An Efficient Approach to Bathymetric SLAM

Stephen Barkby, Stefan Williams, Oscar Pizarro, Michael Jakuba
ARC Center of Excellence for Autonomous Systems
School of Aerospace Mechanical and Mechatronic Engineering
University of Sydney, Sydney, NSW, 2006, Australia
s.barkby@acfr.usyd.edu.au

***Abstract*— In this paper we propose an approach to SLAM suitable for bathymetric mapping by an Autonomous Underwater Vehicle (AUV). AUVs typically do not have access to GPS while underway and the survey areas of interest are unlikely to contain features that can easily be identified and tracked using bathymetric sonar. We demonstrate how the uncertainty in the vehicle state can be modeled using a particle filter and an Extended Kalman Filter (EKF), where each particle maintains a 2D depth map to model the seafloor. Efficient methods for maintaining and resampling the joint maps and particles using Distributed Particle Mapping are then described. Our algorithm was tested using field data collected by an AUV equipped with multibeam sonar. The results achieved by Bathymetric distributed Particle SLAM (BPSLAM) demonstrate how observations of the seafloor structure improve the estimated trajectory and resulting map when compared to dead reckoning fused with USBL observations, the best navigation solution during the trials. Furthermore, the computational run time to deliver these results falls well below the total mission time, providing the potential for the algorithm to be implemented in real time.**

I. Introduction

Bathymetric maps have proven to be an invaluable resource in marine applications such as seafloor monitoring, pipeline surveys, marine habitat monitoring and salvage missions [1]. Utilizing a multibeam depth profiler provides the high resolution, high coverage bathymetry needed to map out large areas of seafloor quickly and accurately, assuming an accurate navigation solution is available [11]. On survey ships and Autonomous Underwater Vehicles (AUVs) this is most commonly achieved using a high precision Inertial Navigation System (INS) coupled with an Attitude Heading Reference System (AHRS), though this is often an expensive option. While AUVs cannot receive GPS observations of their location underwater, an Ultra Short Baseline (USBL) acoustic positioning system can be used to yield range and bearing measurements between the AUV and a support vessel. By using the GPS/Inertial on the ship these observation can be georeferenced to provide an observation of the vehicle's position while it is underway. However, such a setup provides position estimates less accurate than an equivalent GPS fix and requires the support ship to actively maintain the AUV within the USBL's range for the duration of the mission. This problem is further compounded in deep water deployments where the distances between the ship and the vehicle are large.

An alternative is to use the map being built to improve the navigation solution by performing Simultaneous Localization And Mapping (SLAM). This is commonly undertaken using feature based techniques that rely on identifying distinct landmarks whose uncertainty in position can be modeled and parameterized accurately [12]. Unfortunately the unstructured nature of the underwater environment does not lend itself well to feature-based modeling techniques. Features, such as peaks and troughs in the seabed, are difficult to identify and model reliably and are typically of low spatial density.

With this in mind we present an algorithm that performs bathymetric SLAM efficiently in an open underwater environment without needing to explicitly identify features in the seabed. Instead a more general gridded map representation is implemented. The algorithm is capable of running in real time and at high altitude, lending itself well to large scale mapping efforts undertaken by AUVs where high precision localization through GPS is unavailable.

The remainder of this paper is organized as follows. Section II begins by outlining related work. Section III then details the Bathymetric distributed Particle SLAM (BPSLAM) algorithm. Section IV goes on to present the results of applying our algorithm to a real bathymetric survey. Finally, Section V summarizes our findings.

II. Related Work

Bathymetric maps are traditionally built using a gridded or point cloud model of the seafloor with a deterministic model of the vehicle pose estimate [1]. In this case it is sufficient to generate the map by estimating the depth at any given location with the mean of the depth observations there.

One approach to bathymetric SLAM that uses a point cloud model divides the temporal sequence of bathymetry into submaps, assumed to be error free [9]. Pairwise matching of overlapping submaps further constrains the vehicle trajectory and submap origins using a delayed state Kalman filter. While this technique is effective, it is incapable of addressing errors within individual submaps which limits the overall accuracy of the final map. In addition, the tradeoffs in complexity, accuracy and matching performance based on the size and number of submaps have not been fully explored.

