On the use of Particle Filters for Terrain Based Navigation of sensor-limited AUVs

José Melo, Aníbal Matos
INESC TEC (formerly INESC Porto) and
Faculty of Engineering, University of Porto
Rua Roberto Frias, s/n, 4200-465 Porto
Portugal
{jose.melo, anibal}@fe.up.pt

***Abstract*—Different Terrain Based Navigation systems for underwater vehicles have already been presented, with experimentally validated results and consistent performance. However, these results are mostly based on the use of both high accuracy inertial navigation systems and high quality sonars.**

This article presents a study on Particle Filter algorithms that cope with peculiarities of Terrain Based Navigation for sensor limited systems. The focus is on the influence on several parameters, namely the process noise, the measurement noise and the number of the particles, and how these can improve the obtained results. Based on the results obtained by simulation, we present some conclusions relevant for the design of future implementation of the algorithms.

I. Introduction

The problem of localization is one of the most fundamental tasks for the navigation of mobile robotics. For outdoor ground-based mobile vehicles localization is generally addressed by using GPS-related techniques. However, in GPS denied environments, like underwater, different techniques must be derived in order not to compromise the navigation of the vehicles.

The traditional approach for the navigation systems of Autonomous Underwater Vehicles (AUVs) is the combination of both Acoustic Navigation with Dead-Reckoning. Acoustic Navigation embraces a number of techniques that rely on the computation of ranges or bearings to acoustic beacons, with known and pre-defined positions. The main drawback of Acoustic Navigation is the need for the deployment of the acoustic beacons or, in alternative, the need for a support vessel, which can be at times extremely inconvenient, or even unfeasible for some applications.

In this context, the interest for Terrain Based Navigation naturally arises. The main idea in Terrain Based Navigation (TBN) is to use information of the variability of the terrain to bound the errors of the dead-reckoning systems, thus overcoming the need of external aiding devices and, in a sense, making navigation truly autonomous. Generally speaking, TBN produces vehicle horizontal position estimates x_t by matching range measurements of the terrain against an *a priori* Digital Terrain Map (DTM). In the last decades there has been extensive research focused on TBN methods for both manned and unmanned vehicles, like missiles and jet-propulsion aircraft. For underwater environments, however, the use of this technique is quite recent, mostly due to the scarcity of DTMs of large areas.

Early correlation based terrain navigation strategies relied on the evaluation of arbitrary matching functions across the reference map, giving no statistically justified measure of uncertainty in position estimates [1]. However, recent efforts have been directed to the development of Bayesian Estimation methods to tackle this problem, by implementing one of the different realisations of the Bayesian Filter. Different Terrain Based Navigation systems for underwater vehicles have been presented, with experimentally validated results and consistent performance, most notably in [2] and [3], among others. However, these results rely on the use of both high-accuracy inertial navigation systems (INS) and high-quality sonars such as multibeam echosounders.

Building up on the results of Meduna [4], this article considers the performance Terrain Based Navigation algorithms for sensor limited systems - systems using lower-accuracy inertial systems and low-performance sonar systems - thus enabling its use on a broader range of vehicles. In particular, we are focused on the different Particle-Filter alternatives that can be used to implement TBN with a decent realtime performance. Some preliminary findings in this subject are presented that aim to illustrate the achievable performances of Bayesian Terrain Based Navigation algorithms based on Particle Filters. For this, a simulation framework was implemented responsible for simulating vehicle sensor readings, including the IMU, DVL and sonar measurements, as well as vehicle trajectory. Since it was not possible to have access to suitable bathymetric maps of an actual site, the maps under use were also artificially generated.

The remainder of this article is organized as follows. On Section II the general problem of Terrain Based Navigation is introduced, and Section III introduces the concepts of Bayesian Estimation. Then, in Section IV a brief description of the Simulation Framework is provided, and Section V describes the Particle Filter implemented. Finally, Section VI provides the preliminary simulation results obtained under the proposed methods and finally, in Section VII, we present the conclusions of this study.

II. Problem Statement

A basic state-space model for an INS-based AUV system, expressed in terms of vehicle state variables, vehicle observa-

978-1-4799-0002-2/13/$31.00 ©2013 IEEE

tions and state transitions, can usually be stated as:

$$x_{t+1} = x_t + u_t + v_t \tag{1a}$$
$$z_t = \mathcal{H}(x_t) + w_t \tag{1b}$$

Equation (1a) represents the state update model, where x_t, is the state vector, assumed to be Markovian, u_t contains the position updates as calculated from the INS and v_t represents the noise associated to the updates, usually dominated by the INS drift error. The system measurement equation is given by (1b), where z_t refers to the observation vector, $\mathcal{H}(x_t)$ is the non-linear map function, and w_t the stochastic measurement error.

$$x = [x^N \quad x^E \quad x^D \quad \phi \quad \theta \quad \psi]^T \tag{2}$$

For the course of the work here presented, the state variable x_t is supposed to be 6-dimensional, and composed by both position and orientation of the vehicle, as in (2). Additionally, it is assumed that for navigation purposes, the vehicle is equipped with both a MEMS based INS, capable of measuring both translational accelerations and angular rotation, and a Doppler Velocity Log (DVL) so that the velocity of the vehicle can directly measured. Sometimes, the bias of odometry sensors in also included the state vectors [4].

