Efficient and Featureless Approaches to Bathymetric Simultaneous Localisation and Mapping

Stephen Alexander Barkby

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

ARC Centre of Excellence in Autonomous Systems
Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

Submitted March 2011

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Stephen Alexander Barkby

March 2011

Abstract

Stephen Alexander Barkby
The University of Sydney

Doctor of Philosophy
March 2011

Efficient and Featureless Approaches to Bathymetric Simultaneous Localisation and Mapping

This thesis investigates efficient forms of Simultaneous Localization and Mapping (SLAM) that do not require explicit identification, tracking or association of map features. The specific application considered here is subsea robotic bathymetric mapping. In this context, SLAM allows a GPS-denied robot operating near the sea floor to create a self-consistent bathymetric map. This is accomplished using a Rao-Blackwellized Particle Filter (RBPF) whereby each particle maintains a hypothesis of the current vehicle state and map that is efficiently maintained using Distributed Particle Mapping. Through particle weighting and resampling, successive observations of the seafloor structure are used to improve the estimated trajectory and resulting map by enforcing map self consistency.

The main contributions of this thesis are two novel map representations, either of which can be paired with the RBPF to perform SLAM. The first is a grid-based 2D depth map that is efficiently stored by exploiting redundancies between different maps. The second is a trajectory map representation that, instead of directly storing estimates of seabed depth, records the trajectory of each particle and synchronises it to a common log of bathymetric observations. Upon detecting a loop closure each particle is weighted by matching new observations to the current predictions. For the grid map approach this is done by extracting the predictions stored in the observed cells. For the trajectory map approach predictions are instead generated from a local reconstruction of their map using Gaussian Process Regression. While the former allows for faster map access the latter requires less memory and fully exploits the spatial correlation in the environment, allowing predictions of seabed depth to be generated in areas that were not directly observed previously. In this case particle resampling therefore not only enforces self-consistency in overlapping sections of the map but additionally enforces self-consistency between neighboring map borders.

Both approaches are validated using multibeam sonar data collected from several missions of varying scale by a variety of different Unmanned Underwater Vehicles. These trials demonstrate how the corrections provided by both approaches improve the trajectory and map when compared to dead reckoning fused with Ultra Short Baseline or Long Baseline observations. Furthermore, results are compared with a pre-existing state of the art bathymetric SLAM technique, confirming that similar results can be achieved at a fraction of the computation cost.

Lastly the added capabilities of the trajectory map are validated using two different bathymetric datasets. These demonstrate how navigation and mapping corrections can still be achieved when only sparse bathymetry is available (e.g. from a four beam Doppler Velocity Log sensor) or in missions where map overlap is minimal or even non-existent.

Acknowledgements

This research was undertaken in collaboration with the Australian Research Council, with funding support from the Integrated Marine Observing System (IMOS) facility and Commonwealth Environment Research Facilities (CERF) Marine Biodiversity Hub.

First and foremost I would like to thank my supervisor Stefan Williams and associate supervisors Oscar Pizarro and Michael Jakuba for their support, guidance and sage advice that has seen me through to the end of my candidature. They welcomed me into the Oceanographic community of which I now feel part of, and have made my time here truly memorable.

I'd also like to thank the people I have met along the way that have contributed to my research, including Chris Roman for generously supplying bathymetry and navigation for the TAG hydrothermal mission, as well as his support in its processing and visualisation. Special thanks must also go to Dana Yoerger for generously supplying the Haakon Mosby dataset and introducing me to the taste experience that is Mississippi Oysters. The support of these two people has been instrumental in showcasing the capabilities of the BPSLAM filter, without which the results presented here would not nearly have been as compelling.

While collaborating with these people abroad has been a rewarding experience I would be amiss to not mention all the help and support I have also received at home from the people at the Australian Centre for Field Robotics. In particular I would like to acknowledge the unwavering support of my C++ guru Ian Mahon, whose technical support and software modules were a boon to BPSLAM's implementation. I would also like to thank Asher Bender for all of our crazy discussions on Gaussian Processes, along with other members of the GP braintrust at the ACFR including Fabio Ramos, Arman Melkumyan, Simon O'Callaghan and Alistair Reid.

Back to the water I would like to thank the AUV team and all those who have helped contribute to the development and operation of the AUV including George Powell, Matthew Johnson-Roberson, Ritesh Lal, Paul Rigby, Lashika Medagoda, Daniel Steinberg, Ariel Friedman, Jeremy Randle, Bruce Crundwell and the late Duncan Mercer and Alan Trinder. I would also like to acknowledge the Captain and crew of the R/V Challenger. Their sustained efforts were instrumental in facilitating the successful deployment and recovery of *Sirius*.

Finally I would like to thank my family and friends who have supported me in this long journey through university, my days as a student now finally at an end.

Contents

List of Figures

List of Notation

Acronyms

AUV	Autonomous Underwater Vehicle	1
ROV	Remotely Operated Vehicle	1
UUV	Unmanned Undersea Vehicle	1
INS	Inertial Navigation System	2
DVL	Doppler Velocity Log	2
GPS	Global Positioning System	2
USBL	Ultra-Short Baseline	2
LBL	Long Baseline	2
SLAM	Simultaneous Localisation and Mapping	1
RBPF	Rao Blackwellized Particle Filter	5
EKF	Extended Kalman Filter	9
EIF	Extended Information Filter	9
SEIF	Sparse Extended Information Filter	9
SIFT	Scale Invariant Feature Transform	9
TAN	Terrain Aided Navigation	12
DR	Dead Reckoning	13
BPSLAM	Bathymetric Particle SLAM	5

Convention

Scalars are represented by regular symbols e.g. x, X

Vectors are represented by bold lowercase symbols e.g. $\mathbf{x}$

Matrices are represented by bold uppercase symbols e.g. $\boldsymbol{X}$

General Geometry

${}^{i}\mathbf{p}_{j}$	Pose of frame j relative to frame i
${}^{i}x_{j}$	X-axis coordinate for the position of j in frame i
${}^{i}y_{j}$	Y-axis coordinate for the position of j in frame i
${}^{i}z_{j}$	Z-axis coordinate for the position of j in frame i
${}^{i}\boldsymbol{\psi}_{j}$	Euler angles describing the orientation of j relative to frame i
${}^{i}\phi_{j}$	Roll Euler angle of j relative to frame i
${}^{i}\theta_{j}$	Pitch Euler angle j relative to frame i
${}^{i}\psi_{j}$	Yaw Euler angle j relative to frame i
${}^{i}\mathbf{v}_{j}$	Velocity of frame j relative to frame i
${}^{i}\dot{x}_{j}$	X-axis velocity of j relative to frame i
${}^{i}\dot{y}_{j}$	Y-axis velocity of j relative to frame i
${}^{i}\dot{z}_{j}$	Z-axis velocity of j relative to frame i
${}^{i}\dot{\boldsymbol{\psi}}_{j}$	Rate of change of Euler angles of j relative to frame i
${}^{i}\dot{\phi}_{j}$	Rate of change of roll Euler angle of j relative to frame i
${}^{i}\dot{\theta}_{j}$	Rate of change of pitch Euler angle of j relative to frame i
${}^{i}\dot{\psi}_{j}$	Rate of change of yaw Euler angle of j relative to frame i
${}^{j}_{i}\mathbf{F}_{\mathbf{i}}$	Frame of Reference i
${}^{j}_{i}\mathbf{R}$	Matrix rotating frame i to be oriented with frame j
${}^{j}_{i}\mathbf{E}$	Matrix transforming local body rates of i to Euler rates in frame j
${}^{j}_{i}\mathbf{T}$	Homogenous Transformation Matrix converting poses in frame i to frame j

General Estimation

$\mathbf{x}$	State vector
μ	State estimate
$\mathbf{P}$	State covariance
ξ	Information vector
$\mathbf{\Omega}$	Information matrix
$\mathbf{z}$	Observation
$\hat{\mathbf{z}}$	Predicted observation
$f[\cdot]$	State transition model
$\mathbf{F}$	State transition Matrix
$\mathbf{w}$	Process model noise
$\mathbf{Q}$	Process noise Covariance

Observation Models

$\mathbf{h}[\cdot]$	Observation model
$\nabla_x\mathbf{h}$	Jacobian of the observation model with respect to all estimated states
$\mathbf{v}$	Observation model noise
$\mathbf{R}$	Observation model noise covariance
r	Range observation
α	Bearing of range observation
β	Along track angle of range observation
b	Along track coordinate of observation in sensor frame
a	Across track coordinate of observation in sensor frame
d	Depth track coordinate of observation in sensor frame

Particle Filter

N	Number of particles in filter
$\mathbf{S}$	Current Particle Set
P_{ID_i}	ID of particle in current set at index i
w_i	Assigned weight of particle in current set at index i

Gaussian Process Regression

$\mathbf{g}[\cdot]$	Underlying function
$\mathcal{D}$	Training Set
$\mathbf{X}$	Training Inputs
$\mathbf{Y}$	Training Outputs
M	Number of training points
$\mathbf{W}$	Training model noise covariance
$\mathbf{L}$	Cholesky Decomposition

$\mathbf{k}[\cdot]$	Covariance function of Gaussian Process model
$\mathbf{m}[\cdot]$	Mean function of Gaussian Process model
l	Length hyperparameter of covariance function
σ_0	Signal variance hyperparameter of covariance function

$\mathbf{X}_*$	Test Inputs
$\mathbf{g}_*$	Test Outputs
$\mathbf{cov}(\mathbf{g}_*)$	Predictive variance of underlying function
$\mathbf{cov}(\mathbf{Y}_*)$	Predictive variance of test targets

Chapter 1

Introduction

This thesis addresses the problem of improving navigation and mapping in underwater vehicles by providing corrections to navigation through the mapping sensor, in this case a multibeam sonar. This is achieved by using multibeam observations to localise the vehicle (thereby improving navigation) based on the surrounding area. The ability for the vehicle to do this accurately is therefore dependent on the accuracy of the map. Conversely the accuracy of the map generated by a mobile agent is dependent on the accuracy of the navigation solution. An interdependence between the navigation solution and the map is therefore introduced if both operations are to be attempted simultaneously, as is desired here. This problem, known as Simultaneous Localisation and Mapping (SLAM) allows a bounded positional error to be maintained without requiring additional infrastructure. In this way current state of the art techniques for underwater navigation and mapping that do not provide this correction can be improved upon.

1.1 Motivation

The use of Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs) as mapping platforms is becoming evermore prevalent in research and industry, due to their advantages over more traditional shipborne and towed sensor systems (Singh, Whitcomb, Yoerger & Pizarro 2000). Such advantages include the ability to operate away from the oceans's surface, allowing these Unmanned Undersea Vehicles (UUVs) to maintain close proximity to the seafloor regardless of depth (Kirkwood 2007). This allows for higher map resolutions to be achieved and other sensors, such as stereo cameras, to be utilized (Negahdaripour & Madjidi 2003)(Williams, Pizarro, Mahon & Johnson-Roberson 2009). UUVs also have the benefit of requiring less personnel for operations (particularly with AUVs), as well as allowing mission operations to be conducted with smaller support ships. Lastly UUVs can maintain operations in rough seas, whereas this can affect the quality of shipborne surveys and may even cause them to be aborted.

For these reasons UUVs have proven to be an invaluable resource in marine applications such as geological surveying, biodiversity assessments, pipeline surveys and prospecting/salvage missions (Chapman, Wills, Brookes & Stevens 1999)(Blasco 2002). With the advancement of computer processor design and high yield power supplies the capability of UUVs to operate for longer periods underwater is also increasing. However this capability is hindered by the navigation error that accumulates in the vehicle's location estimate while operating in this GPS denied environment, continuing to increase until either the UUV surfaces or has position fixes relayed to it. Prolonging mission operations therefore requires that this error be corrected for if precise navigation and mapping is to be maintained. SLAM provides a method of fulfilling this requirement, though is by no means the only option available to produce navigation corrections. However an additional benefit of SLAM is its ability to infer corrections in the map observations themselves. Without handling these mapping errors blur or inconsistency in the generated map can still occur, even if navigation is carried out with absolute precision.

1.2 Underwater Navigation

Inertial Navigation Systems (INS), Attitude Heading Reference Systems (AHRS), a Doppler Velocity Log (DVL), GPS, Ultra Short Baseline (USBL) and/or Long Baseline (LBL) acoustic positioning systems all provide options for improving navigational accuracy, each with varying levels of attainable precision (Fairfield & Wettergreen 2008). Combining a high precision INS with an AHRS and DVL provides stand-alone navigation with an unbounded error in position typically < 0.1 % of the total distance traveled (Fairfield & Wettergreen 2008), though this is often an expensive option and will naturally degrade with increasing mission time. This also assumes that the UUV is operating close enough to the seafloor to maintain consistent DVL bottom lock. To provide navigation with a bounded error GPS observations can be used to constrain the uncertainty in the vehicle's position while it is travelling on the ocean's surface, typically within 12 m of the true position (Fairfield & Wettergreen 2008). While UUVs cannot receive GPS observations of their location underwater, USBL acoustic positioning can be implemented to yield range and bearing measurements between the UUV and a support vessel. By using GPS/Inertial measurements of the ship the range/bearing observations 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 UUV within the USBL's range for the duration of the mission. This problem is further compounded in deep water deployments as the accuracy of the USBL fix is dependent on the angular accuracy of the USBL head, contributing an error of approximately 0.5 % of range (Fairfield & Wettergreen 2008). LBL acoustic positioning also provides navigation estimates with bounded error but this requires additional infrastructure to be in place, as well as the LBL transponder net to be accurately surveyed in. Furthermore LBL transponder nets are subject to a tradeoff between accuracy and coverage. High frequency (300kHz) LBL provides sub-centimeter localisation but restricts operations to a maximum range of 100m from the beacons (Fairfield & Wettergreen 2008). Alternatively low frequency LBL (12kHz) can be used, which provides ranges of up to 10 km, but this comes with the tradeoff of only achieving 0.1 m to 10 m accuracy, depending on the beacon geometry and accuracy of the estimated speed of sound profile. In addition LBL systems deployed on seafloor with complex structure may suffer from multipath and/or occlusions.

Compared to these systems SLAM has the potential to provide position corrections that are of the order of the resolution of the mapping sensor, provided that opportunities for reobserving terrain or "loop closures" are available. This in addition to not requiring additional infrastructure makes SLAM an attractive option. As will be discussed in Chapter 2 there are many methods of implementing SLAM, the most suitable of which is determined by the characteristics of the mapping sensor and the environment to be mapped.

1.3 Underwater Mapping

Bathymetric maps represent the height of the earth's surface underwater within a given geographical region. These can be generated by several methods, depending on the resolution and coverage desired. In shallow waters bathymetric mapping can be carried out quickly and efficiently through an airborne Lidar system or through Airborne Visible-Infrared Imaging Spectrometry. However these methods are restricted to mapping depths of 50 m or less, depending on water clarity (Danson 2006)(McIntyre, Naar, Carder, Donahue & Mallinson 2006).

To provide bathymetric maps at greater depths a shipborne multibeam depth profiler can be utilized, offering the high coverage bathymetry needed to map out large areas of seafloor quickly and accurately (Singh et al. 2000). However ships equipped with this technology are restricted to operating at the ocean's surface and so provide maps whose resolution decreases as seabed depth increases. Wave induced motion is also problematic for these systems as errors in attitude can translate to significantly large errors in the northing/easting coordinate of the observation, particularly for large grazing angles and depths. For these reasons, and those mentioned previously, UUVs are often used instead for these deeper water surveys.

Bathymetric maps are traditionally built using a gridded or point cloud model of the seafloor with a deterministic model of the vehicle or vessel pose estimate (Blasco 2002). In this case it is sufficient to generate the map by estimating the depth at any given location with the mean of the depth measurements observed there. However, assuming a deterministic navigation solution introduces misalignment in the map if the navigation is subject to errors such as drift or unmodeled biases.

SLAM allows misalignments such as these to be resolved by correcting the navigation solution from which the map was generated from, and can be classified as either feature-based or featureless in its approach. In the feature-based approaches, appropriate features are identified using the vehicle's sensors (camera, laser, sonar, etc.) and the location of the features in space are estimated. Reobservations of particular features are then used to refine the estimate of the vehicle location. Featureless techniques on the other hand exploit map representations that do not require features to be explicitly identified in the sensor data. The unstructured nature of the majority of the seafloor suggest that such an approach may be more appropriate for mapping using sonar. Features, such as peaks and troughs in the seabed, are scale dependent, sensitive to viewing angle and are typically of low spatial density when compared to the footprint of the sonar swath, making them difficult to identify and model reliably.

1.4 Approach

The aim of this thesis is to provide a method of SLAM suitable for underwater bathymetric mapping that can be used by vehicles such as UUVs to improve their navigation and mapping when compared to using state of the art navigation without SLAM. From the discussion so far it is clear that a featureless approach to SLAM is the most suitable option to achieve this.

The key framework upon which this thesis is built is the Rao Blackwellized Particle Filter (RBPF) (Thrun, Burgard & Fox 2005). This is a non-parametric version of the Bayes filter that represents the vehicle state probability distribution by a collection of *particles*, each maintaining a hypothesis of the current vehicle state and map. Through particle weighting and resampling, successive observations of the seafloor structure are used to improve the estimated trajectory and resulting map by enforcing map self consistency, thereby providing a featureless method of Bathymetric Particle SLAM (BPSLAM).

The map representation used by the filter should be chosen based on the requirements of the vehicle, where the the computational memory requirements and run time of the filter are often of primary concern. As such this thesis presents two map representations, each focusing on optimising one of these requirements.

1.5 Contributions

The work presented in this thesis focuses on applying an existing framework to featureless SLAM (the RBPF filter) to a novel domain i.e. the underwater environment. Initially a 2.5D gridded map representation, which can be considered an extension of the 2D occupancy grid into 2.5D, is coupled with this framework to form an approach to BPSLAM. Research and implementation of this map representation then motivated the formulation of an entirely different and novel map representation that addresses the shortcomings of the grid map approach. These two map representations form the main theoretical contributions of this thesis, which through their implementation also led to several practical contributions. These contributions, both theoretical and practical, are listed below:

- A novel 2.5D grid map representation whose run time and memory requirements scale linearly with the number of particles used. In practice the computational speed attained by this form of BPSLAM is on the order of $\approx 2\%$ of the mission time.

- A novel trajectory map representation whose run time and memory requirements scale linearly with the number of particles used. In practice the memory usage of this approach allows for the processing of very large datasets ($\approx 1.65\ km^2$) while attaining a computational speed on the order of $\approx 25\%$ of the mission time. In addition this approach can provide corrections to navigation and mapping even when all areas in the map are only observed once.
- The implementation of both map representations to several real mission scenarios, resulting in a consistent improvement in navigation and map self-consistency.
- A novel online and offline technique for ranking the map quality produced by different navigation solutions based on map self-consistency.
- A method to provide interpolation and extrapolation of terrain predictions in unobserved regions by learning and exploiting the spatial correlation in the seabed using Gaussian Processes and an online version of covariance function training.
- A principled method of transforming uncertainty in a set of range observations to an equivalent set of observations with uncertainty only in depth.

1.6 Thesis Outline

This section provides an overview of the thesis. Chapters 2 and 3 present background and related work while Chapters 4-5 are novel contributions and results.

Chapter 2 provides background information on the current state of research for SLAM in underwater vehicles.

Chapter 3 describes the basic theory behind RBPF SLAM.

Chapter 4 presents the development of the grid map representation used by our BPSLAM filter.

Chapter 5 presents the development of the trajectory map representation that can alternatively be used by our BPSLAM filter.

Chapter 6 provides conclusions and directions for future work.

Chapter 2

SLAM Review

2.1 Introduction

The origins of the SLAM problem can be traced back to the 1986 IEEE Robotics and Automation Conference held in San Francisco, California. At this time the use of probabilistic methods in robotics and Artificial Intelligence (AI) was just emerging, sparking an interest from researchers Cheeseman, Crowley and Durrant-Whyte who realised these methods could be applied to mapping and localisation problems. In essence the problem they sought to solve was as follows: upon placing a robot at an unknown location in an unknown environment, how can the robot incrementally build a consistent map of its environment while at the same time determine its location within this map. This soon came to be known as the Simultaneous Localisation And Mapping (SLAM) problem, whose structure was first presented at the International Symposium on Robotics in 1995. At the theoretical level the SLAM problem has since been solved, though much research still exists concerning its practical implementation, as is the case here. For a more thorough investigation into the history of SLAM and its general framework Durrant-Whyte & Bailey (2006*a*) and Durrant-Whyte & Bailey (2006*b*) provide an extensive introduction.

The remainder of this chapter describes the current state of the art SLAM methods available, classifiable as either feature-based or featureless in their approach. This serves to highlight the strengths and weaknesses of each technique, and in doing so motivates the need for a new approach to SLAM that is suitable for large scale underwater bathymetric mapping.

2.2 Feature-Based SLAM

Feature-based SLAM methods represent the environment by a map of the positions (and associated uncertainties) of several distinguishable landmarks, while the accuracy of the navigation solution is represented by the uncertainty in each of the vehicle's states. By augmenting the vehicle state vector with the feature position estimates, a Bayesian filter such as the Extended Kalman Filter (EKF)[1] can be used to simultaneously track and reduce the uncertainty in the augmented state vector upon re-observing features. This basic approach to SLAM has been shown to work with various mapping sensor modalities, such as sidescan sonar (Tena, de Raucourt, Petillot & Lane 2004) and single vision cameras (Garcia, Puig, Ridao & Cufi 2002). However, this approach becomes computationally expensive for large numbers of features, as the update time of the EKF scales quadratically with the dimension of the augmented state vector. In this situation the EKF can be replaced by an Extended Information Filter (EIF), which is capable of maintaining computational efficiency for large numbers of features if we further approximate the SLAM posterior with a sparse EIF representation (SEIF)(see Thrun et al. (2005) for a review). However, this results in a loss of accuracy, depending on what degree of sparseness is enforced in the information matrix to improve the computational efficiency.

Alternatively Eustice & Singh (2005) presents an approach that casts the SLAM problem into a delayed state framework, also known as view-based SLAM, where past vehicle state estimates are tracked instead of features. By using a single downward looking camera to collect images of the environment, relative pose measurements between past estimates are generated by pairwise registration of images with common overlap, using identifiers such as Harris Corners and Scale Invariant Feature Transform (SIFT) keypoints. This technique also uses an EIF to store past poses. However, in this framework the information form is *exactly* sparse, and so removes the approximation error that the SEIF approach is subject to. Mahon, Williams, Pizarro & Johnson-Roberson (2008) has also shown that this approach can be extended to 3D using a stereo camera and has illustrated how a modified Cholesky factorisation allows for efficient prediction and update depending on the variable ordering of the state vector.

FastSLAM (see Thrun et al. (2005) for a review) is another approach to feature-based

[1]For more information Thrun et al. (2005) provides an extensive review and introduction to Bayesian Filters such as the EKF and their application to probabilistic robotics.

SLAM which reduces the computational complexity of the EKF method described earlier by exploiting the conditional independence of the features being tracked, given the vehicle trajectory. In other words, if the vehicle state is known then the estimation of feature positions becomes independent of each other, allowing them to be tracked instead with separate low-dimensional EKF's. A particle filter can therefore be used to represent the vehicle state, provided that each particle (which treats its vehicle state hypothesis as a known quantity) maintains its own set of EKF's describing its uncertainty in each feature location. The act of separating the vehicle state from the map in this way is an example of *Rao-Blackwellization* (though the map need not be represented by EKF's) and has become a popular method of conducting feature-based SLAM (Doucet, de Freitas, Murphy & Russell 2000) (Dong, Wijesoma & Shacklock 2007).

All of the feature-based techniques discussed so far have one common approximation that reduces the accuracy of each filter, this being the need to linearize the vehicle and observation models upon every state prediction/update. In addition techniques such as the SEIF do not attempt to solve the *full* SLAM problem i.e. they only calculate the posterior over the current pose instead of computing the joint posterior over the whole path of the robot, which allows corrections made in the current pose to propagate back to previous poses. The delayed state framework mentioned earlier can be considered a solution to the full SLAM problem. However when updating past poses this method is bound to use the same linearisations as it did initially, retaining the original error introduced by this approximation.

GraphSLAM provides a solution to the full SLAM problem by framing it as a non linear least squares minimisation problem (see Thrun et al. (2005) for a review). Specifically GraphSLAM retains the full robot path and uses measurements to define constraints between robot poses and sensed features. Likewise control inputs are used to define constraints between consecutive robot poses. In this way the information retained by GraphSLAM is naturally sparse. Upon receiving all measurements and poses the maximum likelihood estimate of the robot path and map is calculated by minimising the function formed from these constraints, usually cast in information space. By performing batch processing the linearisation errors described previously can also be reduced by relinearising during each iteration of the optimisation procedure. While this solves the full SLAM problem the main disadvantage of GraphSLAM is that it cannot run online, whereas all the filters mentioned above are specifically tailored to this, with the exception of FastSLAM and view-based

SLAM that are capable of both.

The work of Dellaert is similar to GraphSLAM in that it also treats SLAM as a least squares problem (Dellaert & Kaess 2006). However in this case the information matrix is factorised into square root form that allows for fast extraction of the robots optimal trajectory and map, as well as resulting in a more stable and accurate algorithm. This method called square root simultaneous Smoothing And Mapping ($\sqrt{SAM}$) can be run as both an offline batch process or an online version, as the factorised matrix is capable of being updated incrementally. However one of the main limitations of this approach is a computational cost thats grows unbounded (as the full trajectory is always used). This suggests that its implementation in large datasets is limited.

Up until now an important requirement of feature-based SLAM has not been discussed, this being the accurate identification of features in the environment that can be used for localisation. Furthermore the algorithm must be capable of determining whether or not two features observed at different points in time correspond to the same feature in the physical world, known as the data association problem. As mentioned earlier a common solution to this problem in vision-based SLAM is to use SIFT keypoint features extracted from the visual imagery (Lowe 2004). These features are invariant to both changes in scale and rotation and provide descriptors that can be used to uniquely identify a feature when it is reobserved.

Discerning what constitutes a feature in bathymetric data however remains a non trivial issue. Previous work has shown that steep gradient contours can be used as map features (Lucido, Opderbecke, Rigaud, Deriche & Zhang 1996), though the matching techniques described involve several processing steps and are sensitive to noise, scale, orientation and displacement. Furthermore, the seabed often lacks the relatively sparse and unambiguous landmarks that are required for feature-based SLAM, leading to instances where features may have multiple possible associations. Methods such as incremental maximum likelihood data association (Thrun et al. 2005) attempt to handle these multiple hypotheses but cannot guarantee that the correct match will be made. Consequently any loop closures performed using incorrect data associations can corrupt the navigation solution, potentially reducing its accuracy below that obtained by simple Dead Reckoning.

To avoid this we assume dense sensor data which individually are not very distinctive, but taken together may form recognizable trends in the seabed. This leads us to examine methods of SLAM that attempt to resolve misregistrations in the map by manipulating the surface generated as a whole, rather than explicit features identified therein.

2.3 Featureless SLAM

Featureless SLAM does not require features to be explicitly identified or tracked and so bypasses the problem of feature estimation entirely. Many approaches to featureless SLAM stem from earlier work in Terrain Aided Navigation (TAN), where position fixes are derived by seeking a position offset from the current estimate that minimizes a cost function. This function is related to the difference between the measured and expected ranges to the local terrain (referencing a prior map) at each candidate offset (Golden 1980). The work of Burgard, Fox, Jans, Matenar & Thrun (1999) shows how this approach can be extended to SLAM by tracking a collection of candidate offsets with a particle filter, where each particle builds its own occupancy grid map that it references during the mission, as opposed to requiring that all particles reference a common prior map. More recent work has examined methods to efficiently maintain detailed grid maps using similar particle based techniques (Grisetti, Stachniss & Burgard 2007, Eliazar & Parr 2003).

An approach to bathymetric SLAM which uses these foundations has been reported by Fairfield, Kantor & Wettergreen (2006). Here a RBPF is combined with an occupancy grid-based volumetric 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 in real time, where the benefit of continuous localisation via measurement of the AUV's proximity with the surrounding cave walls, using both fore and aft mounted sonar arrays, is fully utilized. This approach has also been shown to perform well in an open marine environment when equipped with a single multibeam sonar and a large scale high resolution prior map (Fairfield & Wettergreen 2008). However in this case localisation was only performed using the prior map. For bathymetric mapping missions where no prior map information is available opportunities to localise are typically not as abundant as in these two trials.

Alternatively another approach reported by Roman & Singh (2005) uses a point cloud map model to divide a temporal sequence of bathymetry into submaps that are assumed to be error free. Pairwise matching of overlapping submaps constrains the vehicle trajectory and submap origins using a delayed state Kalman filter. This technique has been shown to produce more accurate maps than Dead Reckoning (DR) navigation alone or LBL filtered navigation. However it cannot address errors within the individual submaps that link together to form the overall map. In addition, the tradeoffs in complexity, accuracy and matching performance based on the size and number of submaps are only partially understood.

The approach presented in this thesis, hereby named Bathymetric Particle SLAM (BP-SLAM), also uses a RBPF to account for the uncertainty in the vehicle's navigation solution and its effect on map-making, where particle resampling is performed based on the self-consistency of each particle's map. This acts to reduce the uncertainty in the trajectory upon re-observing previously explored terrain.

2.4 Summary

This chapter has presented the current state of research in SLAM applicable to bathymetric mapping. Feature-Based methods based on Bayesian frameworks such as the EKF and EIF were investigated, in addition to techniques that keep such approaches tractable when dealing with large sets of features, such as sparse representations. The Rao-Blackwellized Particle Filter was also introduced, a non-parametric version of the Bayes filter which offers desirable computational characteristics by exploiting the conditional independence of the features being tracked. This was followed by a discussion into GraphSLAM, demonstrating how this least squares optimisation approach can yield improvements to solving the full SLAM problem, at the cost of moving to an offline implementation.

The problem of reliable detection, association and tracking of features in an unstructured environment was then discussed, motivating the use of a featureless method of SLAM that does not require these actions to be performed. Following this the foundations of featureless SLAM research were presented. Lastly the current state of the art methods of featureless bathymetric SLAM were discussed, reporting the strengths and weaknesses of each which motivate the new form of SLAM that is presented here.

Chapter 3

RBPF SLAM

3.1 Introduction

Particle filters are a nonparametric implementation of the Bayes filter that can be used to approximate the probability distribution of a state (not necessarily Gaussian) by a set of state hypotheses sampled from the distribution (see Thrun et al. (2005) for an extensive review). Rao-Blackwellized particle filters provide a framework for conducting SLAM by additionally providing each particle with its own map to build (Doucet et al. 2000). The general framework for the RBPF SLAM algorithm with a particle set size (N) is given in Algorithm 1.

As each particle hypothesis can be treated as a concrete instantiation of the true state the associated uncertainty in navigation can be removed entirely from the data association problem, allowing the procedure to simplify to a deterministic referencing of observations into the map. This is a valid approximation when the accuracy of the mapping sensor is significantly greater than that of the navigation, as is the case here.

For a straightforward implementation, using RBPF SLAM for bathymetric mapping would require entire maps to be copied and destroyed each time a particle is resampled, thereby becoming computationally expensive for large numbers of particles. Distributed Particle Mapping (DPM) (Eliazar & Parr 2004, Eliazar & Parr 2005) addresses this issue by efficiently maintaining a joint distribution over maps and robot poses. The key idea behind DPM is to retain the original particle's map and have any new particles (*children*) that are

Algorithm 1 RBPF SLAM Framework

```
1: **Initialise** $N$ particles with poses sampled from some initial distribution and a map
   with prior information that may exist about the world.
2: **repeat**
3:     **Receive** new observation.
4:     **for** $i = 1$ to $N$ **do**
5:         **Propagate** each particle pose to the time of the new observation by sampling
           from the vehicle motion model.
6:         **Weight** each particle based on how well the new observation agrees with its
           map.
7:     **end for**
8:     **Resample** the $N$ particles from the current set with replacement. Perform this
       based on the particle weights so that particles with low weights are likely to be
       discarded while particles with high weights are likely to be duplicated.
9:     **Update** the $N$ maps of the new particle set with the new observation.
10: **until** end of mission
11: **Select** the best surviving particle and corresponding map.
```

resampled from the original particle (*parent*) point to the parent's map rather than copy the map themselves. Extracting a particle's map is then achieved by examining its estimates as well as those inherited by it through the *particle ancestry*.

An example of this filter structure with a general map is shown in Figure 3.1. Here the bottom level of the tree represents the current particle set tracking the vehicle state, where incoming observations are used to build each of their respective maps. Each particle maintains a map generated using the observations received after it was created (which occurs during resampling) and so does not encompass the whole mission. However by tracing back through the particle's ancestry its full map can be reconstructed.

Particles that are not resampled are removed from the ancestry tree along with their map elements. In addition particles that do not possess any siblings are merged with their parent without any loss of information. 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, thus requiring itself to also be merged or removed respectively. In doing so the number of ancestor particles is guaranteed never to exceed $(2N - 1)$ (Eliazar & Parr 2004). Pruning the particles and maps in this manner reduces the asymptotic complexity of DPM to constant/linear (amortized) time per iteration of the filter, keeping it an efficient means of performing featureless SLAM.

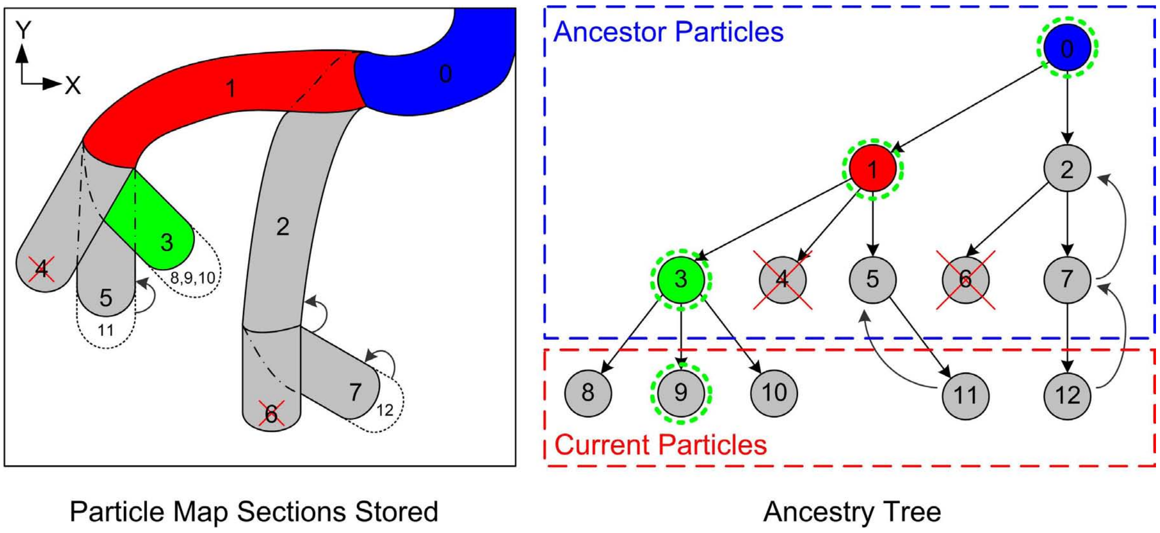


Figure 3.1: An example of the ancestry tree structure used in DPM. Here the tree is undergoing a resampling event that has allowed Particles 5 and 7 to survive and Particle 3 to triplicate. These resampled particles are given new IDs and form the current particle set, indicated by the bottom layer of the tree. Particles that are not resampled (indicated with a cross) or only possess one child (indicated by the curved arrows) are discarded and merged respectively, along with their map sections. While the particles in the new set have yet to create a map section of their own, each inherits the map sections of its ancestors. This is shown for Particle 9, reconstructing its full map by concatenating the map sections of the particles along its lineage (circled with a broken line). Note that in this example Particle 0 is also called the *root particle*, as any map sections stored by this particle are available to all particles in the current set, due to the common lineage.

This concludes a general description of the RBPF SLAM algorithm using DPM. What follows is the specific frame geometry, particle structure and RBPF procedures used to apply RBPF SLAM to bathymetric mapping.

3.2 Vehicle and Sensor Setup

Figure 3.2 presents the frames of reference used to define the vehicle and the placement of its sensors, as well as the geometrical relationship between range observations and their global coordinates. Note that while a specific vehicle is shown the following generalises to other vehicle designs as well.

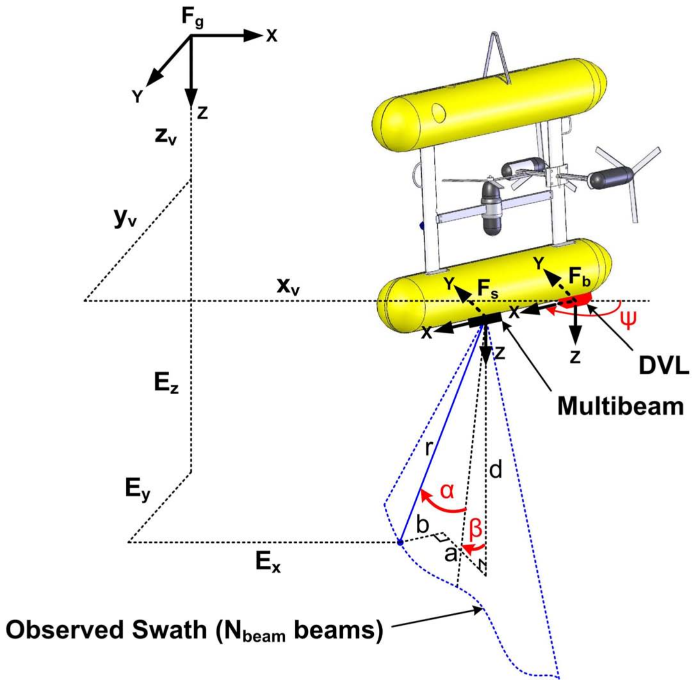


Figure 3.2: Relationship between a multibeam observation of range (r), bearing (α), along track angle (β), the state hypothesis ($\vec{\mathbf{x}}_v(t_k)$) and the location of the seabed patch observed ($\mathbf{E}$)

Observations from the multibeam sensor are received in the form of range observations (r) for a given along track angle (β) and bearing (α). Relative to the multibeam sensor frame

$\mathbf{F_s}$ the coordinates of these observations are given as:

$$^{s}P_{obs} = \begin{pmatrix} b \\ a \\ d \end{pmatrix} = \begin{pmatrix} rcos(\alpha)sin(\beta) \\ rsin(\alpha) \\ rcos(\alpha)cos(\beta) \end{pmatrix} \tag{3.1}$$

where b, a and d are referred to as the along track, across track and depth of the observation respectively. For this application the body frame $\mathbf{F_b}$ is used to represent the vehicle's pose, defined here as the location and orientation of the main navigation sensor i.e. the DVL. $^{s}P_{obs}$ can then be transformed into the body frame through the homogenous transformation $^{b}_{s}\boldsymbol{T}$, which specifies the fixed translational and angular offsets of the sensor frame relative to the body frame. To calculate an observation's location ($\mathbf{E}$) in the global frame $\mathbf{F_g}$ (with XYZ defined along North/East/Down) $^{s}P_{obs}$ is additionally multiplied by $^{g}_{b}\boldsymbol{T}$, calculated from the current pose of the vehicle:

$$\begin{pmatrix} E_x \\ E_y \\ E_z \\ 1 \end{pmatrix} = {}^{g}_{b}\boldsymbol{T}{}^{b}_{s}\boldsymbol{T} \begin{pmatrix} ^{s}P_{obs} \\ 1 \end{pmatrix} \tag{3.2}$$

The origin of the global frame can be set to zero or to a georeferenced location, depending on whether a form of absolute positioning is available at the start of the mission, e.g. GPS.