The approach utilized in this paper is a gridded model that maintains a single estimate of seabed depth in each grid cell, along with an uncertainty. While this enforces a maximum resolution on the map it also allows it to be

quickly updated with observations of varying uncertainty. Provided that observations of seabed depth can be accurately modeled as Gaussian, each estimate and uncertainty can be quickly updated using a single state Extended Information Filter (EIF) whenever a grid cell is given a new observation.

To account for uncertainty in navigation we implement a particle filter. Particle resampling can then be performed based on the self-consistency of each particle's map, reducing the uncertainty in the trajectory upon re-observing previously explored terrain. One of the disadvantages of this approach is that it requires entire maps to be copied and destroyed each time a particle is resampled, therefore becoming computationally expensive for large numbers of particles. Fortunately a new map representation called Distributed Particle Mapping [3] addresses this issue by efficiently maintaining a joint distribution over maps and robot poses using a single grid-like structure. The work of Eliazar et al. [4] further describes techniques that can reduce the asymptotic complexity of Distributed Particle SLAM (DPSLAM) to constant/linear (amortized) time per iteration of the filter, keeping it an efficient means of performing non-feature based SLAM. Furthermore, to avoid the inefficiency associated with sampling in high dimensions, states that possess a linear substructure, subject to gaussian noise, can be marginalized out and tracked instead with an EKF, a technique referred to as Rao-Blackwellization [7].

Distributed Particle Mapping has already been adapted to the problem of performing 3D SLAM in underwater tunnels in real time [5]. Here 3D occupancy grids are used as the map representation, efficiently managed with Deferred Reference Counting Octrees. This method has proven to be successful in generating a consistent 3D bathymetric map in a closed cave environment, where the benefit of continuous localization via measurement of the AUV's proximity with the surrounding cave walls is fully utilized. However, in an open marine environment opportunities to localize are often far more sparse. Furthermore, the algorithm relies on additional bathymetry "looking back" along the path of the vehicle being available, information that many AUV platforms do not have access to.

Recently this approach was extended to mapping in a deep sea environment [6]. While this new version was successful in minimizing the navigation error accumulated during descent, localization was only available after the map building process had been carried out. This prohibited any corrections in navigation during the map building process, suggesting that there will be inconsistencies in the map that will not be corrected by this technique.

Our technique adapts Distributed Particle Mapping to the open underwater domain, utilizing an entirely different map representation, observation model, weighting scheme and final particle selection scheme to produce a self consistent navigation solution and bathymetric map. These are the principal contributions of this paper.

III. BPSLAM

A. AUV trajectory representation

The particle set S_t at timestep t is shown as follows:

$$S_t = \begin{pmatrix} \mathbf{x_{1_t}} & \dots & \mathbf{x_{N_t}} \\ p_{ID_1} & \dots & p_{ID_N} \\ i_{1_{t-1}} & \dots & i_{N_{t-1}} \end{pmatrix} \quad (1)$$

where $\mathbf{x_{i_t}} = (x_{i_t}, y_{i_t})$ is the hypothesized state vector for particle i at time t, N is the number of particles used, p_{ID} is the particle's identification number and i is the particle's index into the particle set during the previous timestep (this allows the extraction of all past poses predicted by any given particle). Depth (z_t), Euler angles $(\phi_t, \theta_t, \psi_t)$, Velocity $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ and Euler angular rotation rates $(\dot{\phi}_t, \dot{\theta}_t, \dot{\psi}_t)$ are tracked using an EKF.

B. BPSLAM map representation

The key idea behind BPSLAM is to retain the original particle's map and have any new particles (*children*) that are resampled from the original particle (*parent*) to point to the parent's map rather than copy the map themselves. With this in mind the idea of extracting a particle's map by tracking the *particle ancestry* follows on naturally. Instead of each particle adding/updating estimates in its own grid, these estimates are first keyed with the particle's ID and then entered into a single global grid. Each cell in the global grid thus contains an estimate for every particle or particle ancestor that has observed that cell. The storage requirements of this approach therefore rivals the simpler approach of providing each particle with its own map. However the latter requires the entire map to be copied whenever it is resampled. For large numbers of particles and large maps this process becomes inefficient, increasing the computational run time exponentially ($O(n^2)$) as more particles are used in the filter.

