Active Localization on the Ocean Floor with Multibeam Sonar

Nathaniel Fairfield David Wettergreen

***Abstract*— We present a method for infrastructure-free localization of underwater vehicles with multibeam sonar. After constructing a large scale (4 km), high resolution (1 m) bathymetric map of a region of the ocean floor, the vehicle can use the map to correct its gradual dead-reckoning error, or to re-localize itself after returning from the surface. This ability to re-localize is particularly important for deep-operating vehicles, which accumulate large amounts of error during the descent through the water column. We use a 3D evidence grid, stored in a efficient octree data structure, to fuse the multibeam range measurements and build maps that do not rely on particular features and are robust to noisy measurements. We use a particle filter to perform localization relative to this map. Both map and filter are general, robust techniques, and both run in real-time. Localization convergence and accuracy are improved, particularly over sparsely varying terrain, by deliberately selecting actions that are predicted to reduce the vehicle's position uncertainty. Our approach to this active localization is to select actions that are expected to generate sonar data that maximally discriminates between the current position hypotheses. Maximal discrimination is a very fast proxy for standard particle filter entropy-based active localization. These methods are demonstrated using a dataset provided by the Monterey Bay Aquarium Research Institute from their mapping AUV, collected near the Axial Seamount in the Juan de Fuca Ridge. Though it depends on the situation, the vehicle's position estimate typically converges to within 2 m of the true position in less than 100 s of traverse, or 150 m at 1.5 m/s. We explore the limitations of our approach, particularly with a smaller number of range sensors: although performance is degraded, satisfactory results are achieved with just four sonars.**

I. Introduction

Autonomous underwater vehicles (AUVs) have the potential to allow scientists to unlock the secrets of the world's oceans by collecting data on temporal and spatial scales and densities that are not feasible by conventional means, such as ships or buoys. One of the great challenges to the deployment of useful deep-sea vehicles is the problem of localization, or the vehicle's self-knowledge of its position. It is a paradox of the underwater domain that localization is straightforward on the surface and within a few hundred meters of the bottom but difficult in the middle of the water column. On the surface, an integrated GPS provides $\sim$ 15 m absolute position information, and once the vehicle has dived to within Doppler sonar range of the bottom, about 200 m, onboard dead-reckoning localization systems based on Doppler velocity log (DVL) and inertial measurement unit (IMU) sensors typically provide $<$ 0.1% of distance traveled dead reckoning position [1].

As a result, localization is straightforward for vehicles that operate in shallow water, where the DVL can reach the bottom while the vehicle is on the surface: they can periodically return to the surface to correct the gradual accumulation of dead reckoning error with GPS. But there is no such simple strategy for vehicles that operate at depths of more than a few hundred meters and must dive down through the water column. If the vehicle relies only on onboard sensors, it must use the DVL to measure the vehicle velocity relative to particles suspended in the water column, a more noisy velocity measurement that must be corrected for currents in the water. As a result, localization based on water-velocity measurements has been shown to be 0.35% of distance traveled [2]. Although localization performance improves once the vehicle is within range of the bottom, the higher error in the middle water column–compounded with the fact that the vehicle can't afford the time or energy to repeatedly return to the surface–has meant that deep-water AUVs have needed to rely on additional, external localization infrastructure: long baseline (LBL) or ultra-short baseline (USBL) acoustic beacons.

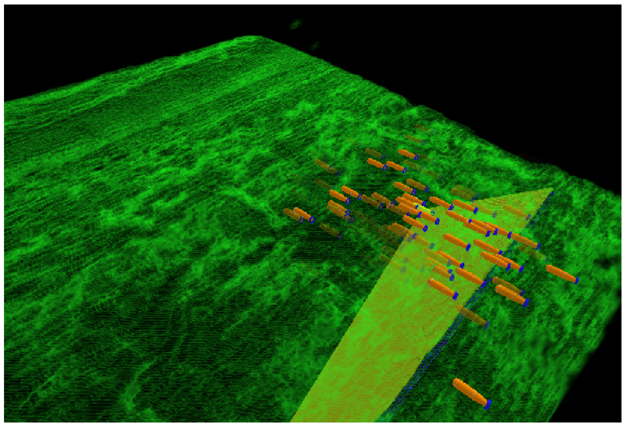
Fig. 1. The particle cloud of possible vehicle locations converging on the true vehicle position (in yellow), showing a portion the evidence grid map of the ocean floor near Axial Seamount and the multibeam sonar swath that is being used to perform localization.