3.3 Particle Structure

To represent the vehicle state we adopt a similar approach to that advocated by Fairfield et al. (2006), in which states that are directly observable using the vehicle's sensors are removed from the particle filter and tracked instead with a single EKF that is shared by all particles. In this thesis we assume that observations of body velocity, attitude and depth (but not x, y position) are available to the vehicle such that the state vector tracked by the EKF is given as:

$$\mathbf{x}_v(t_k) = [^{g}z_v(t_k), ^{g}\boldsymbol{\psi}_v(t_k), ^{b}\mathbf{v}_v(t_k), \dot{\boldsymbol{\psi}}_v(t_k)] \tag{3.3}$$

where $^{g}z_v(t_k), ^{g}\boldsymbol{\psi}_v(t_k), ^{b}\dot{\mathbf{v}}_v(t_k), \dot{\boldsymbol{\psi}}_v(t_k)$ is the respective depth, attitude, body velocity and angular velocity (in terms of Euler angles) of the vehicle. Here the superscript g denotes states that are relative to the global frame and b for those relative to the vehicle's body frame.

Although this is an approximation and decouples the estimation of the vehicle position to some degree from the estimation of attitude and velocity, in practice we find that the observations available with our sensor suite have low noise and that the dynamics of the vehicles presented here, which are relatively slow moving and stable in roll and pitch, are well suited to decoupling the estimation in this manner. The results presented in Chapters 4 and 5 serve to validate this approach. This allows us to use the particle filter to track the x, y position of the vehicle, leading us to define the particle set $\mathbf{S}(t_k)$:

$$\mathbf{S}(t_k) = \begin{pmatrix} {}^g x_1(t_k) & ... & {}^g x_N(t_k) \\ {}^g y_1(t_k) & ... & {}^g y_N(t_k) \\ p_{ID_1} & \cdots & p_{ID_N} \end{pmatrix} \tag{3.4}$$

where $[{}^g x_i(t_k), {}^g y_i(t_k)]$ is the hypothesized vehicle state for the ith particle stored in the set and p_{ID_i} is the particle's identification number.

This setup also assumes that the ocean's surface remains at constant height during the period of the survey. For the missions presented in this thesis this remains a valid approximation. However in regions where tidal influences do cause significant fluctuations in sea level the corresponding tidal bias must be corrected for. This can easily be accomplished by including the tidal bias as an additional state in $\mathbf{x}_v(t_k)$ if local tide gauge observations are available. Alternatively if these are not available the tidal bias can be tracked as a particle state in $\mathbf{S}(t_k)$, thereby allowing particle weighting/resampling to additionally learn the tidal bias that creates the most self-consistent map.

3.4 Filter Initialisation

RBPF SLAM begins by initialising each particle with an estimate of the vehicle state relative to F_g (which can be local or georeferenced). The initial probability distribution describing the vehicle state is encoded into the RBPF by randomly sampling N different initial state hypotheses from the distribution of the initial estimate.

The RBPF can also be initialised with prior map information if available. Normally this would involve initialising each particle's map with the prior map information. However by utilising Distributed Particle Mapping this step can be equivalently carried out by initialising only one particle map, that belonging to the ancestor particle located at the base of the ancestry tree, which we call the *root particle* (See Figure 3.1). Since each current particle shares this common ancestor they all automatically inherit this common map containing the prior map information, allowing them to treat it as previously explored terrain and thereby use it to resample $\mathbf{S}(t_k)$, which in turn provides corrections to navigation. However the RBPF can only be initialised in this way if the prior map information is treated as a rigid map with no uncertainty in x, y. If the prior map information is instead another set of navigation and bathymetric logs then these can be used by concatenating them to the front of the respective logs from the current mission. Provided that the state hypotheses held by $\mathbf{S}(t_k)$ are reinitialised each time a new mission is started this allows for the merging of several overlapping datasets.

By supplying a prior map RBPF SLAM solves the prior map localisation problem, with the added benefit of being able to additionally localise off the current map as its being built. However the success of this approach is dependent on the map representation used by the RBPF and the ease to which prior map information (which can be of different resolutions) can be entered into it. As such this step will be discussed further in Sections 4.2.2 and 5.3.2 in terms of how it applies to the two different map representations presented in this thesis.

3.5 Particle Propagation

Particle propagation requires that the vehicle states held in both the particle and EKF sections of the filter be predicted forward to the time of the next observation. This is done using a discrete vehicle model of the form:

$$\dot{\mathbf{x}}_v(t_k) = \boldsymbol{F}_v(t_k)\mathbf{x}_v(t_k) + \boldsymbol{G}_v(t_k)\boldsymbol{w}_v(t_k) \tag{3.5}$$

where $\boldsymbol{F}_v(t_k)$ is the state transition matrix, $\boldsymbol{G}_v(t_k)$ is the noise transition matrix and $\boldsymbol{w}_v(t_k)$ is the vector of vehicle model errors.

Assuming that navigation is, for the most part, carried out with a constant speed as is typical for bathymetric mapping missions (Grasmueck, Eberli, Viggiano & Correa 2006), we can approximate the dynamics of the vehicle with a simple constant velocity/rotation rate model. Equation 3.5 can then be written explicitly as:

$$\begin{pmatrix} {}^{g}z_v(t_k) \\ {}^{g}\boldsymbol{\psi}_v(t_k) \\ {}^{b}\mathbf{v}_v(t_k) \\ \boldsymbol{\omega}_v(t_k) \end{pmatrix} = \begin{pmatrix} {}^{g}z_v(t_{k-1}) + {}^{g}_{b}\boldsymbol{R}_{(\mathbf{3},:)}(t_{k-1})^{b}\mathbf{v}_v(t_{k-1})\Delta t \\ {}^{g}\boldsymbol{\psi}_v(t_{k-1}) + {}^{g}_{b}\boldsymbol{E}(t_{k-1})\boldsymbol{\omega}_v(t_{k-1})\Delta t \\ {}^{b}\mathbf{v}_v(t_{k-1}) \\ \boldsymbol{\omega}_v(t_{k-1}) \end{pmatrix} + \begin{pmatrix} {}^{g}_{b}\boldsymbol{R}_{(\mathbf{3},:)}(t_{k-1})\mathbf{w}_{\dot{\mathbf{V}}}(t_k)\frac{\Delta t^2}{2} \\ {}^{g}_{b}\boldsymbol{E}(t_{k-1})\mathbf{w}_{\dot{\boldsymbol{\omega}}}(t_k)\frac{\Delta t^2}{2} \\ \mathbf{w}_{\dot{\mathbf{V}}}\Delta t \\ \mathbf{w}_{\dot{\boldsymbol{\omega}}}\Delta t \end{pmatrix} \tag{3.6}$$

where $\mathbf{w}_{\dot{\mathbf{V}}}(t_k)$ and $\mathbf{w}_{\dot{\boldsymbol{\omega}}}(t_k)$ represent translational and angular acceleration disturbances to the model at time t_k.

The particle set is also predicted forward, except in this case the uncertainty associated with the prediction step is encoded into $\mathbf{S}(t_k)$ by each particle randomly sampling a hypothesis from the distribution of $\mathbf{x}_v(t_{k-1})$ to base its prediction on. For the current setup this corresponds to:

$$\begin{aligned} \begin{pmatrix} {}^{g}x_{v_i}(t_k) \\ {}^{g}y_{v_i}(t_k) \end{pmatrix} = \begin{pmatrix} {}^{g}x_{v_i}(t_{k-1}) \\ {}^{g}y_{v_i}(t_{k-1}) \end{pmatrix} &+ \left({}^{g}_{b}\boldsymbol{R}_{(\mathbf{1:2},:)}(t_{k-1})\boldsymbol{\kappa}\Delta t \right) \\ &+ \left({}^{g}_{b}\boldsymbol{R}_{(\mathbf{1:2},:)}(t_{k-1})\mathbf{w}_{\dot{\mathbf{V}}}(t_k)\frac{\Delta t^2}{2} \right) \end{aligned} \tag{3.7}$$

where $\boldsymbol{\kappa} \sim \mathcal{N}({}^{b}\mathbf{v}_v(t_{k-1}), \boldsymbol{\sigma}^2_{\mathbf{V}}(t_{k-1}))$ and ${}^{g}_{b}\boldsymbol{R}_{(\mathbf{1:2},:)}$ are constructed from an attitude hypothesis randomly sampled from $\mathcal{N}({}^{g}\boldsymbol{\psi}_v(t_{k-1}), \boldsymbol{\sigma}^2_{\boldsymbol{\psi}}(t_{k-1}))$.

3.6 Particle Weighting

For observations of states contained in $\mathbf{x}_v(t_k)$ the standard EKF prediction/update equations can be used to incorporate new information into the filter. For a thorough description of the simple measurement models used to update the EKF see Mahon (2007).

However for observations that infer information about the states tracked in $\mathbf{S}(t_k)$, such as those from the mapping sensor, particle weighting and resampling must be carried out to achieve the same goal.

This is done by calculating a weight for each particle that measures the likelihood of receiving the new observation, given the particles previous trajectory and observations. i.e. $w_i = p(\mathbf{z}(t_{k+1})|S_i(t_0 : t_k), \mathbf{z}(t_0 : t_k))$. The set of weights produced thereby represent an approximation to the Bayes filter posterior distribution, by which $\mathbf{S}(t_k)$ is shifted towards through resampling its particles based on these weights. In the context of bathymetric SLAM this corresponds to weighting each particle by the likelihood that the current multibeam swath would be observed based on its map. The calculation of this likelihood is therefore heavily influenced by how the map is represented, later discussed in Chapters 4 and 5.

3.7 Particle Resampling

Once particle weighting has been completed the weights are grouped together and normalized. The particles are then randomly sampled to form the new particle set, ensuring that their probability of being resampled remains directly proportional to their normalized likelihood (Gordon, Salmond & Smith 1993). This allows the particles that are most likely to be an accurate hypothesis to propagate while removing those which are unlikely.

Excessive resampling however can often lead to particle depletion, i.e. the premature removal of particles with maps that could end up being the most self-consistent. This is particularly true if resampling is carried out on a set containing similarly weighted particles. In such a case the most likely state hypotheses cannot be discerned and so resampling would result in a purely random draw of particles from the set, thereby running the risk of removing potentially good particles.

To avoid this, resampling is prevented if the *effective particle size*, which provides a measure of the variation in the weights, is greater than half the number of particles (Liu 1996). This is calculated as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}(i))^2} \tag{3.8}$$

where $\tilde{w}(i)$ refers to the normalized weight of particle i.

3.8 Particle Map Updating

Once particle weighting has been completed each particle updates its map with the new multibeam swath observations. This is normally done after particle resampling (as only the surviving particle maps need updating) but can also be done before particle resampling if this helps reduce the number of map queries required. Again this procedure is completely dependent on the map representation and so is discussed in Chapters 4 and 5.

3.9 Final Particle and Map Selection

At the end of the mission the particle filter provides N possible trajectories and corresponding maps to choose from. For particle filters that are only performing pure localisation the problem of determining the best state hypothesis from $\mathbf{S}(t_k)$ reduces to finding the particle with the maximum likelihood at the end of the mission (for an EKF filter this is equivalent to choosing the mean estimate μ_t). However as we are performing full SLAM this method does not ensure the best result, as the particle with the maximum likelihood at the end of the mission does not necessarily attain the maximum likelihood for past poses as well, which are just as important in determining the quality of the map produced.

To this end we choose the best particle from $\mathbf{S}(t_k)$ by analyzing each particle's map and identifying which is the most self-consistent. This is determined, assuming that the bathymetric data has been processed to remove outliers, using an offline method described by Roman & Singh (2006). This method involves splitting the bathymetry into a sequence of submaps and then calculating the maximum registration error between submap portions located in the same cell, as described in Algorithm 2. These registration errors can then be used to determine the self-consistency of the map, provided that map overlap occurs.

Algorithm 2 Final Map Selection

```
1: for $i = 1$ to $N$ do
2:     Extract the $i$th particle's trajectory from $\mathbf{S}(t_0 : t_{end})$.
3:     Segment the trajectory and start a new submap every time the multibeam swath
       is about to overlap (in the $z$ direction) with previously explored terrain.
4:     Transform all submaps into 3D space.
5:     Grid the area and for each cell calculate the minimum difference in depth between
       all submaps present in that cell.
6:     Set the maximum of these measurements as the registration error of the map for
       that given cell.
7: end for
```

An example of calculating this consistency metric over a 2D slice of terrain observed several times is shown in Figure 3.3. While this procedure involves picking points at random from

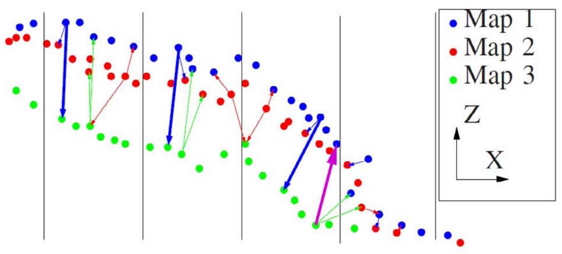


Figure 3.3: An example of calculating the map self consistency in different cells using the metric developed by Roman & Singh (2006) (Image source). The vertical lines represent the boundaries of each bin that are populated by depth observations from three different submaps. In each cell a point from each submap is randomly chosen and matched with the closest point belonging to each of the other submaps, either in the same or neighboring cell (represented by the thin arrows). The maximum registration error between these matches is then returned as the map self-consistency metric for that cell (shown in each cell by the bold arrow). Note that bin size related biases in this calculation are avoided by including the immediate neighboring cells when finding the closest match. This is demonstrated by the magenta arrow that shows how the closest green to blue map pairing is incorrectly chosen when neighboring cells are not considered. Note also that the right most bin does not have any Map 3 pairings as these are only included for consideration when all surrounding bins contain Map 3 points.

each submap, the variance associated by doing this is quickly reduced by performing several instances of this calculation.

As each particle has a different trajectory the amount of swath overlap in each particle's map will be different. Consequently the number of registration error measures available to each particle varies. Given that N_{errors} is the smallest number of registration error measures available to a particle in the final set, we calculate a single measure of self-consistency for each particle's map by taking the average of the N_{errors} largest registration errors present in its grid cells, which prevents this average being biased towards particles with more overlap. The trajectory and corresponding map of whichever particle achieves the lowest error measure is then returned by the filter. Note that this metric assumes that a single large registration error is just as preferable as numerous small registration errors with the same total error. Altering this depending on the requirements of the user is a subject of future work.

This Final Particle Selection (FPS) scheme is only available offline and requires that swath overlap exists in each particle's map. For missions with minimal overlap (as can be attempted with the trajectory map approach) an online FPS scheme is instead used that progressively builds histograms of the observation/estimate differences encountered by each particle during the particle weighting stage. Similar to the offline method the average of the N_{errors} present in each particle's histogram can then be used to estimate the best particle trajectory and map. This is less accurate than the offline method as it only compares the registration error between two submaps, one created by the current observed swath and the other that is a fusion of all previous overlapping swaths. However the benefit of the online FPS scheme is that it can use extrapolated predictions and has the potential to aid in path planning decisions should this algorithm be implemented in real time.

Once the final particle is selected the final map can also be obtained. For online applications this can be done quickly as the final particle's map is accessible from BPSLAM. However, for a more dense and non-discretised version the map can alternatively be reconstructed as a point cloud by georeferencing the raw bathymetry using the final particle's trajectory. This is the method that will be used to generate the maps shown in Chapters 4 and 5.

3.10 Summary

This chapter has presented the underlying mechanics of the particle filter, both in general terms and when applied to bathymetric mapping. For the BPSLAM algorithm the states most prone to drift (x, y) are tracked in the particle filter, while those that are directly observable are tracked with a common EKF accessed by all particles.

The state transition model was also presented, a simple constant velocity/rotation rate model that is used to propagate both RBPF and EKF sections of the filter. This was then followed by a description of the general premise behind particle weighting, map updating and particle resampling, the specific details of the former two presented in the proceeding chapters.

Lastly two FPS schemes were presented, an offline method that explicitly determines the particle with the most self-consistent map and an online method that makes an educated guess of this based on the differences between the observations and estimates encountered during the particle weighting stage.

Chapter 4

Bathymetric Particle Filter SLAM Using Grid Maps

4.1 Introduction

The approach presented in this chapter utilizes the RBPF SLAM technique described in Chapter 3 as its foundation. As each particle represents a deterministic hypothesis of the true location of the vehicle, standard mapping techniques can be used to properly maintain each particle's local map of the environment. Several map representations exist that can capture the 3D structure of an underwater environment. Of these, one of the simplest is to represent the map as a cloud of 3D points.

Alternatively grid based map representations are available which allow for relatively faster map access at the cost of losing resolution through discretisation. In this case to capture a fully 3D environment (e.g a subterranean cave system) a 3D grid must be used, each cell containing a probability of occupancy. However by restricting the scope of operations to marine environments that are approximately 2.5D (as is the case for the majority of the seafloor) the mapping requirement can be relaxed to a digital elevation map, i.e. a 2D grid where each cell contains an estimate and uncertainty in depth. The memory requirements of the map then scale quadratically as either the size or resolution is increased. For the area and resolution of the datasets we wish to map (see Section 4.3) this provides a tractable option. In addition the 2D depth grid map representation allows for direct prior map input and new map output in a format which is standard in the marine surveying industry. To process even larger datasets the implementation of quadtrees may become necessary to handle the growing memory requirements (Fairfield et al. 2006), though this will increase the complexity of the map querying and updating operations.

So far only a general description of the stages in the RBPF SLAM framework involving map operations i.e. the map weighting and map updating procedures, has been given (see Algorithm 1). This was done as the RBPF setup allows for an interchangeable map representation, on which these two procedures are dependent. The remainder of this chapter now goes on to describe the grid map representation and the manner in which it carries out these two tasks. Two case studies are then presented which validate this approach using real mission data.

4.2 Method

4.2.1 Map Structure

Instead of each particle adding/updating estimates in its own grid, the estimates are first keyed with the particle's ID and 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. For this reason the maximum memory used by this approach can sometimes rival the simpler approach of providing each particle with its own private copy of the map. However the latter requires entire maps to be copied whenever it is resampled. This creates $O(NG)$ work per iteration, where G is the number of grid cells in the map, thereby becoming inefficient for large maps and large numbers of particles.

Each particle maintains and updates its estimates of seabed depth stored in the grid map using a 1D EIF, where each estimate is represented by an information vector ξ and an information matrix Ω. 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.

The list of depth estimates in a cell is referred to as the *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. Each particle also retains the following information in the ancestry tree of the particle filter:

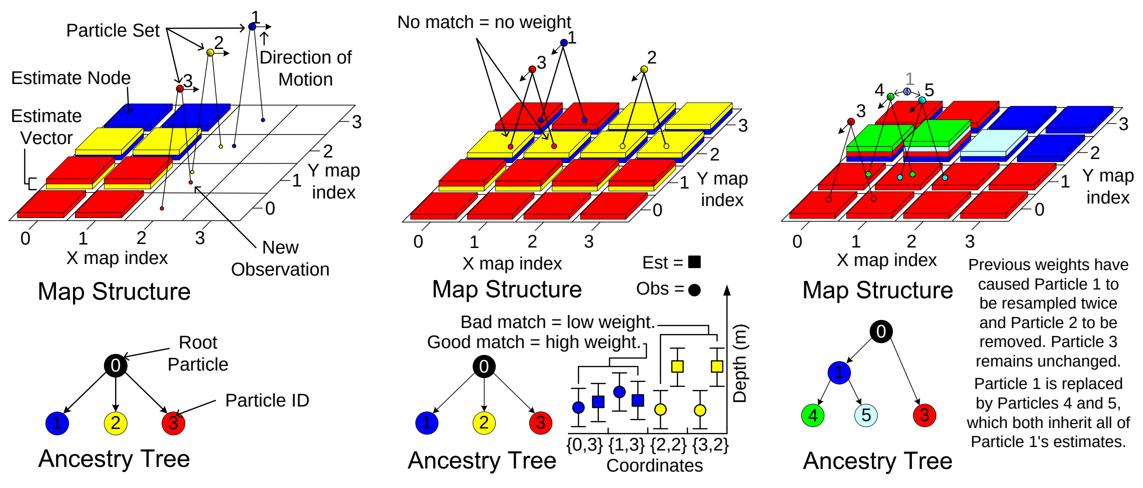
- **Parent ID** - The ID of the parent that the particle was resampled from.
- **Child List** - A list of all the particle's children.
- **Estimates** - A list of all (x,y) locations where the particle made/updated an estimate of seabed depth.

Here each particle maintains a list of the estimates it has made. This is done purely to facilitate their removal from the map structure should the owner not survive the resampling procedure, thus preventing the need to query each cell.

Figure 4.1 provides an example of the map representation and associated ancestry tree used to maintain and extract the maps monitored by each particle. In Figure 4.1a) the vehicle moves over an unexplored patch of seabed from left to right (shown mid-progress). Each particle uses its observations to make its own initial estimates of seabed depth (the estimate nodes are shown with finite thickness for clarity). In Figure 4.1b) the vehicle returns some time later from a different direction and encounters the same seabed patch. Each particle searches the cells they are currently observing for a previous depth estimate belonging to them. If one exists a weighting for the particle, based on the consistency between the new observations and previous estimates of seabed depth, is calculated. These previous estimates are then updated with the new observations using an EIF[1]. In Figure 4.1c) the particle set is shown two timesteps later. Resampling triggered by the weights calculated in 4.1b) has

[1]The maps are updated here (as opposed to after particle resampling) as it requires no extra map accessing operations. This does not effect the particle weights for this iteration as they are calculated before the maps are updated.

caused Particle 1 to be resampled twice while removing Particle 2 from the set. The two samples are renamed Particles 4 and 5 and inherit the map estimates of Particle 1. In the next timestep Particles 4 and 5 are randomly propagated and take on different positions, as shown.



(a) Particles forming initial depth estimates.

(b) Particle weighting procedure upon reobserving seabed.

(c) Particle resampling that results from weighting.

Figure 4.1: An example of the map structure and ancestry tree used to store estimates from different particles with a particle set size of N = 3 and a sonar swath of two observations.

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 if $\mathbf{S}(t_k)$ is progressively saved as the filter propagates, along with each particle's (or parent particle's) index in $\mathbf{S}(t_{k-1})$. In this way the trajectory of a particle can be backtraced through the poses that were held previously by the particle set.

4.2.2 Map Initialisation

As we use a 2D grid structure for our map representation it is also relatively straightforward to initialize these grid cells with depth estimates from a prior map. All that is required is to enter each prior depth estimate, along with its uncertainty, into the grid cell at its x, y location and key it with the ID number of the root particle (see Figure 4.1). If the prior map is of higher resolution than the BPSLAM grid map then multiple initial depth estimates will be available per grid map cell. These are combined into a single depth estimate using the standard EIF update equations (Thrun et al. 2005). As with the estimates that are generated from the current bathymetry this map discretisation limits the precision of the position corrections to the resolution of the grid map.

Prior maps with resolution lower than the BPSLAM grid map structure can also be used to provide corrections to navigation. However in this case attempting prior map localisation will require that particle resampling be allowed when only a fraction of the incoming multi-beam observations (which are assumed to have resolution comparable to the grid map) can be matched to a prior map estimate (as these are relatively sparser). The effect that this allowable fraction, which we refer to as $\gamma_{overlap}$, has on particle resampling is discussed in the next section.

4.2.3 Map Weighting

As mentioned previously, the map weighting stage judges each particle by the likelihood that the current multibeam swath would be observed, given the particle's current map. Algorithm 3 details the steps involved to calculate this.

Algorithm 3 Step 6 from Algorithm 1:Grid Map Weighting

```
Require: Combined particle/EKF state hypothesis $\vec{\mathbf{x}}_{v_i}(t_k)$, Observations $\mathbf{z}$, Map $i$
 1: $ctr = 0$
 2: $B_{min} = N_{beams} * \gamma_{overlap}$
 3: for $b = 1$ to $N_{beams}$ do
 4:     Transform $\mathbf{z}_b$ into coordinates $[\mu_{E_{x.obs}}, \mu_{E_{y.obs}}, \mu_{E_{z.obs}}]$ relative to global frame.
 5:     Access most recent depth estimate $E_{z.est}$ at cell $[\mu_{E_{x.obs}}, \mu_{E_{y.obs}}]$ that belongs to
        current particle or any of its ancestors.
 6:     if ($E_{z.est}$ exists) then
 7:         Calculate the likelihood of $E_{z.obs}$ matching the current estimate:
            $lklhd(ctr) = p(E_{z.obs} - E_{z.est} = 0)$.
 8:         Increment $ctr$.
 9:     end if
10: end for
11: if ($ctr \geq B_{min}$) then
12:     return Joint Likelihood $w_i = \prod_{j=1}^{B_{min}} lklhd(j)$
13: else
14:     return Do not include particle in resampling.
15: end if
```

As shown the observations of range must first be transformed into the global frame. This is calculated using Equations 3.1 and 3.2. In addition a Markovian observation model is used to define the relationship between $\mathbf{z}$ and the depth estimates stored in the map:

$$\mathbf{z} = \begin{pmatrix} r & \alpha & \beta \end{pmatrix}^T = h(\vec{\mathbf{x}}_v(t_k), \mathbf{E}) + \mathbf{v} \tag{4.1}$$

where $\mathbf{h}$ is the measurement function, $\vec{\mathbf{x}}_v(t_k)$ is the combined state hypothesis/shared EKF state of the current particle, $\mathbf{E} = [E_x, E_y, E_z]$ is the location of the seabed patch being observed and $\mathbf{v}$ is zero mean Gaussian noise associated with the observations with covariance:

$$R = diag(\sigma_r^2, \sigma_\alpha^2, \sigma_\beta^2) \tag{4.2}$$

Note that along track angle observations are not provided by the sensor but rather modeled as 0° with an uncertainty that arises from the beamwidth of the sonar aperture. Similarly the bearing observations are also subject to a finite uncertainty.

The measurement function $\mathbf{h}$, and its corresponding Jacobian $\nabla_x \mathbf{h}$, are formulated from the geometrical relationship between $\mathbf{E}$, $\vec{\mathbf{x}}_v(t_k)$ and the expected observation $\hat{\mathbf{z}}$, shown previously in Figure 3.2:

$$\hat{\mathbf{z}} = \mathbf{h}(\vec{\mathbf{x}}_v(t_k), \mathbf{E}) = \begin{pmatrix} \sqrt{b^2 + a^2 + d^2} & \arctan(\frac{a}{d}) & \arctan(\frac{b}{d}) \end{pmatrix}^T \tag{4.3}$$

where b, a and d are derived from $\mathbf{E}$ and $\vec{\mathbf{x}}_v(t_k)$ as:

$$\begin{pmatrix} b \\ a \\ d \end{pmatrix} = {}^s_g\boldsymbol{R} \begin{pmatrix} E_x - x_v(t_k) \\ E_y - y_v(t_k) \\ E_z - z_v(t_k) \end{pmatrix} \tag{4.4}$$

where ${}^s_g\boldsymbol{R}$ is the global to sensor frame Directional Cosine Matrix.

With the observation model defined $\mathbf{z}$ can be used to calculate an equivalent depth observation with mean $\mu_{E_{z.obs}}$. The uncertainty of the observation in this frame $\sigma^2_{E_{z.obs}}$ can also be calculated as a backwards transport (Hartley & Zisserman 2003) of the covariance R through $\mathbf{h}$:

$${\sigma_{E_{z.obs}}}^2 = (\nabla_x \mathbf{h}^T R^{-1} \nabla_x \mathbf{h})^{-1} \tag{4.5}$$

This effectively approximates $E_{z.obs}$ as Gaussian by linearizing about the mean $\mu_{E_{z.obs}}$. The corresponding depth estimate at the observed location can also be quickly extracted from the EIF stored there:

$$\mu_{E_{z.est}(t-1)} = \Omega_{t-1}^{-1}\xi_{t-1} \tag{4.6}$$

$$\sigma^2_{E_{z.est}(t-1)} = \Omega_{t-1}^{-1} \tag{4.7}$$

A data association problem should be highlighted at this stage as, although each particle state hypothesis $[x_{v_i}(t_k), y_{v_i}(t_k)]$ can be treated as truth, the uncertainty in a sonar's range and angular observations will cause uncertainty in its corresponding x, y position, more so at large grazing angles and ranges. The depth observation made can therefore belong to any of the grid cells located within this region of uncertainty. Several data association techniques were investigated (Dezert & Bar-Shalom 1993, Bar-Shalom & Fortmann 1987) to try and take this into account but were found to be too computationally expensive for real time operation with large numbers of particles. Alternatively the observation could be used to update the depth estimate of every cell within this region of uncertainty, though this results in a blurring of the maps. Instead we directly associate the observation to the grid cell at the observation's mean x, y location and set the resolution of the grid to be on the scale of the sensor error in x, y. While this also blurs the map through discretisation it prevents the filter from producing overconfident depth estimates and therefore guards against particle depletion during resampling.

To determine the resolution the grid maps should be set at the 95% confidence bound in an observation's across track and along track coordinate (a_{error}, b_{error}) is analysed for the worst case scenario (maximum range, bearing and along track angle). This can be calculated from the corresponding confidence bounds in the raw observation, given as:

$$\begin{aligned} a_{error} &= 2[r_{max}cos(\alpha_{max})sin(2\sigma_\alpha) + 2\sigma_r sin(\alpha_{max})cos(2\sigma_\alpha)] \\ b_{error} &= 2[r_{max}cos(\beta_{max})sin(2\sigma_\beta) + 2\sigma_r sin(\beta_{max})cos(2\sigma_\beta)] \end{aligned} \tag{4.8}$$

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_{error} or b_{error}, whichever is largest. This quantity can thus be used to dictate a maximum resolution that ensures this error bound does not exceed the size of the grid cells, thereby being on the order of the map error already introduced by map discretisation. As will be shown in Section 4.3

this maximum resolution is more than adequate for the large mapping missions we wish to undertake.

Now that both observation $E_{z.obs}$ and estimate $E_{z.est}$ are in the same frame of reference the map can be weighted based on their difference. As both are modeled as Gaussian the probability distribution of their difference is also Gaussian. The likelihood that this difference is zero therefore provides a principled method by which the particle that made the observation can be weighted. Setting this difference to zero we obtain:

$$likelihood = p((E_{z.est} - E_{z.obs}) = 0) = \frac{e^{-\frac{1}{2}\frac{(\mu_{E_{z.est}} - \mu_{E_{z.obs}})^2}{\sigma^2_{E_{z.obs}} + \sigma^2_{E_{z.est}}}}}{\sqrt{2\pi(\sigma^2_{E_{z.obs}} + \sigma^2_{E_{z.est}})}} \tag{4.9}$$

Note that the likelihood measure described in Equation 4.9 can only be calculated when a prior estimate exists. If a particle has no prior estimates to match its observations to it is assumed that the particle is just as likely to be a good or bad estimate of the true state, and thus is not included in the resampling phase. Since each particle will often have a differing number of observation/estimate matches it is also withheld from the resampling phase if less than $\gamma_{overlap}$ percent of its observations are successful in matching to a prior estimate. If $\gamma_{overlap} = 100\%$ this would ensure that resampling only occurred when the multibeam swath fully overlapped with an area previously explored. Decreasing $\gamma_{overlap}$ will include more particles in the resampling phase (i.e. 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. This reduced threshold can be chosen based on the amount of overlap/loop closures expected during the mission. If L_{min} is the smallest number of likelihood measures belonging to a particle within the subset of those to be included in the resampling phase, then a single weighting factor can be generated for each of these particles by sampling L_{min} of the likelihood measures belonging to them and calculating the joint likelihood, i.e. the product of those likelihoods.

4.2.4 Map Updating

Once a particle has been weighted its map is updated with the new information provided by the observations. As the estimate of seabed depth contained within each grid map cell is tracked using a simple 1D EIF it can be updated with the standard EIF equations, based on the observation model described in Section 3.6. Furthermore if we assume a static environment, where the seafloor does not change with time over the period of a single survey, the prediction step of the EIF can be removed, simplifying the update equations to those shown in Algorithm 4:

Algorithm 4 Step 9 from Algorithm 1:Grid Map Updating

```
1: for $b = 1$ to $N_{beams}$ do
2:     Extract current estimate $(\xi_{t-1}, \Omega_{t-1})$ from observed cell.
3:     $\mu_{t-1} = \Omega_{t-1}^{-1}\xi_{t-1}$
4:     $\Omega_t = \Omega_{t-1} + \nabla_x \mathbf{h}_t^T \mathbf{R}_t^{-1} \nabla_x \mathbf{h}_t$
5:     $\xi_t = \xi_{t-1} + \nabla_x \mathbf{h}_t^T \mathbf{R}_t^{-1}[\mathbf{z}_b - h(\vec{\mathbf{x}}_v(t_k), \mu_{t-1}) + \nabla_x \mathbf{h}_t \mu_{t-1}]$
6:     Store new estimate.
7: end for
```

4.3 Results

In this section we present a thorough analysis of the BPSLAM filter running on the grid map representation for both short scale and long scale missions. All results presented in this thesis were processed using an Intel Xeon 3.00GHz CPU with 16GB of RAM, though the actual memory requirements of this approach often fall well below this capacity.

4.3.1 Case Study 1: Butts Reef Pockmarks

To begin with, the BPSLAM algorithm was tested on a small scale mission scenario using bathymetric and navigation logs from a real survey undertaken by our research class AUV *Sirius* (Williams et al. 2009), shown in Figure 4.2.

The survey was taken off the coast of Tasmania and contains several pockmarks 30 m in diameter and 3 m deep (on average). Two orthogonal grid transects were carried out underwater at an altitude of 20m. *Sirius* possesses inclinometers that provide the AUV with observations of roll and pitch throughout the mission. Depth observations are obtained through a high precision depth sensor whereas along track, across track and depth velocity observations are provided by a 1200 kHz DVL with ± 3 mms^{-1} accuracy. Note that this specification does not take into consideration errors in the speed of sound estimate used by the DVL. However *Sirius* also possesses a high precision conductivity and temperature

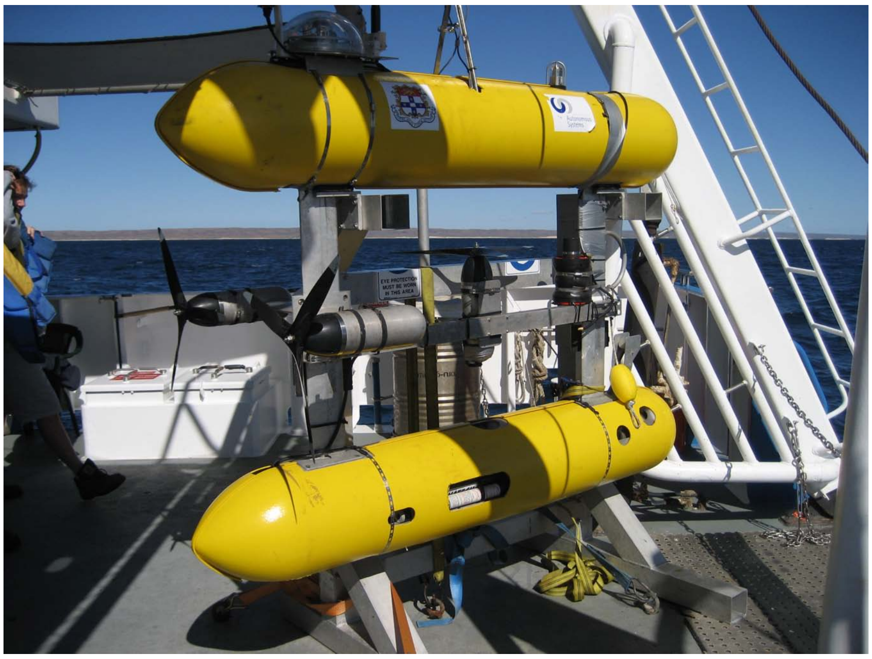


Figure 4.2: The Sirius AUV is a research vehicle capable of exploring the ocean down to 700 meters depth. Sirius's two torpedo shape promotes passive stability in pitch and roll, allowing for high resolution bathymetry and stereo imagery to be obtained.

sensor that it uses to calculate the speed of sound in the surrounding water. Therefore the precision stated earlier (provided by the manufacturer) remains a reasonable error model for our DVL velocity observations.

Heading estimates are received from a magnetic compass and as such can be prone to persistent heading-dependent errors of $\approx 1°$, arising from its sensitivity to the magnetic signature of the rest of the vehicle. Fortunately through an iterative calibration process the heading dependent bias in the compass observations was able to be modeled and corrected for (Jakuba, Williams & Pizarro 2010). This coupled with the design of *Sirius*, which promotes passive stability in pitch and roll, justifies the inclusion of the states x, y into the particle filter, as they are the most prone to drift, while leaving the remaining states to be tracked by the EKF.

USBL observations of the AUV's range and bearing relative to a support ship, along with a GPS fix of the ship, were also available and relayed to the AUV throughout the mission. These were then used as measurement updates in an EKF filter (separate to BPSLAM) to fuse this information into a dead reckoning based navigation solution. This navigation solution is not used by the BPSLAM filter but is reproduced here as a baseline to demonstrate the best possible map that could previously be produced, and how BPSLAM can improve upon these results.

The bathymetric sensor used by *Sirius* is an Imagenex Delta T 260 kHz multibeam profiling sonar, providing 120 beams uniformly across 120° with a 3° beamwidth in along track. This is taken into account when creating the particle maps, as the resolution of the map is limited by the resolution of the mapping sensor. While the spacing between observations on the seafloor is non-uniform the variance in the spacing is reduced when mapping is performed at constant altitude. As such the average separation provides a good measure by which to gauge an appropriate resolution for the grid maps. For this mission the resulting average beam separation in across track and along track was calculated as 0.726 m and 0.065 m respectively. Additionally we impose the maximum resolution approximated by Eq. 4.8, calculated as 0.486 m. Based on this requirement, the size of the survey (150 m by 300 m) and the length of the mission (113 minutes) a grid resolution of 1.0 m was chosen to demonstrate the BPSLAM filter's performance.