In the framework of TBN, both the measurements and the map altitudes refer to the total sea depth at the current vehicle position. The total altitudes, or terrain elevations, are usually observed with respect to the mean sea level (MSL), and computed as the sum of the AUV depth given by a pressure sensor, and the AUV altitude above the see floor, given by a sonar.

III. Bayesian Estimation

Despite the early correlation based TBN strategies, which relied solely on the performance of matching functions across the reference map, Bayesian Filters are now the primary choice for the design of TBN filters. Not only the bayesian approach allows for the fusion of information from multiple sensors, but it also provides a statistical measure of the uncertainty in position estimates produced.

TBN can be implemented using any realization of the Bayesian Filter, including both parametric filters like the Kalman Filter (KF), and non-parametric approaches, like the Particle Filter (PF). However, due to the high non-linearities $\mathcal{H}(x_t)$ present in the model of the system (1b), non-parametric filters like the Particle Filter (PF) or the Point Mass Filter (PMF) are usually preferred, due to their ability to represent strong non-linearities more accurately [5]. Moreover, these filters do not require for the matching function to be Gaussian shaped, making them more appropriate in situations where only a few terrain height measurements are available.

Under first-order Markov assumptions, the recursive form of the Bayes Filter is given by equations (3) and (4). $p(x_k|z_{k\text{-}1}, u_k)$ represents the prior, $p(x_k|z_k, u_k)$ is the posterior, and α is just a normalizing factor, so that the integral sums up to one. u_t explicitly represents the dependency of the system state to the changes in vehicle state measured by odometry sensors. Furthermore, previous knowledge of the model is incorporated by the term $p(x_k|z_{k\text{-}1}, u_k)$, and the likelihood, $p(z_k|x_k)$, represents the measurement model. Calculation or approximation of these terms are the essences of the Bayesian filtering and inference.

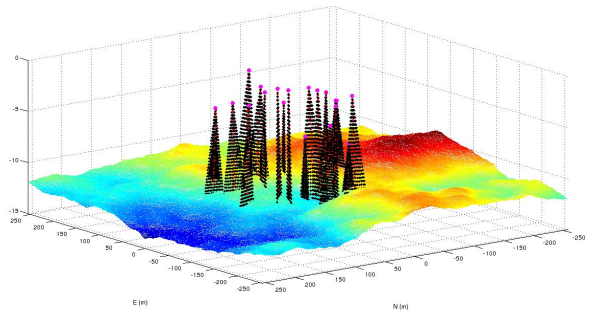

Fig. 1: Simulation of a Particle Filter for Terrain Based Navigation algorithm.

$$p(x_k|z_{k\text{-}1}, u_k) = \int p(x_k|x_{k\text{-}1}, u_k)p(x_{k\text{-}1}|z_{k\text{-}1}, u_{k\text{-}1})dx_{k\text{-}1} \tag{3}$$
$$p(x_k|z_k, u_k) = \alpha p(z_k|x_k)p(x_k|z_{k\text{-}1}, u_k) \tag{4}$$

The Particle Filter (PF) is a numerical approximation to the Bayesian Filter for non-linear and non-Gaussian systems that uses a large number of hypothetical samples of the state vector to estimate its probability distribution. It can be demonstrated that, under certain conditions, the posterior density can be approximated as in (5) [6]. N_s is the total number of samples, and w_k^i is the normalized weight associated with particle i at time instant k. These weights are obtained using the measurement model, $p(z_k|x_k)$.

$$p(x_k|z_k) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{5}$$

Resampling is a crucial step in PF algorithms, as it is the process on which low probability particles are replaced by others with higher probability, thus increasing the concentration of particles in regions of the state-space with higher likelihood. This study starting-point is the Sampling Importance Re-Sampling (SIR) filter, sometimes also referred to as Bayesian Bootstrap Filter. Although one of the most common and simple, it has already been demonstrated in [4] that this type of PF algorithm is able to cope with all the constraints existent in TBN for sensor-limited underwater vehicles with satisfying results.

IV. Simulation Framework

The framework developed for this initial study was implemented in Matlab, as this is an excellent tool to analyse the data. However, this is not compatible with a future implementation on the on-board computational unit of the AUVs. For the course of this work some assumptions were made concerning the vehicle and the sensors in use, always mimicking actual devices.

