1. Contour map and AUV trajectory. The parts of the trajectory from 3000–4000 s and 9000–10000 s are used as example areas in this paper.

A. 2-Dimensional Algorithms

We first present results from the 2D algorithms. In real-time applications the algorithms are typically restarted frequently, and an estimate is computed from a relative short series of MBE pings, so we followed this approach in our tests. Typically the algorithms were run using around 1000 seconds of measurement data each time they were restarted, corre-

sponding to terrain profiles of around 2 kilometers. However, 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 will 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 performance in the flat area was 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. As mentioned in [8], the PMF has a tendency of overconfidence. Though this is also the case to some extent for both the PMF and the PF in these computations, this effect has been successfully reduced using sub-sampling of the measurements both within and between pings, to minimize the correlations between the different MBE beams.

Fig. 2 and Fig. 3 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 $\pm$300meters 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, which turned out to be a favorable number of 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 has a slightly faster convergence to a stable solution. As can be seen in Fig. 2, 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.

A well-known problem with terrain navigation algorithms is false fixes, i.e. erroneous estimates with low covariances. However, in this run, no such false fixes were experienced 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. Both methods were very robust to errors in the initial position. In the experiments shown in Fig. 2 and Fig. 3, 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 that the terrain was suitable for terrain navigation.

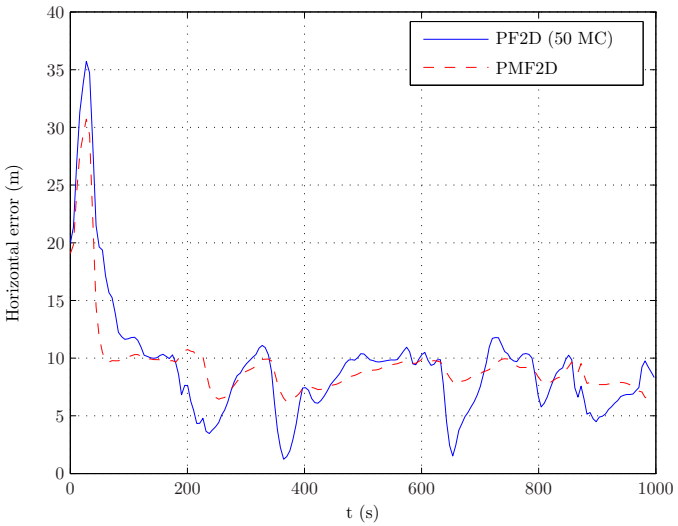

Fig. 2. Horizontal error from PMF and PF Monte Carlo mean error, 3000-4000 s from start of run (rough terrain).

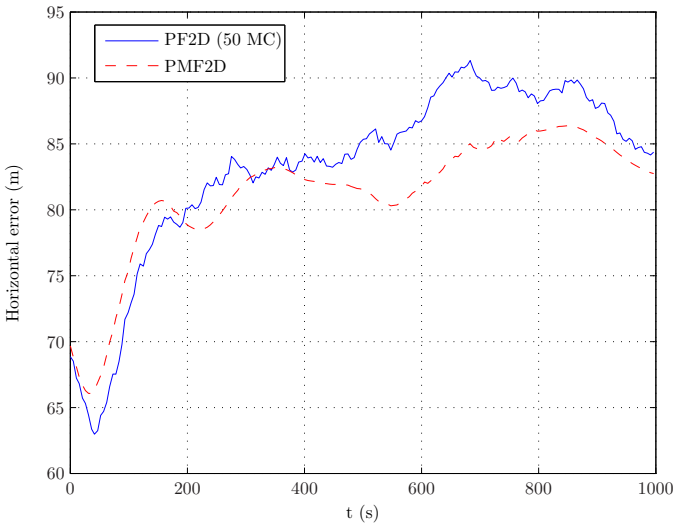

Fig. 3. Horizontal error from PMF and PF Monte Carlo mean error, 9000-10000 s from start of run (flat terrain).