To reduce mapping errors induced by sensor misalignment between the multibeam and the vehicle, a standard calibration run was completed prior to the mission in which the AUV maintained position and rotated on the spot over a test target (Foote, Chu, Hammar, Baldwin, Mayer & Hufnagle 2005). The sensor offsets which minimised the discrepancy in the resulting bathymetry were then applied.

As this survey mission does several complete passes over previously explored terrain, the parameter $\gamma_{overlap}$ can be set to 100 % for resampling.

Navigation and Mapping Comparison

To ensure a good result we demonstrate the BPSLAM filter performance during the mission using the aforementioned parameters with a relatively large number of particles ($N = 640$). The corresponding navigation solution produced by the BPSLAM filter is shown in Figure 4.3 along with the vehicle position confidence in Figure 4.4. Figure 4.5 presents the multibeam data collected during the mission. The hand matched correspondences shown are not used in any way by the filter but are provided here for reference.

Without resampling the point cloud continues to spread out without bound as the uncertainty in the pose grows, as expected. The first resampling event occurs at $t = 73.75$ min

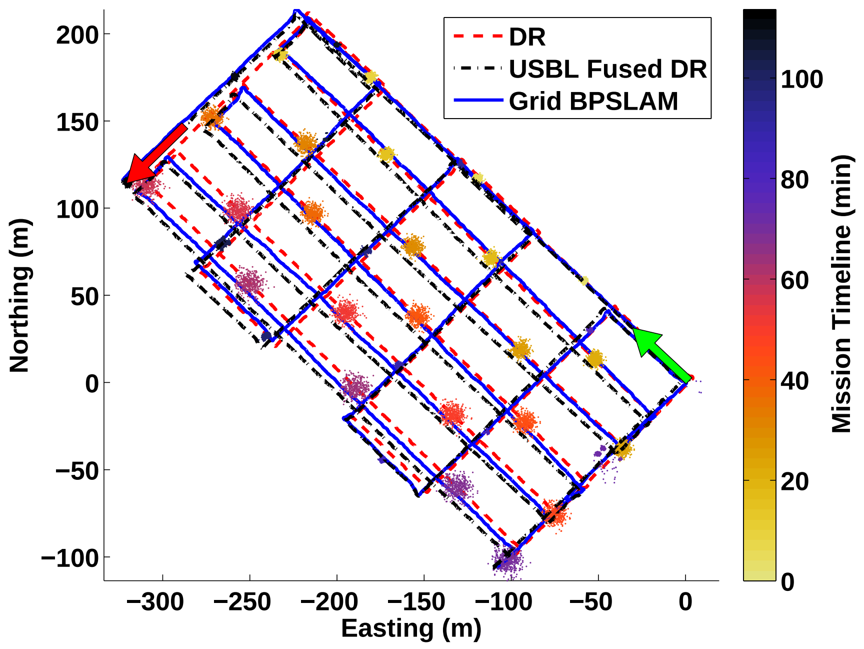


Figure 4.3: Tracklines produced by three different navigation solutions; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively (longest tracklines belong to the initial grid survey). The evolution of the particle cloud is also shown, changing from light to **dark** as the mission progresses.

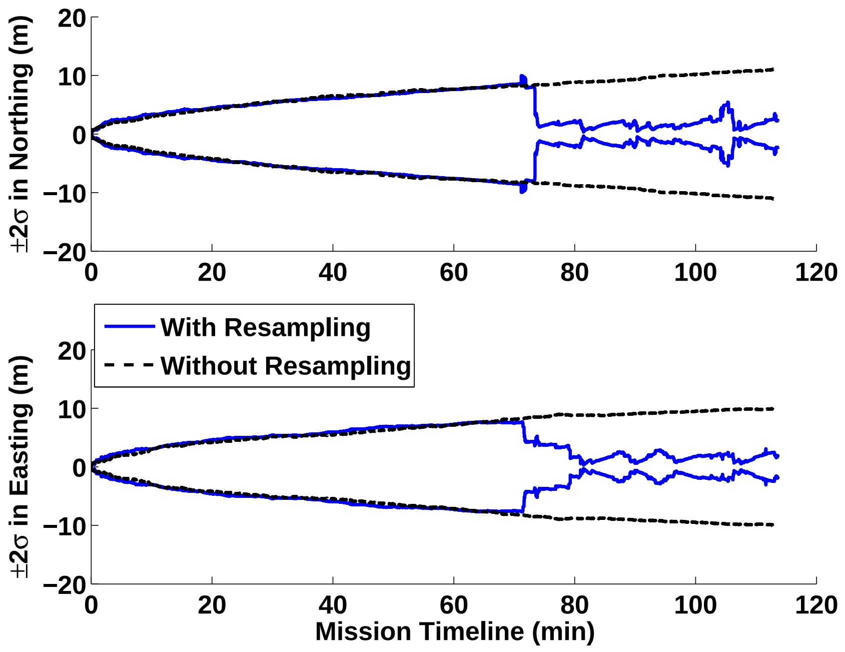


Figure 4.4: Variation in the 95% confidence interval of the vehicle's x, y position, with (solid line) and without (dotted line) particle resampling.

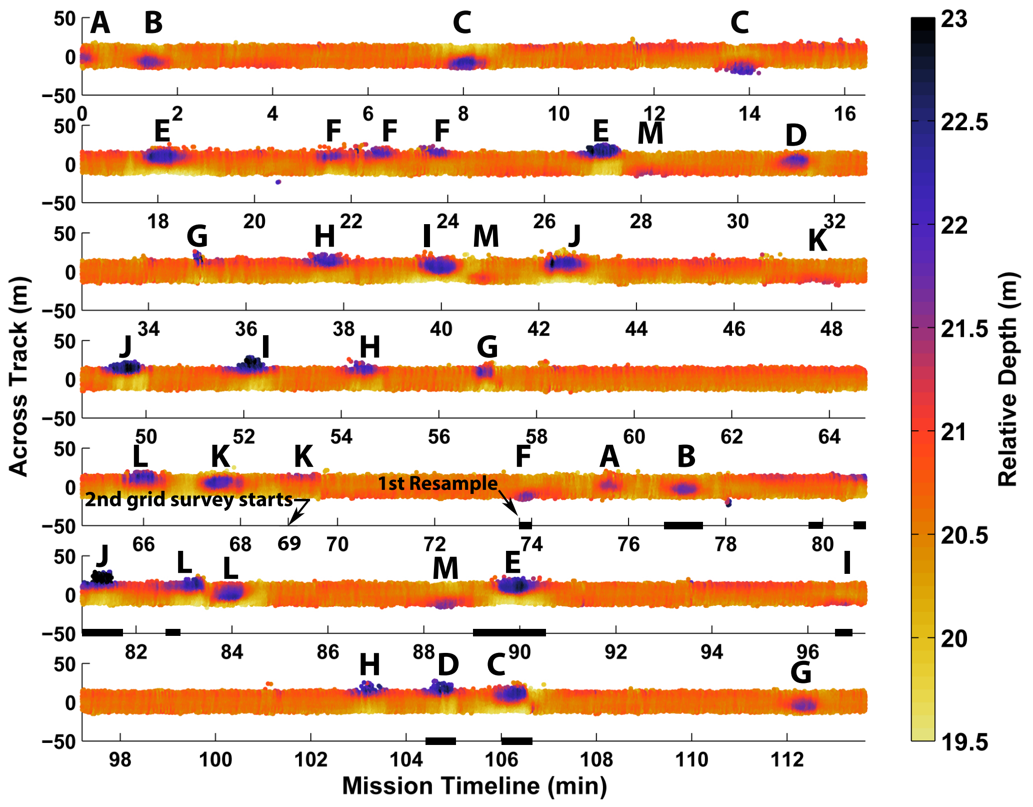
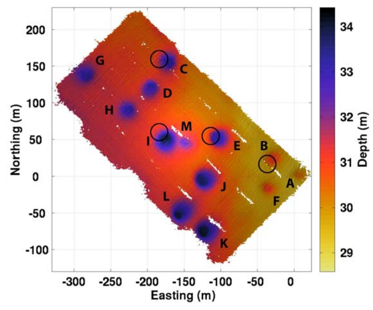


Figure 4.5: Waterfall display of the multibeam observations (depth relative to AUV). Letters A through M correspond to 13 distinct pockmarks uncovered by the survey. The dark bands on the timeline indicate the periods when resampling occurred.

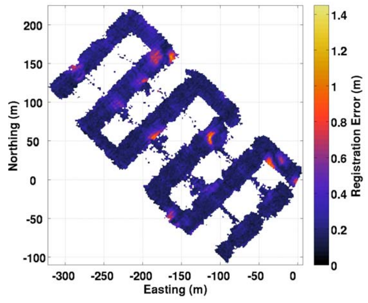
after travelling ≈ 2 km, which corresponds to the reobservation of pockmark F (see Figure 4.5) and the initial contraction in the uncertainty of the particle cloud in Figure 4.4. This contraction is also evident in Figure 4.3 near the east corner of the survey where the particle cloud is in the process of collapsing down into a smaller set of more likely state hypotheses. Further resampling after this time continues to constrain the uncertainty in the current pose, converging the particle cloud towards the solution which possesses the most self-consistent map. Note that resampling only occurs during a very small portion of the 2nd grid transect despite there being an effectively continuous reobservation of terrain. It was found that during these times resampling was possible but was being prevented by the filter as the particle weighting was too uniform to discern the most likely state hypotheses (see section 4.2.3). This suggests that during these times the relative navigation error accumulated between observation and reobservation was still relatively small, or that the majority of the particles had already been resampled and possessed similarly self-consistent maps.

Figure 4.6 presents a comparison of the resulting maps produced by the DR, USBL fused DR and BPSLAM solutions (see Appendix A for larger reproductions). Corresponding maps and histograms of the offline registration error measure described in Section 3.9 are also provided.

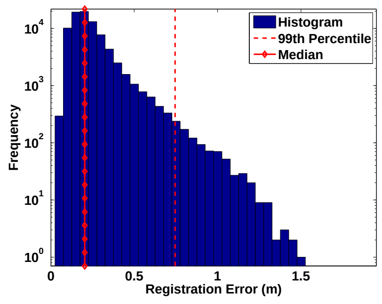
From Figure 4.6(a) the pockmarks discovered during this mission can be seen. However, the use of dead reckoning as a navigation method has resulted in some blur. Errors produced by bad sonar returns can appear anywhere in the map and will not be consistent with nearby swaths. However, inspection of Figure 4.6(b), which plots the maximum separation between overlapping swaths in each cell (note that these are only available in places where overlap occurred i.e. during the 2nd grid survey), reveals that the inconsistencies are localised around pockmarks. This is where sudden changes in depth occur and suggests that the blur is most likely caused by navigation error creating a misalignment between successive observations of each pockmark (i.e. ghosting), as opposed to bad sonar returns. The severity and abundance of these registration errors is also highlighted by Figure 4.6(c), which plots them as a histogram. Figures 4.6(d), 4.6(e) show how fusing USBL observations into the navigation helps reduce ghosting in the map by improving the navigation solution. However Figure 4.6(f) shows that while the overall quantity of these registration errors has decreased (indicated by the slight shift leftwards of the median and 99th percentile in Figure 4.6(f))



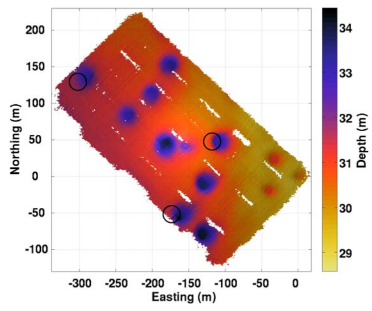
(a) DR Bathy. Map



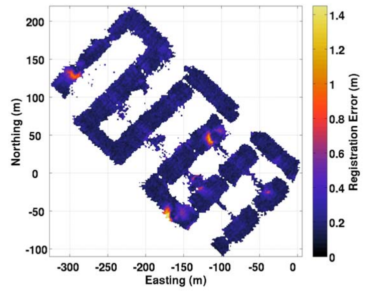
(b) DR Error Map



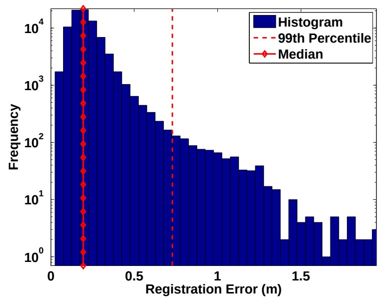
(c) DR Error Hist.



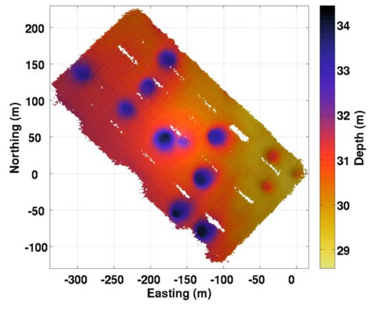
(d) USBL Fused DR Bathy. Map



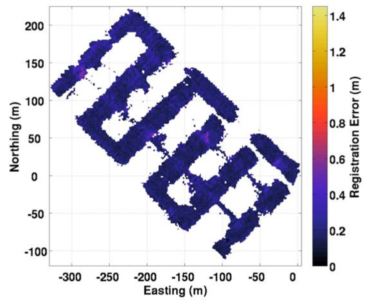
(e) USBL Fused DR Error Map



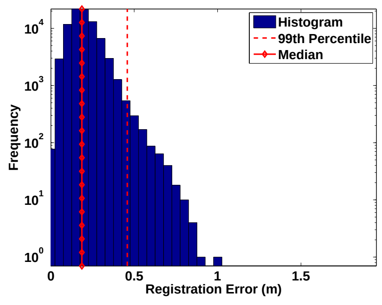
(f) USBL Fused DR Error Hist.



(g) Grid BPSLAM Bathy. Map



(h) Grid BPSLAM Error Map



(i) Grid BPSLAM Error Hist.

Figure 4.6: Bathymetric maps generated using our three different navigation solutions. The corresponding error maps and histograms are also provided, detailing the misregistration between overlapping swaths within each cell. Comparison shows the BPSLAM filter providing a reduction in the mapping error, the most prominent circled in black, when compared to the map produced by DR and the USBL fused solution. This is also reflected by a reduction in the tail of the histogram, which highlights the large registration errors, as well as a shift of the histogram's median and 99th percentile towards lower values as we move from using DR to USBL and then to the BPSLAM solution.

a small number of large scale registration errors have been introduced by this solution, making the map quality worse in some areas. These artifacts in the USBL solution may be owing to multipath in the acoustic signal received.

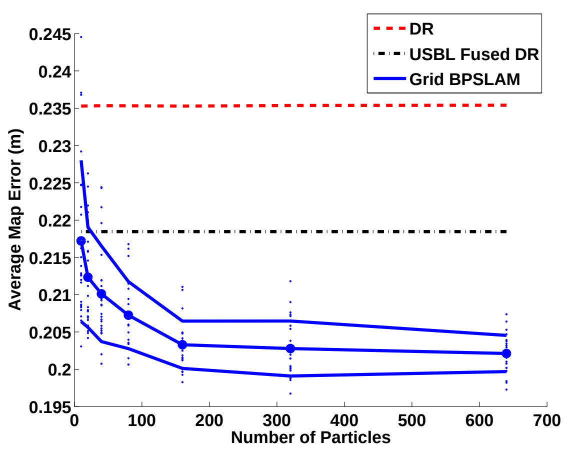
Figures 4.6(g), 4.6(h) and 4.6(i) demonstrate the superior performance of the BPSLAM filter. The ghosting in the map has been reduced significantly, condensing the corresponding histogram even further into the region of low registration error. As expected the BPSLAM filter has identified a navigation solution that aligns all the pockmarks discovered in the mission, without the need for feature detection algorithms or heuristics to identify loop closures.

For the maps produced by DR, USBL fused DR and Grid Map BPSLAM the overall quality, using the same offline FPS scheme described in Section 3.9, were calculated as 0.235 m, 0.216 m and 0.197 m respectively. Furthermore it took the Grid Map BPSLAM filter 3.1 minutes to process this mission, requiring 5.4 GB of RAM. This run time is 2.8 % of the total mission duration, additionally validating the Grid Map BPSLAM filter's computational performance.

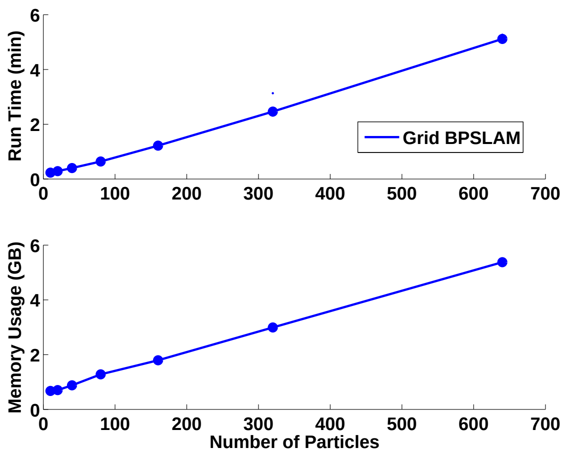
Accuracy, Consistency and Efficiency Analysis

As BPSLAM is a sampling based method the quality of the resulting paths and maps it produces can vary from run to run, the extent of which is heavily governed by the number of particles used in **S**. To investigate this behavior the BPSLAM filter was run repeatedly in batches of 25 using a different fixed particle size for each batch ($N = 10, 20, 40, 80, 160, 320, 640$). For each run the processing time, memory requirements and average registration error of the map were recorded. These results are shown in Figure 4.7.

Figure 4.7(a) demonstrates how increasing the particle set size converges the BPSLAM solution towards a more self-consistent map while also improving the repeatability of this result from run to run. Additionally it can be seen that only a small particle set size ($N > 160$) is required for the BPSLAM filter to repeatedly produce a more self-consistent map than when using DR or the USBL fused DR solution. Increasing the number of particles beyond this number shows significantly little improvement in the self-consistency of the maps produced. Figure 4.7(b) also demonstrates the computational behavior of BPSLAM for different particle sizes. As expected the implementation of DPM allows the



(a) Variation in map error with average particle set size.



(b) Variation in run time and memory usage with particle set size.

Figure 4.7: a) Variation in the map error with a varying particle set size using Grid Map BPSLAM. The large dots represent the mean map error measure in each batch of 25 runs (shown by the smaller dots). The outer solid lines represent the $\pm 2\sigma$ bound in each batch. Results show an increase in the accuracy and consistency of the map produced by BPSLAM when the number of particles is increased. In b) the run time and memory requirement of Grid Map BPSLAM can be seen to scale linearly with the particle set size.

filter to maintain a run time that scales linearly with N (see Section 3.1). The memory usage can also be seen to scale linearly with N.

Error Model Analysis

So far we have shown that the BPSLAM filter using grid maps can yield improvements in navigation and map quality when compared to DR and USBL fused DR solutions. However this is based on a mission where the DR solution is already fairly accurate due to the short mission duration and the use of high precision velocity observations from a DVL sensor (± 3 mms^{-1}). In this scenario the BPSLAM filter was only able to offer relatively small adjustments to the map that are not immediately recognizable when comparing Figures 4.6(a), 4.6(d) and 4.6(g). For an AUV operating with less accurate navigation it remains to be seen whether the BPSLAM filter remains as effective. To investigate this we separately simulate two forms of error in our navigation: a heading dependent bias in our magnetic compass and lower precision observations from our DVL sensor.

As mentioned previously our magnetic compass suffers from a bias that changes with heading, which was able to be corrected for through calibration (Jakuba et al. 2010). Disabling this compass correction reintroduces this navigation error into the tracklines. Alternatively we can artificially simulate a simple heading dependent compass bias of $\approx 1^\circ$ caused by hard iron interference. This is modeled by the first order harmonic:

$$bias = 1.15 * (cos(\psi + \delta) + sin(\psi + \delta)) \tag{4.10}$$

where δ is the phase. This type of navigation error is non linear and so cannot be modeled properly as a Gaussian uncertainty in the heading observation, as is required by the EKF filter. However as the error is dependent on heading and not on time (as is the case for states x, y) it is also difficult to model the error properly with the particle filter. Instead we leave the BPSLAM filter setup unchanged and artificially increase the uncertainty introduced by our constant velocity model. This caters for the navigation error introduced by the compass bias as well as other navigation errors that may still be unmodeled, allowing the particle set to expand appropriately. However this is an adhoc approach that overcompensates for uncertainty, and so requires more particles than would be needed if the compass bias were modeled by a more principled approach. This is the subject of future work.

Figure 4.8 presents both the model of our estimated compass bias and the artificial compass biases described. We then repeatedly ran the BPSLAM filter in batches of 25 to successively test the performance of DR, USBL fused navigation and the BPSLAM filter when subject to each of these compass biases.

Figure 4.9 presents the maps generated by DR and BPSLAM when the compass bias is left uncorrected (the USBL fused DR solution is not shown here as it is indistinguishable from that shown in Figure 4.6). Figure 4.10 presents the results of our successive runs with different simulated compass biases. As expected the uncorrected compass bias results in a more inconsistent map when DR is used as the navigation solution. However BPSLAM is still able to correct for this additional navigation error and produce (on average) a more self-consistent map than the USBL fused DR solution. When the larger simulated heading biases were applied it was found that the USBL fused solution performed (on average) better than BPSLAM. However BPSLAM still consistently outperforms the use of DR as a navigation solution. As this survey consists of two orthogonal grid transects the majority of the mission is undertaken along four dominant headings. As expected the biases which

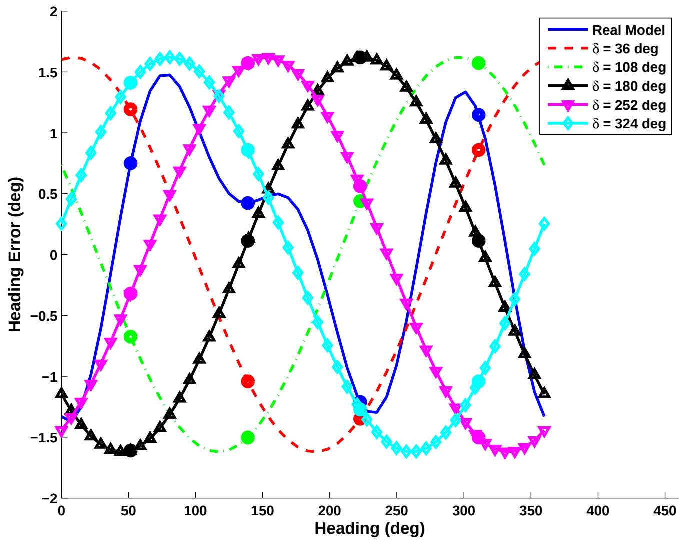
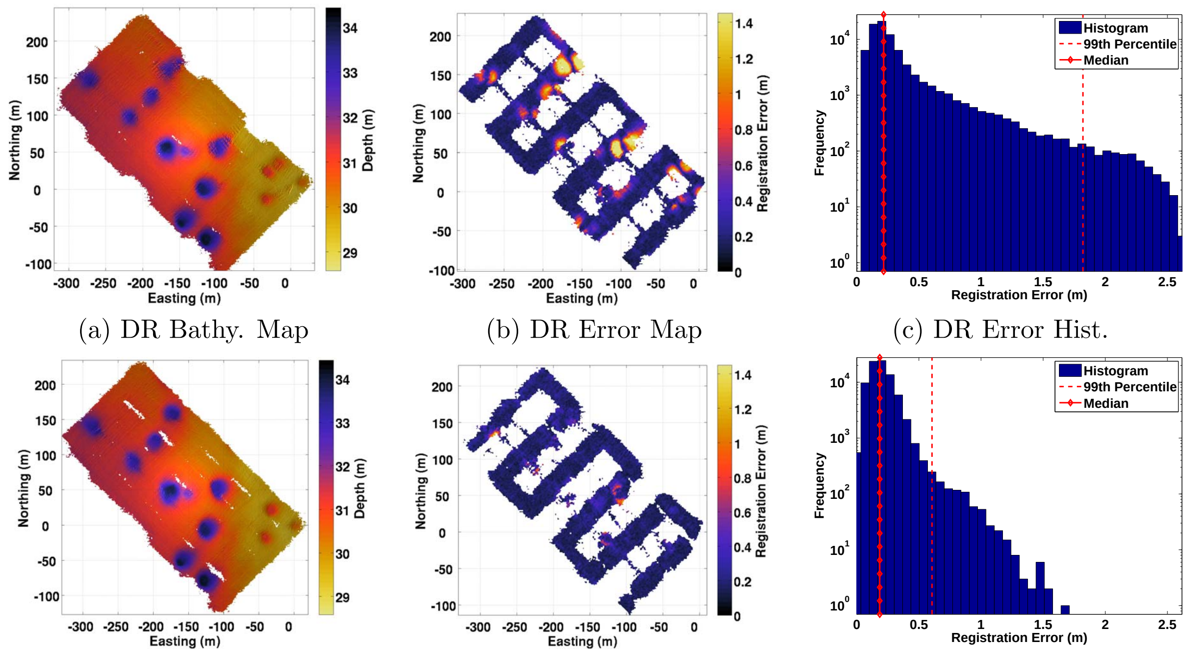


Figure 4.8: Different biases that we simulate in our magnetic compass. Note that the bias shown in dark blue is the estimated compass bias and is achieved by turning off our compass correction. The dominant headings of the grid survey are shown by the large dots.



(a) DR Bathy. Map (b) DR Error Map (c) DR Error Hist.

(g) BPSLAM Bathy. Map (h) BPSLAM Error Map (i) BPSLAM Error Hist.

Figure 4.9: Bathymetric maps generated when our compass bias is left uncorrected. The corresponding error maps and histograms are also provided, detailing the misregistration between overlapping swaths within each cell. While the DR solution suffers from a significant increase in registration error the BPSLAM solution is still able to maintain the same level of consistency.

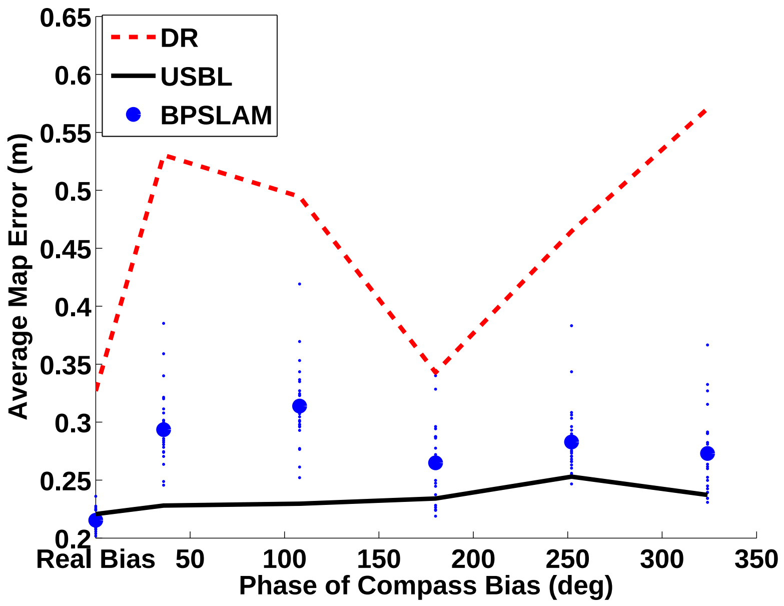


Figure 4.10: Results of simulating different heading dependent compass biases. BPSLAM produces (on average) a better solution than the USBL fused solution when the real compass bias is left uncorrected, though for larger simulated biases USBL navigation becomes the preferred solution. However BPSLAM continues to consistently outperform the DR solution in all cases.

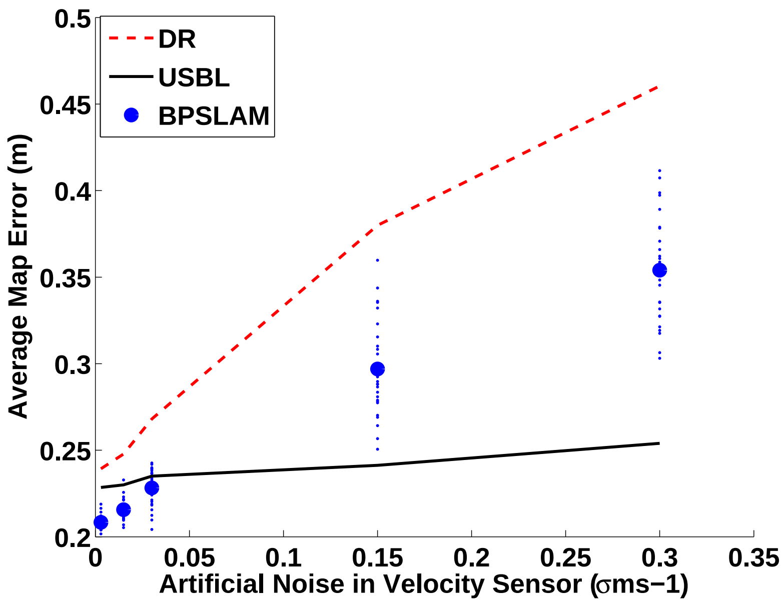


Figure 4.11: Results of simulating a lower precision DVL sensor. BPSLAM consistently out performs the DR solution. However for very high noise levels the USBL fused solution emerges as the most self-consistent.

produced the largest errors along these dominant headings produced the most inconsistent maps.

The effect that the precision of our DVL sensor has on performance of the filter can be investigated by simulating less precise observations through the addition of zero mean Gaussian noise. The levels of noise that we have chosen to use reflect the range of accuracies from commercially available DVL systems (± 0.003 ms^{-1} to ± 0.03 ms^{-1}). As an exploratory measure we also model precisions much lower than this, as might be expected of a model that relies on propeller counts to estimate velocity (± 0.15 ms^{-1}, ± 0.3 ms^{-1}). Again we repeatedly run the BPSLAM filter in batches of 25 for each noise level.

Figure 4.11 shows that decreasing the precision of the DVL sensor results in a map which is progressively more inconsistent. As the USBL solution is a fusion of DR and USBL observations it too suffers from an increase in the average registration error, though this increase is significantly smaller. Here the BPSLAM solution is shown to provide the most self-consistent maps (on average) for the range of precisions that are currently available in commercially available DVL sensors. Increasing the noise further results in a map that is progressively less accurate than the USBL fused solution and is also less repeatable. However the BPSLAM solution still consistently outperforms the DR solution, validating the effectiveness of BPSLAM when dealing with lower precision velocity sensors.

4.3.2 Case Study 2: TAG Hydrothermal Vent

As alternative bathymetric SLAM filters currently exist it is instructive to provide a performance comparison. In this case Roman's Sub-mapping bathymetric SLAM filter (Roman & Singh 2005) was chosen as it is also featureless in its approach and suited to mapping in open underwater environments (see Section 2 for more details). In addition this comparison is made on a large deployment to test the scalability of BPSLAM to a mission timescale more common in industry.

This is achieved by utilising bathymetric and navigation logs from a survey undertaken by JASON, a ROV operated by the US National Deep Submergence Facility (Roman 2005).

The survey covers part of the Trans-Atlantic Geotraverse (TAG) Hydrothermal field and contains several crossing tracklines over a large hydrothermal vent spanning over $17,500$ m^2 in area, collected over a time span of 11 hours. The navigation suite installed on JASON

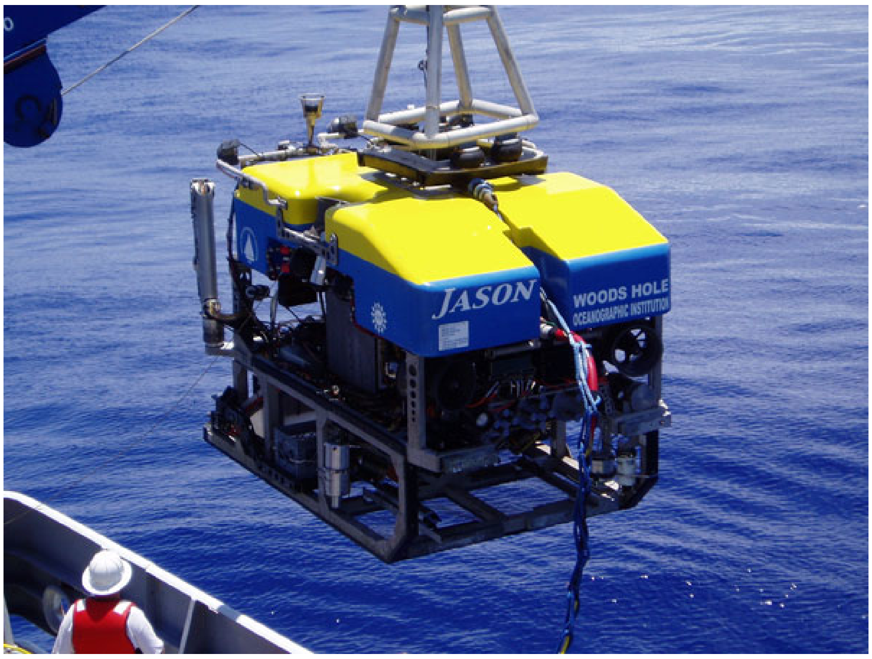


Figure 4.12: The JASON ROV is a research vehicle capable of exploring the ocean down to 6,500 meters depth. Similar to Sirius it can be used to collect high resolution bathymetry and stereo imagery.

is similar to that described in Section 4.3 in that three axis attitude, three axis bottom relative velocity and surface relative depth sensors were available. However JASON also receives heading observations from a Fiber Optic Gyroscope (FOG) that delivers $\approx 0.1^\circ$ accuracy. High precision pitch and roll observations of $\approx 0.1^\circ$ precision were also available. As such the vehicle states x, y remain the most prone to drift and so no change was made to the state setup of the filter.

Acoustic LBL navigation fixes from external beacons were also available at 10 second intervals using a vehicle mounted transponder. These fixes delivered position estimates accurate to ≈ 4 m and are only used by the BPSLAM filter to initialise the ROV's position. Similar to Section 4.3 they were used as measurement updates in an EKF filter separate to BPSLAM, so as to produce a navigation solution by which to evaluate the performance of the BPSLAM filter.

The bathymetric sensor used by JASON is a SM2000 200 kHz multibeam sonar (Kongsberg-Mesotech Ltd), providing 128 beams uniformly across 120°. This corresponded (for this mission) to an average spatial separation of 0.239 m and 0.126 m in across track and along track respectively. Additionally imposing our maximum resolution bound specified by Eq. 4.8 gives us 0.783 m. Based on this we run our BPSLAM filter using a grid size of 1.0 m. The parameter $\gamma_{overlap}$ remained set at 100 %.

Navigation and Mapping Comparison

To ensure a good result we demonstrate the BPSLAM filter performance during the mission using the aforementioned parameters with a relatively large number of particles ($N = 640$). The corresponding navigation solution produced by the BPSLAM filter is shown in Figure 4.13 along with the behavior of the particle set in Figure 4.14. Unfortunately the Sub-mapping algorithm does not currently output a navigation solution so this cannot be shown. Figure 4.15 presents the multibeam data collected during the mission.

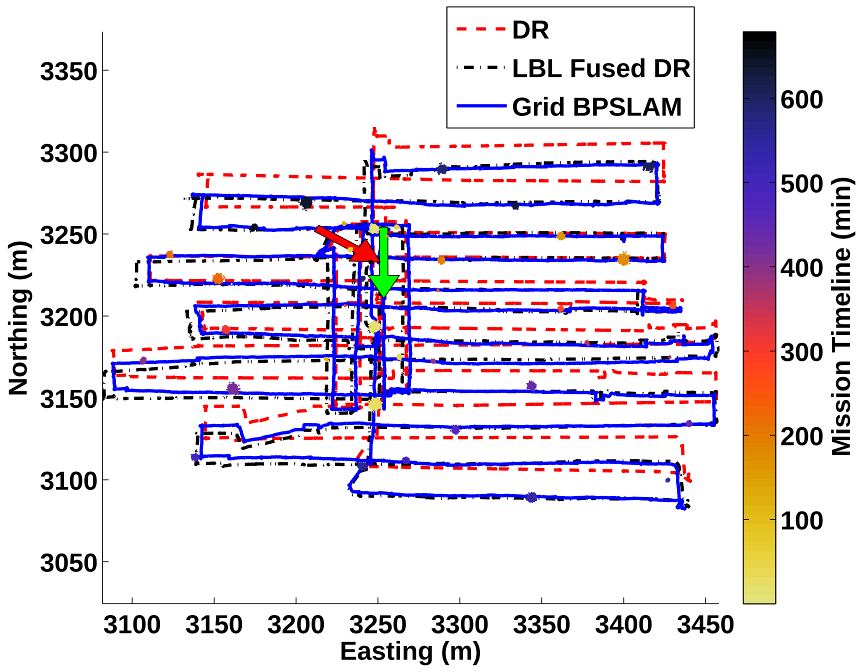


Figure 4.13: Tracklines produced by three different navigation solutions; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively. The evolution of the particle cloud is also shown, changing from light to **dark** as the mission progresses.

Comparing navigation solutions in Figure 4.13 shows how BPSLAM significantly shifts the tracklines away from the DR solution towards the LBL fused DR solution, as much as 25 m in some places. This shift is also evident when viewing the LBL innovations (the difference between the raw vehicle position observations reported by the LBL system

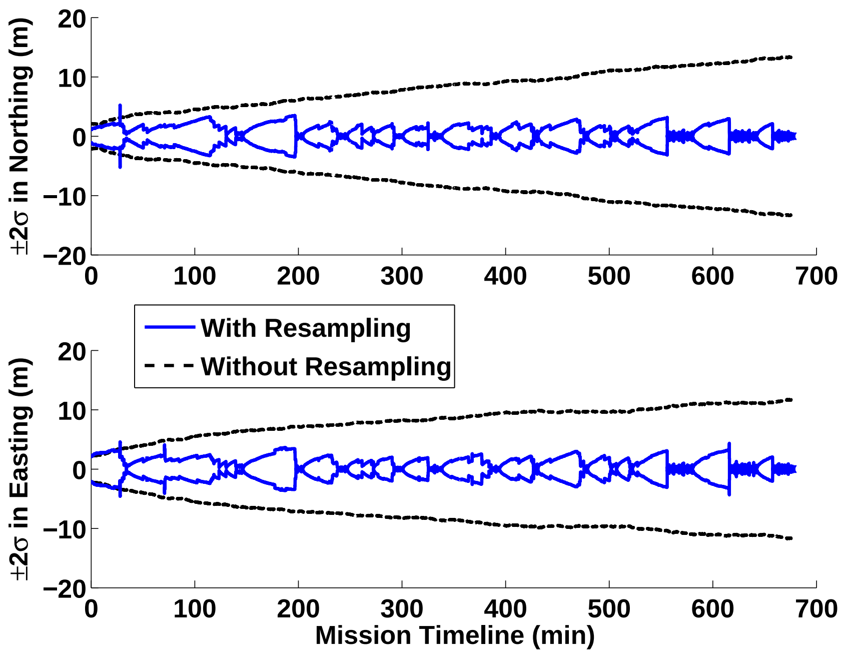


Figure 4.14: Variation in the 95% confidence interval of the vehicle's x, y position, with (solid line) and without (dotted line) particle resampling.

and the predicted position from the trajectory) for each navigation solution, as shown in Figure 4.16. As expected the LBL fused DR solution retains the smallest LBL innovations, as its filter uses these observations as measurement updates. Note though that the LBL observations are subject to error and cannot be treated as ground truth. For this reason the difference in the innovations between the BPSLAM and LBL fused DR solutions are too small relative to the precision of the LBL observations (≈ 4 m) to judge which is more accurate. However the innovations observed by the DR solution are large enough to signify greater navigation error, verifying that LBL fused DR and BPSLAM have improved the accuracy of the navigation solution.