The list of estimates in the cell are referred to as an *estimate vector*, where each entry in the estimate vector, called an *estimate node*, contains the following items:

- **Particle ID** - The ID of the particle that owns the estimate.
- **Information** - The Information Vector (ξ) and Information Matrix (Ω) of the estimate.
- **Timestamp** - The time at which the estimate was last updated.

The estimate at a given cell for a given particle is accessed by searching through the estimate vector contained within that cell for the last estimate that was made/updated by the particle or its ancestors. BPSLAM thus requires that the lineage of each particle be stored in an *ancestry tree* so that the map for any current particle can be reconstructed or modified at any time. The ancestry tree is represented as a vector of *ancestor particles*, the N youngest of these making up the particle set S_t. By recursively pruning our ancestry tree of dead ancestor particles (those that produced no children) and merging single children into their parents, the number of ancestor particles is guaranteed never to exceed $2N-1$ [3]. By also recycling the IDs of dead particles

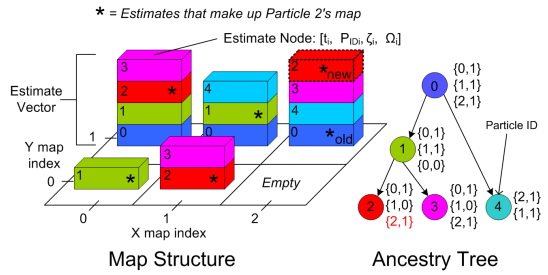


Fig. 1. An example of the map structure and ancestry tree used to store estimates from different particles with a particle cloud size of N = 3. Tracing chronologically backwards through any given particle's ancestry allows extraction of the most recent estimate belonging to it's map, as shown by the starred estimates that make up the map for particle 2. Particle 2 then makes a new observation in cell (2,1), finding ancestor particle 0 holding the most recent update for its map. Using an EIF this estimate is fused with the new observation and added to the estimate vector in that cell, along with the time the estimate was made and the ID of the particle that made it. The ancestry tree is also updated with the new estimate.

when new particles are generated the ID of each particle can be used as a direct pointer into this vector while keeping the size of the ancestry tree fixed. Each ancestor particle retains the following information:

- **Parent ID** - The ID of the parent that the particle was resampled from.
- **Child List** - A list of all the particle's children.
- **Estimates** - A list of all (x,y) locations where the particle made an observation of seabed depth.

Figure 1 provides an example of the map representation and associated ancestry tree used to maintain and extract the maps when $N = 3$.

C. Particle propagation

The BPSLAM algorithm begins by initializing each particle with an estimate of x, y via GPS observations that are available at the start of the mission. As the mission continues the hypothesized state of each particle is propagated based on the current state covariance matrix and the AUV vehicle model. For this application constant velocity and rotation rate models are used to approximate the vehicle dynamics. Observations of velocity, depth and attitude are then used to update the filter.

D. Particle weighting

When an observation is received, the particles are weighted based on how well the observed swath matches with their stored estimates of seabed depth ($\mathbf{E} = [E_x, E_y, E_z]$). Each observation of range, bearing and along track angle $\mathbf{z} = (r, \alpha, \beta)$ can be transformed into an equivalent depth observation z_D that is Gaussian, making the probability distribution of the difference from prediction also Gaussian. The particles can thus be weighted by the likelihood that this difference is zero:

$$weight = P((E_z - z_D) = 0) = \frac{e^{-\frac{1}{2}\frac{(\mu_{E_Z} - \mu_{z_D})^2}{\sigma^2_{z_D} + \sigma^2_{E_Z}}}}{\sqrt{2\pi(\sigma^2_{z_D} + \sigma^2_{E_Z})}} \quad (2)$$