High frequency (300 kHz) LBL localization provides sub-centimeter accuracy, but has a maximum range of only 100 m, while the standard low-frequency (12 kHz) used for long-range localization up to 10 km has 0.1 to 10 m accuracy depending on the range and beacon geometry [3]. Likewise, USBL performance is excellent at short range but long-range performance degrades with distance and depth since the USBL fix is dependent on the angular accuracy of the USBL head, yielding ideal performance of about 0.5% of depth, or 5 m at 1000 m depth [4], [5]. When either LBL or USBL is combined with IMU/DVL based localization onboard the vehicle, the accuracy improves to around 0.2% of depth, or 2 m 1 σ at 1000 m [6].

Although they provide a localization solution for deep-water vehicles, both LBL and USBL require a significant amount of infrastructure and ship time to deploy and operate. LBL networks provide position fixes in a limited area (about 10 km) and must be recovered, redeployed, and surveyed for each new locale, though there has been work on automating the calibration process [7]. USBL is usually attached to a ship, which avoids the redeployment problem, but the ship must periodically return to its station above the vehicle in order to acquire fixes and send them down to the vehicle via acoustic modem. The fact that deep-sea underwater vehicles must be shepherded by a ship vitiates the autonomy of the vehicle, and increases the cost and difficulty of deployment. In some cases, such as under ice, neither LBL nor USBL may be feasible.

We present a method for robust and accurate localization without infrastructure. Instead, the vehicle uses an onboard multibeam sonar to automatically construct bathymetric maps of the sea floor, and in turn uses these maps to re-localize itself – addressing both the large amount of position uncertainty after diving down through the water column, as well as the gradual accumulation dead reckoning error while in range of the bottom (Figure 1. This method is an approach to the Simultaneous Localization and Mapping (SLAM) problem [8]. This problem is a very active area in terrestrial robotics, and has also been addressed in the underwater domain (see Related Work).

Much of the sea floor is flat and featureless – at least from a sonar perspective. It is clear that our basic localization method will not work over truly featureless plains, but when there is at least some variation the vehicle can seek out regions with significant bathymetric variation in order to reduce its position uncertainty. This process of choosing actions in order to aide localization is known as active localization [9]. We present an implementation of active localization that uses very simple metrics to select actions to reduce position uncertainty, when necessary. This capability is crucial during the period immediately after the dive from the surface.

We believe that our main contributions are: 1) a robust real-time localization method, capable of convergence from a large amount of initial uncertainty (such as is inevitable after a dive to 6000 m), 2) automation the map construction process, which can also be done in real-time, 3) scaleability to multiple square kilometers or more, and 4) active localization in using maximally discriminating actions, which makes our approach more robust in regions with little terrain variability.

We demonstrate our methods with an AUV survey dataset that was collected by the Monterey Bay Aquarium Research Institute (MBARI) mapping AUV in 2006 near the Axial Seamount in the Juan de Fuca Ridge.

II. Related Work

There is an enormous amount of work on the SLAM problem, particularly in terrestrial robotics. Since our focus here is on bathymetry-based active localization, we direct the reader to [10] for an excellent survey of underwater navigation, including SLAM methods.

a) Bathymetry-Based Localization: Many SLAM methods depend on the reliable detection of features, such as stop signs, in the environment. For an underwater vehicle, the main task is often to generate maps of the bathymetry. This, coupled with the difficulty of extracting reliable features from sonar data, encourages the use of the bathymetry itself, rather than feature-based approaches, for localization.

Approaches to bathymetry-based localization have become common, largely due to the recent availability of multibeam sonar systems. Roman and Singh [11] use an extended Kalman filter (EKF) and a variant of the Iterative Closest Points algorithm to match submaps of sonar measurements. As in our method, Ura et al. [12] use a particle filter, but their mesh-based map is only 150 m x 150 m. Sarma [13] shows theoretical bounds for bathymetry-based localization.

b) Active Localization: Active localization has long been an area of research, beginning with exploration and action selection in uncertain environments [14] [15] [16]. Generally, taking uncertain actions in an uncertain environment can be modelled as a partially observable Markov decision process (POMDP), as in [17]. POMDPs are theoretically satisfactory but computationally intractable for any realistically complicated scenario, so heuristics (like greedy action selection) are used.