Figure 4.17 presents a comparison of the resulting maps produced by the DR, LBL fused DR, BPSLAM and Sub-mapping filters. Corresponding maps and histograms plotting the offline registration error described in Section 3.9 are also provided. For this mission the Sub-mapping filter was able to maintain self-consistency in its map by temporally dividing the generated point cloud into 62 submaps, whose translation and rotation relative to each other were then tightly constrained through the filter's identification of 130 terrain registrations between the submaps.

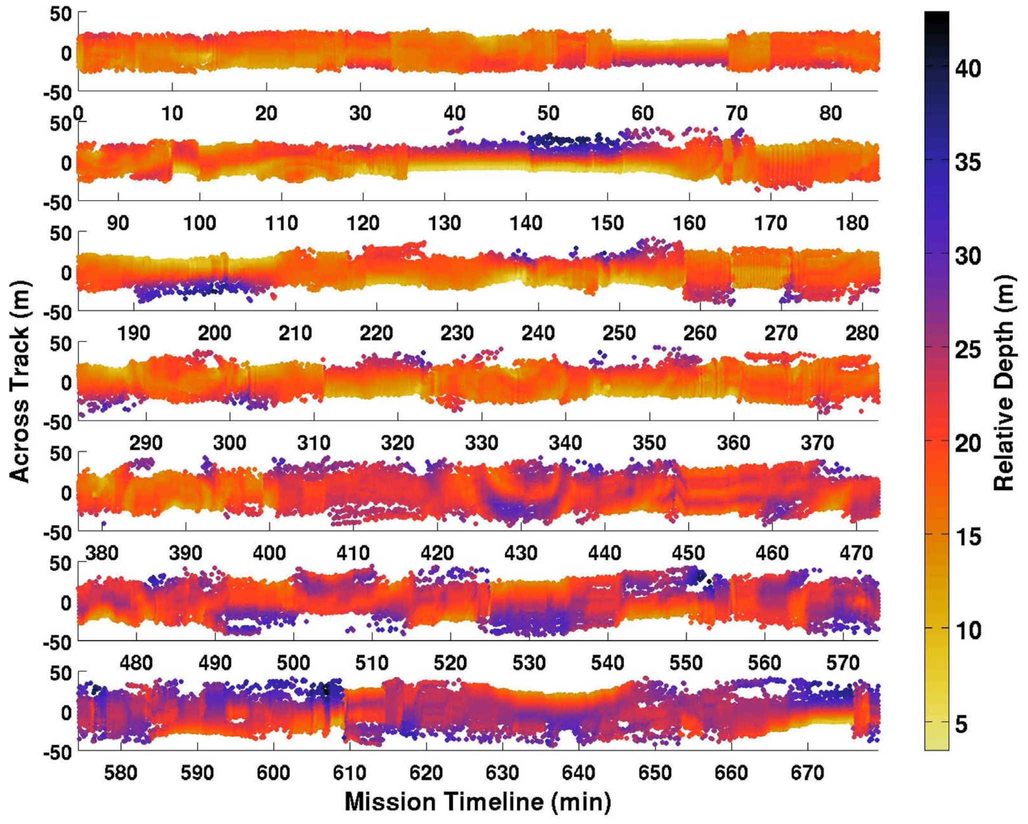
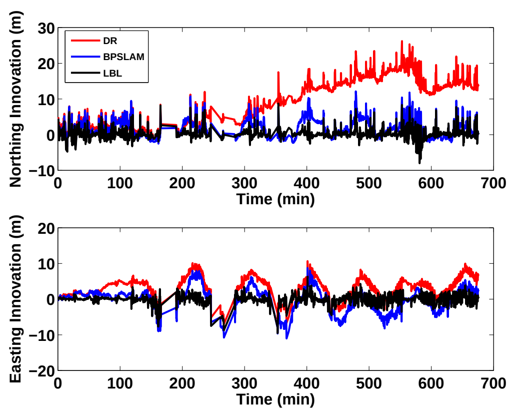


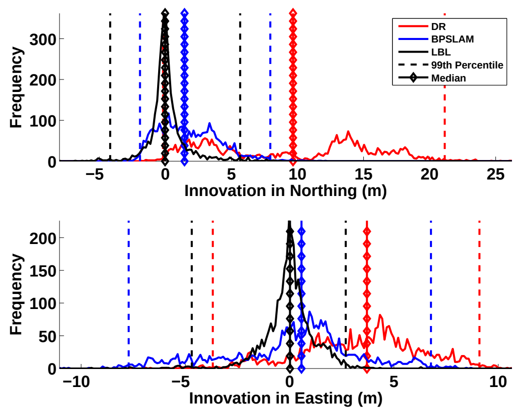
Figure 4.15: Waterfall display of the multibeam observations (depth relative to ROV) collected during the TAG mission. Note that sharp changes in the waterfall are caused by the ROV performing 90° turns in areas of large relief, such as around the rim of the vent.

The use of DR as a navigation solution has produced the map with the most artifacts, as shown in Figure 4.17(a). This is particularly evident along the northern rim of the vent where a crosshatch pattern is clearly visible. Comparing these artifacts to the tracklines shown in Figure 4.13 suggest that they are of the same shape as the scan pattern. This and the tendency for these artifacts to be localised around areas with large depth gradients (see Figure 4.17(b)) further suggest that they are caused by misaligned swaths produced by navigation error. The inclusion of LBL observations into the navigation solution resolves many of these artifacts, as can be seen in Figures 4.17(d), 4.17(e) and 4.17(f). However it has also introduced artifacts in places where there were none before. This highlights the ability for LBL observations to remove low frequency error from navigation while causing problems by introducing high frequency error.

Figures 4.17(g)-4.17(l) demonstrate the superior performance of both the BPSLAM filter



(a) Innovation of LBL observations in time

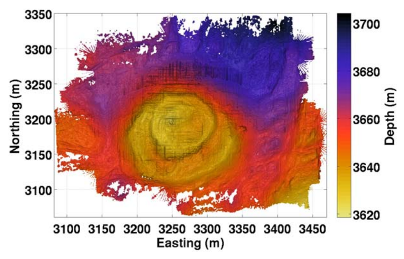


(b) Histogram of Innovations from LBL observations

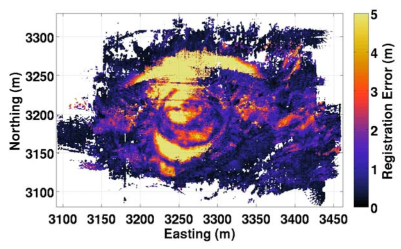
Figure 4.16: Innovation of raw LBL observations during the TAG mission for three different navigation solutions. Comparison shows a distinct divergence of the DR solution away from the LBL observations, whereas the LBL fused DR and BPSLAM solutions retain a much closer proximity, confirming an improvement in navigational accuracy.

and the Sub-mapping filter when compared to LBL aided navigation. All of the prominent artifacts distorting the non SLAM approaches have been successfully resolved, although both approaches still retain a small amount of registration error. For the Sub-mapping filter these correspond to areas where the ROV was "yanked" by its tether, producing vehicle motion that significantly diverges from the constant velocity model it uses within each local submap. This poses a particular problem for the Sub-mapping filter as it creates misregistration within the local submap that cannot be resolved. However the BPSLAM filter is still capable of resolving this type of misregistration, though it was not able to completely resolve the misalignment in the steepest areas of the survey e.g. the tip of the hydrothermal vent. Comparing Figures 4.17(i) and 4.17(l) shows similar performances by both SLAM filters. However when regarding the self-consistency of the whole map it can be seen that the sub-mapping filter offers a slightly better solution. This is also confirmed by the average registration error for the DR, LBL fused DR, BPSLAM and Sub-mapping solutions, calculated as 1.733 m, 1.490 m, 0.585 m and 0.515 m respectively.

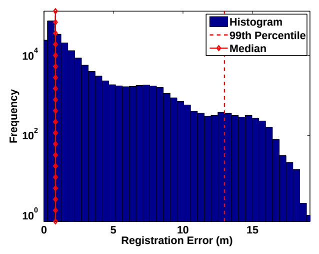
Comparing the computational performance of each SLAM filter however reveals a significant difference. The BPSLAM filter required 9.76 GB of RAM and took 8.7 minutes to process the mission, which is 1.2 % of the mission time. The Sub-mapping algorithm however required overnight processing to provide its mapping corrections for this mission. This effectively demonstrates the run time efficiency of the BPSLAM algorithm and its potential



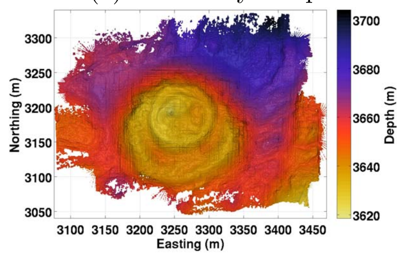
(a) DR Bathy. Map



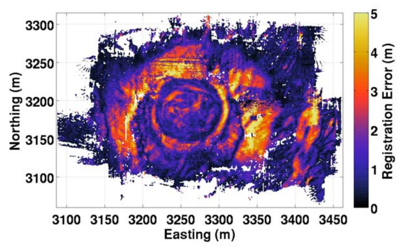
(b) DR Error Map



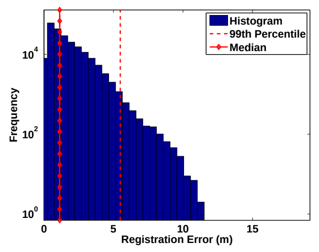
(c) DR Error Hist.



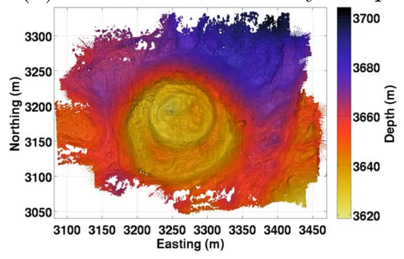
(d) LBL Fused DR Bathy. Map



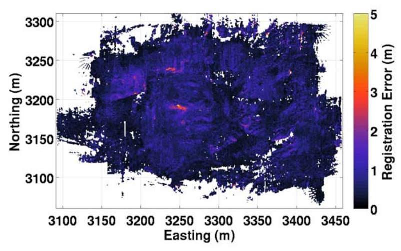
(e) LBL Fused DR Error Map



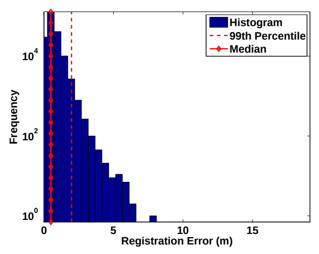
(f) LBL Fused DR Error Hist.



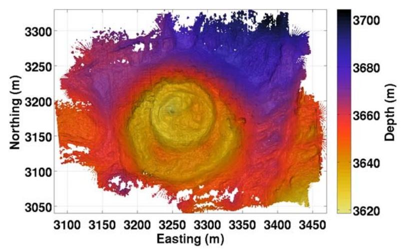
(g) Grid BPSLAM Bathy. Map



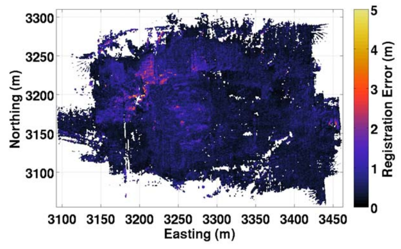
(h) Grid BPSLAM Error Map



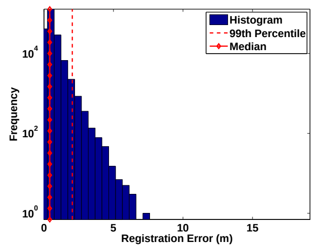
(i) Grid BPSLAM Error Hist.



(j) Sub-mapping Bathy. Map



(k) Sub-mapping Error Map



(l) Sub-mapping Error Hist.

Figure 4.17: Bathymetric maps generated using four different navigation solutions. The corresponding error maps are also provided, detailing the misregistration between overlapping swaths within each cell. Comparison shows the two SLAM filters providing a significant reduction in the mapping error when compared to the map produced by DR and the LBL fused DR solution.

for real time implementation on surveys of this scale or smaller.

4.4 Limitations

While Section 4.3 has successfully demonstrated the ability of the BPSLAM filter to impart corrections to navigation and mapping it has also highlighted some drawbacks to using a grid map representation. In particular the memory requirements of this representation, while scaling well with particle set size, would start restricting its use if missions larger than the ones presented so far were to be attempted. Furthermore, by bounding the survey with a rectangular grid a significant amount of memory can be wasted providing cells in the areas of the grid that the survey does not reach. As mentioned in Section 4.2.1 quadtrees provide a memory efficient approach to this problem but come at the cost of increasing the complexity of the map querying and updating operations.

Another problem with the grid map representation is that some map resolution will always be lost by discretising the map into a collection of cells. In this case the loss of resolution cannot be simply overcome by increasing the resolution, as this increases the memory requirement quadratically but also reduces the chance of incoming observations landing in cells that have been previously observed, thereby reducing the ability to resample. For sparse datasets or for missions with minimal overlap this is particularly problematic and will reduce the performance of the BPSLAM filter back to Dead Reckoning. Resolution is also limited by the data association problem mentioned in Section 4.2.3, whereby an observation may be incorrectly matched to a cell due to its uncertainty in range creating a corresponding uncertainty in its x, y location.

This motivates the creation of a new map representation that is memory efficient and better models the uncertainty that is introduced in the map by range observations. Most importantly though this discussion highlights compelling reasons to move to a map representation that accommodates for the spatial correlation in the environment instead of treating each estimate as independent, as has been done here. While this is more than likely to increase the computational run time of the approach the afore-mentioned characteristics will serve to make BPSLAM suitable for a much wider range of mission scenarios. In this way a new map representation would complement the grid map approach rather than replace it, as the computational run time of the latter has great potential for real time operations.

4.5 Summary

This chapter has presented the grid map representation that is paired with RBPF SLAM to form the BPSLAM filter. A description of the map structure along with the procedures used to weight and update each particle's map were provided.

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, both in small scale and large scale missions, as well as demonstrating a robustness to compass biases and noisy velocity observations. Its performance was also compared against Roman's Sub-mapping approach to SLAM (Roman 2005). While only a slight difference was found in the map qualities produced by both SLAM filters, the BPSLAM algorithm was able to demonstrate a significant improvement in computational efficiency. BPSLAM also achieved this with fewer tuning parameters, which were shown to translate well between the small and large scale missions presented here. These parameters included the number of particles, map resolution and $\gamma_{overlap}$.

Lastly a discussion of the limitations faced by the grid map representation were presented, motivating the creation of a new map representation that addresses these shortcomings.

Chapter 5

Bathymetric Particle Filter SLAM Using Trajectory Maps

5.1 Introduction

In Chapter 4 a grid map representation was presented that could be paired with the RBPF SLAM technique described in Chapter 3 to form a means of SLAM with superior computational run time, provided that the mission contained dense bathymetry and maximal overlap with previously explored areas.

In this chapter the BPSLAM approach is extended by presenting an alternative map representation that is not bound by these limitations. This is achieved by utilising Gaussian Process(GP) Regression to provide loop closures in areas where little to no overlap with previously explored terrain is present. In this way the spatial correlation in the environment is fully exploited, allowing the filter to not only enforce self-consistency in overlapping sections of the map but additionally enforce self-consistency between neighboring map borders.

In itself this does not form a new map representation but rather a new method of particle weighting that could still be implemented using the current grid map representation. However to make the memory requirements of this approach more tractable to datasets larger than the ones presented so far, as well as to recover the map resolution lost through discretisation, we forego the grid map representation for another more memory efficient approach.

As discussed in Section 4.1, grid map representations offer relatively fast map access at the cost of losing map resolution through discretisation. Alternatively, storing each particle map as a cloud of 3D points does not incur this loss of resolution but unfortunately imposes even greater memory requirements for storage when compared to the grid map approach.

An important observation is now made about both of these map representations, namely that each particle map is generated from an identical log of bathymetric observations. It is only the underlying particle trajectories that are different. A considerable amount of redundancy can therefore be removed from the maps if, instead of storing each particle's point cloud or grid map, the trajectory of each particle is stored and simply linked to a corresponding log of observations shared by all particles. The memory requirements of this map representation is therefore approximately N times less than using separate point clouds for each particle. Technically this in itself is not a true "map" representation but rather an efficient manner in which the raw components to build a particle's map can be stored and accessed without loss of information. Once constructed a particle's map is simply a 2.5D point cloud. However, in the interests of highlighting the novel methods by which the particle maps are maintained we refer to this setup as the trajectory map representation.

This new type of map representation will be paired with GP Regression and RBPF SLAM to perform BPSLAM. While this will mean that computational run time will be sacrificed (as a particle's point cloud will need to be reconstructed locally every time particle weighting is required) the results presented in Section 5.4 serve to validate this tradeoff.

In the remainder of this chapter an introduction to Gaussian Process Regression is provided, along with a description of the trajectory map representation and the map weighting and updating procedures particular to this approach. The two case studies presented in Chapter 4 are then revisited to validate the performance of BPSLAM using trajectory maps, and to compare the new performance against that previously attained from using the grid map representation. Lastly a new case study is presented that investigates the added capabilities of Trajectory Map BPSLAM in missions with minimal map overlap.

5.2 Gaussian Process Regression

As each particle's map (once reconstructed from its trajectory) is a point cloud with uncertainty in depth, a form of regression will be required to extract depth predictions from it.

These predictions need to be made at the precise locations of the new multibeam observations so that particle weighting can occur. In addition the uncertainty in these predictions will also be required if the consistency between observation and prediction is to be properly judged during this stage.

To perform these predictions some assumptions about the seabed must be made. Firstly it is assumed that, to a good approximation, the seafloor can be modeled as a continuous function $g(x, y)$. As with the grid map representation this restricts the scope of operations to marine environments that are approximately 2.5D (as is the case for the majority of the seafloor). If it was further assumed that $g(x, y)$ could be represented by a specific model (e.g. an nth order polynomial or spline) then data fitting techniques (such as least squares minimisation) could be used to learn $g(x, y)$ and hence provide predictions at any given x, y coordinate. However choosing the most appropriate model is a non-trivial problem and can lead to under or over-fitting the data if done incorrectly. This approach also requires heuristics to additionally calculate the uncertainty in the predictions. For these reasons a Gaussian process is instead used to model $g(x, y)$ as it places less assumptions on the function, allowing an appropriate model to be learnt by letting the data "speak for itself".

A Gaussian process is a multivariate gaussian distribution generalised to infinitely many variables. Using this as a model for $g(x, y)$ effectively represents the function as an infinite 2D matrix (both in range and resolution) of correlated variables, where each cell specifies the distribution of $g(x, y)$ for that given x, y coordinate. Naively, utilising this infinite dimensional model would be computationally impractical. Fortunately GPs possess an important marginalisation property that bypasses this issue i.e. if the distribution of $g(x, y)$ is only required at a finite number of points (variables), then inference in the Gaussian process will give the same answer whether or not all the other infinitely many points are taken into account (Rasmussen & Williams 2006). Inference can therefore be made computationally tractable by safely ignoring all points that are not of interest.

While this model is more flexible than polynomial or spline fitting it still requires some more subtler assumptions to be made in terms of the correlation between neighboring predictions. This is specified in the form of a covariance function $k(x, y, x', y')$, of which many different models are available. In addition covariance functions often contain a set of *hyperparameters* that can be learnt to improve the accuracy of the model, based on the dataset and some optimisation criteria.

The mean function $m(x, y)$ of the GP also needs to be specified (though is normally set to zero). This function describes the expected value of $g(x, y)$ a priori i.e. the value predictions will converge to in the absence of training data within proximity. Once chosen these two functions completely define the GP.

Once the GP model is defined regression can be performed to predict the function value of $g(x, y)$ at the locations of the new observations. To do this the particle's point cloud map is treated as a training set $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ with M input points $\mathbf{X} = \{x_i, y_i\}_{i=1}^{M}$ and noisy output points $\mathbf{Y} = \{z_i\}_{i=1}^{M}$ with noise covariance matrix $\mathbf{W} = diag(\sigma_1^2 ... \sigma_M^2)$. The predictive distribution of the underlying function $g(\mathbf{X}^*)$ at test points $\mathbf{X}^*$ is then calculated based on our covariance function $k(X, X')$ and mean function $m(X)$. This is done using the standard equations for GP Regression as given by Algorithm 5.

Algorithm 5 GP Regression Equations

```
1: $\mathbf{L} := cholesky(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \mathbf{W})$
2: $\alpha := \mathbf{L}^T \backslash (\mathbf{L} \backslash \mathbf{Y})$
3: $\mathbf{g}(\mathbf{X}^*) := \mathbf{k}(\mathbf{X}, \mathbf{X}^*)^T \alpha$
4: $\mathbf{v} := \mathbf{L} \backslash \mathbf{k}(\mathbf{X}, X^*)$
5: $Cov(g(X^*)) := k(X^*, X^*) - \mathbf{v}^T \mathbf{v}$
6: $log\ p(\mathbf{Y}|\mathbf{X}) := -\frac{1}{2}\mathbf{Y}^T \alpha - \Sigma_i log\ L_{ii} - \frac{n}{2} log 2\pi$
   (Note: For multiple test points lines 4,5 are repeated.)
```

In summary GP Regression is a very powerful tool as it allows depth predictions to be made in unobserved areas as well as providing a theoretically sound method of calculating the corresponding uncertainty $Cov(g(\mathbf{X}^*))$ in the predictions made. For a more detailed description of the theory and applications of GPs, Rasmussen & Williams (2006) provides an extensive review.

5.3 Method

5.3.1 Map Structure

To perform Distributed Particle Mapping an ancestry tree must be maintained. Similar to Section 4.2.1 this is structured in the form of a list, where each particle (both current and ancestral) has a record in this list that contains 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.

In addition to this, each particle's trajectory map can also be stored by appending the following information:

- **Trajectory** - A list of poses which forms the trajectory branch of this particle.
- **Observations** - A list of indexes (synchronized to the trajectory) that point to sonar swaths in the multibeam log.

The ID of the particle is not stored as a field in this record but is encoded as the position of the record in the list itself, allowing for fast access. A particle's trajectory can therefore be reconstructed at any time by backtracing through its lineage. A particle's map is similarly reconstructed by backtracing through a particle's observations and then using the trajectory (whose poses are synchronised to the observations) to transform them into the global frame. This is demonstrated in Figure 5.1.

5.3.2 Map Initialisation

At the beginning of the mission each particle is initialised with an empty trajectory and sonar swath index list. If a prior map is available this can be entered into the filter by first calculating the location of each prior map estimate relative to the sensor frame at the start of the mission. These transformed estimates can then be modeled as a single large 2.5D multibeam swath, with each beam observation having an along track, across track and depth component. The uncertainty in depth is also stored. If this "prior map swath" is stored in the bathymetric log as the first entry then all particles can automatically inherit and reconstruct the prior map if the starting pose and index of 1 is added to the root particle's trajectory list and sonar swath index list respectively.

Compared to the grid map representation this approach allows for prior map localisation without losing precision in the navigational corrections through map discretisation. In addition this approach is also less sensitive to differences between the prior map resolution and resolution of the current mapping sensor, as the use of GP Regression allows more observations to be matched to prior estimates through interpolation/extrapolation.

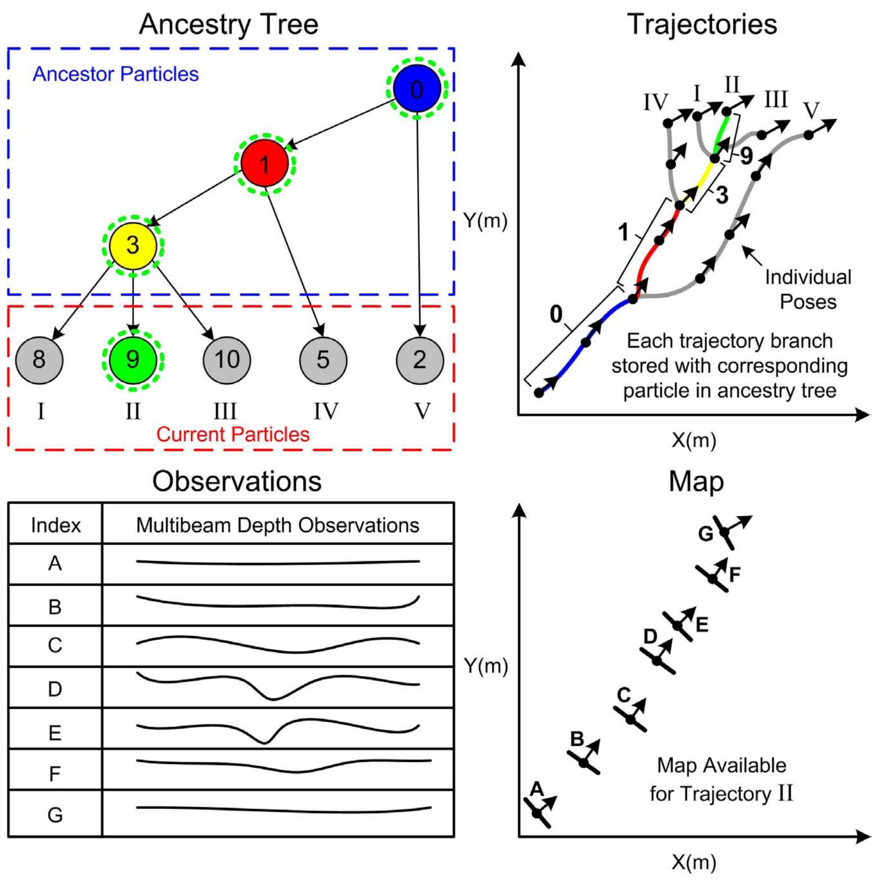


Figure 5.1: An example of the trajectory map structure with N=5 particles and 7 observations (A-G). The trajectory stored at index *II* in the particle set is reconstructed by backtracing through the ancestry of the particle located at this index and connecting together each trajectory segment owned by the particle and its ancestors (shown by the broken circles). This trajectory is then used to transform a list of observations, synchronised to the poses contained within it, into the global frame to create the particle's map.

5.3.3 GP Model Choice and Learning Scheme

In BPSLAM each particle's map is progressively generated, meaning that the GP model must be periodically relearnt as new bathymetry is received. Furthermore, each particle will need to perform a separate regression based on its own map during particle weighting. This poses a problem as one of the major drawbacks of using GPs is the need to invert potentially large covariance matrices during learning and regression, corresponding to a $\mathcal{O}(M^3)$ computational cost. In terms of computational tractability this precludes the use of more complex non-stationary covariance functions such as the neural network (Vasudevan, Ramos, Nettleton & Durrant-Whyte 2009)(Rasmussen & Williams 2006) that can model both large scale trends and local anomalies relating to discontinuous data. Instead a simpler stationary covariance function $k(X, X'; l, \sigma_0)$ developed by Melkumyan & Ramos (2009) is used. This is given below:

$$k(X, X'; l, \sigma_0) = \begin{cases} \sigma_0[\frac{2+cos(2\pi\frac{d}{l})}{3}(1-\frac{d}{l}) + \frac{1}{2\pi}sin(2\pi\frac{d}{l})] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases} \quad (5.1)$$

where the distance $d = |X - X'|$. The properties of this covariance function are similar to the squared exponential kernel (Rasmussen & Williams 2006). However in this case the covariance function falls to exactly zero for distances greater than the length hyperparameter l. This means that the covariance matrices generated by this function are intrinsically sparse, which allows for exact inference while providing faster computation using sparse methods. It should also be noted that stationary covariance functions cannot account for variable smoothness in the terrain without needing to be relearned. However, this is already a requirement of our approach and Section 5.4 will serve to validate our choice.

The set of hyperparameters for this covariance function should be chosen to produce output that matches the variances observed in the training data, while at the same time create a model that is not too complex. Fortunately the first two terms of the log marginal likelihood $log\ p(\mathbf{Y}|\mathbf{X})$ respectively provide quantitative measures of these criteria, presented in Algorithm 5 on Line 6. The third term here is a normalising constant. Optimising on the marginal log likelihood therefore produces a set of hyperparameters that provides a good model for regression. However, as mentioned previously the hyperparameters will need to be relearned periodically to ensure that they remain a good model for the local terrain. This is accomplished by taking a square patch of the most recent bathymetry (with along track distance equal to the swath width so as not to bias any particular direction) and transforming it into the global frame using the Dead Reckoning trajectory. GP learning is then carried out using this training subset.

What remains to be chosen is the mean function that will be coupled with the covariance function to perform GP Regression. To improve the predictions outside the realm of the training data the mean function is set to the mean of the training output i.e. $m(X) = \sum_{i=1}^{M} \frac{Y_i}{M}$.

5.3.4 Map Weighting

As mentioned in Chapter 3, the map weighting stage judges each particle by the likelihood that the current multibeam swath would be observed, given the particle's current map. Algorithm 6 details the steps involved to calculate this.

Algorithm 6 STEP 6 FROM ALGORITHM 1:TRAJECTORY MAP WEIGHTING

```
Require: Combined particle/EKF state hypothesis $\vec{\mathbf{x}}_{v_i}(t_k)$, Observations $\mathbf{z}$, Map $i$
 1: if Time since GP hyperparameters were relearnt $\geq \tau_{relearn}$ then
 2:     Relearn GP hyperparameters using recent observations and DR trajectory.
 3: end if
 4: if (GP length hyperparameter $l \leq l_{max}$) then
 5:     $ctr = 0$
 6:     $B_{min} = N_{beams} * \gamma_{overlap}$
 7:     for $b = 1$ to $N_{beams}$ do
 8:         Transform $\mathbf{z}_b$ into equivalent depth observation $E_{z.obs}$ at coordinates
            $[E_{x.obs}, E_{y.obs}]$ relative to global frame.
 9:         if ($[E_{x.obs}, E_{y.obs}]$ is within $\tau_{lc}l$ proximity of past observations) then
10:             Perform GP Regression at $[E_{x.obs}, E_{y.obs}]$ using past observations (be-
                longing to both the current particle and its ancestors) within $\tau_{train}l$ prox-
                imity as training data, returning depth estimate $E_{z.est}$.
11:             Calculate the likelihood of $E_{z.obs}$ matching the current estimate:
                $lklhd(ctr) = p(E_{z.obs} - E_{z.est} = 0)$.
12:             Increment $ctr$.
13:         end if
14:     end for
15:     if ($ctr \geq B_{min}$) then
16:         return Joint Likelihood $w_i = \prod_{j=1}^{B_{min}} lklhd(j)$
17:     else
18:         return Do not include particle in resampling.
19:     end if
20: else
21:     Do not attempt particle resampling.
22: end if
```

This is very similar to the map weighting procedure used by the grid map representation in Section 4.2.3. However, in this case GP Regression is used to predict estimates of seabed depth, rather than recalling them from grid map cells.

Following from Algorithm 6 an observation model is required to first transform the observations **z** into the global frame. For the trajectory map representation we model each range observation (r) with zero mean Gaussian noise:

$$\mathbf{z} = r = h(\vec{\mathbf{x}}_v(t_k), \mathbf{E}) + \mathrm{v}, \quad \mathrm{v} \sim N(0, \sigma_r^2), \quad \sigma_r^2 = (\lambda r)^2 \tag{5.2}$$

where r is the range and λ is a constant value given by the precision of the multibeam sensor. The measurement function h is identical to the one described in Section 4.2.3 except that in this case the bearing (α) and along track angle (β) are treated as known quantities, rather than observations.

Using Equations 3.1 and 3.2 each observation r can be converted to an equivalent depth observation $E_{z.obs}$. As was the case for the grid map representation the uncertainty in range introduces uncertainty in the observations x, y location ($E_{x.obs}, E_{y.obs}$) through this transformation, particularly for observations with large grazing angles. This in turn means that uncertainty will exist in the training inputs used by our GP model, as these observations form the training data for GP Regression in future calls of the map weighting procedure. Methods of dealing with uniform uncertainty in the training input do exist (Girard & Murray-Smith 2003) but this problem is further compounded by the xy uncertainty being non-uniform and dependent on the vehicle pose at the time of observation.

Instead of directly modeling uncertainty in $E_{x.obs}, E_{y.obs}$ this issue is bypassed using a principled method that transforms a swath of multibeam observations, with uncertainty in range, into an equivalent set of depth observations with uncertainties only in depth. This is achieved by modeling the multibeam swath as a GP in the polar domain, allowing for the angular correlation between observations to be exploited. The multibeam swath is therefore represented as a single entity by this model, and through GP Regression can be raytraced to determine the uncertainty of intersecting the seafloor along any given path within the swath, which in this case is specified as a path that follows the z axes of the original observations. For a full review of this technique see Section 5.3.6. Note that while this method provides a principled approach to converting the range uncertainty it has been found to produce a negligible improvement in the datasets shown when compared to the simpler approach used by the grid map representation (see Section 4.2.3). However it is still implemented here for completeness, as the corrections it may provide in future (such as in more depth-discontinuous datasets) could prove to be very compelling. This investigation is the subject of future work.

It should also be noted that this approach assumes no uncertainty in the bearing and along track angle of the observation, which is often not the case as a small uncertainty arises in these angles from the finite beamwidth of the sonar aperture. As mentioned before there are methods available for handling this uncertainty (Girard & Murray-Smith 2003) but they were found to be computationally intractable. Despite this the results presented in Section 5.4 serve to validate the corrections that can still be achieved in navigation and mapping without taking this uncertainty into consideration.

With the swath of observations now transformed into the global frame the decision as to whether to perform a loop closure or not i.e. particle weighting/resampling, can now be made. As GP Regression allows for the prediction of depths in places that have not been directly observed, this naively allows loop closures to be performed at any time during a mission. However, regression far away from the training set will not be useful as all predictions tend toward the mean function at this limit. The rate at which this occurs decreases as the length scale hyperparameter of the GP model l increases, meaning that extrapolated predictions can be performed further away from the training set when the surrounding terrain is more spatially correlated. However performing particle weighting using observations of highly correlated terrain, such as a flat plane, is less likely to discern the particles with the most likely state hypotheses. A loop closure is therefore not attempted if the current length scale is above the user defined threshold l_{max}. For length scales below this threshold particles are only weighted and included in the resampling phase if they possess at least B_{min} observations that fall within $\tau_{lc}l$ distance of each of their respective maps, where τ_{lc} is a scale factor. This effectively allows us to dynamically increase or decrease the acceptable range of extrapolation based on the correlation in the local seabed.

For the observations that pass this criteria an estimate of seabed depth ($\mu_{E_{z.est}}$) and corresponding uncertainty ($\sigma^2_{E_{z.est}}$) is calculated at the observations x, y location using GP Regression based on training data taken from the particle's map (see Algorithm 5). However, while the GP covariance function provides the desired computational speedup for regression in large sparse datasets this benefit reduces as the dataset gets smaller and denser relative to l. In this case the posterior distribution of the GP is approximated by selecting a subset of training points that are within proximity of the locations to be tested, the size of this subset window equal to $\tau_{train}l$, where τ_{train} is a user-defined constant. This ensures that any errors introduced from this approximation are bounded during the operation, approaching zero as τ_{train} is increased.

Once the GP regression has been performed the observation $E_{z.obs}$ can be weighted based on how consistent it is with $E_{z.est}$ using the same formula implemented in the grid map representation:

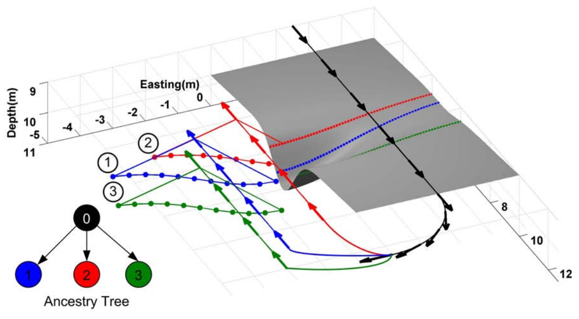
$$likelihood = p((E_{z.est} - E_{z.obs}) = 0) = \frac{e^{-\frac{1}{2}\frac{(\mu_{E_{z.est}} - \mu_{E_{z.obs}})^2}{\sigma^2_{E_{z.obs}} + \sigma^2_{E_{z.est}}}}}{\sqrt{2\pi(\sigma^2_{E_{z.obs}} + \sigma^2_{E_{z.est}})}} \tag{5.3}$$

As was the case for the grid map representation each particle included in the resampling phase will have a different number of observations that have been successfully assigned likelihood measures, the minimum number in this case tracked by the variable L_{min}. A single weighting factor can therefore be generated for each particle by calculating the joint likelihood of L_{min} of the likelihood measures available to each particle i.e. the product.

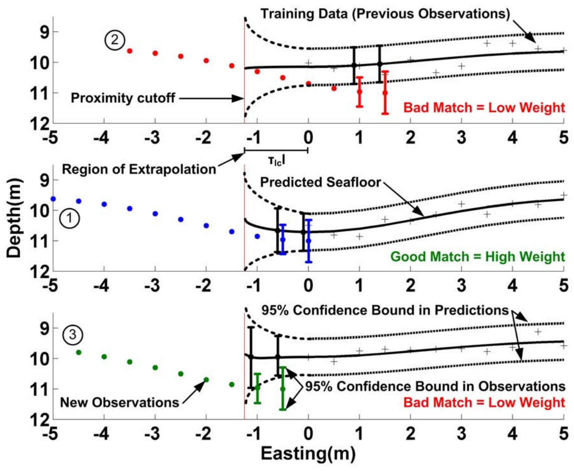
An example of the new map weighting procedure is shown in Figure 5.2(a). Here previous particle resampling has caused the set to collapse down onto a single past trajectory belonging to the root ancestor Particle 0. Upon receiving the new multibeam swath a loop closure is detected, as the particles have observations that fall within proximity to their maps, shown in Figure 5.2(b). Each particle then uses a local patch of their own map to generate a prediction (even when no overlap exists) and compares it against the observation. The joint likelihood of these observation/estimate pairs is then calculated. Note that the joint likelihood is a relative measure that only holds meaning if it is generated from the same number of observations for each particle. As such only two observations are used in this case, as increasing this number would require some particles to extrapolate predictions beyond the window where they remain accurate enough for use. This effectively shows how loop closures can be performed with little to no overlap by using GP Regression to generate predictions.