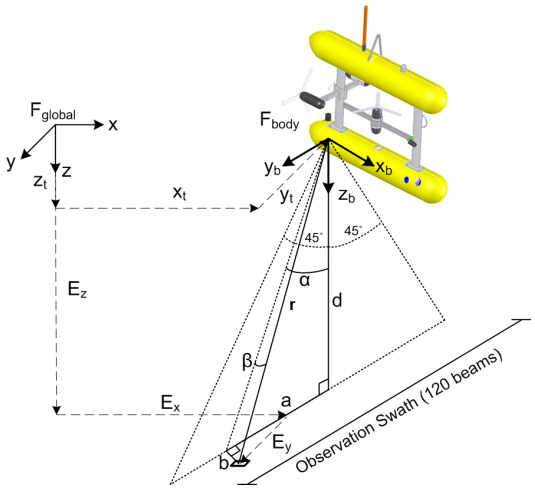


Fig. 2. Relationship between observed Range (r), bearing (α), along track angle (β) and the estimates of AUV state ($\mathbf{x_t}$) and location of the seabed patch observed ($\mathbf{E}$)

To achieve this, errors in the range, bearing and along track angle observations are treated as Gaussian using the Markovian observation model:

$$\mathbf{z} = \begin{pmatrix} r & \alpha & \beta \end{pmatrix}^T = h(\mathbf{x}, \mathbf{E}) + w \quad (3)$$

where w is noise associated with the measurements and is assumed to be independent and Gaussian with covariance:

$$R = diag(\sigma_r^2, \sigma_\alpha^2, \sigma_\beta^2) \quad (4)$$

and h is the measurement function. Note that the uncertainty in bearing and along track angle are often quite small, arising from the beamwidth of the sonar aperture. Figure 2 shows the relationship between an observation $\mathbf{z}$, the corresponding along track (b), across track (a) and depth (d) observation (with respect to the body frame), and the coordinate set of the patch of seabed being observed ($\mathbf{E}$).

From these relationships the measurement function h is given as:

$$h = \begin{pmatrix} \sqrt{b^2 + a^2 + d^2} & tan^{-1}(\frac{a}{d}) & tan^{-1}(\frac{b}{d}) \end{pmatrix}^T \quad (5)$$

The estimated observation, b, a and d can be calculated using the global to body frame Directional Cosine Matrix C_{bg} with the current estimates of roll ($\hat{\phi}$), pitch ($\hat{\theta}$) and heading ($\hat{\psi}$) for that particle.

$$\begin{pmatrix} b \\ a \\ d \end{pmatrix} = C_{bg} \begin{pmatrix} E_x - x_t \\ E_y - y_t \\ E_z - z_t \end{pmatrix} \quad (6)$$

The mean of the depth observation μ_{z_D} can be calculated for a given observation $\mathbf{z}$ and vehicle state $\mathbf{x_t}$ by inverting h to take the observation $\mathbf{z}$ as input. σ_{z_D} is a backwards transport [8] of the covariance R through h and is calculated using:

$${\sigma_{z_D}}^2 = (H^T R^{-1} H)^{-1} \quad (7)$$

where H is the jacobian of the measurement function. This effectively approximates z_D as Gaussian by linearizing about the mean μ_{z_D}.

The weighting factor described above can only be used to govern whether the particle is resampled or not if a prior estimate exists. If one does not exist we assume that the particle is just as likely to be a good or bad estimate of the true state, and thus do not include it in the resampling phase.

However, each particle has an entire swath of observations that can be used to weight the particle. Furthermore, each particle will often have a differing number of observations that can be matched with an existing prior estimate. With this in mind we use the rule that for a swath with B observations a particle should only be included in the resampling phase if it has more than B_{thresh} observations for which a prior estimate can be found. B_{thresh} is dynamically set to equal the percentage of cells within the observation's swath window that get updated in that ping multiplied by the number of observations in that ping. This ensures that resampling will occur only when the observation's swath fully overlaps with an area previously explored. Decreasing B_{thresh} will include more particles in the resampling phase (e.g. those with swaths that only partially overlap previously explored terrain) but comes with the tradeoff of using a less informative likelihood to weight each particle against. This reduced threshold can be chosen based on the amount of overlap/loop closures expected during the mission.