B. 3-Dimensional Algorithms

Similar tests to the above, but with added depth errors, showed that the 2D methods were very sensitive to such depth errors. 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 errors of this magnitude are very realistic 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 errors are known to be relatively small, the vertical search area in the 3D methods was typically set to $\pm$10 meters. Even with no added artificial depth errors, the 3D methods were generally more accurate than the 2D methods. However,

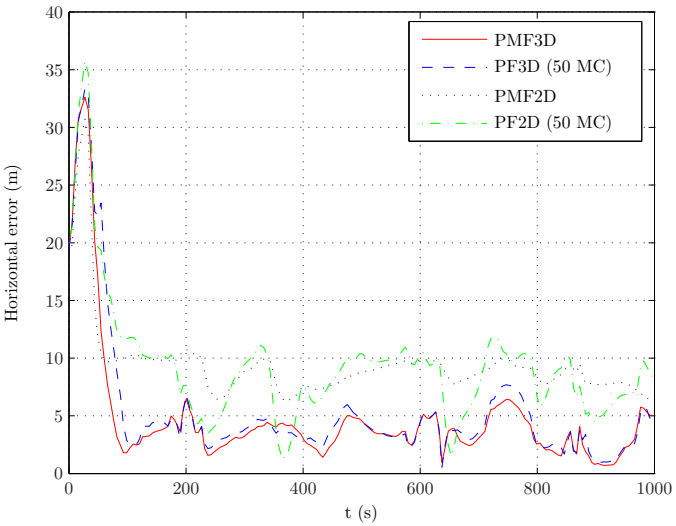

Fig. 4. Horizontal errors from 3D and 2D methods $3000-4000$ s from start of run. No added depth error.

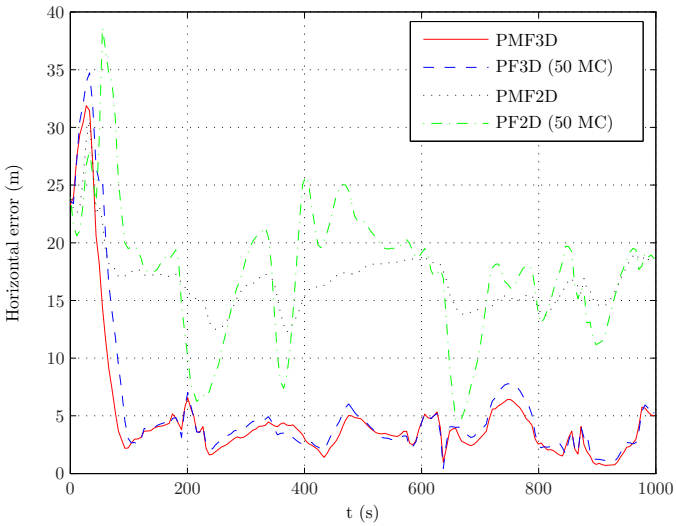

Fig. 5. Horizontal errors from 3D and 2D methods $3000-4000$ s from start of run. 1 m added depth error.

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.

Fig. 4 and Fig. 5 show typical errors for both the 2D and 3D methods in the rough area, without and with an added depth error 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 errors is clearly shown. Even with such small depth errors as 1 meter, the performance of the 2D methods are 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.

Fig. 6 and Fig. 7 show corresponding results from the flat area. Though the errors are quite high for all methods, the 3D methods are again more accurate, both with and without added depth errors. We emphasize that the covariance matrices in this case 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 our straight-forward MATLAB implementation of PMF3D, processing 1000 seconds of real data, takes around 6 hours on a 3.6 gigahertz computer with 2.0 gigabytes of RAM. This yields a real-time factor of around 20. In comparison, both the adaptive PMF2D and PF2D (with 1000 particles) typically have real-time factors of around 0.05. For PF3D, the computational demands do not increase significantly when extending the 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, in our case 1000 particles turned out to be sufficient also in the 3D case. This is probably due to the relatively small vertical search area, and 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. We emphasize 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 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 full 3D models is, as mentioned in Section II, to use relative profiles, by subtracting the mean 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. Fig. 8 and Fig. 9 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, in self-similar terrain, relative methods are known to perform poorly, since they are not able to distinguish between similar terrain at different depths [5]. This results in a high risk of convergence to erroneous position estimates in such terrain, and relative methods should therefore be used with

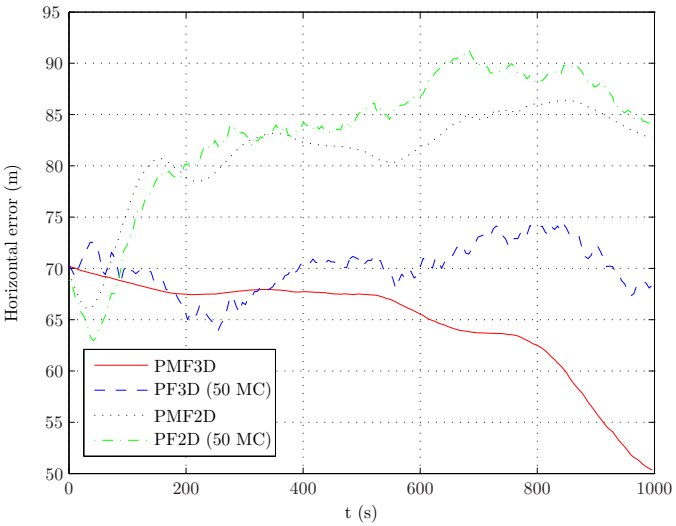

Fig. 6. Horizontal errors from 3D and 2D methods $9000 - 10000$ s from start of run. No added depth error.