Inspired by the vehicle MARES [7], we assume a vehicle able to control horizontal and vertical motion independently, and able to hover and maintain a constant depth. Moreover, we assume only surge velocity and neglect sway velocity. As for the navigation sensors, the assumptions are that the vehicle is equipped with a 3-axis IMU, a pressure depth sensor and a Doppler Velocity Logger (DVL). By mimicking existing sensor available on the market, we consider that the DVL is able to provide direct vehicle velocity measurements in all the 3 axis of the vehicle, and the depth of the vehicle is given by the depth sensor. Inspired by sensors like the Xsens MTI or the PNI Trax, it was considered that the IMU is able to computed the orientation of the vehicle. In fact, the aforementioned sensors use advanced proprietary sensor fusion algorithms that use the Earth's magnetic field to stabilize Heading, but also keep track of the biases affecting the different sensors, thereof bounding the errors in a dramatic way.

This vehicle configuration is similar to the proposed in several publications concerning TBN studies [4], [5] and the set of sensors considered is, in a way, the standard for entry-level modern commercial AUVs. These class of AUVs, usually referred to as sensor-limited, and are characterized by the poor performance of the navigation system, mainly as a result of low-accuracy inertial sensors and simple low-information range sensors.

The generation of suitable trajectories for the vehicles is a 3-stepped approach. First the navigation sensor measurements, from both the IMU and the DVL are artificially generated. Using the orientation estimates of the IMU, the the DVL output values are then transformed from the sensor frame of reference to the vehicle frame of reference; Finally the position of the vehicle is obtained by suitable integration of the velocity of the vehicle.

The trajectory obtained in this way is going to be the ground truth for all the further calculations. However, in the real world it is obvious that noise always affects sensor readings in different ways. In this way, additive noise will also be introduced to the simulated sensor measurements in a convenient way. In specific, the noise of DVL velocities v_x^{DVL} and v_y^{DVL}, and the heading ψ were affected by first-order Markov Processes, while to the remaining sensor measurements were affected by Gaussian white noise.

A. Process Model

Considering the state-space model in (2), for this initial assessment of the performance of the TBN algorithms, the state vector will be $x_t = [x_k^N \ x_k^E \ x_k^D \ \phi_k \ \theta_k \ \psi_k]$. The increase of dimension of the state space, when comparing to the more basic algorithms, is justified by the use of less accurate sensors, that lead to the need of estimating not only the position, but also the orientation of the vehicle. Inspired by the results presented by Meduna in [4], the resulting process model is then given by (6). Comparing both approaches, one can notice that here the biases of gyroscopes are not included in the state vector, as the IMU used already applies proprietary sensor fusion algorithms responsible for tracking the biases of the individual gyroscopes, accelerometers and magnetometers, and subtract its effects to the output orientation values.

$$x_{k+1} = x_k + \left\{ \begin{array}{c} v_k^N \mathrm{d}t \\ v_k^E \mathrm{d}t \\ \Delta z_k \\ \Delta \phi_k \\ \Delta \theta_k \\ \Delta \psi_k \end{array} \right\} + r_k \tag{6}$$

In (6), the control elements Δz_k is given by the depth pressure sensor, and $\Delta\phi_k$, $\Delta\theta_k$ and $\Delta\psi_k$ are values provides by the IMU. The vehicle velocities along the North and East components of the frame of reference, v_k^N and v_k^E respectively, are computed by using the local orientation matrix $R(\phi(t), \theta(t), \psi(t))$ to transform the DVL velocities to the appropriate frame of reference of the vehicle. As for the process noise, r_k is assumed that is normally distributed with zero mean and standard deviation given by Σ (7).

$$\Sigma = \begin{bmatrix} \sigma_{v_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{v_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\phi^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\psi^2 \end{bmatrix} \tag{7}$$

B. Observation Model

Observation Models for Terrain Based Navigation are usually one of two kinds: ray-tracing based or projection based models. Ray tracing models are known to be more accurate but they are computationally very expensive and, because of that, not adequate for online processing. For this reason, the projection based models are preferred.

As the name might suggest, projection-based observation models consist on projecting the measured ranges r given by a sonar, into the 3-dimensional space and according to the vehicle orientation. From this three-dimensional position it is possible to extract a projected depth or terrain height, y, which is then compared with the expected terrain height for that specific location, $\hat{h}(x_N, x_E)$, as given by a digital map of the location. The observation model used on the remaining of this work is given by (8). As the gridded map only contains elevations of the terrain at specific points, $h(x^N, x^E)$ implements a billinear interpolation of the 4 nearest neighbours and includes a term to account for the uncertainties introduced by the terrain model and the interpolation process.

$$p(z_k|x_k) = \alpha \ exp\left[-\frac{1}{2}\sum_{i=1}^{M} \beta_i (y_i - \hat{h}_i)^2\right] \tag{8}$$

where α is a weighting factor, and β is a term to account for the effect of modelling errors that during map registration, and errors present in the projection model. Similar measurements models have also commonly used in related areas [8], [9]. A more detailed derivation of this model can be found in [10].