E. Limitations on Mapping Resolution

A data association problem should be highlighted at this stage as, although each state vector $\mathbf{x_t}$ is treated as truth, the uncertainty in range and bearing causes uncertainty in the observation's x, y location, more so at large grazing angles and depths. This can result in matching an observation to a neighboring incorrect grid cell and thus incorrect $\mathbf{E}$ coordinate set if the observation is associated to the cell at its mean x, y location. Several Data Association techniques were investigated [2] to try and take into account the uncertainty in the x, y plane but each were found to be too computationally expensive for real time operation with large numbers of particles.

Without handling this uncertainty in the x, y location, the resolution of the bathymetric maps that can be generated is limited. For any given observation we have a 95% confidence that the error in r, α and β do not exceed $2\sigma_r$, $2\sigma_\alpha$ and $2\sigma_\beta$ respectively. For a mission with expected maximum range, bearing and along track angle $r_{max}, \alpha_{max}, \beta_{max}$ this corresponds to across and along track error bounds of:

$$\begin{pmatrix} a_{ebound} \\ b_{ebound} \end{pmatrix} = 2[r_{max} cos \begin{pmatrix} \alpha_{max} \\ \beta_{max} \end{pmatrix} sin \begin{pmatrix} 2\sigma_\alpha \\ 2\sigma_\beta \end{pmatrix} + 2\sigma_r sin \begin{pmatrix} \alpha_{max} \\ \beta_{max} \end{pmatrix} cos \begin{pmatrix} 2\sigma_\alpha \\ 2\sigma_\beta \end{pmatrix}] \quad (8)$$

Assuming the stability of our platform provides minimal disturbances in roll and pitch, we can approximate the error bound in the x, y location of the observation with a_{ebound} or b_{ebound}, whichever is largest. This quantity can thus be used to indicate the best resolution possible for mapping before errors in our data association technique begin to cause problems. Fortunately this maximum resolution is more than adequate for the large mapping missions we wish to undertake, as shown in section IV.

F. Particle resampling

Once the likelihood of each particle having the correct pose has been calculated they are grouped together and normalized. To avoid particle depletion, resampling is prevented if the *effective particle size* is too large [7]. Particles that are not resampled are removed from the ancestry tree along with their estimates from the map structure. Each ancestor particle maintains a list of the observations it has made, facilitating their removal without needing to query each cell. The particle removal and merging process is also done recursively, as often a parent particle may have only one or no children left after resampling, and thus requiring itself to be merged or removed respectively.

G. Updating the map

After resampling, the surviving particles use their observations to update their respective maps[1]. Assuming a static environment, where the seafloor does not change with time over the period of a single survey, the update equations for the EIF filter simplify to:

$$E_{z(t-1)} = \Omega_{t-1}^{-1} \xi_{t-1} \quad (9)$$

$$\Omega_t = \Omega_{t-1} + H_t^T R_t^{-1} H_t \quad (10)$$

$$\xi_t = \xi_{t-1} + H_t^T R_t^{-1} [\mathbf{z} - h(\mathbf{x_t}, \mathbf{E_{t-1}}) + H_t E_{z(t-1)}] \quad (11)$$

H. Extracting the trajectory and map

The map for any given particle can be retrieved at any time during the mission by extracting the estimates in each map cell made by that particle. If no estimate is available then the cell is iteratively checked for estimates made by the particle's ancestors, most recent first. The trajectory of each particle can also be retrieved by backtracing through the past poses of the particle's ancestors, using the pointers stored within S_t. At the end of the mission the best map and associated trajectory is chosen by analyzing each remaining particle and identifying the one which has the most self consistent map [10]. Naively this error metric would bias those maps with less overlap. However, by restricting analysis of map self consistency to only the overlapping regions common to all maps, this bias is removed.

IV. Results

To test our algorithm on a real mission scenario, bathymetric and navigation logs from a survey undertaken by our research class AUV *Sirius* were utilized [13]. The survey was taken off the coast of Tasmania and contains several pockmarks 30 metres in diameter and 3 metres deep

[1]The advantage of using an EIF over an EKF here is the ability to specify zero information at initialisation, as opposed to infinite uncertainty. The EIF is also computationally superior in this scenario as information is additive and, as the depths are assumed stationary, there is no prediction used in the filter.