5.3.5 Map Updating

As each particle stores its map as a simple list of trajectories and associated sonar swaths it is relatively straightforward to update each particle's map once particle weighting/resampling has been completed. All that is required is to simply add the particle's current pose and



(a) Loop Closure Geometry



(b) Particle Observations and Generated Predictions

Figure 5.2: An example of the GP method used to weight each particle given a set of $N = 3$ with a swath of 11 multibeam observations. In part a) the trajectories for Particle's $1, 2, 3$ are shown in blue, red, green respectively. The trajectory of ancestor Particle 0, which they all inherit, is shown in black. The local map section that each particle uses to match its observations against is shown in grey. This map section is different for each particle if it is generated from a trajectory section in the particles ancestry that is not shared. However, in this case the neighboring trajectory within proximity belongs to Particle 0, so the corresponding map is common to all (useful for illustrative purposes). The colored broken lines indicate the slices of the predicted map that are within the plane of each particle's observations, displayed in Part b). Here GP regression is used by each particle to generate predictions to compare against their observations. For illustrative purposes the GP regression is only shown in 2D, where in practice the full 3D map is utilised to generate these predictions. Equation 5.3 is then used to weight the particles based on these observation/estimate pairs.

swath index to the back of these respective lists, as described in Algorithm 7.

Algorithm 7 Step 9 from Algorithm 1:Grid Map Updating

```
1: **Access** ancestry tree record for current particle.
2: **Add** current state hypothesis of particle $[x_{v_i}, y_{v_i}]$ to end of trajectory list held by
   particle.
3: **Add** index of sonar swath to end of observation list held by particle.
```

5.3.6 Raytracing Uncertainty from Correlated Observations

This section details a technique by which range observations from a multibeam swath with uncertainty in range are converted into equivalent depth observations with uncertainty in depth through the use of GP Regression in the polar sensor frame (range/bearing/along track angle). To begin with the GP model is learned using the multibeam swath as training data. In this case the training inputs are the bearing and along track angle of each observation $\mathbf{\Theta} = \{\alpha_i, \beta_i\}_{i=1}^{B}$ and the training outputs are the range observations $\mathbf{Z} = \{r_i\}_{i=1}^{B}$ with covariance matrix $\mathbf{W} = diag(\sigma_{r1}^2 ... \sigma_{rB}^2)$. This is demonstrated in Figure 5.3(a) where a multibeam swath has been modeled by a GP in the polar domain and GP Regression has been used to generate the probability distribution of the range to the seafloor for every angular coordinate Θ^* within the swath. For clarity each beam in this example possesses zero along track angle, allowing us to represent the swath in 2D.

Each vertical slice of the plot in Figure 5.3(a) corresponds to the probability distribution of the predicted range at that angular coordinate Θ^* i.e.:

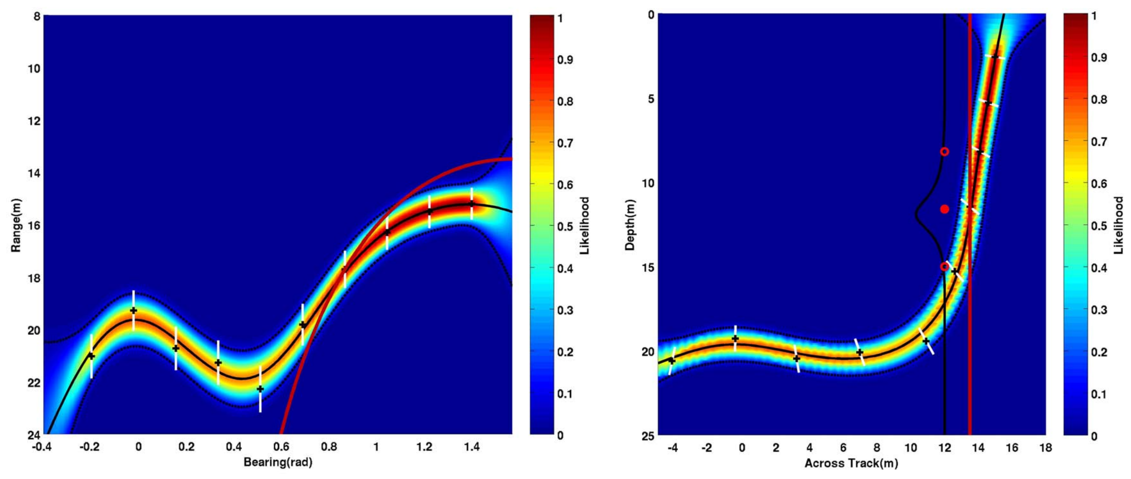
$$P(Z^*|\mathbf{\Theta}, \mathbf{Z}, \Theta^*) \sim N(\mu_{Z^*}, cov(Z^*)) \tag{5.4}$$

where μ_{Z^*} and $cov(Z^*)$ are calculated using the standard regression equations described in Algorithm 5.

By further specifying a particular range value r_{test} the probability that the seafloor intersects between r_{test} and $r_{test} + dl$ for that angle Θ^* can be estimated:

$$P(Z^* \in \{r_{test}, r_{test} + dl\}|\mathbf{\Theta}, \mathbf{Z}, \Theta^*) = \frac{dl}{\sqrt{2\pi cov(Z^*)}} e^{\frac{-(r_{test} - Z^*)^2}{2cov(Z^*)}} \tag{5.5}$$

where dl is the distance between successive test inputs in polar space.



(a) GP Regression in the polar domain

(b) GP Regression mapped into the cartesian domain

Figure 5.3: An example of how polar observations with uncertainty in range are converted to depth observations with uncertainty only in depth, using GP Regression in the polar domain. In part a) the observations received from the multibeam are shown by black crosses, the corresponding 95% confidence bounds indicated in white. These are used to train a GP, resulting the predicted seafloor and 95% confidence bounds indicated by the black solid and broken lines respectively. Overlayed is the corresponding likelihood measures of the seafloor intersecting at any given range and polar coordinate. In part b) these trends are mapped from the polar domain into the cartesian domain. The uncertainty in depth assigned to each original range observation is calculated by sampling the probability of seafloor intersection up the depth line that passes through it. This is shown for the 7th observation by the vertical red line, corresponding to sampling along the red contour in a). Conditioning these sampled probabilities on one seafloor intersection along this contour results in the probability distribution shown by the vertical black line, scaled and reflected in the plane for clarity. A Gaussian is then fitted to this distribution, the mean and 95% confidence bounds of which are indicated by the solid and empty red circles respectively.

If the probabilities of the swath intersecting the seafloor at different range values were sampled while keeping Θ^* fixed, a discrete probability distribution would be produced that converged to $P(Z^*|\mathbf{\Theta}, \mathbf{Z}, \Theta^*)$ as dl was decreased. It is here that the main observation that is the basis of this approach is made, namely that these calculated probabilities are still meaningful if sampling instead occurs along the contour $\mathcal{C}$ that maps to the vertical line at the x, y location of any given observation in the local-level frame. For the $7th$ observation in the current example this corresponds to sampling along the red contour shown in Figure 5.3(a) that maps to the red depth line shown in Figure 5.3(b). For clarity no pitch or roll is present in this example, which allows the scenario to be represented in 2D as any given depth line will map to a contour that is within the range/bearing plane.

In instances where pitch and roll does exist the mapping of the depth line $[x_{obsll}, y_{obsll}, z_{test}]$ (defined in the local level frame where only z_{test} varies) into the sensor frame is achieved by first transforming the current sample point into along track (b), across track (a) and depth values (d). Similar to Equation 4.4 this is given as:

$$\begin{pmatrix} b \\ a \\ d \end{pmatrix} = {}^{s}_{l}\mathbf{R} \begin{pmatrix} x_{obsll} \\ y_{obsll} \\ z_{test} \end{pmatrix} \tag{5.6}$$

where ${}^{s}_{l}\mathbf{R}$ is the local-level to sensor frame Directional Cosine Matrix derived from the pose of the vehicle. These are then transformed into range, bearing and along track angles using the geometry described by the measurement function h:

$$(r, \alpha, \beta)^T = \begin{pmatrix} \sqrt{b^2 + a^2 + d^2} & \arctan(\frac{a}{d}) & \arctan(\frac{b}{d}) \end{pmatrix}^T \tag{5.7}$$

Incrementing z_{test} therefore corresponds to sampling along $\mathcal{C}$ which produces a set of probabilities that describe the chance of the seafloor intersecting this depth line at different depths. However it is important to note that this is not a true probability distribution. The GP in the polar sensor frame models range observations as a continuous function of bearing, providing a one to one mapping. This still allows for the possibility of a surface with overhang i.e. one that could intersect a chosen depth line multiple times. As such integrating this probability set along $\mathcal{C}$ will often result in a value greater than one.

Fortunately this problem is overcome by enforcing the condition already assumed by the GP model in the cartesian frame F_{global} i.e. that the depth of the surface can also be modeled as a continuous function of x, y, which assumes no overhang. This criteria is therefore enforced in the probability set by conditioning it on the event that only a single intersection along the chosen depth line occurs. For the unconditioned discrete probability set P_z consisting of D samples at depths $\{z_i\}_{i=1}^{D}$ this corresponds to:

$$\begin{aligned} P_{zi}|single\ intersection &= \frac{P_{zi} \cap P_{single}}{P_{single}} \\ &= \frac{P_{zi} \prod_{j=1 \neq i}^{D} (1 - P_{zj})}{\sum_{k=1}^{D} P_{zk} \prod_{j=1 \neq i}^{D} (1 - P_{zj})} \end{aligned} \tag{5.8}$$

Note that the denominator of Equation 5.8 is simply the sum of the numerator over all samples and hence normalises the new probability set, creating a valid probability distribution. This distribution is then approximated by fitting a Gaussian to it, which forms the observation (μ_{z_O}) with uncertainty ($\sigma^2_{z_O}$), now transformed into the cartesian frame along the depth axis. Note that this also invokes a correction to the observation itself, inferred from the angular correlation between observations.

Referring back to Figure 5.3(b) demonstrates the benefit of using this technique. At the highlighted observation there is significant slope that, due to the position of the multibeam, creates significant uncertainty in the slope's across track coordinate. Intuitively this should therefore create a large uncertainty in depth if a test point within this region is specified, as at this point it is uncertain whether the top or bottom of the slope will be observed. Using this new procedure a probability distribution can be extracted that encodes this behavior. While this distribution is not Gaussian (e.g. there is a very slight increase in probability of surface intersection at zero depth) modeling it as such can be seen to be a good approximation, which is necessary if this observation is to be used for weighting and later as training data.

In summary the uncertainty that the range observations create in x, y is effectively accounted for by modeling the swath as a collection of inter-dependent observations. This allows them to work together to infer probability distributions along any arbitrary contour within the observed space, provided that the chosen contour is guaranteed to only intersect the observed surface once. Sharp edges and discontinuities in depth are effectively handled as these often map to smooth continuous trends in the range domain where GP Regression is performed. Extending this approach to instances where overhang does occur is the subject of future work and is discussed in Chapter 6.

5.4 Results

In the previous chapter the BPSLAM algorithm utilised a grid map representation and demonstrated how observations of the seafloor structure improved the estimated trajectory and resulting map when compared to dead reckoning fused with USBL or LBL observations (Barkby, Williams, Pizarro & Jakuba 2011). Results were also compared with a pre-existing state of the art bathymetric SLAM technique, confirming that similar results could be achieved at a fraction of the computational run time. Here the same trials are repeated using the new trajectory map representation to see how the performance of BPSLAM differs. Descriptions of the missions can be found in Section 4.3. For these trials the setup of the particle filter remains unchanged.

5.4.1 Case Study 1: Butts Reef Pockmarks

As this survey mission does several complete passes over previously explored terrain the scaling factor τ_{lc} of the trajectory map is set to zero, meaning that particle weighting based on extrapolated predictions will not be attempted. This is done to increase the computational speed of the filter, as there is already ample opportunities for resampling available (the extrapolative ability of the trajectory map is tested separately in Section 5.4.3). In addition the maximum log length scale allowed for loop closing is set to ($l_{max} = 7$). This value was chosen as it effectively prevents the filter from attempting loop closures within flat plane regions and has shown to translate well across missions.

Based on the noise characteristics of the multibeam sensor, the uncertainty in range will be modeled as 1.5% of the measured range to account for attenuation in the water column and possible uncertainty in the speed of sound.

Navigation and Mapping Comparison

Figure 5.4 demonstrates how similar navigation solutions are achieved for both variants of the BPSLAM filter, using a particle set size of $N = 640$. This is due in part to their similar resampling behavior, shown in Figure 5.5. Here both BPSLAM variants share the same cloud growth rate and collapse during similar time periods.

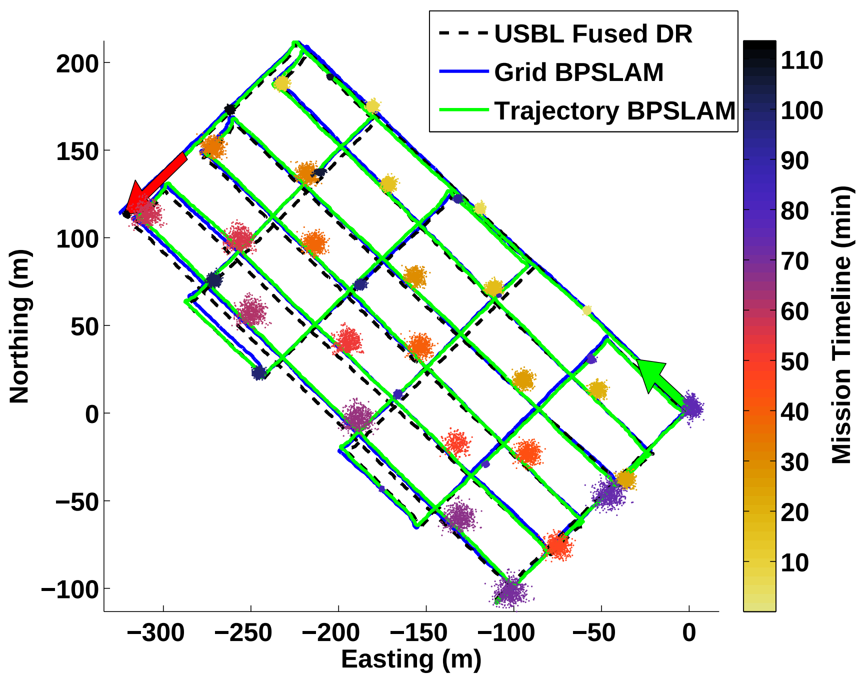


Figure 5.4: The tracklines produced by three different navigation solutions are shown; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively (longest tracklines belong to the initial grid survey). The evolution of the particle cloud for BPSLAM with trajectory maps is also shown, changing from light to **dark** as the mission progresses.

Figure 5.6 presents a comparison of the resulting maps produced by USBL fused navigation and the two BPSLAM solutions (see Appendix A for larger reproductions). Corresponding maps and histograms of the registration error described in Section 3.9 are also provided. Figures 5.6(a)-(f) correspond to Figures 4.6(d)-(i) from Section 4.3 respectively. Comparing Figures 5.6(g)-(i) with the previous results produced by Grid Map BPSLAM show how the same level of correction is produced by Trajectory Map BPSLAM, condensing the corresponding histogram into the region of low registration error. For the USBL fused DR, Grid map BPSLAM and Trajectory map BPSLAM solutions the average mapping error were calculated as 0.216 m, 0.197 m and 0.195 m respectively. For this run Trajectory Map BPSLAM required 1.2 GB of memory and took 45 min to run, which is 40 % of the total mission time.

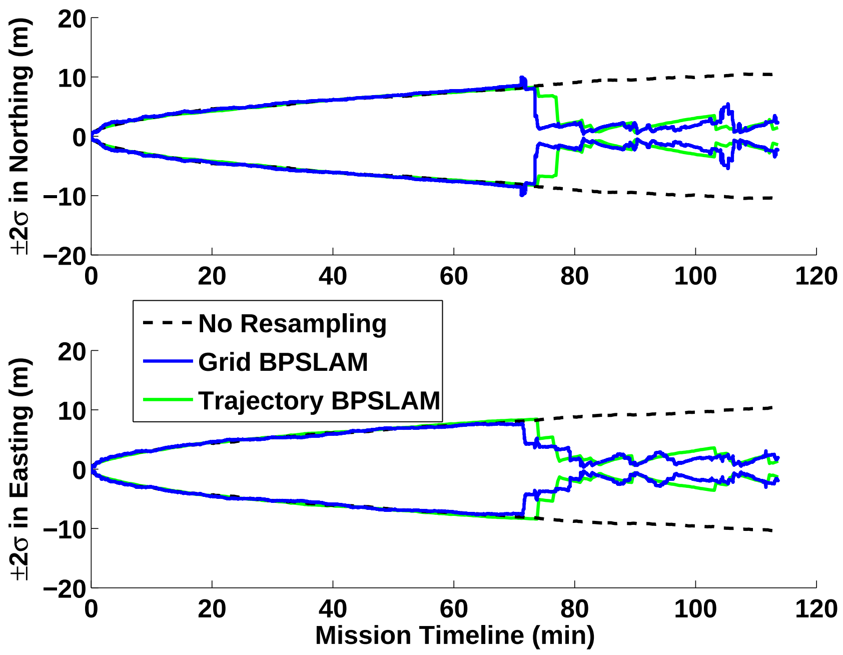


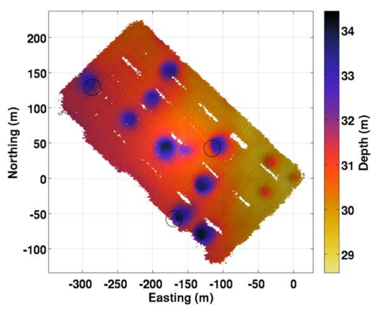
Figure 5.5: Variation in the 95% confidence interval of the vehicle's x, y position for both variations of BPSLAM.

Accuracy, Consistency and Efficiency Analysis

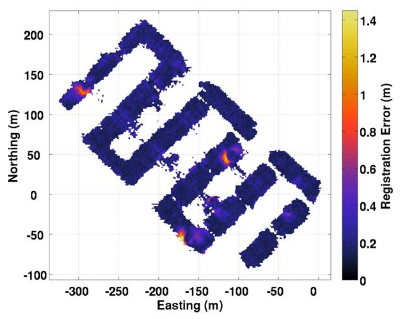
To investigate the consistency and repeatability of these results the BPSLAM was run repeatedly in batches of 25 using a different fixed particle size for each batch ($N = 10, 20, 40, 80, 160, 320, 640$). For each run the processing time, memory usage and average registration error of the map were recorded. This procedure was repeated for both map variations of the BPSLAM filter, producing the results shown in Figure 5.7.

Figure 5.7(a) demonstrates how increasing the particle set size converges the BPSLAM solution towards a more self-consistent map while also improving the repeatability of this result from run to run. Additionally it can be seen that only a small particle set size ($N > 160$) is required for the BPSLAM filter to repeatedly produce a more self-consistent map than when using DR or the USBL fused DR solution.

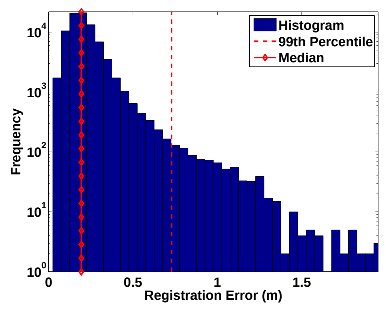
Comparing both map variants of BPSLAM shows no significant difference in the consistency and repeatability of the maps deliverable. However Figure 5.7(b) demonstrates a significant difference in the computational resources required for each approach to run. Whereas BPSLAM using grid maps is significantly faster than the new trajectory map approach,



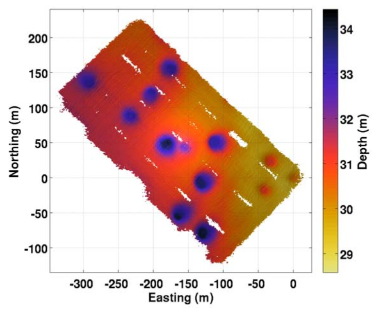
(a) USBL Fused DR Bathy. Map



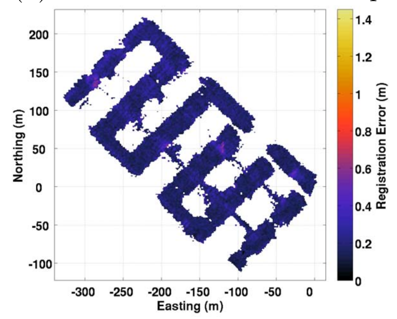
(b) USBL Fused DR Error Map



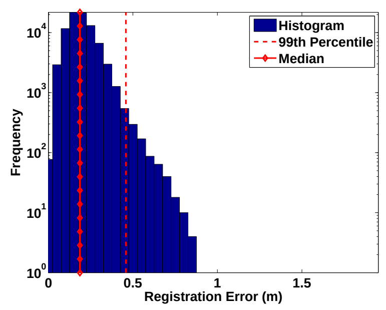
(c) USBL Fused DR Error Hist.



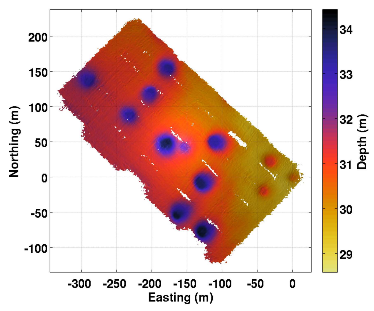
(d) BPSLAM Grid Bathy. Map



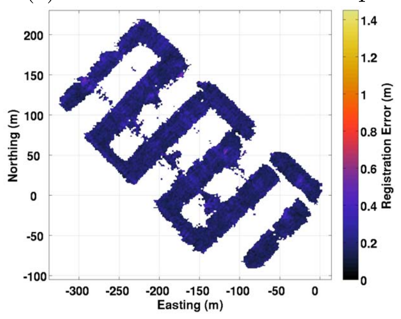
(e) BPSLAM Grid Error Map



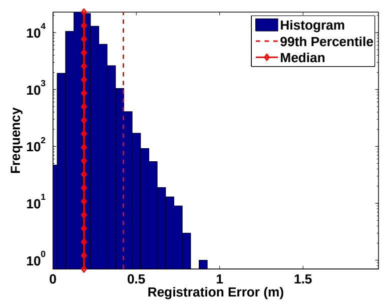
(f) BPSLAM Grid Error Hist.



(g) BPSLAM Traj. Bathy. Map

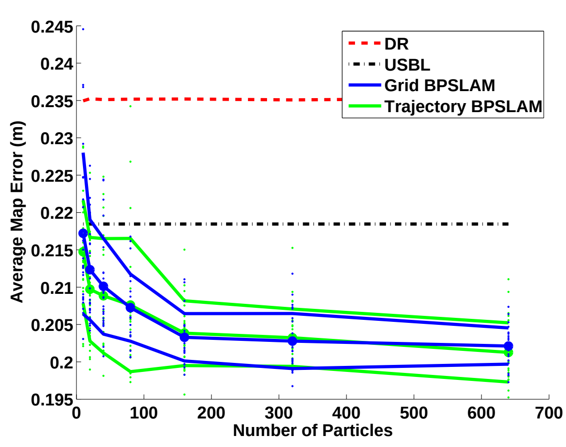


(h) BPSLAM Traj. Error Map

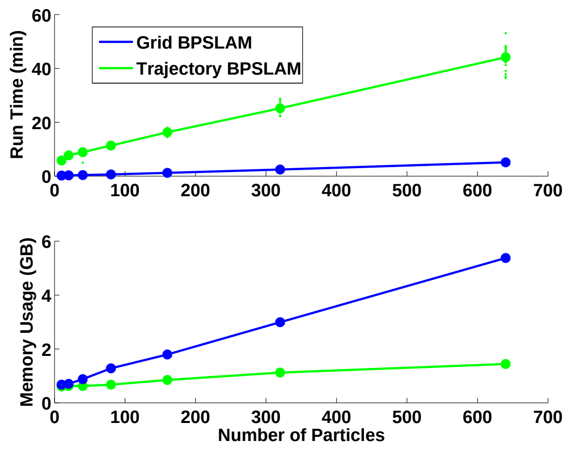


(i) BPSLAM Traj. Error Hist.

Figure 5.6: Bathymetric maps generated using three different navigation solutions. The corresponding error maps and histograms are also provided, detailing the misregistration between overlapping swaths within each cell. Comparison shows both variants of the BP-SLAM filter providing a reduction in the mapping error, the most prominent circled in black, when compared to the map produced by the USBL fused solution. This is also reflected in the histograms, shifting the registration error measurements of the map towards lower values.



(a) Variation in map error with particle set size.



(b) Variation in run time and memory usage with particle set size.

Figure 5.7: a) Variation in the map error with a varying particle set size using both grid map and trajectory map representations. The large dots represent the mean error used in each batch of 25 runs (shown by the smaller dots). The outer solid lines represent the $\pm 2\sigma$ bound in each batch. Results show an increase in the accuracy and consistency of the map produced in both variations of the filter when the number of particles is increased, though no significant improvement can be seen between the two methods. In b) BPSLAM with the trajectory map demonstrates a significant increase in computational run time when compared to the grid map approach. However this is compensated by a relatively lower memory requirement.

this is countered by the former requiring significantly larger memory. In this way the two approaches complement each other in their abilities. For this mission using BPSLAM with grid maps can be thought of as the best choice, since full overlap is available and the memory requirements ($\sim$4GB) are achievable by most computers available today. However for larger scale missions, or for missions containing less overlap, the proceeding case studies will serve to validate the use of the trajectory map as a more appropriate choice.

5.4.2 Case Study 2: TAG Hydrothermal Vent

As was done in Section 5.4.1 the scaling factor τ_{lc} of the trajectory map is set to zero for this mission as abundant map overlap exists for this mission (some areas are revisited by the ROV up to ten times). Based on the noise characteristics of the multibeam sensor the percentage uncertainty in the range observations will be modeled as 1.0%.

Navigation and Mapping Comparison

Figure 5.8 demonstrates again how similar navigation solutions are achieved for both variants of the BPSLAM filter, using a particle set size of $N = 640$. The resampling behavior shown in Figure 5.9 is also similar, sharing many of the resampling events that cause large reductions in the size of the particle cloud.

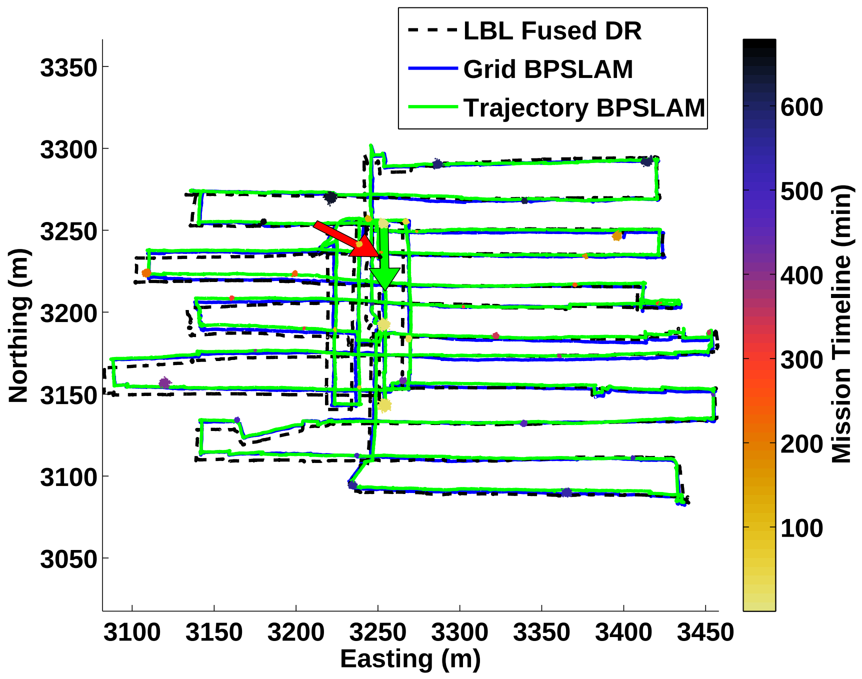


Figure 5.8: Tracklines produced by three different navigation solutions for the TAG mission; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively. The evolution of the particle cloud for BPSLAM with trajectory maps is also shown, changing from light to **dark** as the mission progresses.

To investigate the error in the Trajectory Map BPSLAM navigation solution the LBL innovations are again analysed, shown in Figure 5.10. As discussed in Section 4.3.2 the precision of the LBL observations ($\approx$ 4 m) is not accurate enough to discern the most accurate solution between LBL fused DR and the two BPSLAM solutions in this case. However it is accurate enough to conclude that the Trajectory Map BPSLAM navigation solution is more accurate than that produced by DR.

Figure 5.11 presents a comparison of the resulting maps produced by the LBL fused DR, Grid Map BPSLAM, Trajectory Map BPSLAM and Sub-mapping filters. Corresponding maps and histograms plotting the offline registration error measure described in Section 3.9

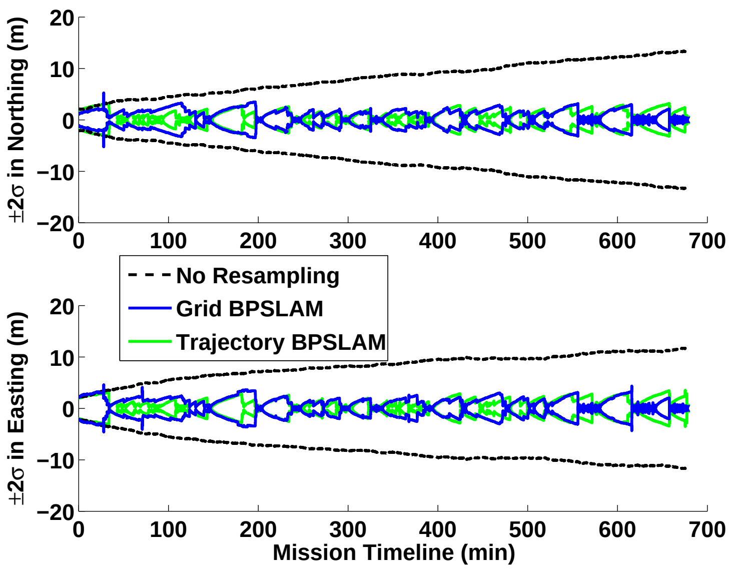
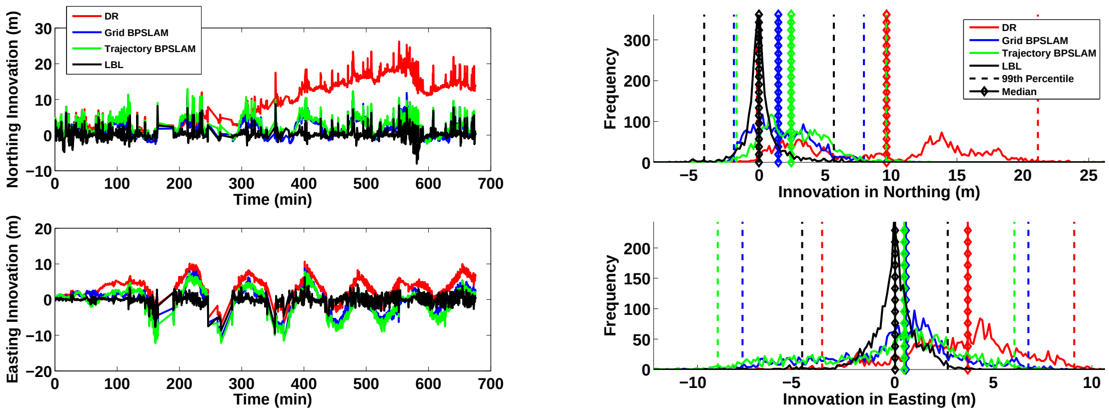


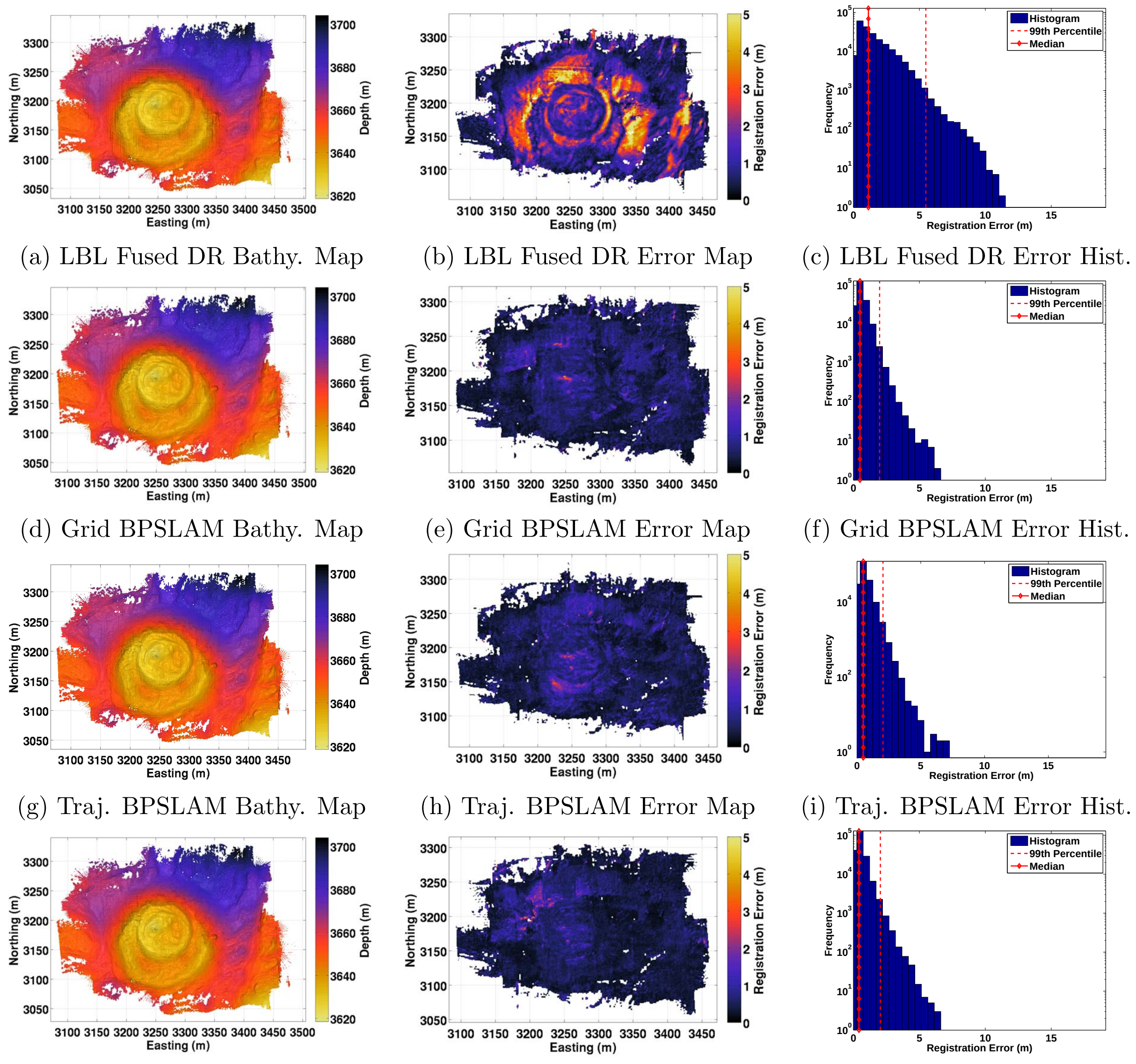
Figure 5.9: Variation in the 95% confidence interval of the vehicle's x, y position for both variants of the BPSLAM filter.



(a) Innovation of LBL observations in time (b) Histogram of Innovations from LBL observations

Figure 5.10: Innovation of raw LBL observations during the TAG mission for four different navigation solutions. Comparison shows a distinct divergence of the DR solution away from the LBL observations, whereas the LBL fused DR and both BPSLAM solutions retain a much closer proximity, confirming an improvement in navigational accuracy.

are also provided. As expected both BPSLAM variants were equally successful in removing mapping artifacts created by errors in navigation, providing superior performance when compared to LBL aided navigation. This is also confirmed by the average registration error for the LBL fused DR, Grid map BPSLAM, Trajectory map BPSLAM and Sub-mapping solutions, calculated as 1.490 m, 0.585 m, 0.590 m and 0.515 m respectively.



(a) LBL Fused DR Bathy. Map (b) LBL Fused DR Error Map (c) LBL Fused DR Error Hist.

(d) Grid BPSLAM Bathy. Map (e) Grid BPSLAM Error Map (f) Grid BPSLAM Error Hist.

(g) Traj. BPSLAM Bathy. Map (h) Traj. BPSLAM Error Map (i) Traj. BPSLAM Error Hist.

(j) Sub-mapping Bathy. Map (k) Sub-mapping Error Map (l) Sub-mapping Error Hist.

Figure 5.11: Bathymetric maps generated using both BPSLAM map variations. The corresponding error maps are also provided, detailing the misregistration between overlapping swaths within each cell. Comparison shows the three SLAM filters providing a significant reduction in the mapping error when compared to the map produced by the LBL fused DR solution.

The above results show no significant difference in the level of map corrections attained when running the BPSLAM filter with different map representations. However in terms of computational run time the grid map representation took 8.7 minutes to run, whereas the trajectory map took 156.8 minutes (22.4% of mission time). While this new approach is significantly longer its memory requirement was found to be only 2.45 GB whereas the former required 9.76 GB. As this memory requirement significantly exceeds the capabilities of most computers the trajectory map can be thought of as the best choice for this mission, whose computational run time is still far improved over the Sub-mapping algorithm which required overnight processing to achieve its result.

BPSLAM with Sparse Bathymetry

While the preceding section has shown how BPSLAM can improve navigation and mapping it remains to be seen whether similar corrections can still be achieved without requiring a dedicated multibeam depth profiler, as this sensor can be absent from vehicles with mission goals other than bathymetric mapping.