V. PARTICLE FILTER

There has been a growing interest on the use of non-parametric filters for Terrain Based Navigation in underwater vehicles. The interest on these filters, in particular on Particle Filters and Point Mass Filters arises due to their ability to better track strong non-linearities, as the ones present on TBN problems, when compared to traditional linear filters. Both PFs and PMFs have been successfully implemented for underwater Terrain Based Navigation but it has been show that while the PMF is more robust and accurate than the PF, the latter allows higher dimensional search without the computational expense of the PMF [3]. Because of this, Particle Filters have been more extensively used, specially when dealing with state-space models with higher dimensions. In this section, we provide some insights on the theoretical background of the Particle Filters used, but a thorough derivation can be found in [11], [12], [13].

The Particle Filter is an approximation to the recursive Bayes Filters. In specific, the objective is to use an approximate form to solve for the integrals of the general Bayesian Filter recursion, (3) and (4). The idea is to approximate these equations by sampling a set of randomly chosen state particles $\mathcal{X}_t = \{x^{[1]}, x^{[2]}, \ldots, x^{[M]}\}$ from an appropriate distribution. If this is the case, then the weighted set of of particles is enough to approximate the system state, and a solution to the to the Minimum Mean Square estimate of the state can be expressed according to (9).

$$\hat{x}_k = \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{9}$$

Such a representation is approximate, but it is non-parametric, and therefore can represent a much broader space of distributions than, for example, Gaussians [14]. Moreover, as the number of sampled particles N_s increases this approximation is guaranteed to converge to the true solution. In (9) δ is the Dirac impulse function, and w_k^i is the weight associated to each particle such that $\sum_{i=1}^{N_s} w_k^i = 1$.

A. Resampling

A key aspect on Particle Filters is the resampling step. The Resample step is in fact crucial in the Particle Filter algorithm, as it allows replace particles with low importance weights $w_k^{[m]}$, or low-probability samples, to be substituted by particles with higher importance weights. In most resampling strategies the number of samples is maintained constant. An interpretation of this, according to Thrun [14], is that the resampling step is a probabilistic implementation of the Darwinian idea of the survival of the fittest. There are different resampling approaches and there is a lot of literature devoted to the study of efficient resampling strategies. Different resampling strategies have been proposed, like Multinomial resampling, Stratified resampling, Systematic resampling among others.

In what follows we will use the Minimum Variance Sample, or Systematic Resampling. This algorithm presents a very low complexity, when compared to others, and it was also considered to be the one providing the most favourable sampling scheme for Terrain Navigation applications [15]. This algorithm uses a single random number to sample from and has a minimal complexity, making it the preferential resampling algorithm for numerous applications. The Minimum Variance Sample is detailed on Algorithm 1.

Algorithm 1 Minimum Variance Sample algorithm

function LOWVARIANCERESAMPLE($\mathcal{X}_t$, $\mathcal{W}_t$)
  $\overline{\mathcal{X}_t} = \emptyset$
  $r = rand(0; N_s^{-1})$
  $c = w_t^{[1]}$
  $i = 1$
  for $m = 1$ to N_s **do**
    $U = r + (m-1).N_s^{-1}$
    while $U > c$ **do**
      $i = i + 1$
      $c = c + w_t^{[i]}$
    end while
    add $x_t^{[i]}$ to $\overline{\mathcal{X}_t}$
  end for
  return $\overline{\mathcal{X}_t}$
end function

B. SIR Filter

The particle filter formulation used to support this initial study on Terrain Based Navigation is a Sampling Importance Resampling (SIR) filter, also known as a Bayesian bootstrap filter or condensation algorithm. This formulation, which is one of the simplest ones, uses both the process model, $p(x_{k+1}|x_k)$, and the observation model, $p(z_k|x_k)$, derived in previous sections, to obtain an approximate solution for the Bayesian recursion. Algorithm 2 details this formulation.

For every time step, the algorithm starts with a prediction or time uptade step, where new predictions are generated according to the proposal distribution, which in this case is the process model. Then there is a measurement update step, on which new weights for every particle are evaluated according to the observation model. After weights normalization, the resample step takes place if the effective number of samples, and indicator of the degree of depletion, is below a given threshold. As there is usually no prior knowledge on the distribution of the particles once the algorithm is initiated, $p(x_0)$, it is a common practice to sample new particles from an uniform distribution covering the whole space. For the SIR filter, resampling is done whenever the effective sample size, N_{eff} goes below a given threshold, usually between $0.5N_s$ and $0.8N_s$.

C. Regularized Particle Filter

Resampling is a very effective method to avoid the degeneracy of the particle filter set, or sample depletion, which happens whenever all but one of the importance weights are close to zero. However, resampling is also known to introduce the problem of loss of diversity among the particles, which in some cases can lead to "particle collapse", when all particles are just a copy of each other. This phenomenon is particularly present when the noise of the dynamical system is low.