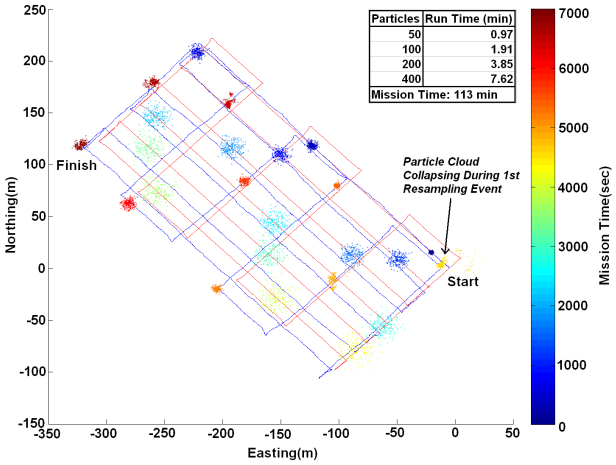


Fig. 3. Tracklines produced by Dead Reckoning (red) and the BPSLAM Filter (blue) using 400 particles. The evolution of the particle cloud is also shown, changing from dark blue to dark red as the mission progresses. The run times for this mission are also shown when different numbers of particles are used in the filter.

(on average). Two orthogonal grid transects were carried out underwater at an altitude of 20m. USBL observations were also available and relayed to the AUV throughout the mission. These observations are not used by the BPSLAM filter but are reproduced here as a baseline to demonstrate the best possible map that we can currently produce, and how BPSLAM can significantly improve our results.

The maximum resolution that we can correctly map is approximated by Eq. 8. Substituting in the mission parameters limits the mapping resolution to no smaller than $0.53m$. However based on the size of the survey (335m by 350m), the quality of our navigation sensors and the length of the mission (113 minutes), a grid resolution of 1m is sufficient for the BPSLAM filter to demonstrate its performance. Additionally, as this survey mission does several complete passes over previously explored terrain, the parameter B_{thresh} can be set to require full overlap over previously explored seabed for resampling.

Sirius possesses an AHRS that provides the AUV with observations of roll, pitch and heading. Depth observations are obtained through a high precision pressure depth sensor whereas along track, across track and depth velocity observations are provided by a Doppler Velocity Log (DVL). This coupled with the design of *Sirius*, which promotes passive stability in pitch and roll, justifies the inclusion of x, y states into the particle filter, as they are the most prone to drift, while leaving the remaining states to be tracked by the EKF.

Figure 3 demonstrates the BPSLAM filter performance during the mission, as well as its computational performance when different numbers of particles are used. Resampling is able to remove pose solutions that cause inconsistency in their maps while allowing those that agree to be propagate and multiply. The uncertainty in the pose solution reduces and the point cloud converges to a self-consistent solution. This is particularly evident in Figure 3 near the East corner of the map where the first resampling event occurs after the vehicle has traveled nearly 2km. Here the particle cloud is in the process of collapsing down (shown in light orange) into a smaller set of more likely state hypotheses. Figure 4 presents the results of our filter running on our mission with 400 particles. Ground truthing for an AUV in an underwater environment requires costly infrastructure, such as a Long Base Line (LBL) acoustic transponder net, which we do not have at our disposal. Instead the map self consistency metric as described in section III-H is the error metric we use to compare the map quality generated by the BPSLAM filter to those generated with dead reckoning or USBL fused navigation solutions.