To investigate this the TAG survey was repeated, except in this case the multibeam depth profiler was replaced by the DVL as the mapping sensor. In contrast to the multibeam (which provides 120 beams over 120°) the DVL bathymetry is extremely sparse, consisting of 4 beams as shown in Figure 5.12. To select an appropriate τ_{lc} scaling factor we specified

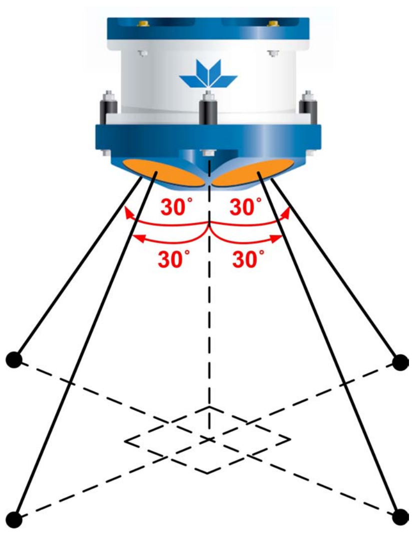


Figure 5.12: Four beam pattern provided by DVL.

that predictions should be at most no further than 1.25 m from their training set. Given l_{max} this corresponded to a scaling factor of $\tau_{lc} = 0.18$. For this run $N = 320$ particles were used.

Figure 5.13 presents the navigation solution generated by Trajectory Map BPSLAM using DVL bathymetry for mapping. As can be seen BPSLAM is still successful in providing corrections to navigation, creating a clear visible shift of the tracklines away from the DR solution and towards the corrected solutions. The resampling behavior seen in Figure 5.14 demonstrates that the particle cloud in this case is significantly less constrained than when using the multibeam, due to the sparse bathymetry providing less opportunities to resample. However some resampling events are still evident, which are enough to produce the corrections shown.

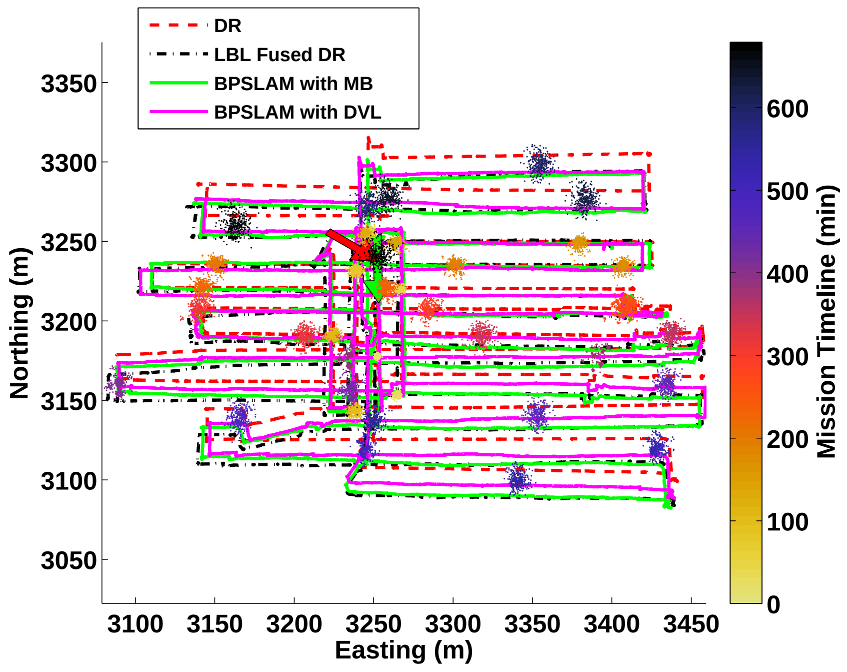


Figure 5.13: Tracklines produced by four different navigation solutions for the TAG mission; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively. The evolution of the particle cloud for BPSLAM with trajectory maps is also shown, changing from light to **dark** as the mission progresses.

Figure 5.15 analyses the LBL innovations observed for Trajectory Map BPSLAM when only DVL bathymetry is available. As shown the innovations generated by this new trajectory is larger than that produced by Trajectory Map BPSLAM when the full multibeam is available, bordering on magnitudes that suggest a reduction in navigational accuracy when compared to the latter. Despite this Figure 5.15 confirms that the accuracy of this navigation solution still exceeds that produced by DR.

Figure 5.16 presents a comparison of the resulting maps produced by the DR, LBL fused

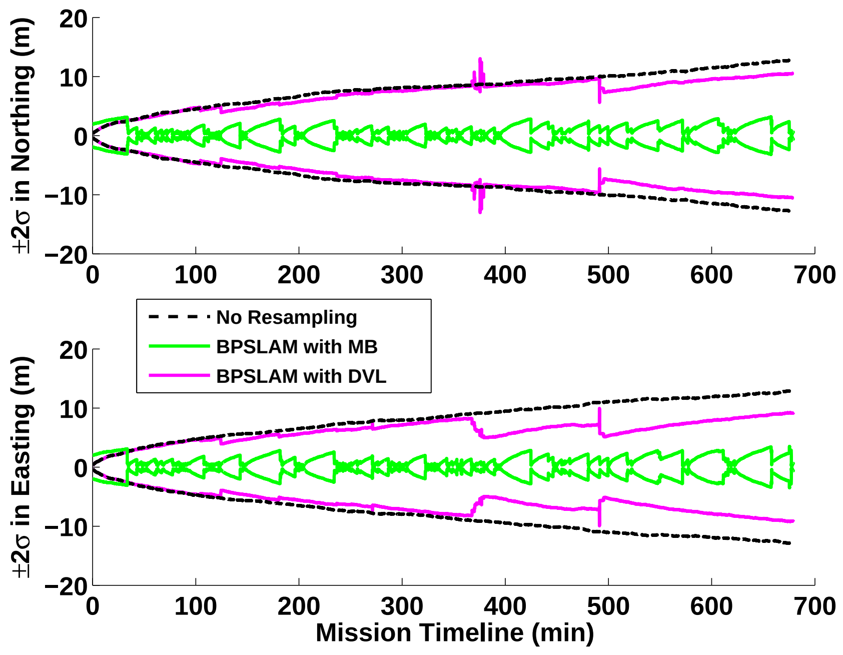
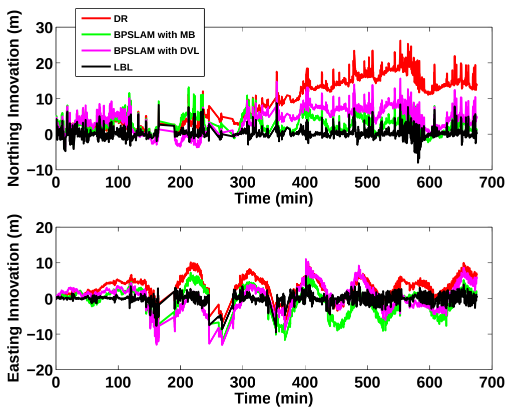
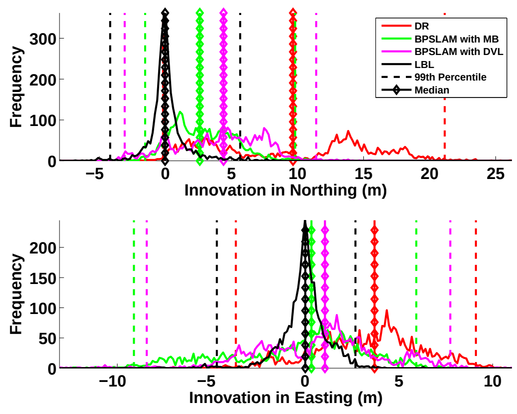


Figure 5.14: Variation in the 95% confidence interval of the vehicle's x, y position for BPSLAM with different mapping sensors.



(a) Innovation of LBL observations in time



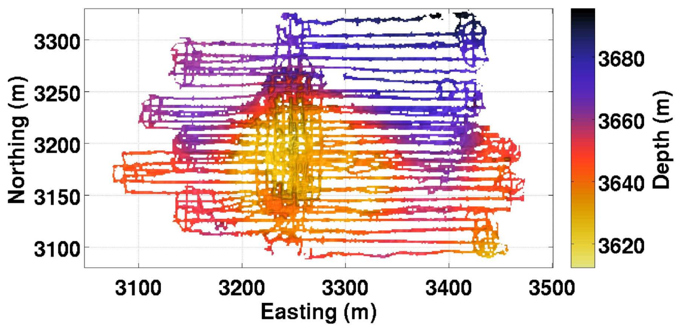
(b) Histogram of Innovations from LBL observations

Figure 5.15: Innovation of raw LBL observations during the TAG mission for four different navigation solutions. Comparison shows a distinct divergence of the DR solution away from the LBL observations, whereas the LBL fused DR and both BPSLAM solutions retain a much closer proximity, confirming an improvement in navigational accuracy.

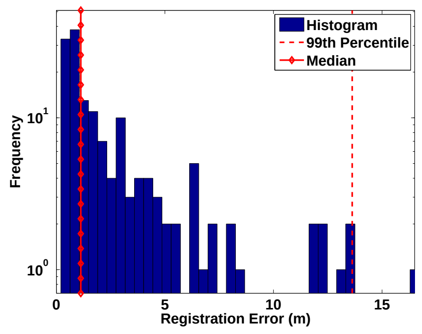
DR and BPSLAM solutions using the DVL bathymetry. In addition we compare the map that is produced when the previous Trajectory Map BPSLAM navigation solution (created using the multibeam) is applied to the DVL bathymetry. In Figure 5.16(a) the registration error in the map caused by DR can be seen. In comparison to Figures 5.16(c),(e) and (g) a significant amount of this error is resolved. However due to the sparse bathymetry it is not clear how self-consistent these maps are compared to each other. Figures 5.16(b),(d),(f) and (h) provide a more clear comparison of these registration errors. As can be seen BPSLAM using DVL bathymetry creates less registration error in these sparse maps than compared to DR or LBL fused DR. However the using multibeam bathymetry still provides the most self-consistent map, suggesting that its navigation solution is the most accurate.

Note that using DVL bathymetry has created maps with significantly less registration error measurements on which to base comparisons on. To additionally verfiy the performance of the Trajectory Map BPSLAM filter running off DVL bathymetry, its navigation solution is applied to the multibeam bathymetry, creating a dense map that can be compared to those previously presented in Section 5.4.2. These are shown in Figure 5.17, which provide the same conclusions concerning performance between navigation solutions, though the differences between the maps created by BPSLAM based on multibeam bathymetry, LBL fused DR and BPSLAM based on DVL bathymetry are more distinguishable. This validates the use of BPSLAM with DVL bathymetry as a means of improving navigation during missions where multibeam bathymetry is not available, providing a level of correction comparable to using LBL fused DR.

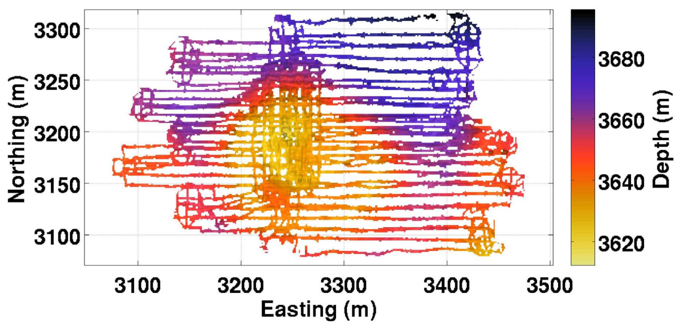
For this trial the BPSLAM filter took 13.3 min to process the mission (1.9% of mission time). This is a significant improvement when compared to the 156.8 min that was required when using the full multibeam, due mainly to the reduction of observations that had to be processed. Similarly the memory requirements were also reduced from 2.45 GB to 0.81 GB when compared to using full multibeam bathymetry.



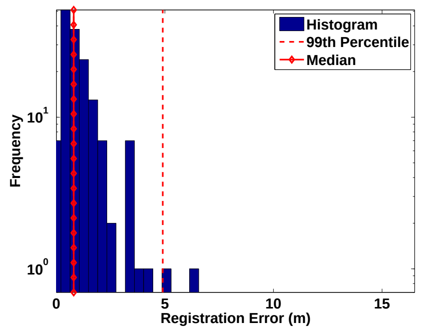
(a) DR Bathy. Map



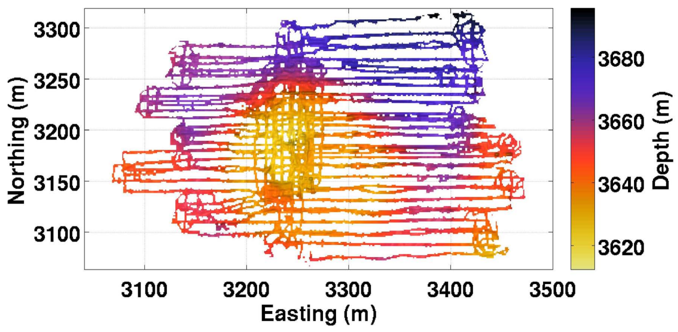
(b) DR Error Map



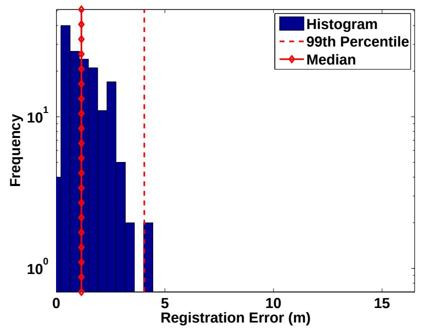
(c) LBL Fused DR Bathy. Map



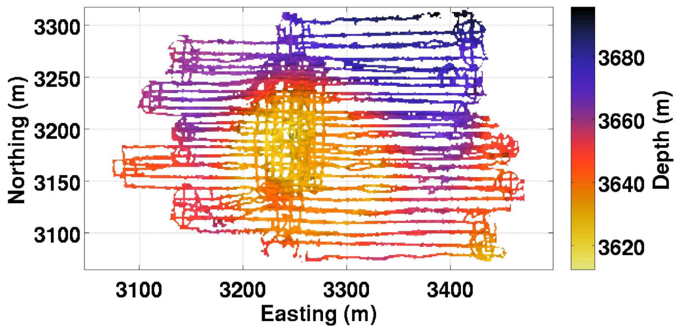
(d) LBL Fused DR Error Map



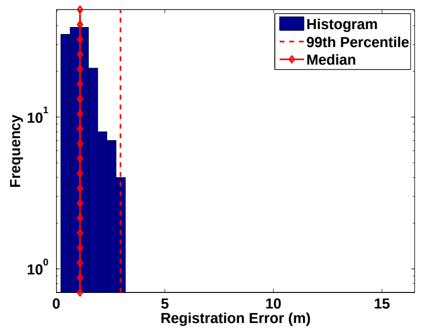
(e) BPSLAM DVL Bathy. Map



(f) BPSLAM DVL Error Hist.

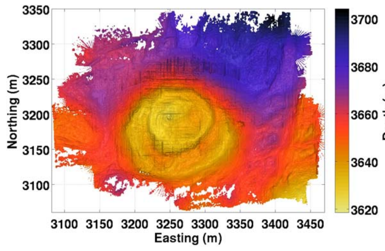


(g) BPSLAM MB Bathy. Map

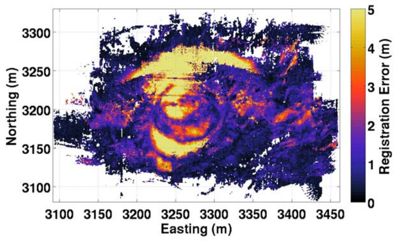


(h) BPSLAM MB Error Hist.

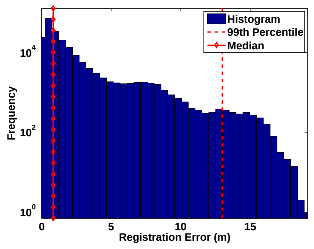
Figure 5.16: Bathymetric maps generated using four different navigation solutions with DVL bathymetry from TAG mission. The corresponding error histograms are also provided, detailing the misregistration between overlapping swaths within each cell.



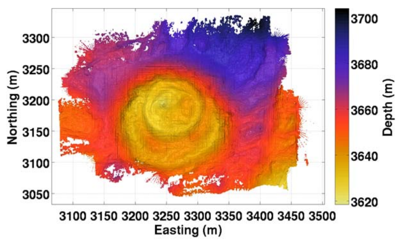
(a) DR Bathy. Map



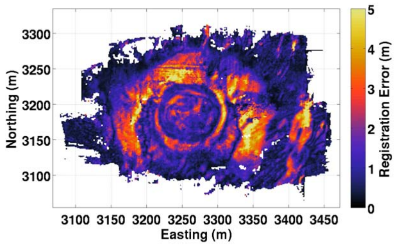
(b) DR Error Map



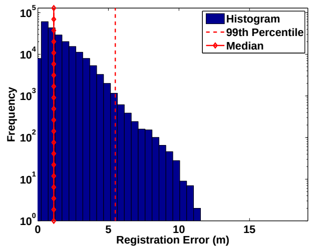
(c) DR Error Hist.



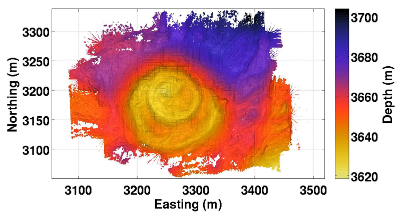
(d) LBL Fused DR Bathy. Map



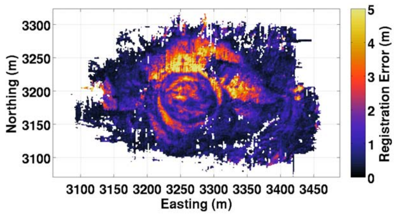
(e) LBL Fused DR Error Map



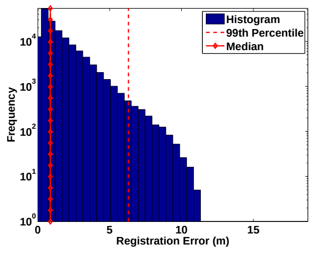
(f) LBL Fused DR Error Hist.



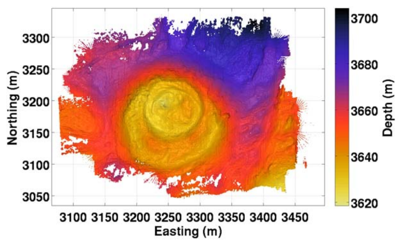
(g) BPSLAM DVL Bathy. Map



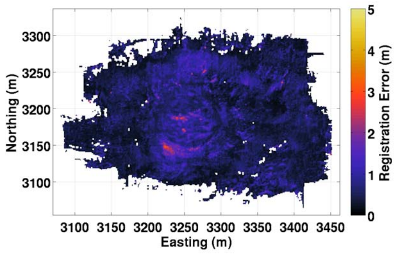
(h) BPSLAM DVL Error Map



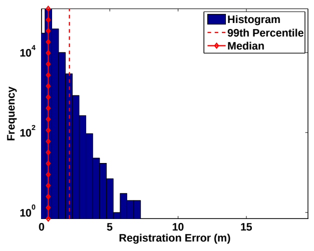
(i) BPSLAM DVL Error Hist.



(j) BPSLAM MB Bathy. Map



(k) BPSLAM MB Error Map



(l) BPSLAM MB Error Hist.

Figure 5.17: Bathymetric maps generated using four different navigation solutions with multibeam bathymetry from TAG mission. The corresponding error maps are also provided, detailing the misregistration between overlapping swaths within each cell. Comparison shows that performing BPSLAM based only on DVL bathymetry provides navigation corrections comparable to LBL fused DR, in terms of mapping quality. However performing BPSLAM based on multibeam bathymetry still provides the most self-consistent map, suggesting superior navigational accuracy.

5.4.3 Case Study 3: Haakon Mosby Mud Volcano

The purpose of this case study is to investigate the ability of BPSLAM to provide corrections to navigation and mapping when the amount of map overlap available during a mission is progressively reduced. To this end we test our algorithm on a large scale mission using bathymetric and navigation logs from a real survey undertaken by the research class AUV *Sentry*, operated by the National Deep Submergence Facility (NDSF) at the Woods Hole Oceanographic Institution (WHOI) (Catanach & German 2011).

Figure 5.18: The Sentry AUV is a research vehicle capable of exploring the ocean down to 4,500 meters (14,764 feet) depth. Sentry's hydrodynamic shape allows faster ascents and descents and is also capable of collecting oceanographic and benthic data for a wide range of applications. It features an extensive suite of sensors, including multibeam sonar, CTD, camera's and magnetometers (*Image Source:www.whoi.edu*)

The survey is of the Haakon Mosby Mud Volcano, located 1250 m deep south of Svalbard. Figure 5.19 presents a bathymetric map of this region generated from an older survey. As indicated by the black box in Figure 5.19 the desired size of the new survey is $\approx 1.65\ km^2$. To achieve full coverage of this area the tracklines navigated for this mission are a standard "lawnmower" pattern, running primarily North to South and carried out underwater at 20 m altitude to produce swath widths of $\approx$ 60 m. To ensure full coverage the survey was designed with trackline spacings of 50 m and trackline lengths of $\approx$ 1.65 km. This took 10.3 hours to complete and produced on average 11.5 m of overlap between multibeam

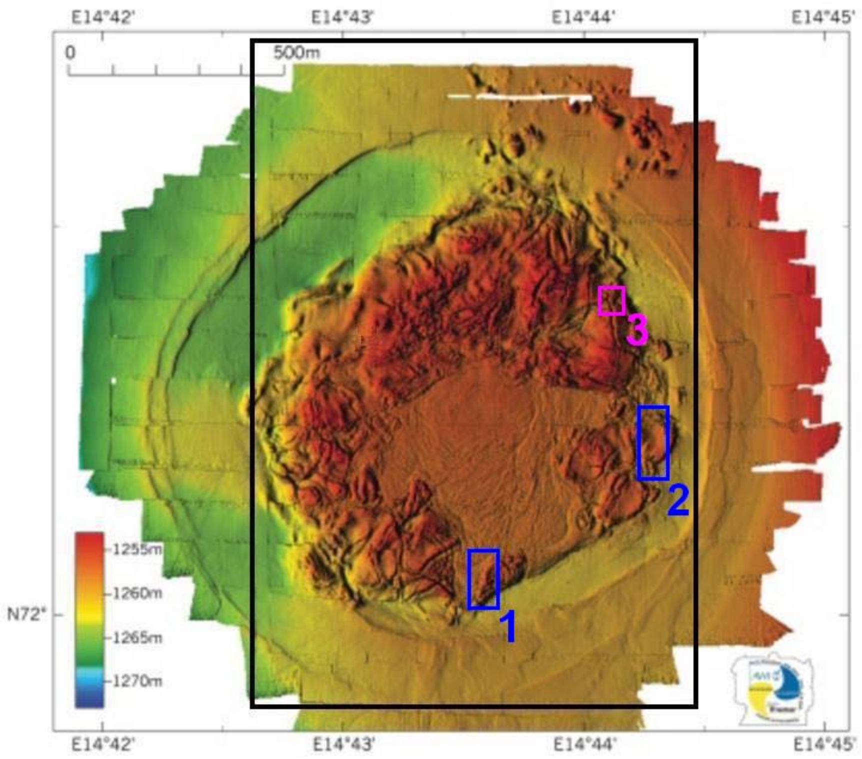


Figure 5.19: Previous bathymetric survey of the Haakon Mosby Mud Volcano taken by the ROV Victor 6000. The size of the new survey is indicated by the solid black box, carried out using a lawnmower pattern designed to provide 122% coverage (major tracklines in the North/South direction). The small colored boxes indicate regions where the map corrections produced by BPSLAM are more closely investigated in Figures 5.23 and 5.25. (*Image Source:www.uib.no/geobio*)

swaths from neighboring tracklines, which corresponds to 122% coverage.

The navigation suite installed on *Sentry* is similar to that described in Section 4.3 three axis bottom relative velocity (± 3 mms^{-1} precision) and surface relative depth sensors were available. However *Sentry* receives pitch, roll and heading observations from a PHINS Ring Laser Gyro (RLG) that delivers $\approx 0.01°$ accuracy. As such the vehicle states x, y remain the most prone to drift and so no change was made to the state setup of the filter.

Acoustic LBL navigation fixes were also available. These fixes delivered position estimates accurate to ≈ 4 m and are only used by the BPSLAM filter to initialise the AUV's position and for measurement updates in an EKF filter separate to BPSLAM, so as to produce a navigation solution by which to evaluate the performance of the BPSLAM filter. The bathymetric sensor used by *Sentry* is a Reson 7125 400 kHz multibeam sonar, providing 480 beams uniformly across 120°.

The bathymetric sensor used by *Sentry* is a Reson 7125 400 kHz multibeam sonar, providing 480 beams uniformly across 120°. Based on the noise characteristics of the multibeam sensor the percentage uncertainty in the range observations will be modeled as 1.0%.

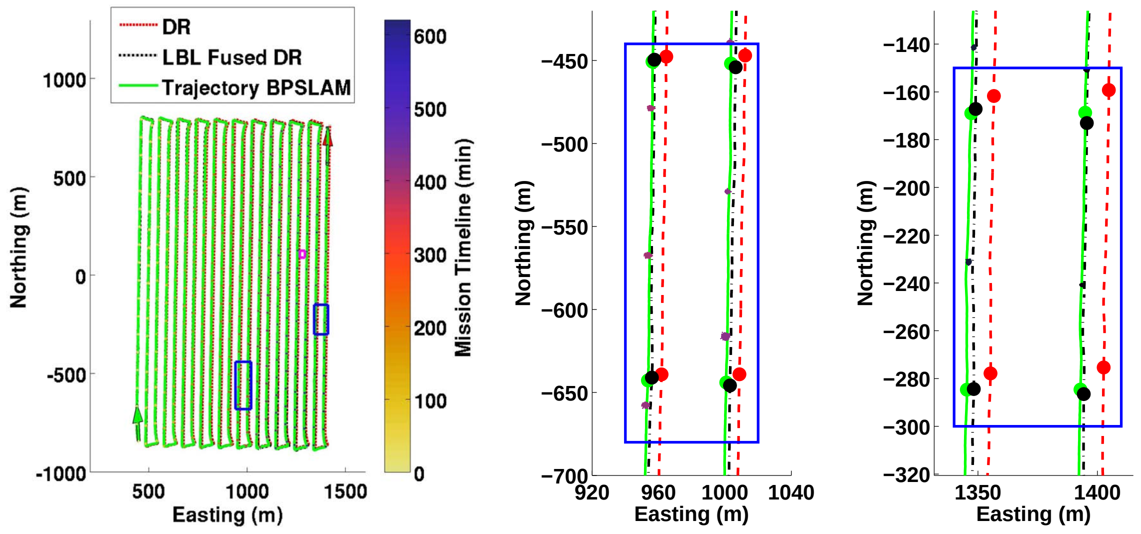
Navigation and Mapping Comparison

Figure 5.20 demonstrates the navigation solutions achieved by DR, LBL fused DR and BPSLAM with trajectory maps, using a particle set size of $N = 160$. This was chosen based on the precision of the navigation estimates available and the duration of the mission. As this survey retains consistent overlap (albeit partially) with previously explored terrain the scaling factor τ_{lc} of the trajectory map is set to zero for this run, meaning that particle weighting based on extrapolated predictions will not be attempted.

The resulting resampling behavior of the BPSLAM filter is shown in Figure 5.21, consistently constraining the size of the particle cloud through subsequent resampling.

In Figure 5.20 the use of BPSLAM has resulted in a navigation correction that slightly contracts the trackline spacings of the mission when compared to the DR solution. This is also evident in the LBL fused DR solution, producing a similar large scale correction to that achieved by BPSLAM.

At the end of the mission BPSLAM provides a final position estimate that is 14.5 m SW of the DR solution. Given that the AUV has traveled ≈ 36 km over a period of 10.3 hours



(a) Navigation Solutions (b) Zoomed in View 1 (c) Zoomed in View 2

Figure 5.20: In (a) Tracklines produced by three different navigation solutions for the Haakon Mosby mission are presented; the start and end position/direction of the BPSLAM solution is shown by the light and dark arrows respectively. The evolution of the particle cloud is also shown, changing from light to **dark** as the mission progresses. In (b) and (c) the zoomed in sections of the navigation solution bordered in blue are provided. For both figures four arbitrary BPSLAM poses were chosen and matched to the corresponding DR and LBL fused DR estimates at that time, shown by the large colored dots.

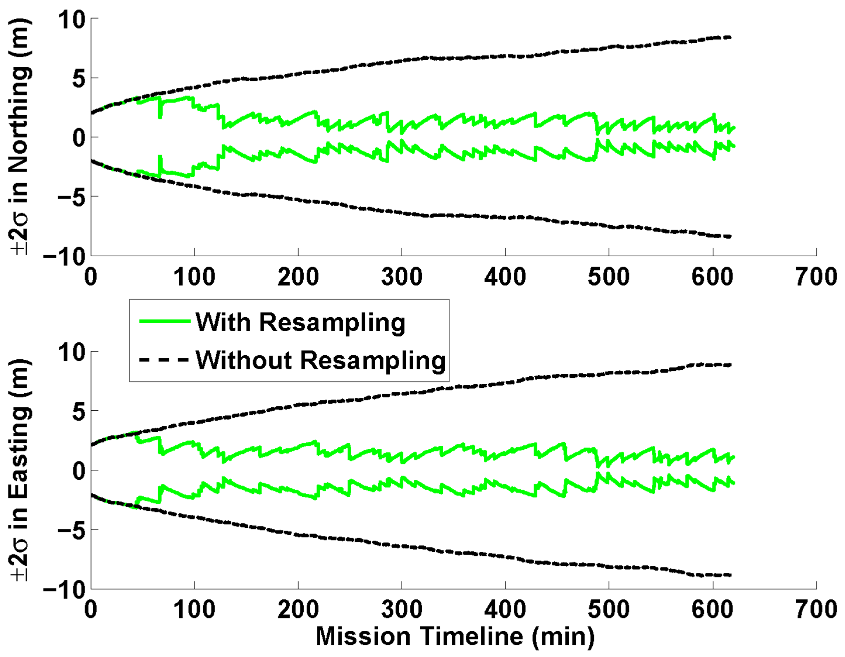


Figure 5.21: Variation in the confidence bound of the particle cloud with and without resampling for the Haakon Mosby Mission.

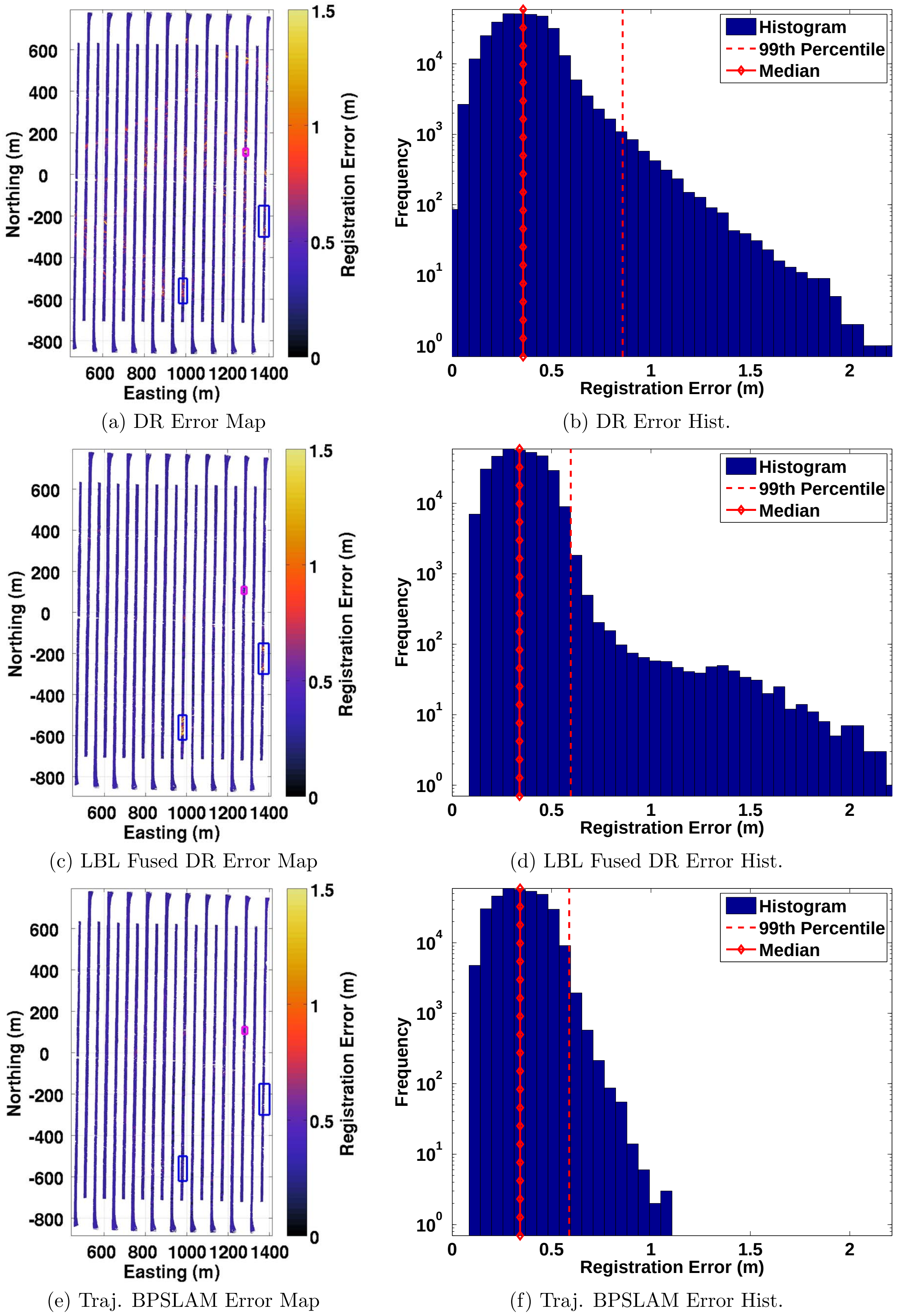
this error is relatively small, demonstrating an exceptional performance by the DVL/RLG navigation suite to produce accurate velocity estimates.

Despite this, the accumulation of this small error in DR navigation over time is enough to produce significant error in the corresponding map. Unfortunately, Sentry's bathymetric map and its geological interpretation has yet to be published in a scientific journal. Because of this, we can not show the full AUV-based map. However the registration errors detected in the full map for each case can be presented, along with zoomed in bathymetric maps of two regions where the registration errors in the DR and LBL fused DR map solutions were largest. These are shown in Figures 5.22 and 5.23 respectively.

From Figures 5.22(a),(b) it can be seen that the use of DR as a navigation solution produces the map with the most artifacts, again localised around areas of steep descent, which suggests misalignment between overlapping swaths caused by navigation error. The inclusion of LBL observations into the navigation solution resolves many of these artifacts, shown in Figures 5.22(c),(d). However, as seen in Chapter 5, this can also introduce artifacts in places where there were none before. This is evident in Figures 5.23(b) and 5.23(e), which highlight two main regions in the map where the inclusion of LBL observations has had a negative effect on mapping accuracy.

Figures 5.22(e),(f) demonstrate the performance of the BPSLAM filter when compared to LBL aided navigation. All of the prominent artifacts distorting the non-SLAM approaches have been successfully resolved. This is particularly evident in the tail of the histogram in Figure 5.22(f) which is significantly reduced when compared to the two non-SLAM approaches.

A close examination of Figure 5.23 demonstrates how BPSLAM achieved this correction. In Figure 5.23(c) BPSLAM has identified a navigation solution that, compared to the DR solution in Figure 5.23(a), shifts its East trackline 2.28 m southwards relative to its West trackline, creating the correct match between neighboring swaths. In comparison the LBL fused DR solution has overcompensated, shifting its East trackline 5.45 m southwards relative to its West trackline. As can be seen in Figure 5.23(b) this in turn creates a more pronounced registration error than originally seen in the DR solution. These artifacts in the LBL solution may be owing to the difficult acoustic environment present around the main slope of the volcano. The same effect is also seen in Figure 5.23(f), where the use of BPSLAM shifts its East trackline 3.73 m southwards relative to its West trackline, whereas

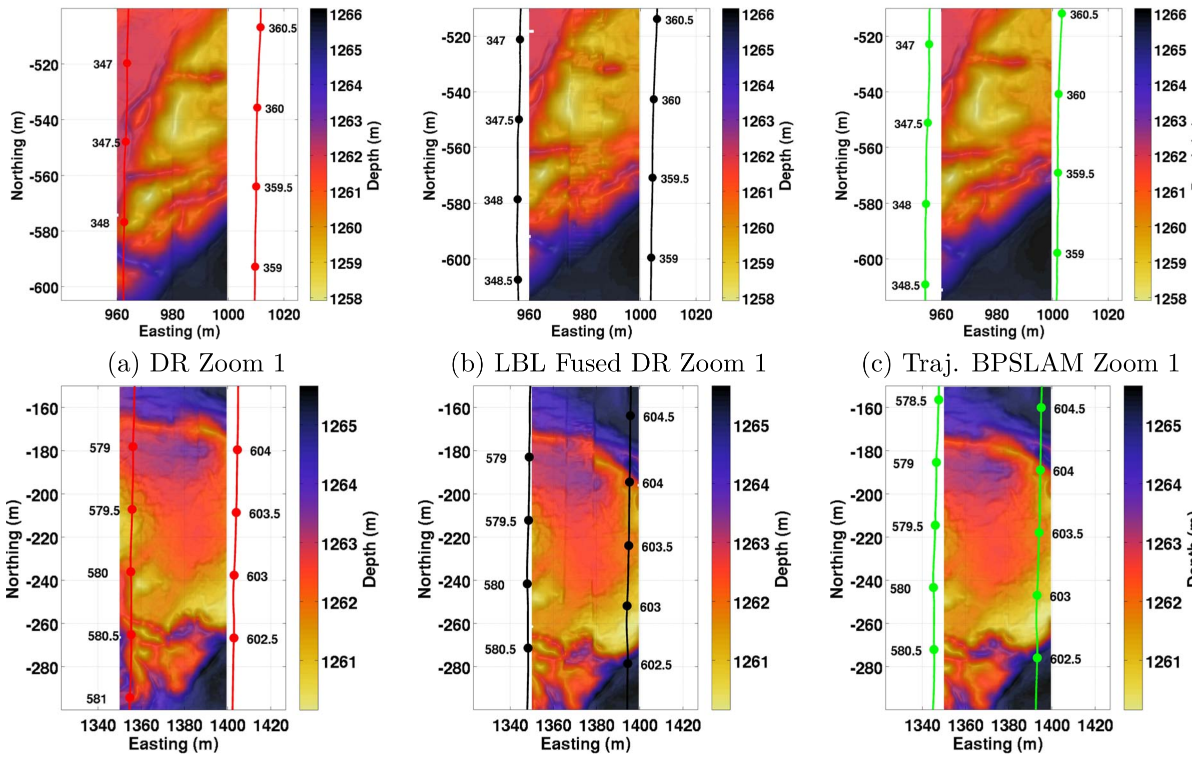


(a) DR Error Map (b) DR Error Hist.