One obvious solution for stopping sample impoverishment is to add some random-noise to each particle before propagating it onto the next time step, a simple strategy proposed

Algorithm 2 The SIR Particle Filter algorithm

Initialization
for all m **do**
  $x_0^{[m]} \approx p(x_0)$
end for
for all k **do**
  Prediction:
  for all m **do**
    $x_k^{[m]} = p(x_{k+1}|x_k^{[m]}, r_k)$
  end for
  Measurement Update:
  for all m **do**
    $w_k^{[m]} = p(z_k|x_k^{[m]})w^{[m]}$
  end for
  Weight Normalization:
  for all m **do**
    $w_k^{[m]} = \frac{w_k^{[m]}}{\sum_{j=1}^{N_s} w_k^{[j]}}$
  end for
  Resample:
  $N_{eff} = \frac{1}{\sum_{m=1}^{N_s}(w_k^{[m]})^2}$
  if $N_{eff} < \alpha N s$ **then**
    RESAMPLE
  end if
end for

in [16] and known as jittering. However, and because the variance of the random noise introduced is freely determined by the user, this becomes a very *ad-hoc* solution. A similar, but slightly elaborate technique was introduced in [17], by proposing the Regularized Particle Filter (RPF).

The problem of loss of diversity arises due to the fact that in the resampling stage, samples are drawn from a discrete distribution rather than a continuous one [13]. By sampling new particles from a continuous approximation of the posterior density $p(x_k|z_k)$, the RPF introduces the necessary diversity of the particle set. The continuous approximation of $p(x_k|z_k)$ is obtained according to (10), where $K(.)$ is a multivariate kernel estimator.

$$p(x_k|z_k) \approx \sum_{i=1}^{N_s} w_k^i K(x_k - x_k^i) \tag{10}$$

Under certain conditions, when the all particles are equally weighted, it can be shown that the Epanechnikov kernel, in (11), is the optimal choice with respect to the mean integrated square error of the density estimation [17]. For the general case, these will lead to a suboptimal filter, but still valid. In (11) c_d refers to the volume of the unit sphere with dimension d.

$$K(x) = \begin{cases} (2c_d)^{-1}(d+2)(1-x'x), & \text{if } x'x < 0. \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Using a kernel estimator to approximate the posterior requires minimal changes to Algorithm 2. According to what was stated above, the RPF only needs an additional step so that after resampling, some extra noise is added to the samples, according to (12), where H is the smooth parameter, or bandwidth, D_k is the Cholesky factorization of the covariance matrix of the particles before resampling, and ϵ is generated from the Epanechnikov Kernel.

$$x_{k+1}^i = x_{k+1}^i + HD_k\epsilon \tag{12}$$

Generating from the Epanechnikov Kernel means consists on generating from $\sqrt{d+4}\sqrt{\beta(d/2,2)}T_d$, where β follows a beta distribution with parameters and T_d is uniformly distributed over the unit sphere of $\mathcal{R}^d$ [18]. H is computed according a generalization of the Scott's rule of thumb for general multivariate distributions, $H = n^{\frac{-1}{d+4}}$ [19].

VI. SIMULATION RESULTS

Even though several authors presented valid contributions in the field of Terrain Based Navigation, there hasn't been much working concerning the tuning of the filters. The purpose of the simulation results presented in this section is to understand how the variation of some parameters of the Particle Filter influences the final result in terms of the accuracy of the estimation. The number of particles, the process noise and the sensor noise are obviously the parameters deserving more attention. Additionally, we also study the influence of sampling from a continuous approximation of the posterior density, by using a Regularized Particle Filter.

The performance of the algorithms in terms of processing time is of uttermost importance, as we are interested on running the filter in real-time. Therefore, the first limitation imposed to our system will be on the time it takes for every iteration of the filter to process. This constitutes and indirect upper bound on the computational complexity involved. For this reason, we are primarily interested in the simpler versions of the Particle Filter, like the previously describer SIR Filter, or the Regularized Particle Filter.

The processing power available for the onboard computational systems of AUVs, like the MARES, is likely to be lower than what was used on the simulations. On the other hand the simulation framework was build on MATLAB, which adds extra complexity to the whole system. Consequently it is assumed that results obtained in simulation are comparable what could be achieved in a mission scenario. Additionally, for the subsequent simulations, it is assumed that the initial pose of the vehicle is known with some accuracy, and we are only interested on the tracking of the vehicle.

A. Number of Particles

The number of particles of a Particle Filter has always been a design choice critical to the overall performance and convergence of the filter. It is empirically known that a large number of particles usually means better accuracy of the filter, and this is due to a more efficient covering of the whole sample space. This is specially true when there is no prior information of the position of the vehicle and global localization is needed. However, when the filter is only tracking the vehicle, as in the case here considered, the number of particles needs decreases dramatically. By using the Kullback-Leibler metric, Fox demonstrated in [20], that filters with a small number of samples track a vehicle in a very satisfactory way.