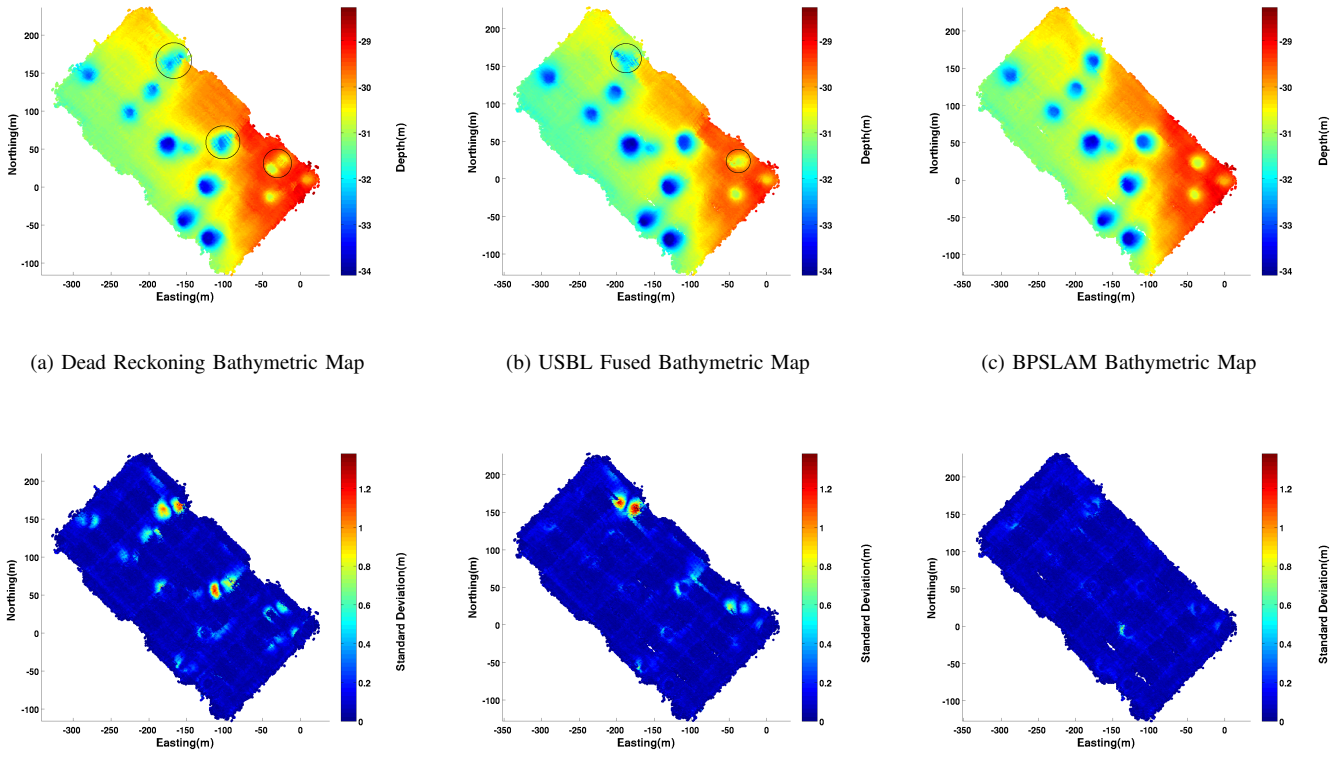
From Figure 4(a) the pockmarks discovered during this mission can be seen. However, the use of dead reckoning as a navigation method has resulted in some blur. Errors produced by bad sonar returns can appear anywhere in the map and will not be consistent with nearby swaths. However, inspection of Figure 4(d), which plots the standard deviation in the observations used to create this map, reveals that the inconsistencies are localized around the pockmarks where sudden changes in depth occur. This suggests that the blur is most likely caused by navigation error creating a misalignment between successive surveys of each pockmark i.e. ghosting. Figures 4(b) and 4(e) show how fusing of USBL observations helped reduce ghosting in the map by improving the navigation solution. However, map blur is still evident, particularly along the northeast border.

Figures 4(c) and 4(f) demonstrate the power of the BPSLAM filter. The ghosting in the map has been reduced significantly. As expected the BPSLAM filter has identified a navigation solution that aligns all the pockmarks discovered in the mission, without the need for specifying loop closure techniques or feature detection algorithms. The average binned standard deviation of overlapping areas common to all maps is $0.132m, 0.119m$ and $0.087m$ using dead reckoning, USBL observations and the BPSLAM filter respectively, further validating the BPSLAM filter's superior performance.