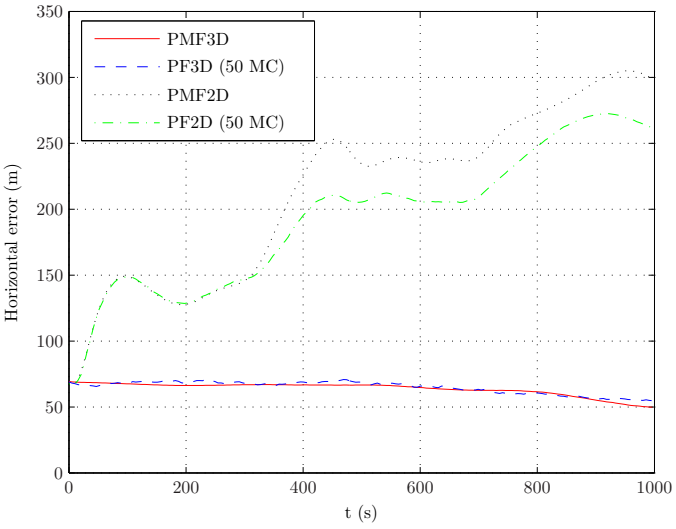

Fig. 7. Horizontal errors from 3D and 2D methods $9000 - 10000$ s from start of run. 1 m depth error.

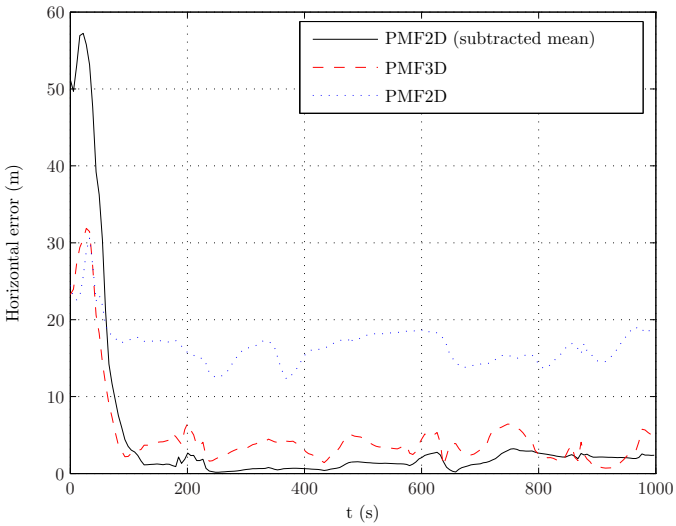

Fig. 8. Horizontal errors from PMF3D and PMF2D with and without subtracted mean, $3000 - 4000$ s from start of run. 1 m added depth error.

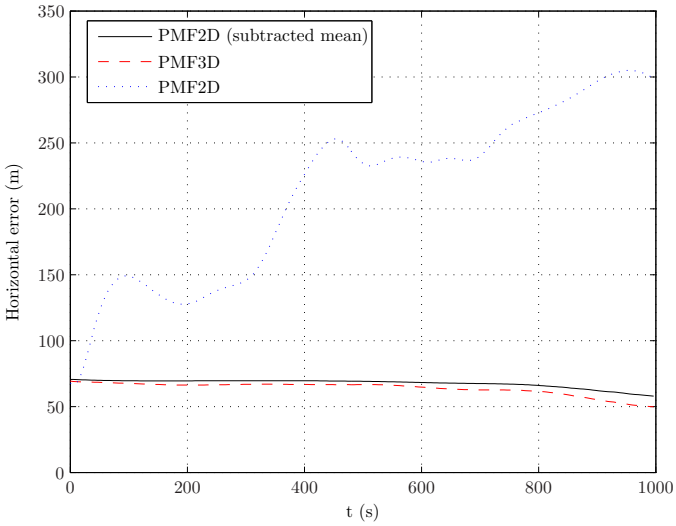

Fig. 9. Horizontal errors from PMF3D and PMF2D with and without subtracted mean, $9000 - 10000$ s from start of run. 1 m added depth error.

care. Another disadvantage with relative methods is that they do not estimate the depth bias, which in itself can be useful.