In our simulations we incremented the number of particles from 250 to up to 6000 and compared the errors in position to the true position. The upper bound on the number of particles, 6000, corresponds the the maximum number of particles able to be processed between the arrival of 2 measurements of the DVL, that we considered to occur every 5 seconds. The lower bound, on the other hand, corresponds to the minimum number of particles needed to have a consistent convergence of the filter to the true position.

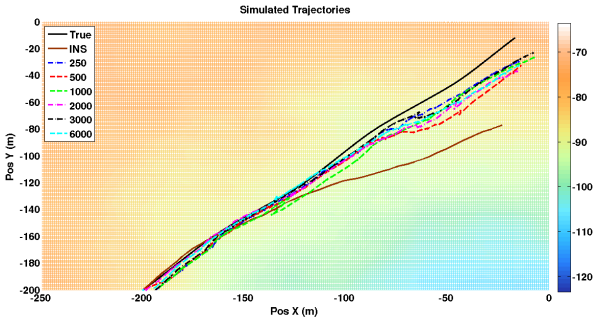

Fig. 2: Simulated Trajectories with different numbers of particles. The filter clearly improves position estimates even when the number of particles is small.

In Figure 2 the trajectories for several simulations of a SIR PF are depicted, where only the number of particles was varied. It is clear that the filter provides better position estimates that traditional INS-based solutions. However, the filter is unable to closely track the true position of the vehicle.

The results obtained in terms RMS of distance to the true trajectory are depicted on Figure 3. As expected, the more accurate tracking of the true position happens when the filter is using an higher number of particles. However, comparing the final position of all the simulations, and the average RMS distance to the true trajectory, it is possible to verify that increasing the number of particles is not necessarily reflected on the performance of the filter. By comparing the average RMS of the error position between the filter using 3000 particles, and the filter using 6000 particles, the results are fairly similar. Considering that by choosing 3000 particles we are saving half of the processing power, we consider this to be the best option. In this specific situation, trading processing power for an higher number of particles is not a particularly efficient trade-off, as the increase in accuracy doesn't pay off. Consequently, and for the subsequent simulations, we set the number of particles to 3000.

As it can be seen on Figure 4, the states concerning the position of the vehicle, x^N, x^E and x^D are considered to be satisfactorily estimated; it can be seen the filter is able to overcome the effect of noisy control inputs. However, the same doesn't happen for the roll, pitch and yaw of the vehicle, and the estimated values for these states present a non negligible deviation from the ground truth. This is specially noticeable for the heading of the vehicle, and this is likely to be the cause for growing position errors visible at the final part of the simulation.

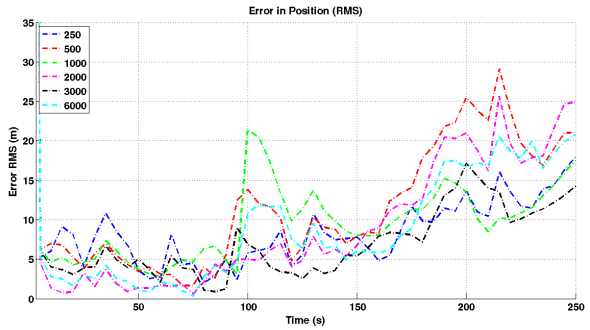

Fig. 3: RMS Error in Position when comparing with the true position of the vehicle, for simulations with different number of particles.

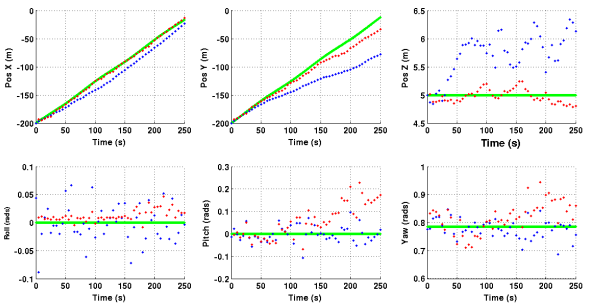

Fig. 4: Evolution of the different state variables along the time. The true simulated state is depicted on green, traditional INS estimates are on blue, and the particle filter estimates are in red.

B. *Process Noise and Measurement Noise*

The process noise and measurement noise are of paramount importance in the particle filter. In fact, if the process noise is small, or in extreme cases zero, the particle set will rapidly collapse [21]; on the other hand, it is also known that increasing the noise levels, so that so that they appear larger than are in reality, is a common trick to improve the performance. Increasing the noise level in the process model is supposed to increases the support of the sampled particles, while increasing the noise level in the observation model implies that the particles whose weights are smaller are more likely to be resampled [22]. To study the influence of both process and measurement noise on the performance, we performed various simulations while only changing this two sources of noise.