Finally, the run times in Figure 3 verify that the algorithm retained an asymptotic complexity of constant/linear (amortized) time per iteration. Additionally only 7.62 minutes were needed to process a mission 113 minutes long, validating its potential use in real time applications.

V. Conclusions

In this paper we have presented an algorithm that utilizes the BPSLAM filter to improve the accuracy of maps and trajectories generated during bathymetric mapping missions without having to explicitly model seabed features. Results showed an improvement in both the map and the trajectory when compared to using state of the art fused navigation and low cost sensors without SLAM. In addition our algorithm maintains an asymptotic computational complexity that scales linearly with the number of particles used. The run time of the algorithm was 15 times faster than that of the mission itself, offering potential to be implemented in real time.



(a) Dead Reckoning Bathymetric Map

(b) USBL Fused Bathymetric Map

(c) BPSLAM Bathymetric Map

(d) Dead Reckoning Deviation Map

(e) USBL Fused Deviation Map

(f) BPSLAM Deviation Map

Fig. 4. Bathymetric maps generated using our three different navigation solutions. A corresponding map of the standard deviation in observed depth for each grid cell is also provided. Comparison shows the BPSLAM filter significantly reducing the mapping errors, the most prominent circled in black and also highlighted by a decrease in standard deviation, when compared to the maps produced by pure dead reckoning and USBL fused navigation solutions.

Acknowledgements

This work is supported by the ARC Center of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government and the Integrated Marine Observing System (IMOS) through the DIISR National Collaborative Research Infrastructure Scheme. The authors would like to thank the Tasmanian Aquaculture and Fisheries Institute (TAFI) and the Captain and crew of the R/V Challenger for facilitating the data collection used to validate this work. We would also like to acknowledge the help of the students and technical staff of the ACFR who keep our AUV operational, including Ian Mahon, Matthew Johnson-Roberson, Duncan Mercer, George Powell, Ritesh Lal, Paul Rigby, Jeremy Randle and Bruce Crundwell.

References

[1] S. Blasco. Application of Multibeam Surveying to Resource Mapping. *FIG XXII International Congress*, TS4.3 Hydrographic Surveying I, 2002.

[2] J. Dezert and Y. Bar-Shalom. Joint probabilistic data association for autonomous navigation. *Aerospace and Electronic Systems, IEEE Transactions on*, 29(4):1275–1286, Oct 1993.

[3] A.I. Eliazar and R. Parr. DP-SLAM 2.0. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 2:1314–1320 Vol.2, 26-May 1, 2004.

[4] A.I. Eliazar and R. Parr. Hierarchical Linear/Constant Time SLAM Using Particle Filters for Dense Maps. *Neural Information Processing Systems Conference*, 2005.

[5] N. Fairfield, G.A. Kantor, and D. Wettergreen. Towards particle filter SLAM with three dimensional evidence grids in a flooded subterranean environment. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3575 – 3580, 2006.

[6] Nathaniel Fairfield and David Wettergreen. Active localization on the ocean floor with multibeam sonar. In *Proceedings of MTS/IEEE OCEANS*, 2008.

[7] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *Robotics, IEEE Transactions on*, 23(1):34–46, Feb. 2007.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

[9] C. Roman and H. Singh. Improved vehicle based multibeam bathymetry using sub-maps and SLAM. *Intelligent Robots and Systems (IROS)*, pages 3662–3669, 2005.

[10] C. Roman and H. Singh. Consistency based error evaluation for deep sea bathymetric mapping with robotic vehicles. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3568–3574, 15-19, 2006.

[11] H. Singh, L. Whitcomb, D. Yoerger, and O. Pizarro. Microbathymetric Mapping from Underwater Vehicles in the Deep Ocean. *Computer Vision and Image Understanding*, 79:143–161, 2000.

[12] S.B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, Australian Centre for Field Robotics - The University of Sydney, 2001.

[13] S.B. Williams, O. Pizarro, I. Mahon, and M. Johnson-Roberson. Simultaneous Localisation and Mapping and Dense Stereoscopic Seafloor Reconstruction Using an AUV. *Springer Tracts in Advaned Robotics*, 54:407–416, 2009.