IV. Conclusions

Our tests have shown that both the point mass filter and the Bayesian bootstrap particle filter are suitable methods for underwater terrain navigation, and both methods have an estimation accurracy around the horizontal resolution of the terrain map. The PMF is in general slightly more accurate and robust than the PF, but also more computationally expensive. Both methods are able to assess the quality of their own estimates through their error covariance matrices, though a tendency of overconfidence was experienced in our test results. However, this problem has been successfully reduced by using a sub-sampling procedure on the MBE data.

To make the methods more robust to depth errors, e.g. due to erroneous tidal estimates, 3-dimensional versions of the methods can effectively be used. For the PMF the computational expense is dramatically increased when extending the methods from 2D to 3D. For the PF, however, the increase in computation time is insignificant, and the 3D PF should therefore be preferred to the 2D PF. As an alternative to 3D methods, 2D methods with relative profiles can be used. In our tests this approach yielded results comparable to those of the 3D methods. However, relative profiles are known to work poorly in self-similar terrain.

Possibilities for future work include implementation of an adaptive grid version of the 3D PMF and a more sophisticated analysis of both the process and measurement errors.

ACKNOWLEDGMENTS

This work is part of the UNaMap project, funded by the Norwegian Research Council, Kongsberg Maritime AS, and Kongsberg Defence & Aerospace AS. The authors would like to thank Are B. Willumsen and Øyvind Hegrenæs for useful comments and Ove Kent Hagen at FFI for, among other things, suggesting the use of the MBE sub-sampling procedure.

REFERENCES

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

[2] J. Golden, "Terrain contour matching (TERCOM): A cruise missile guidance aid," in *Image Processing for Missile Guidance*, T. Wiener, Ed., vol. 238. San Diego, CA: The Society of Photo-Optical Instrumentation Engineers, 1980, pp. 10–18.

[3] L. Hostetler, "Optimal terrain-aided navigation systems," in *AIAA Guidance an Control Conference*, Palo Alto, CA, 1978.

[4] N. Bergman, "Recursive bayesian estimation - navigation and tracking applications," Ph.D. dissertation, Department of Electrical Engineering, Linköping University, Sweden, 1999.

[5] M. Mandt, "Terrengreferert posisjonering av undervannsfarkoster," Norwegian Defense Research Establishment, Tech. Rep. FFI/RAPPORT-2001/05900, 2001.

[6] I. Nygren and M. Jansson, "Terrain navigation using the correlator method," in *Position Location and Navigation Symposium, PLANS 2004*, Monterey, CA, April 2004.

[7] O. Hagen and P. Hagen, "Terrain referenced integrated navigation system for underwater vehicles." in *SACLANTCEN Conference Proceedings CP-46*, E. Bovio, R. Tyce, and H. Schmidt, Eds. La Spezia, Italy: NATO SACLANT Undersea Research Centre, 2000, pp. 171–180.

[8] K. B. Ånonsen, O. Hallingstad, O. Hagen, and M. Mandt, "Terrain aided AUV navigation - a comparison of the point mass filter and terrain contour matching algorithms," in *Proceedings from UDT Europe 2005*, Amsterdam, the Netherlands, June 2005.

[9] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[10] A. Doucet, J. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer Verlag, 2001, ch. An Introduction to Sequential Monte Carlo Methods.

[11] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, 1993, pp. 107–113.

[12] B. Jalving, "Depth accuracy in seabed mapping with underwater vehicles," in *Proceedings from Oceans '99*, Seattle, WA, 1999.

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

[14] R. Bucy and K. Senne, "Digital synthesis of non-linear filters," *Automatica*, vol. 7, no. 3, pp. 315–322, May 1971.

[15] K. Gade, "Navlab, a generic simulation and post-processing tool for navigation," *European Journal of Navigation*, vol. 2, no. 4, pp. 51–59, November 2004. [Online]. Available: www.navlab.net

[16] K. Vestgård, R. Hansen, B. Jalving, and O. Pedersen, "The HUGIN 3000 survey AUV," in *Proceedings from ISOPE-2001: Eleventh International Offshore and Polar Engineering Conference*, Stavanger, Norway, June 2001, pp. 679–684.