Increasing the measurement noise causes the particles with low weights to take more time to be replaced with newer particles. What this means is that when tracking a vehicle, the filter will have a slower response when trying to compensate for disturbances. From our experience when varying the measurement noise, we observed only small variations of the estimated variables, but this behaviour is likely to be caused by the inherent stochastic nature of the filter. Furthermore, we prefer to keep the measurement noise to levels similar to the ones introduced by the actual sensor, and then fine-tune the performance of the filter by changing on the β_i

parameter that is present in (8). While β_i is identified with the covariance of the errors of the map and the errors of measurement, it's value will also considerably affect $p(z_k|x_k)$. Moreover, we are confident that using β_i to weight more the ranges corresponding to more central beams of the sonar, and at the same time weighting less the more distant beams will yield improved results.

Our simulations confirmed that the level of process noise needs to be high so the tracking is successful, and this is depicted in Figure 5. This is particularly important if the noise that corrupts the sensor readings is significant. During our simulations we realized that the filter could successfully track the position of a vehicle if the disturbances present were small. However, when disturbing the sensor readings with random-walk like noise, then the process noise needs to be at higher levels. For our simulations, we concluded having the affecting the estimated states x_k^E and x_k^D with noise with magnitudes comparable to the corresponding inputs provides good results. High levels of process noise however, cause the cloud of propagated particles to widespread, which can have negative results; if the particles spread over a too wide area in the space, then too many particles will have, so care should be taken to avoid this situation. Maybe due to this, when the process noise is high, the estimated trajectory tends to be much less smooth as the estimated position tends to bounce around the true position.

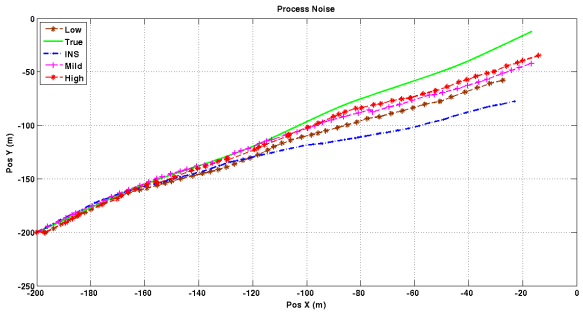

Fig. 5: Effect of diferent levels of process noise in the filter, comparing with true trajectory and traditional INS-estimated trajectory

When comparing a traditional INS based navigation with a Particle Filter approach here proposed, it can be concluded that the latter produces much more accurate results in terms of tracking a position of a vehicle with sensor limited sensors. As can be seen in Figure 6, the error in the final position when calculated with traditional INS integration methods was of around 65 meters, and when using a Particle Filter this error decreased to almost 20 meters.

At this point a closer look to the estimated variables concerning the attitude of the vehicle, namely roll, pitch and yaw is needed. While we are satisfied with the estimated state variables concerning the roll and pitch movements of the vehicle, the same doesn't happen with the heading estimate, and it most likely happens because yaw is more subject to noise than the other two. This inability to accurately track the heading of the vehicle happened throughout the simulations, even though a higher process noise seems to improve the results. If one is only interested in position corrections, the Particle Filter approach here proposed can be considered satisfying. However, if there is a need for an accurate estimate of the heading of the vehicle, then other methods must be sought.

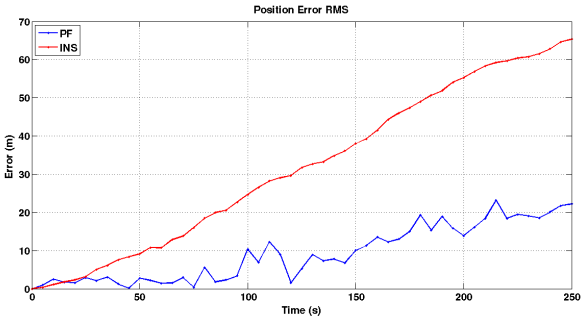

Fig. 6: RMS Error in position along the time: comparison between Particle Filter and traditional INS integration.

C. Regularized Particle Filter

As stated before, high levels of process noise are desirable, but if the process noise is too high this can cause some sort of degeneracy. To overcome this issue, we tried applying a Regularized Particle Filter to the TBN problem. The idea behind it was to be able to be able reduce the process noise while maintaining a satisfactory tracking performance. We also intended to verify if the resampling strategy of the RPF could help on improving the tracking accuracy of the heading of the vehicle, ψ.

Despite the popularity of this kind of Particle Filter, the only improvements in the simulations we performed were obtained only when the number of particles was small, but we were unable to determine if this was caused by the intrinsically stochastic nature of the process, or by the regularization factor introduced. When increasing the number of particles to a number allows for a close tracking of the position of the vehicle, the benefits of using such filter were negligible. This is presumably caused by the levels of noise present in the system, and required to be high as stated in previously.