(c) LBL Fused DR Error Map (d) LBL Fused DR Error Hist.

(e) Traj. BPSLAM Error Map (f) Traj. BPSLAM Error Hist.

Figure 5.22: Registration error maps generated using three different navigation solutions, detailing the misregistration between overlapping swaths within each cell. Comparison shows the LBL fused DR and BPSLAM filters providing a significant reduction in the mapping error when compared to the map produced by DR. The rectangles indicated by the broken lines indicate the positions of the zoomed in plots shown in Figure 5.23.



(a) DR Zoom 1 (b) LBL Fused DR Zoom 1 (c) Traj. BPSLAM Zoom 1

(d) DR Zoom 2 (e) LBL Fused DR Zoom 2 (f) Traj. BPSLAM Zoom 2

Figure 5.23: Zoomed in bathymetric maps generated using three different navigation solutions within Area 1 and 2. The section of the AUV trajectory which creates each map is also shown and timestamped (by minutes from mission start) for reference. Comparison shows a visible misalignment in the maps produced by DR and LBL Fused DR. The map produced by BPSLAM however successfully resolves these errors.

the LBL fused DR solution in Figure 5.23(e) has its East trackline shifted by 11.40 m southwards in this regard, again creating a more pronounced registration error than the original DR solution, shown in Figure 5.23(d).

The average of the registration errors in the maps produced by DR, LBL Fused DR and Trajectory Map BPSLAM were calculated as 0.371 m, 0.348 m and 0.344 m respectively. Lastly, for these trials the BPSLAM filter required 2.1 GB of RAM and took 79 minutes to process the mission, which is 13.2 % of the mission time.

Accuracy and Consistency Analysis with Reduced Map Overlap

The preceding section has demonstrated how BPSLAM with trajectory maps can perform corrections to navigation and mapping in large scale missions when provided with only $\approx$ 11.5 m of overlap between multibeam swaths from neighboring tracklines, corresponding to 122% coverage. What remains to be seen is how the BPSLAM filter performs if the amount of overlap is progressively reduced.

To investigate this behavior the BPSLAM filter was run repeatedly in batches of 25 for a particle set size of $N = 160$. For each batch the maximum width of the multibeam swath was trimmed to a different value, corresponding to 11.5 m, 4.0 m and 0.0 m of swath overlap between neighboring tracklines. Lastly, as an exploratory measure the swaths were trimmed even further so that an average *separation* of 1.0 m between swaths (i.e. -1 m overlap) was achieved. For these trials the scaling factor τ_{lc} was set to 0.3. Given l_{max} this corresponds to a minimum proximity of 2.1 m that had to be achieved between observation and map before a depth prediction was allowed. To measure the performance of each run the offline registration error measure proposed in Section 3.9 is again used. So as not to bias the runs with less overlap the full swath is made available at the end of all runs to compare the maps generated by BPSLAM, DR and LBL fused DR navigation. However it is important to note that since the FPS process is part of the filter, rather than an evaluation of it, only the trimmed swaths are used in this step to properly simulate the reduction in mapping coverage. For this reason the online FPS scheme is utilised instead as its offline counterpart requires map overlap (see Section 3.9). The performance of the filter is also shown when using the offline FPS scheme operating on the untrimmed swaths, though this is only done to demonstrate the performance of the filter if the actual best particle had been correctly chosen from the final set during each run.

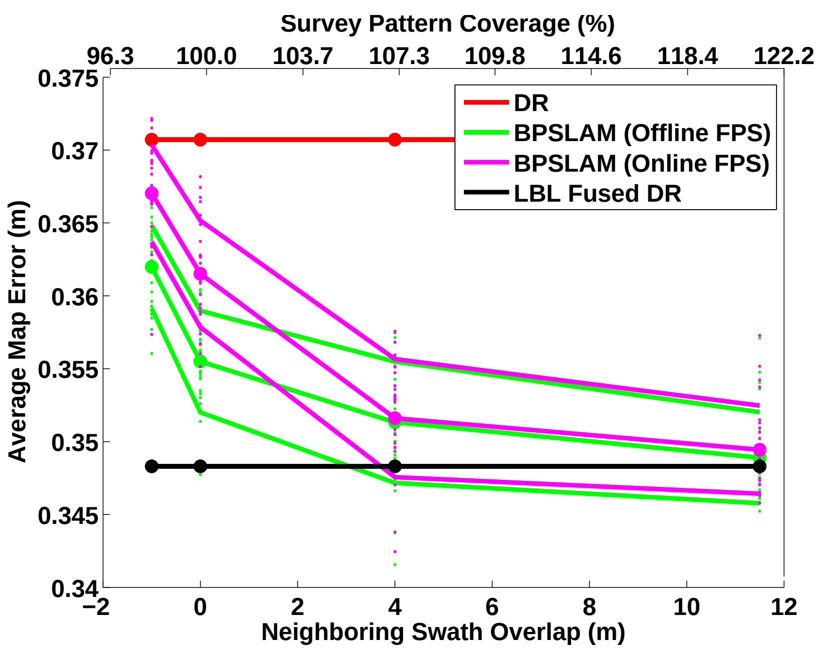
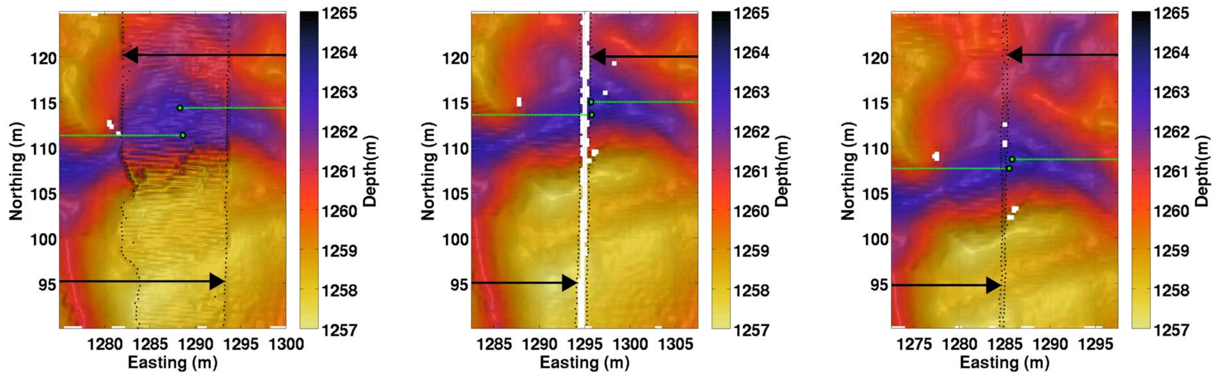


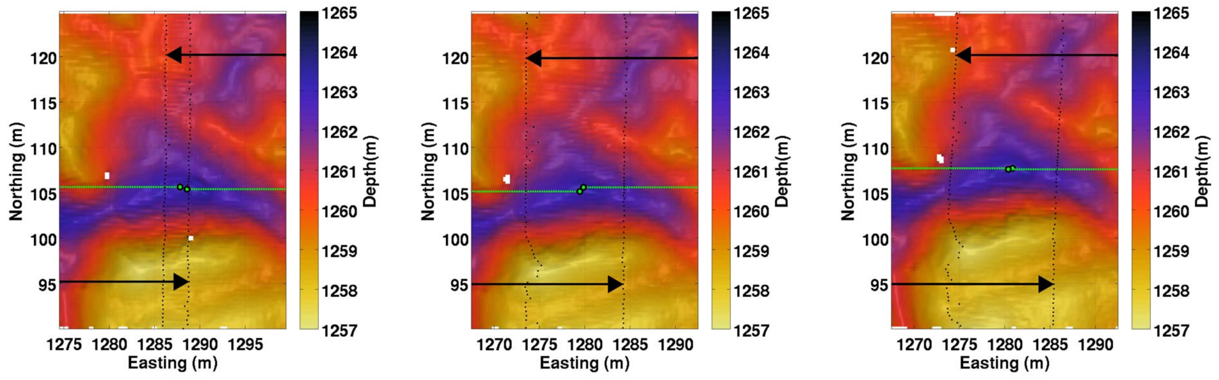
Figure 5.24: Variation in the map error as the amount of swath overlap between neighboring tracklines is reduced. The large dots represent the mean error used in each batch of 25 runs (shown by the smaller dots). The outer solid lines represent the $\pm 2\sigma$ bound in each batch. For reference the results of using the offline Final Particle Selection (FPS) scheme based on the full multibeam is also shown. This extracts the actual best particle that has survived in the set, as opposed to the predicted best particle based on the online FPS scheme, which must be used when overlap is minimal or non-existent.

Figure 5.24 presents the results of this investigation. For 11.5 m overlap between swaths the map self-consistency produced by BPSLAM rivals and often exceeds that produced by LBL fused DR. For this amount of overlap the online version of the FPS scheme also performs well in estimating the best particle to extract (the actual best particle pointed to by the offline version). As the amount of swath overlap is reduced the BPSLAM mapping solution starts to converge towards the map quality produced by DR. However, as shown the BPSLAM filter is still capable of producing mapping corrections even when no overlap between neighboring swaths exists, consistently out performing Dead Reckoning in this case.

This is further demonstrated in Figure 5.25, which presents a comparison of the maps produced during this investigation within the third zoomed-in area highlighted in Figure 5.19. Here four maps created by the BPSLAM filter are presented, corresponding to the runs from Figure 5.24 that possessed the median error measure (using the online FPS scheme) of each batch. The corresponding maps produced by DR and LBL fused DR are also provided. As an added measure the center of the trough in this region was hand matched to the beams that observed it, once when the vehicle was traversing southwards on the western side of this trough, and again when it was traversing northwards on the eastern side. The locations of these beams based on the navigation solution therefore provides a good indicator of the relative misalignment in this section of the map.



(a) DR 11.5 m overlap (b) BPSLAM 1.0 m separation (c) BPSLAM 0.0 m overlap



(d) BPSLAM 4.0 m overlap (e) BPSLAM 11.5 m overlap (f) LBL Fused DR 11.5 m overlap

Figure 5.25: Zoomed in bathymetric maps generated using six different navigation solutions around Area 3. Four of these navigation solutions are from the BPSLAM filter, taken from the median of each batch of runs shown in Figure 5.24. The green circles indicate where both eastern and western swaths map (or would map for those with reduced overlap) the deepest part of the trough in this area. The edge of each multibeam swath is shown by the small black dots, the black arrows indicating whether the edge line belongs to those from the eastern or western side.

In Figure 5.25(a) the inconsistency in the overlapping swaths is clearly evident, creating pronounced artifacts. This is also demonstrated by the misalignment in the green circles, indicating a 3.06 m difference between where eastern and western portions of the map believe the center of the trough to be. Figure 5.25(b) shows how BPSLAM is able to reduce this alignment error to 1.42 m , even when a 1.0 m gap between neighboring swaths exist. Figures 5.25(c)-(e) demonstrate how this misalignment is further reduced to 1.06 m, 0.80 m and 0.60 m respectively when the overlap between neighboring swaths is increased. Finally Figure 5.25(f) demonstrates LBL fused DR creating a misalignment of 0.47 m. For this section of the map LBL fused DR therefore provides a slightly better result in terms of map self-consistency, though as was shown in Figures 5.22 and 5.23 creates gross misalignments elsewhere that BPSLAM fortunately resolves.

5.5 Limitations

Currently the major limitation faced by the new trajectory map representation is its computational efficiency in terms of run time. This arises from the need to potentially invert large covariance matrices each time GP learning or GP regression is to be attempted. For the framework presented here this problem is overcome by restricting the training data used to points within close proximity to the places we wish to test. In doing so we introduce an approximation error into the resulting predictions. Computing the inverse of these large covariance matrices using sparse methods may provide the computational speed up required to remove the need for this approximation. Implementing a neural network covariance function also has the possibility of speeding up computation, provided that the neural network only be learned once with all the multibeam data at hand. This would also allow for a more principled treatment of large scale trends and local anomalies in the seabed but comes at the cost of reducing the scope of the BPSLAM filter to a post-processing algorithm.

Lastly BPSLAM using the trajectory map representation is restricted to environments that can model seabed depth as a continuous function of x, y location and hence cannot properly take into account overhang or ceilings. Implementing BPSLAM in a full 3D environment using this approach is the subject of future work and will be discussed in Chapter 6.

5.6 Summary

This chapter has presented the trajectory map representation that is paired with RBPF SLAM to form another variation of the BPSLAM filter. A description of the map structure along with the procedures used to weight and update each particle's map were provided.

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, both in small scale and large scale missions. Its performance was also compared against Roman's Sub-mapping approach to SLAM (Roman 2005).

Comparing the performance of the BPSLAM filter with trajectory maps against the BPSLAM filter with grid maps demonstrated that both were equally proficient at providing corrections to navigation and mapping. While the grid map approach demonstrated a superior computational run time the trajectory map approach required significantly less memory. The ability of the trajectory map approach to run off sparse bathymetry obtained from the DVL sensor was also presented. Results demonstrated that such a setup could yield significant improvements to navigation, similar to what was achieved using LBL fused DR. Finally the performance of Trajectory Map BPSLAM was tested in a mission where little to no map overlap was present, verifying that corrections to navigation and mapping can still be produced even when no map overlap exists.

Chapter 6

Conclusions and Future Work

6.1 Introduction

This thesis has presented an in-depth analysis of the BPSLAM algorithm, demonstrating how the map and navigation solution generated during bathymetric surveys can be improved by enforcing map self-consistency. The main theoretical framework that BPSLAM used as its foundation was the Rao-Blackwellized Particle Filter, efficiently maintained using Distributed Particle Mapping. New map representations were then described that coupled with this framework to form a method of SLAM that did not require features in the seabed to be explicitly identified or tracked. The first was a 2.5D grid map representation efficiently stored by exploiting redundancies between different maps. The second was a trajectory map representation that, instead of directly storing estimates of seabed depth, recorded the trajectory of each particle and synchronised it to a common log of bathymetric observations.

The contributions of this thesis are summarised in Sections 6.2. Section 6.3 then proposes directions for future work.

6.2 Summary of Contributions

6.2.1 Primary Contributions

2.5D Grid Map Representation

A new 2.5D grid map representation for BPSLAM was proposed. In this approach any new map estimates were first keyed with the ID of the particle that made them and then entered into a single global grid. Each particle then maintained and updated their estimates of seabed depth using a 1D Extended Information Filter. Through the implementation of Distributed Particle Mapping the need to copy maps during resampling was removed, as any new particles could simply point to the map of the particle they were resampled from instead. Applying BPSLAM with this map representation to the missions described in Section 6.2.2 demonstrated that the corresponding map weighting and updating operations could be carried out with exceptional computational speed, allowing BPSLAM to process each mission in significantly less time than it took to execute.

Trajectory Map Representation

A new trajectory map representation for BPSLAM was proposed. In this framework a significant amount of memory was saved by storing each particle map as a trajectory linked to a bathymetric log of raw observations that is shared by all particles. Each particle then updated its map by progressively adding its current pose to the end of its trajectory list. Applying BPSLAM with this map representation to the missions described in Section 6.2.2 demonstrated that the corresponding map weighting operation was significantly slower than that of the grid map representation, made computationally expensive by requiring Gaussian Process Regression to be performed for each particle. However the trajectory map approach still retained a computational speed faster than the original mission time and provided exceptional memory requirements that allowed for the processing of very large datasets. In addition it allowed BPSLAM to additionally enforce self-consistency between neighboring map borders in situations where no map overlap existed.

6.2.2 Secondary Contributions

BPSLAM Implementation on Real Mission Scenarios

Investigations into the use of BPSLAM to improve navigation and mapping during bathymetric surveys were carried out on three different case studies. The first focused on an AUV deployment at Butts Reef over a short timescale ($\sim$ 2 hours) in a region that contained minimal relief but possessed several distinct pockmarks in the seabed. The second case study then focused on an ROV deployment within the TAG hydrothermal field that in comparison observed significantly higher relief in the seabed over a much longer timescale ($\sim$ 11 hours). Applying the BPSLAM filter to these first two case studies with either map representation demonstrated 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, as well as demonstrating a robustness to compass biases and noisy velocity observations. For the second case study the performance of the BPSLAM filter was also compared against Roman's Sub-mapping approach to SLAM (Roman 2005). While only a slight difference was found in the map qualities produced by both SLAM filters, the BPSLAM algorithm was able to demonstrate a significant improvement in computational efficiency. BPSLAM also achieved this with fewer tuning parameters, which were shown to translate well between the small and large scale missions presented here.

These trials revealed no significant differences in the navigation and mapping corrections provided by both map variations of the BPSLAM filter. The computational performances of the two map variants were however significantly different. Whereas the grid map representation used by BPSLAM possessed a computationally superior run time the new trajectory map representation required significantly less memory.

The added ability of the trajectory map representation to produce map corrections using sparse bathymetry was then tested by repeating the TAG survey using only the bathymetry provided by DVL sensor (4 beams) as mapping observations. Results showed that in this setup the BPSLAM filter was still able to improve upon the navigation and map produced by the LBL fused DR solution, though as expected did not produce as good a result as when the full multibeam sonar was utilised.

Finally a third case study was investigated that focused on an AUV deployment over the Haakon Mosby Mud Volcano, carried out over 10 hours. In this survey no loop closure manoeuvres were performed. However partial overlap between swaths from neighboring tracklines did exist. Applying the BPSLAM filter to this case study demonstrated an improvement in both the map and the trajectory when compared to state of the art navigation fused with external position estimates from an LBL transponder net. By successively trimming the width of the multibeam swaths it was also shown that the BPSLAM filter using trajectory maps could still provide some corrections to navigation and mapping even when no overlap existed between neighboring swaths, outperforming DR (but not LBL fused DR) in this situation.

Map Self-Consistency Measures

A technique for selecting the final navigation and map solution from BPSLAM was presented. This was based on a method described by Roman & Singh (2006) and involved splitting the bathymetry into a sequence of submaps and then calculating the maximum registration error between submap portions located in the same cell. An alternative method of selecting the best particle map was also proposed for situations where little to no map overlap existed, where the best particle map is instead chosen based on the average registration error observed so far by each particle during the map weighting procedure.

Online Gaussian Process Regression for SLAM

An online method of extracting depth estimates in unobserved regions using Gaussian Process Regression was proposed and, through suitable approximations and choices of GP models, was made computationally tractable for SLAM. This method implemented a stationary covariance function proposed by Melkumyan that allowed for exact Gaussian Process Inference in large datasets, should the entire training set be used. Periodic relearning was then undertaken to continuously update the GP model based on incoming bathymetry. Particle weighting was then performed or prevented based on the proximity of new observations to the current map and the amount of spatial correlation in the surrounding seabed.

Uncertainty Raytracing using Correlated Observations

A principled method of transforming uncertainty in a set of range observations to an equivalent set of observations with uncertainty only in depth was introduced. This was achieved by modeling each multibeam swath as a collection of inter-dependent observations that could be used to train a GP in the polar domain. By exploiting the angular correlation between observations in this way GP Regression was used to infer the probability distribution of the observations along each of their depth axes.

6.3 Future Research

Throughout this thesis several issues have presented themselves, providing potential future research.

6.3.1 Modeling Heading Error in the Particle Set

For UUV operations that rely on a magnetic compass for navigation the heading state must be included in the particle filter if the compass is poorly calibrated. This is of particular concern as compasses are easily corrupted by local anomalies in the magnetic field that can change with the heading of the vehicle. Since this type of error varies with heading it cannot be modeled in the same way as the x, y states, whose error varies with time. Instead the heading error is more suitably modeled as a 3rd order harmonic in heading. Initialising each particle with a random sampling of coefficients to this harmonic could therefore provide a way of modeling this heading error in the particle filter, as well as providing a calibrated heading sensor model that can be extracted at the end of the mission.

6.3.2 Iterative BPSLAM

BPSLAM utilises the same sensor observations used to create DR navigation, creating particle trajectories that remain densely sampled around this solution until the first resampling event occurs. An opportunity therefore exists to improve the sampling strategy of the BPSLAM filter by recording the navigation observations sampled by the winning particle. A new run of the BPSLAM filter can then be performed, using these augmented sensor observations instead of the original observations that created the DR solution. This creates a denser sampling of trajectories around the previous winning solution, allowing the navigation and mapping solutions to be iteratively refined through subsequent runs.

6.3.3 Full 3D BPSLAM

Both the current grid map and trajectory map representations used by BPSLAM assume that no overhang exists in the environment, restricting its use to missions where the seabed can be modeled as 2.5D. Extending BPSLAM to a full 3D environment would require that the grid map approach be extended to a 3D grid, though this is computationally expensive. However there is the potential to extend the trajectory map approach to 3D environments if we represent each trajectory map in polar space, where the seafloor would instead be modeled as a continuous function in angular coordinates, thereby allowing cliffs and ceilings to be accounted for. However this approach in turn cannot model features that cause “overhang” in the polar domain, such as observing troughs from a large grazing angle. Finding a way to combine or interchange GP Regression between these two domains has the potential to remove this limitation.

6.3.4 Real Time Operations

There is a great potential to implement the BPSLAM filter, particularly with the grid map representation, in real time. This will first require an investigation into the additional computational overhead involved in beamforming the multibeam observations, detecting/removing bad pings and potentially reducing the memory requirements further. Real time BPSLAM will also require that a corrected navigation solution be available on the fly, as opposed to using our offline final particle selection scheme. As such a more thorough investigation into different online techniques for finding the particle with the most accurate navigation solution would be useful.

This capability would also allow for active SLAM to be additionally performed, whereby the UUV periodically directs itself towards previously explored areas to induce loop closures and thus improve its navigation and map. Additional metrics such as the choice of candidate regions for loop closures based on map saliency and uncertainty in the vehicle state would need to be explored.

6.3.5 Investigation into Map Saliency

The ability of BPSLAM to provide corrections to navigation is completely dependent on there being enough variation in the particle weights generated during a loop closure attempt, as this allows the most likely state hypotheses to be discerned. This variation naturally decreases as the seabed being mapped becomes less “feature rich” i.e. less salient. As such it would be instructive to investigate and quantify how much the saliency in the seabed can be reduced before the corrections provided by BPSLAM become insignificant.

To this end running BPSLAM on simulated test data may offer the best approach to this investigation, as it allows the saliency of the seabed to be precisely controlled, along with the size of the maps and the sensor resolution and accuracy. However choosing a quantitative metric to represent map saliency is a non trivial problem. Spatial correlation (such as that measured by the length hyperparameter l in the GP model) is one such potential candidate, though is only a good measure if the spatial correlation in the simulated seabed is roughly uniform. To simulate a more natural environment, where the spatial correlation is itself spatially dependent, a more complex metric for map saliency will be needed.

6.3.6 Investigation into Tuning Parameters

While the main tuning parameter of the BPSLAM filter i.e. the number of particles, was thoroughly investigated for its effect on mapping, there exists other tuning parameters that could also warrant investigation. These include the minimum overlap required for particle weighting ($\gamma_{overlap}$), the grid resolution, the proximity required for loop closure (τ_{lc}) and the maximum spatial correlation allowed for resampling (l_{max}). Furthermore it may also be useful to investigate other sampling/resampling strategies than that currently used, as this is still an active area of research that proposes many approaches. Lastly the effects that these tuning parameters have on navigation could also be thoroughly explored with the implementation of a ground truth system, either through simulation or in a controlled and accurately surveyed test chamber.

6.4 Summary

This thesis has addressed the problem of performing SLAM in unstructured underwater environments using bathymetric observations. It has contributed to featureless SLAM theory and has provided practical guidelines as to how UUVs can improve their navigation and mapping by enforcing self-consistency in the bathymetric maps they create. For small scale missions with ample overlap BPSLAM with a grid map representation was shown to be the best choice for providing these corrections. However for larger scale missions, or for missions that contain minimal overlap, performing BPSLAM with a trajectory map representation became the best choice. This was all accomplished without requiring the additional infrastructure, time and expense of setting up USBL or LBL acoustic positioning systems, even outperforming these methods of navigation in regards to the quality of the maps produced.

Appendix A

Auxiliary Maps

In the preceding chapters there have been several bathymetric maps presented that contain rich structure but were reduced in size so as to allow for a proper comparison between mapping methods. These maps are now reproduced here in full size, so that this structure can be more easily viewed.