VII. Conclusions

In this article we discussed how the different parameters of a particle filter can influence the result of a Terrain Based Navigation algorithm. We started with a SIR Particle Filter, and independently varied the number of particles, the process noise and the measurement noise in order to understand the influence of this parameters on the filter.

Increasing the number of particles improves the overall performance of the filter, but only up to some point. Our simulations suggest that there is no interest on having a very large set of particles, as this is not translated into better estimates. There is a trade-off between available processing power and size of the particle set and, under the conditions of our study, we found that increasing the number of particles above 3000 doesn't improve the obtained results. The measurement noise was one of the parameters under study, but in this case we

couldn't observe any significant improvement even by varying it over a wide range of values. On the other hand, we concluded that some of the terms of $p(z_k|x_k)$ can have an important role on the fine-tune of the filter, specially when the process noise is high. The process noise was in fact the parameter that changed the behaviour of the filter in a more dramatic way. If the process noise is small enough, the filter is dominated by the noisy inputs, but if its magnitude is high enough, than the filter is able to compensate this deviations.

While the particle filters under study were very effective on tracking the true position of a vehicle when subject to non-linear behaviours, tracking the heading of the vehicle was not so successful. If a complete solution for the navigation problem is required, including the full pose of the vehicle, than different strategies must be found. Considering this, a natural extension to this work is the use of a Rao-Blackwellized particle filter applied to Terrain Based Navigation for underwater sensor-limited AUVs.

ACKNOWLEDGMENT

This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP-01-0124-FEDER-022701».

The first author was supported by the Portuguese Foundation for Science and Technology through the Ph.D. grant SFRH/BD/70727/2010.

REFERENCES

[1] C. Morice, S. Veres, and S. McPhail, "Terrain referencing for autonomous navigation of underwater vehicles," in *OCEANS 2009-EUROPE*. IEEE, May 2009, pp. 1–7.

[2] I. Nygren and M. Jansson, "Terrain Navigation for Underwater Vehicles Using the Correlator Method," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 3, pp. 906–915, Jul. 2004.

[3] K. Anonsen and O. Hallingstad, "Terrain Aided Underwater Navigation Using Point Mass and Particle Filters," in *2006 IEEE/ION Position, Location, And Navigation Symposium*. IEEE, 2006, pp. 1027–1035.

[4] D. K. Meduna, S. M. Rock, and R. S. McEwen, "Closed-loop terrain relative navigation for AUVs with non-inertial grade navigation sensors," in *2010 IEEE/OES Autonomous Underwater Vehicles*. IEEE, Sep. 2010, pp. 1–8.

[5] K. B. Anonsen and O. K. Hagen, "Recent developments in the HUGIN AUV terrain navigation system," in *OCEANS 2011*, 2011, pp. 1–7.

[6] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[7] N. Cruz and A. Matos, "The mares auv, a modular autonomous robot for environment sampling," in *OCEANS 2008*, 2008, pp. 1–6.

[8] S. Barkby, S. Williams, O. Pizarro, and M. Jakuba, "An efficient approach to bathymetric SLAM," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2009, pp. 219–224.

[9] N. Fairfield, D. Wettergreen, and G. Kantor, "Segmented SLAM in three-dimensional environments," *Journal of Field Robotics*, vol. 27, no. 1, pp. 85–103, Jan. 2010.

[10] D. K. Meduna, "Terrain Relative Navigation for Sensor-Limited Systems with Application to Underwater Vehicles," PhD, Standord University, 2011.

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

[12] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, Jul. 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1008935410038

[13] 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.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005.

[15] J. Hol, T. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, 2006, pp. 79–82.

[16] P. Fearnhead, "Sequential monte carlo methods in filter theory," Ph.D. dissertation, Merton College, University of Oxford, 1998.

[17] C. Musso, N. Oudjane, and F. Gland, "Improving regularised particle filters," in *Sequential Monte Carlo Methods in Practice*, ser. Statistics for Engineering and Information Science, A. Doucet, N. Freitas, and N. Gordon, Eds. Springer New York, 2001, pp. 247–271.

[18] L. Devroye and L. Györfi, *Nonparametric density estimation: the L1 view*, ser. Wiley series in probability and mathematical statistics. Wiley, 1985. [Online]. Available: http://books.google.pt/books?id=ZVALbrjGpCoC

[19] I. Ahamada, E. Flachaire, and A. Clark, *Non-Parametric Econometrics*, ser. Practical Econometrics. OUP Oxford, 2010. [Online]. Available: http://books.google.pt/books?id=WW2Q7nXC3iMC

[20] D. Fox, "Kld-sampling: Adaptive particle filters," in *In Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 713–720.

[21] D. Salmond and N. Gordon, "An introduction to particle filters," September 2005, incomplete Draft.

[22] F. Gustafsson, "Particle filter theory and practice with positioning applications," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 25, no. 7, pp. 53–82, 2010.