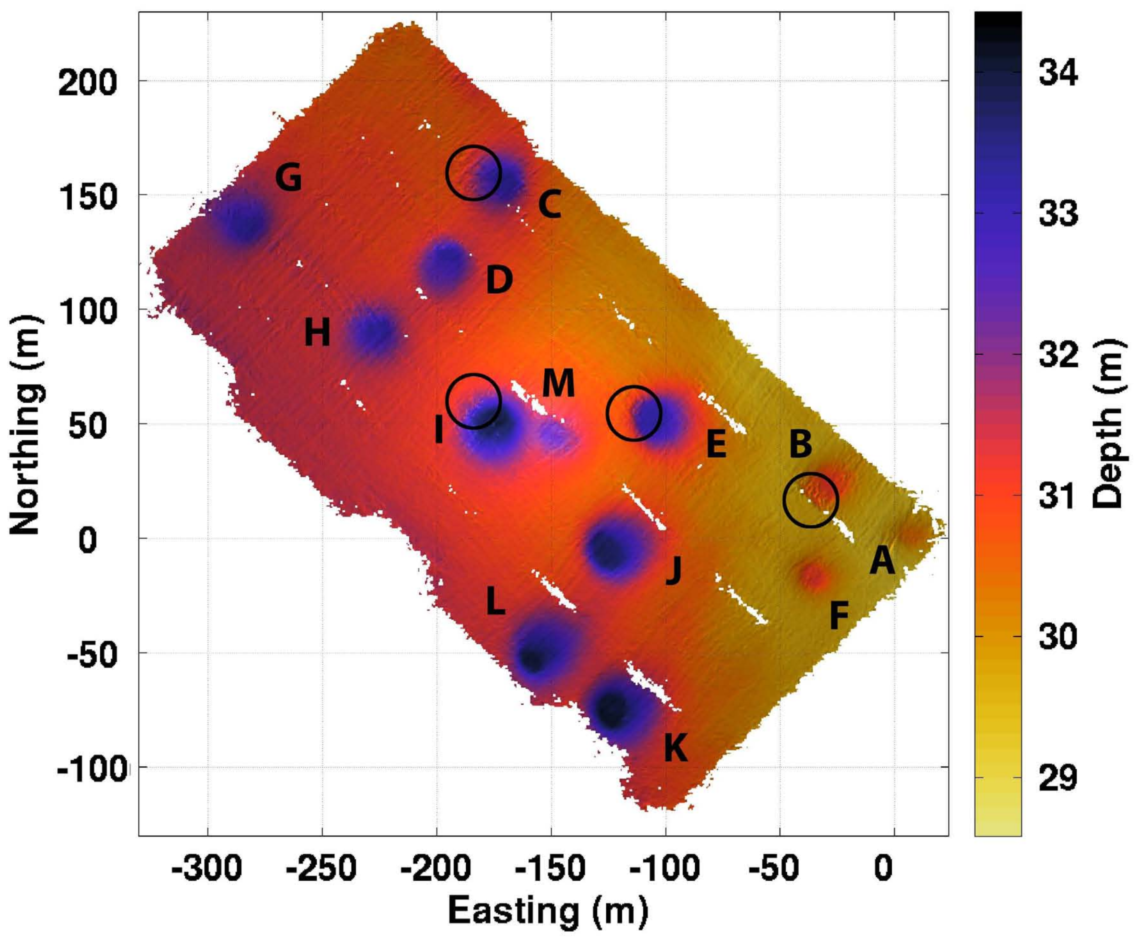


Figure A.1: Bathymetric map of Pockmark mission using DR

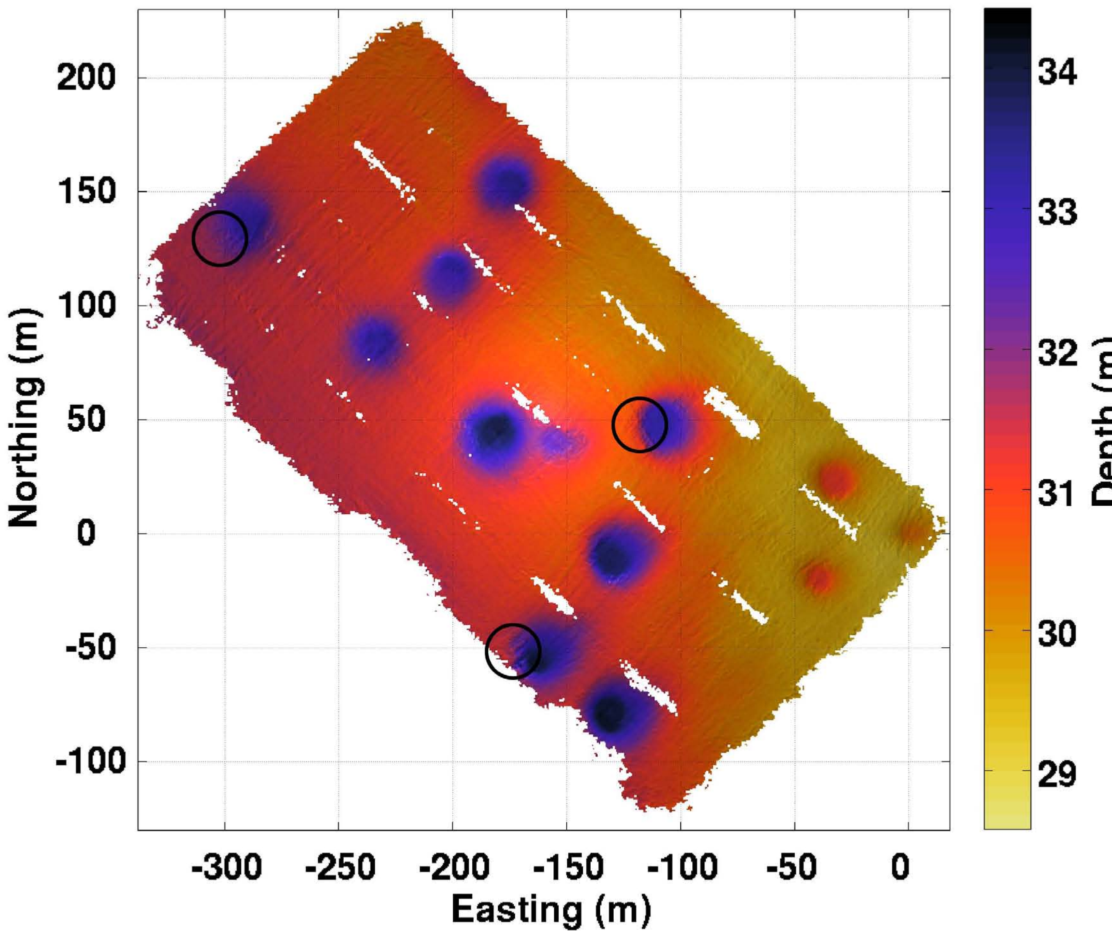


Figure A.2: Bathymetric map of Pockmark mission using USBL fused DR

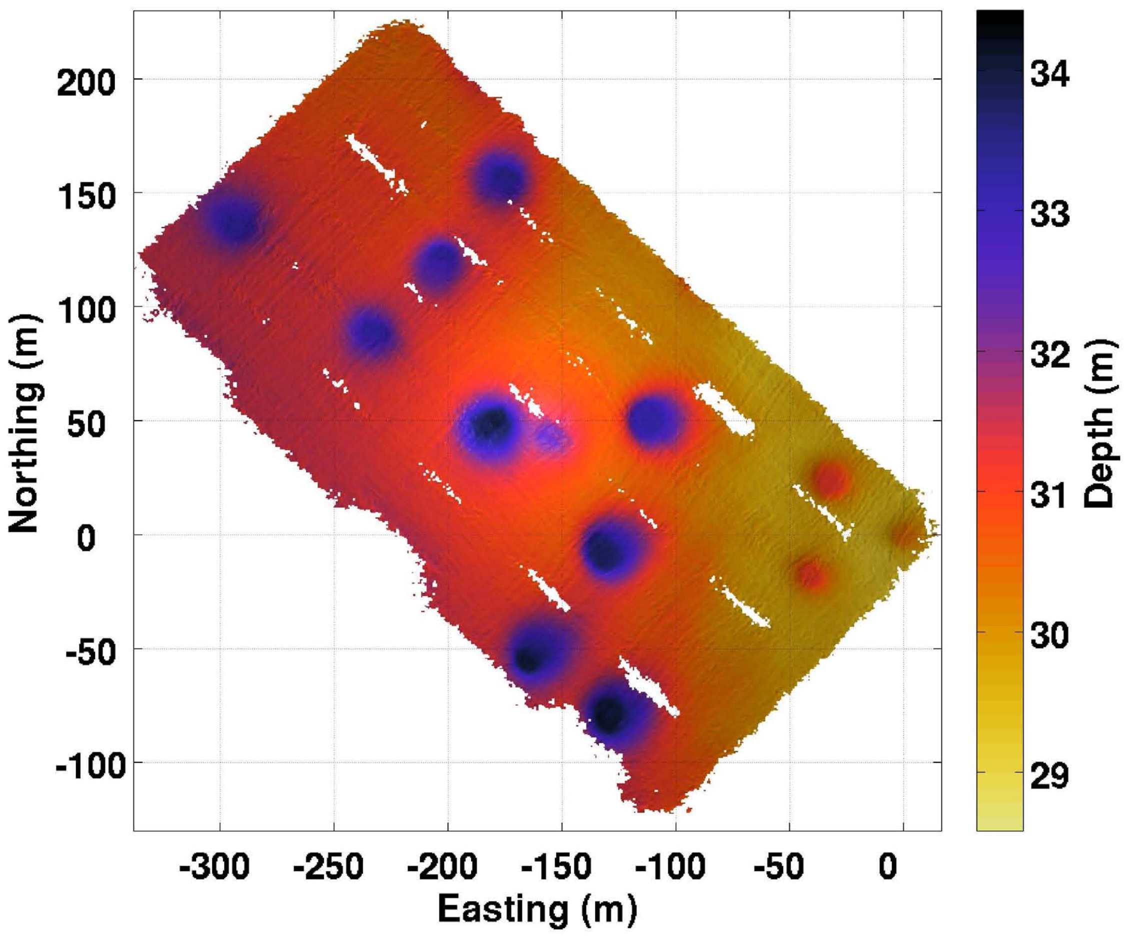


Figure A.3: Bathymetric map of Pockmark mission using Grid Map BPSLAM

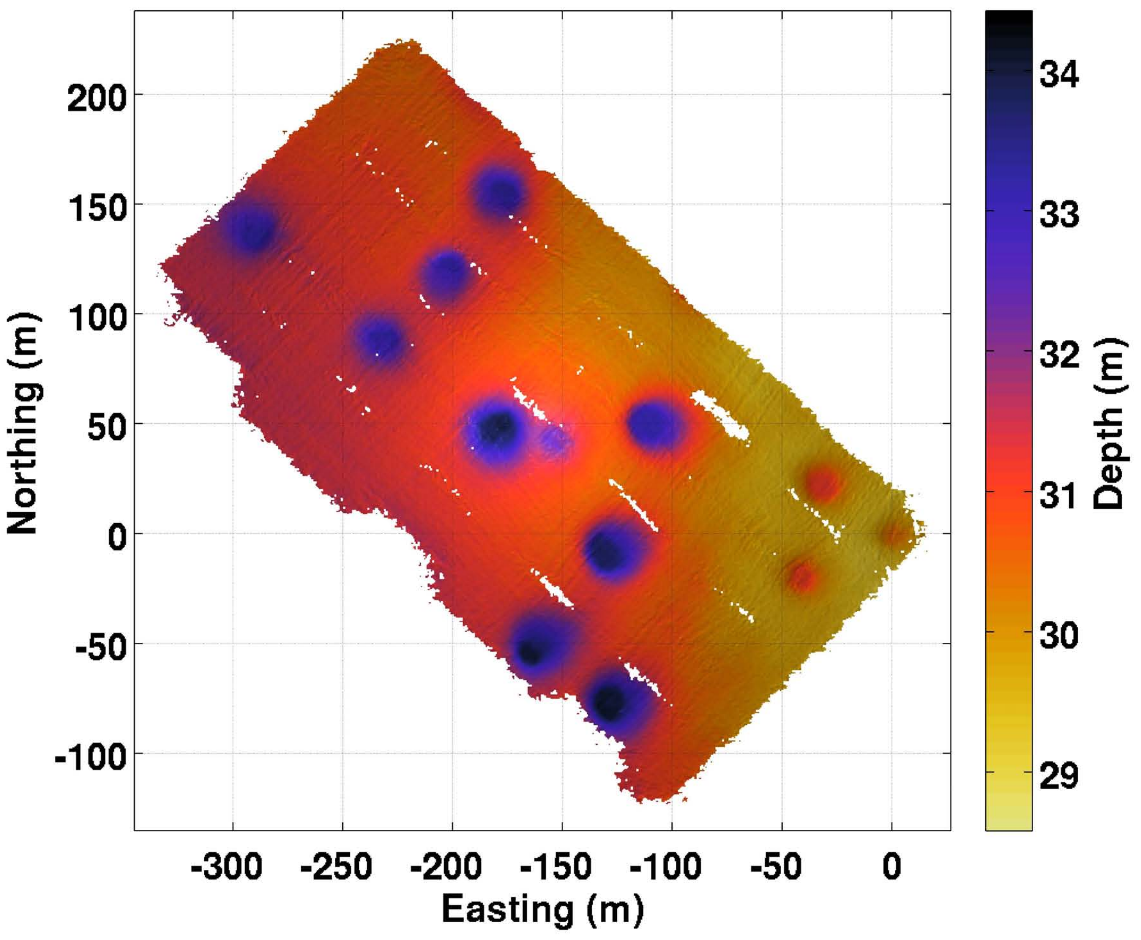


Figure A.4: Bathymetric map of Pockmark mission using Trajectory Map BPSLAM

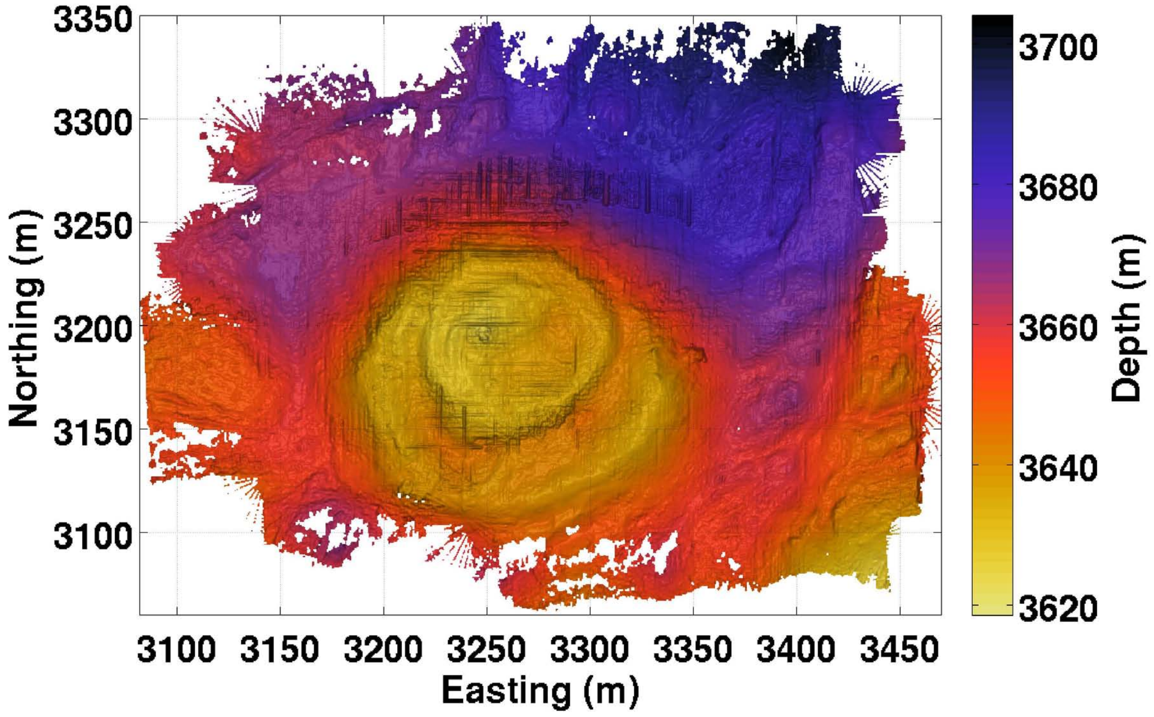


Figure A.5: Bathymetric map of TAG mission using DR

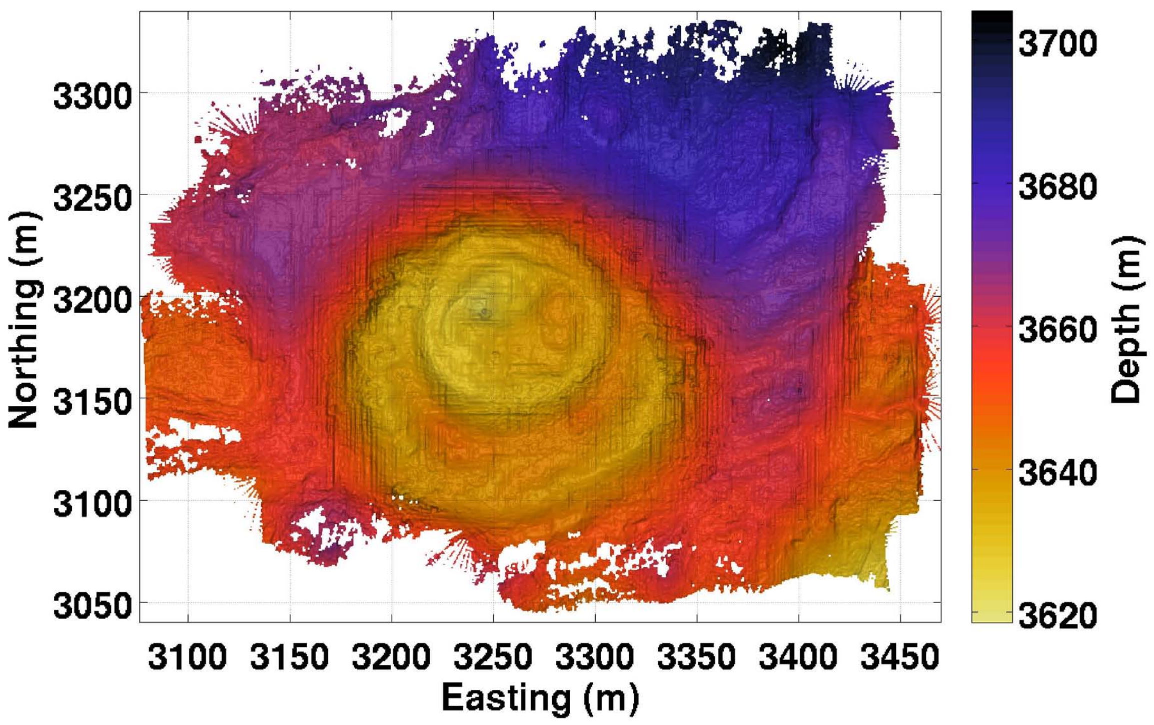


Figure A.6: Bathymetric map of TAG mission using LBL fused DR

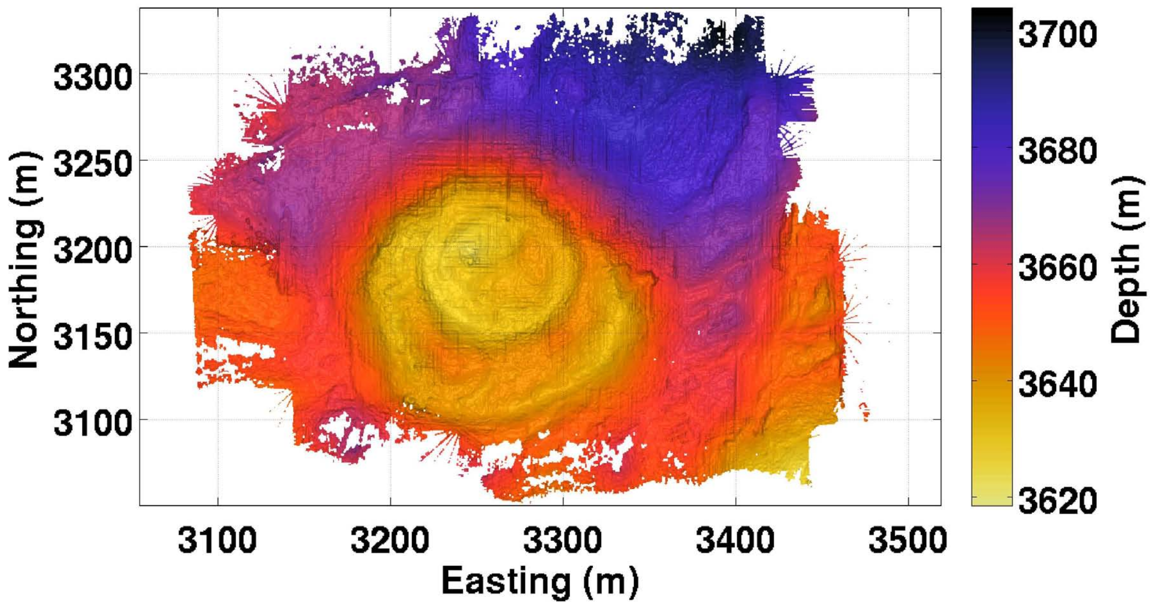


Figure A.7: Bathymetric map of TAG mission using DVL based Trajectory Map BPSLAM

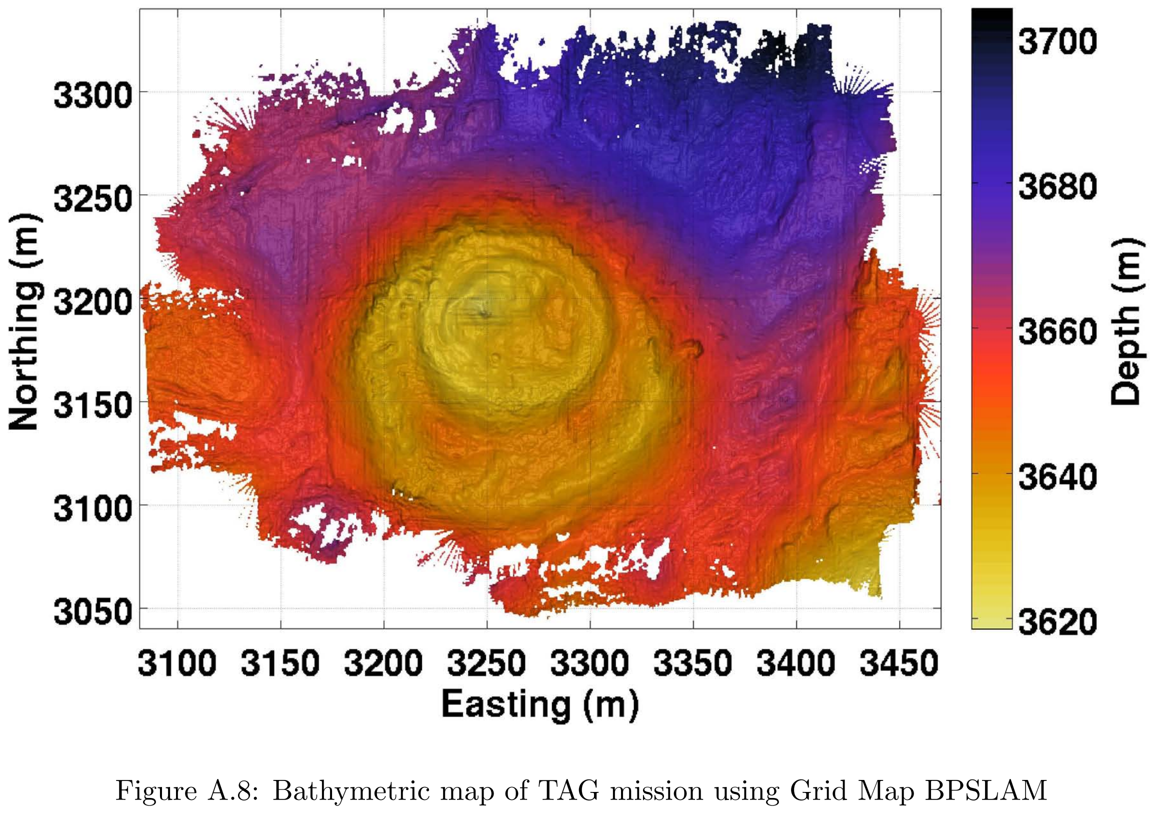


Figure A.8: Bathymetric map of TAG mission using Grid Map BPSLAM

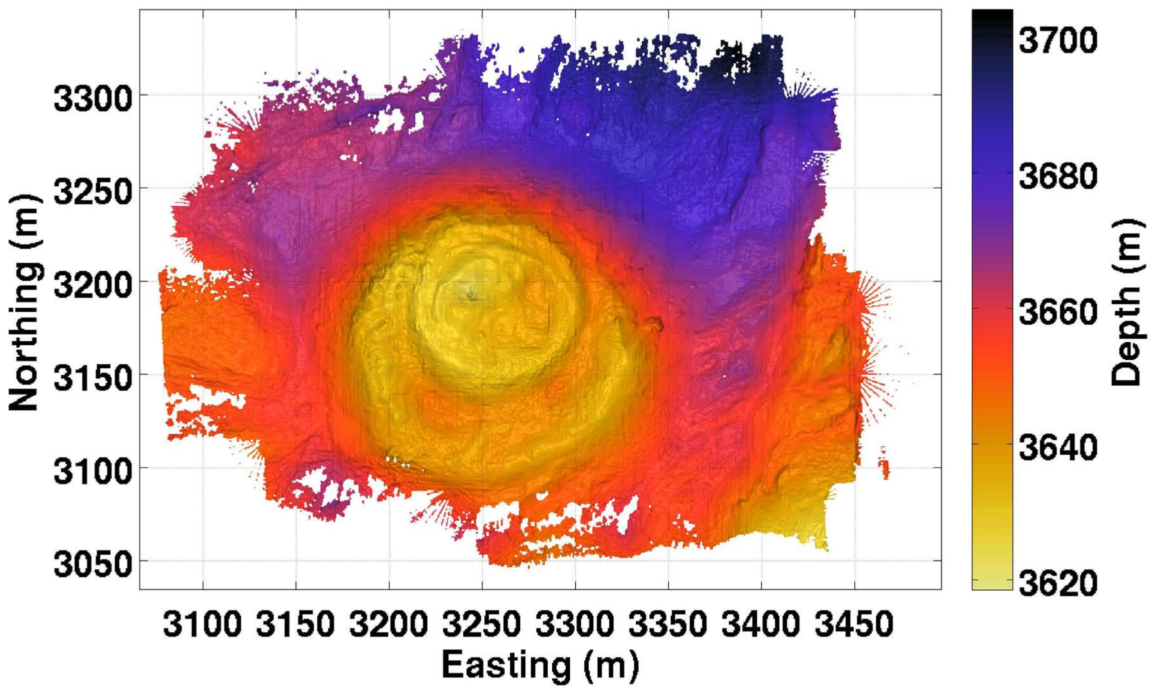


Figure A.9: Bathymetric map of TAG mission using Trajectory Map BPSLAM

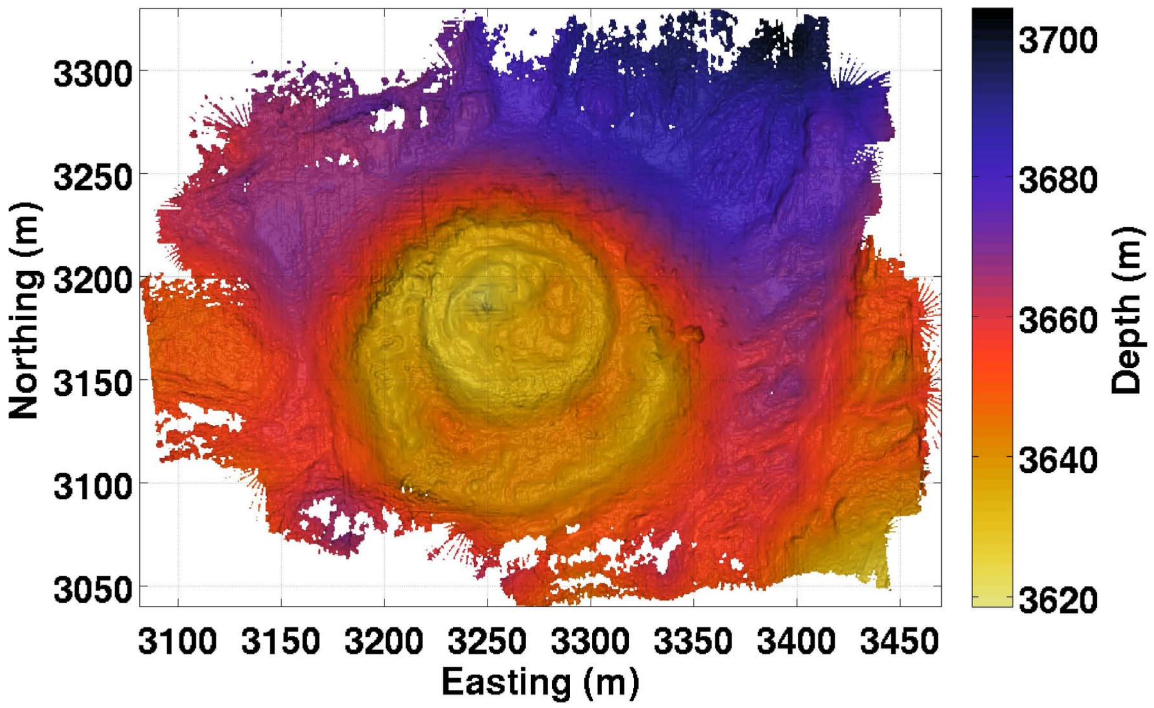


Figure A.10: Bathymetric map of TAG mission using Submapping Filter

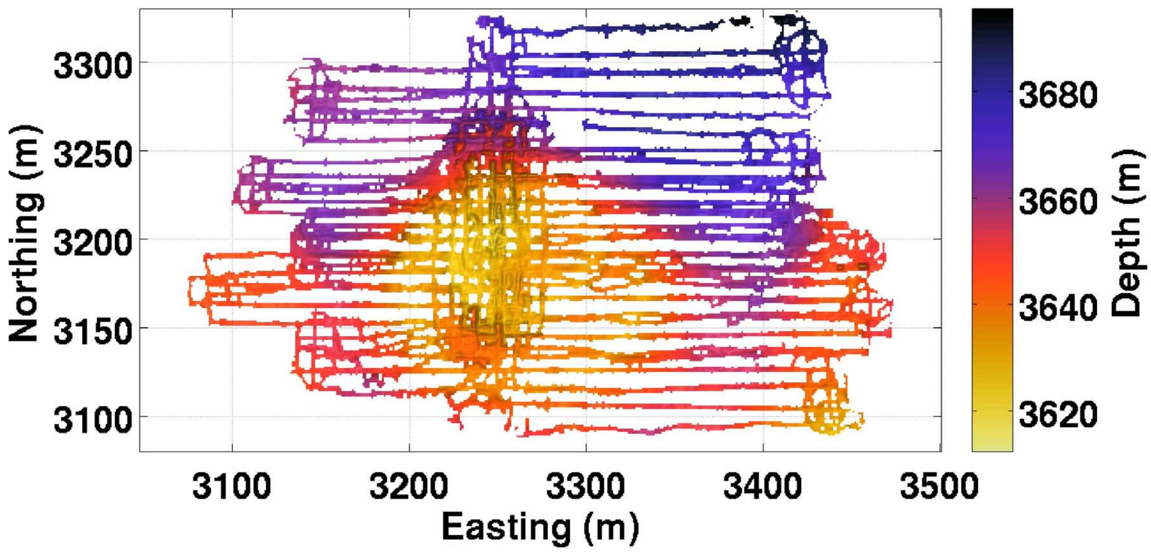


Figure A.11: DVL Bathymetric map of TAG mission using DR

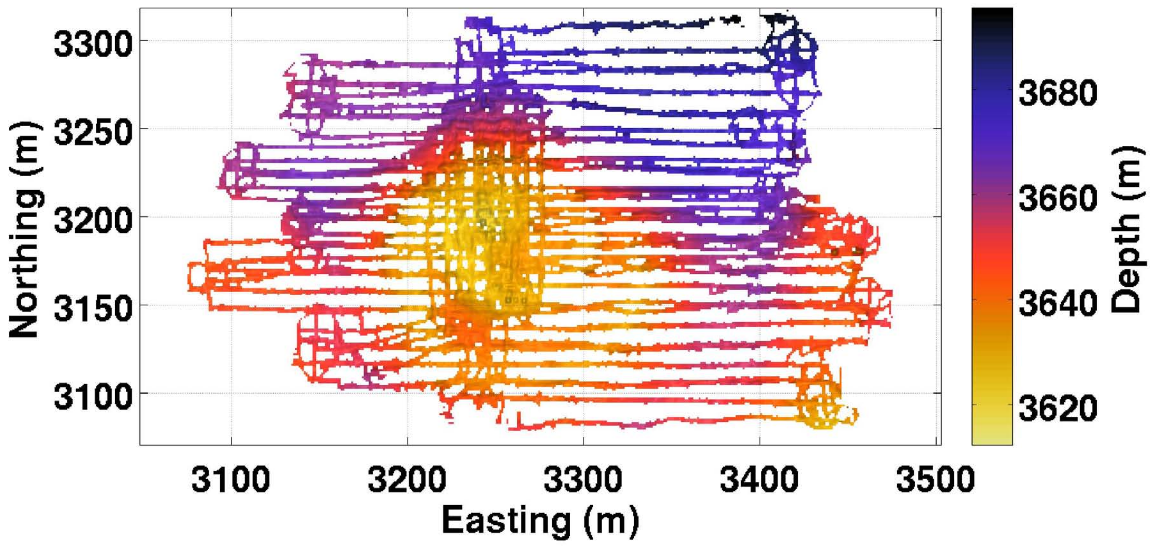


Figure A.12: DVL Bathymetric map of TAG mission using LBL Fused DR

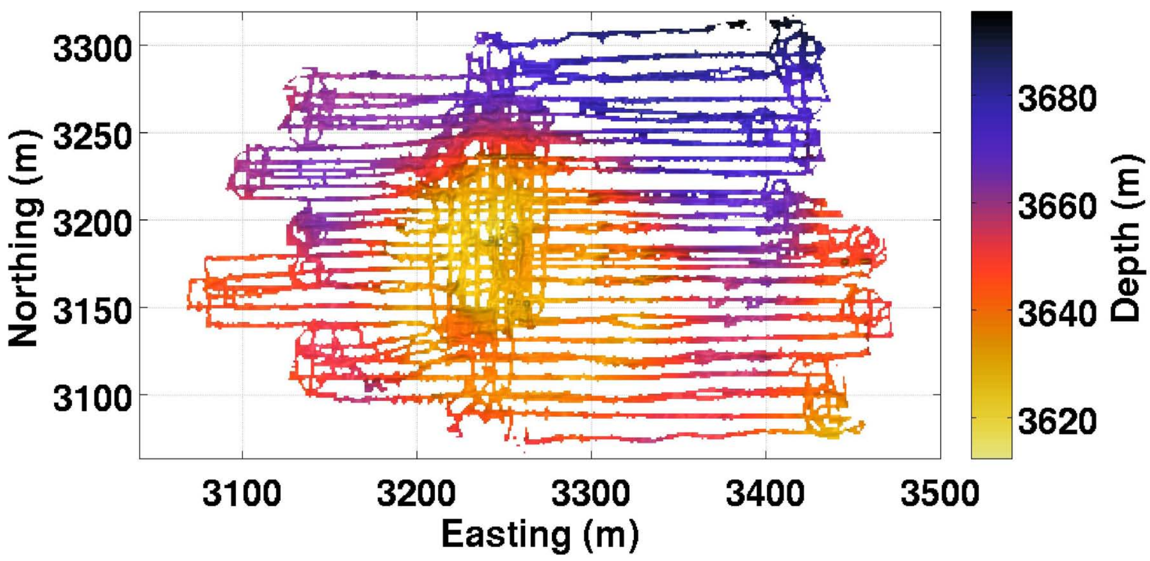


Figure A.13: DVL Bathymetric map of TAG mission using DVL based Trajectory Map BPSLAM

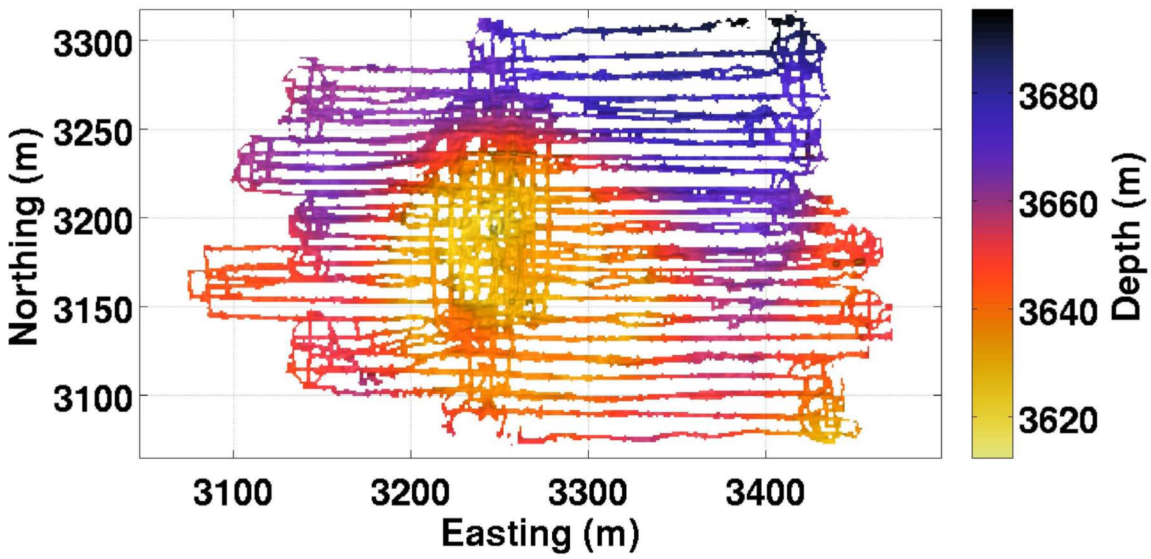


Figure A.14: DVL Bathymetric map of TAG mission using Multibeam based Trajectory Map BPSLAM

Appendix B

Adaptive Particle Sizing

B.1 Method

Adaptive particle sizing attempts to reduce the computational complexity of the particle filter by adaptively adding or removing particles from $\mathbf{S}(t_k)$ as the uncertainty (represented by the volume of the particle set in state space) grows or shrinks respectively. To provide a method of sizing $\mathbf{S}(t_k)$ we can enforce the condition that the probability distribution represented by $\mathbf{S}(t_k)$ remain a good approximation of the true distribution. More formally we specify that their Kullback-Leibler Divergence (KLD) (Fox 2001), which is a (non-symmetric) measure of the difference between two probability distributions, remain within a given threshold (ε). Although the true distribution is unknown the minimum number of particles required to achieve this can still be approximated using the formula:

$$N \approx \frac{k-1}{2\varepsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right)^3 \tag{B.1}$$

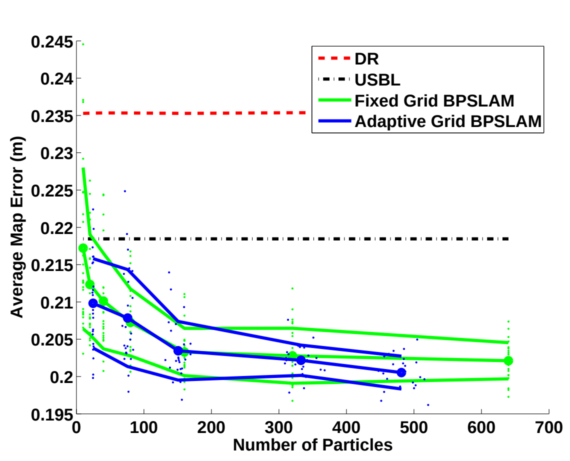
where k is the number of bins (user defined resolution) the particles occupy in state space and $z_{1-\delta}$ is the upper $1-\delta$ quartile of the standard normal distribution. Fox (2001) provides a full derivation of this result. This provides a principled approach to using particles efficiently while also allowing the particle filter to naturally adapt to different mission scenarios.

Note that Equation B.1 estimates the number of particles required to adequately sample the *current* state probability distribution. In the case of pure localisation, i.e. where each particle is referencing a prior map, this is acceptable as the filter need only consider the current state distribution and observations to localise. However in the case of SLAM each particle's map is governed by their own past trajectory. Therefore, in addition to ensuring the current state distribution is adequately sampled we need to further ensure that all past state distributions also remain adequately sampled, i.e. maintain a diverse sample of maps. For this reason it is incorrect to reduce the number of particles using this technique as doing so will often violate this additional condition, resulting in the premature removal of maps which may prove to be the most self-consistent. Adaptive downsampling of the particle set could be properly achieved but would require that the KLD be recalculated for every previous iteration of the filter using only the particles that are currently surviving. Doing this in an efficient way is the subject of future work. The resizing of $\mathbf{S}(t_k)$ is therefore prevented when Equation B.1 dictates a reduction in the number of particles (increasing the number of particles is still allowed as it does not affect the map diversity of the current particle set).

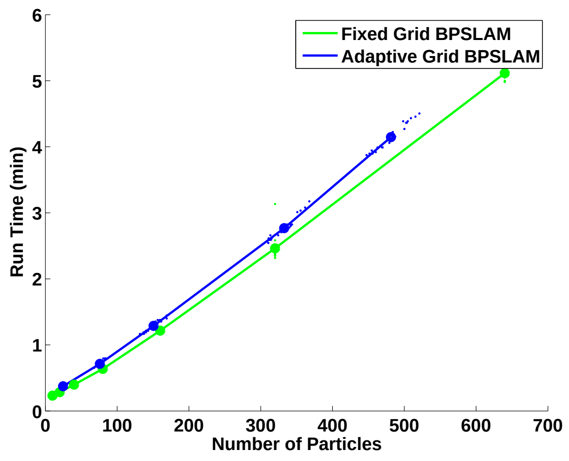
B.2 Results

To analyse the effect adaptive particle sizing has on the performance of the Grid Map BPSLAM filter the Pockmark mission detailed in Section 4.3 is repeatedly run in batches of 25, gridding the state space with 1.0 m resolution bins and using a different KLD threshold for each batch $\varepsilon = [0.85, 0.7, 0.55, 0.3, 0.25]$. Note from Equation B.1 that lower values of ε encourage greater numbers of particles in $\mathbf{S}(t_k)$. For each run the average particle set size, processing time and average registration error of the map were recorded. The performance of the adaptive particle filter was then directly compared to the fixed particle size case in Section 4.3.1 by plotting their performance measures against the average particle set size used in each run. These results are shown in Figure B.1.

Comparing the results achieved from fixed particle sizing to that achieved using adaptive particle sizing confirms that the repeatability and accuracy of the BPSLAM filter is still maintained when less particles are used during periods of low uncertainty in the vehicle state. However using adaptive particle sizing has not created a significant difference in the accuracy and repeatability of the maps produced, though it has increased the run time of the filter slightly, due to the added complexity of the algorithm, as seen in Figure



(a) Variation in map error with average particle set size.



(b) Variation in run time and memory usage with particle set size.

Figure B.1: a) Variation in the map error with a varying average particle set size using both particle set sizing methods (a smaller ε threshold produces a larger average particle set size). The large dots represent the mean error and mean average particle set size used in each batch of 25 runs (shown by the smaller dots). The outer solid lines represent the $\pm 2\sigma$ bound in each batch. Results show an increase in the accuracy and consistency of the map produced in both variations of the filter when the average particle size is increased. b) Variation in the run time with a varying average particle set size using both particle set sizing methods.

1(b). However this comes with the advantage of removing particle size selection from the initial calibration process (the tuning parameter ϵ is much more robust to multiple mission scenarios). This potentially justifies its use in future trials, provided an efficient method of adaptively downsampling the particle set can be found.

Bibliography

Bar-Shalom, Y. & Fortmann, T. (1987), *Tracking and Data Association*, San Diego, CA.

Barkby, S., Williams, S., Pizarro, O. & Jakuba, M. (2011), 'A featureless approach to efficient bathymetric slam using distributed particle mapping', *Journal of Field Robotics* **28**(1), 19–39.

Blasco, S. (2002), 'Application of multibeam surveying to resource mapping', *FIG XXII International Congress, TS4.3 Hydrographic Surveying I*.

Burgard, W., Fox, D., Jans, H., Matenar, C. & Thrun, S. (1999), Sonar-based mapping with mobile robots using EM, *in* 'Proceedings of the International Conference on Machine Learning', Bled,Slovenia, pp. 67–76.

Catanach, R. & German, C. (2011), http://www.whoi.edu.

Chapman, P., Wills, D., Brookes, G. & Stevens, P. (1999), 'Visualizing underwater environments using multifrequency sonar', *IEEE Computer Graphics and Applications* **19**(5), 61–65.

Danson, E. (2006), 'Understanding lidar bathymetry for shallow waters and coastal mapping', *FIG XXIII International Congress*.

Dellaert, F. & Kaess, M. (2006), 'Square root sam: Simultaneous localization and mapping via square root information smoothing', *The International Journal of Robotics Research* **25**(12), 1181–1204.

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

Dong, J., Wijesoma, W. & Shacklock, A. (2007), An efficient rao-blackwellized genetic algorithmic filter for slam, *in* 'Proceedings of IEEE International Conference on Robotics and Automation (ICRA '07)', Roma, Italy, pp. 2427–2432.

Doucet, A., de Freitas, N., Murphy, K. & Russell, S. (2000), 'Rao-blackwellised particle filtering for dynamic bayesian networks', *Proceedings of the 16th Conference on Uncertainty in AI* pp. 176–183.

Durrant-Whyte, H. & Bailey, T. (2006*a*), 'Simultaneous localisation and mapping:part i', *Robotics & Automation Magazine, IEEE* **13**(2), 99.

Durrant-Whyte, H. & Bailey, T. (2006*b*), 'Simultaneous localisation and mapping:part ii', *Robotics & Automation Magazine, IEEE* **13**(3), 108.

Eliazar, A. & Parr, R. (2003), Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks, *in* 'Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)', Vol. 18, Acapulco, Mexico, pp. 1135–1142.

Eliazar, A. & Parr, R. (2004), 'Dp-slam 2.0', *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)* **2**, 1314–1320.

Eliazar, A. & Parr, R. (2005), 'Hierarchical linear/constant time slam using particle filters for dense maps', *Neural Information Processing Systems Conference* (18), 339–346.

Eustice, R. & Singh, H. (2005), Exactly sparse delayed-state filters, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05)', Vol. 3, Barcelona, Spain, pp. 2417–2424.

Fairfield, N., Kantor, G. & Wettergreen, D. (2006), Towards particle filter slam with three dimensional evidence grids in a flooded subterranean environment, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)', Orlando, FL, pp. 3575–3580.

Fairfield, N. & Wettergreen, D. (2008), Active localization on the ocean floor with multibeam sonar, *in* 'Proceedings of the OCEANS '08 Conference', Quebec City, Quebec, pp. 1–10.

Foote, K., Chu, D., Hammar, T., Baldwin, K., Mayer, L. & Hufnagle, L. (2005), 'Protocols for calibrating multibeam sonar', *Journal of the Acoustical Society of America* **117**(4), 2013 – 2027.

Fox, D. (2001), Kld-sampling: Adaptive particle filters, *in* 'Neural Information Processing Systems Conference', Vol. 1, Vancouver, BC, Canada, pp. 713–720.

Garcia, R., Puig, J., Ridao, P. & Cufi, X. (2002), Augmented state kalman filtering for auv navigation, *in* 'Proceedings of IEEE International Conference on Robotics and Automation (ICRA '02)', Vol. 4, Washington DC, USA, pp. 4010–4015.

Girard, A. & Murray-Smith, R. (2003), Learning a gaussian process model with uncertain inputs, *in* 'Technical Report TR-2003-144', Department of Computing Science, University of Glasgow.

Golden, J. (1980), 'Terrain contour matching (tercom): A cruise missile guidance aid', *Image Processing for Missile Guidance* **238**, 10–18.

Gordon, N., Salmond, D. & Smith, A. (1993), 'Novel approach to nonlinear/non-gaussian bayesian state estimation', *Radar and Signal Processing, IEE Proceedings Part F* **140**(2), 107–113.

Grasmueck, M., Eberli, G., Viggiano, D. & Correa, T. (2006), 'Autonomous underwater vehicle (auv) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the straits of florida', *Geophysical Research Letters* **33**(23), L23616.

Grisetti, G., Stachniss, C. & Burgard, W. (2007), 'Improved techniques for grid mapping with rao-blackwellized particle filters', *IEEE Transactions on Robotics* **23**(1), 34–46.

Hartley, R. & Zisserman, A. (2003), *Multiple View Geometry*, Cambridge University Press, Cambridge, England.

Jakuba, M., Williams, S. & Pizarro, O. (2010), High resolution, consistent navigation and 3d optical reconstructions from auvs using magnetic compasses and pressure-based depth sensors, *in* 'Proceedings of the OCEANS '10 Conference', Sydney, Australia, pp. 1–9.

Kirkwood, W. (2007), 'Development of the dorado mapping vehicle for multibeam, subbottom, and sidescan science missions', *Journal of Field Robotics* **24**(6), 487–495.

Liu, J. (1996), 'Metropolized independent sampling with comparisons to rejection sampling and importance sampling', *Statistics and Computing* **6**(2), 113–119.

Lowe, D. (2004), 'Distinctive image features from scale-invariant keypoints', *International Journal of Computer Vision* **60**(2), 91–110.

Lucido, L., Opderbecke, J., Rigaud, V., Deriche, R. & Zhang, Z. (1996), 'A multiscale approach for terrain referenced underwater navigation', *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis* pp. 393–396.

Mahon, I. (2007), Vision-based Navigation for Autonomous Underwater Vehicles, PhD thesis, Australian Centre for Field Robotics - The University of Sydney.

Mahon, I., Williams, S. B., Pizarro, O. & Johnson-Roberson, M. (2008), 'Efficient view-based SLAM using visual loop closures', *IEEE Transactions on Robotics and Automation* **24**(5), 1002–1014.

McIntyre, M., Naar, D., Carder, K., Donahue, B. & Mallinson, D. (2006), 'Coastal bathymetry from hyperspectral remote sensing data: comparisons with high resolution multibeam bathymetry', *Marine Geophysical Researches* **27**, 129–136.

Melkumyan, A. & Ramos, F. (2009), 'A sparse covariance function for exact gaussian process inference in large datasets', *International Joint Conferences on Artificial Intelligence (IJCAI '09)* pp. 1936–1942.

Negahdaripour, S. & Madjidi, H. (2003), 'Stereovision imaging on submersible platforms for 3-d mapping of benthic habitats and sea-floor structures', *IEEE Journal of Oceanic Engineering* **28**(4), 625–650.

Rasmussen, C. & Williams, C. (2006), *Gaussian Processes for Machine Learning*, The MIT Press, MIT, MA.

Roman, C. (2005), Self Consistent Bathymetric Mapping from Robotic Vehicles in the Deep Ocean, PhD thesis, Massachusetts Institute of Technology / Woods Hole Oceanographic Joint-Program.

Roman, C. & Singh, H. (2005), Improved vehicle based multibeam bathymetry using submaps and slam, *in* 'Proceedings of the Intelligent Robots and Systems (IROS '05)', Edmonton, Canada, pp. 3662–3669.

Roman, C. & Singh, H. (2006), Consistency based error evaluation for deep sea bathymetric mapping with robotic vehicles, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)', Orlando, FL, pp. 3568–3574.

Singh, H., Whitcomb, L., Yoerger, D. & Pizarro, O. (2000), 'Microbathymetric mapping from underwater vehicles in the deep ocean', *Computer Vision and Image Understanding* **79**, 143–161.

Tena, R., de Raucourt, S., Petillot, Y. & Lane, D. (2004), 'Concurrent mapping and localization using sidescan sonar', *IEEE Journal of Oceanic Engineering* **29**(2), 442–456.

Thrun, S., Burgard, W. & Fox, D. (2005), *Probabilistic Robotics*, The MIT Press, MIT, MA.

Vasudevan, S., Ramos, F., Nettleton, E. & Durrant-Whyte, H. (2009), 'Gaussian process modeling of large scale terrain', *Journal of Field Robotics* **26**(10), 812–840.

Williams, S., Pizarro, O., Mahon, I. & Johnson-Roberson, M. (2009), 'Simultaneous localisation and mapping and dense stereoscopic seafloor reconstruction using an auv', *Springer Tracts in Advanced Robotics* **54**, 407–416.