Burgard et al. [9] distinguishes between active navigation, in which the entire robot moves, and active sensing, in which only the sensors are pointed or focussed on particular targets. Kümmerle at al. [18] do active sensing, clustering the particles into groups and calculating the total expected entropy for the particle filter by a weighted average of the expected entropy for each group. This is an example of the general technique of estimating the expected entropy of the entire position filter (not always a particle filter) after taking different actions [19], but we turn the problem around by directly comparing the data (range measurements) that would be generated by different actions. Intuitively, actions that generate different data for different particles will allow the particle filter to discriminate amongst the particles, and simulating range measurements is far faster than simulating the behavior of the entire particle filter.

There is not much work in underwater active localization; despite the name of ”coastal navigation” Roy and Thrun [20] construct an entropy map for a museum robot by simulating the change in entropy of the particle filter for each possible point in the map. The entropy map is computationally intensive, and is usually precomputed. A further limitation is that the simulated particle filter is started in some default state, which may or may not be similar to the actual particle filter state, and thus the precomputed entropy values may be incorrect. For example, the map may have been computed for a particular heading and a Gaussian particle distribution, but the actual heading may be different or the particle cloud may have a multimodal distribution. Our method selects actions based on the current state of the map and the particle filter.

Fig. 2. The MBAUV aboard the R/V Zehpyr. *Image: Duane Thompson (c) 2005 MBARI*

III. Vehicle

MBARI has developed the 6000 m rated Multibeam Mapping AUV (MBAUV) (Figure 2), based on the modular 21" diameter Dorado AUV design [21]. The navigation system is built around a Kearfott SEADeViL integrated IMU/DVL. The primary mapping system is a Reson 7125 200 kHz multibeam sonar. The multibeam sonar array provides an array of 256 1° by 1° beams spread in a downward facing 150° fan perpendicular to the AUV's direction of travel. The update rate depends on the range to the bottom, but is generally about 2 Hz. The mapping AUV has demonstrated very successful survey operations [22], including a survey of Axial Seamount in 2006. The MBARI mapping team has generously provided this dataset, which includes both the navigation data from the onboard IMU/DVL and sonar data from the multibeam sonar system.

IV. Method

Our method is a general approach to SLAM in large-scale natural environments using range sensors. The major components of our implementation are a map representation and a particle filter. The map representation is a 3D evidence grid stored in an efficient data structure (described below), that can represent arbitrary 3D geometry. The particle filter is likewise a general technique, which can be adapted to any vehicle by plugging in the appropriate vehicle motion and range sensor models. Our basic method was originally developed for the DEPTHX vehicle [23], and has been applied, in unpublished work, to several other aerial, terrestrial, subterranean, and underwater robots.

A. Particle Filter SLAM

The goal of a SLAM system is to estimate the probability distribution at time t over all possible vehicle states s and world maps Θ using all previous sensor measurements Z_t and control inputs U_t (for a complete list of notation, see Table I):

$$p(s, \Theta | Z_t, U_t)$$

$s_t^{(m)}$	vehicle state of the m-th particle at time t
$S_t^{(m)}$	trajectory of m-th particle from time 0 to t $= \{s_0^{(m)}, s_1^{(m)}, s_2^{(m)}, \ldots, s_t^{(m)}\}$
z_t	sonar measurements at time t
Z_t	history of measurements from time 0 to t $= \{z_0, z_1, z_2, \ldots, z_t\}$
u_t	vehicle dead-reckoned innovation at time t
U_t	history of dead-reckoning from time 0 to t $= \{u_0, u_1, u_2, \ldots, u_t\}$
$\Theta^{(m)}$	map of m-th particle
$w_t^{(m)}$	m-th particle weight at time t

TABLE I
Particle filter notation.

This distribution is called the *SLAM posterior*. The recursive Bayesian filter formulation of the SLAM problem is straightforward (see, for example, [24] for a derivation) but is usually computationally intractable to solve in closed form. Particle filters are a Monte Carlo approximation to the Bayesian filter. The particle filter maintains a discrete approximation of the SLAM posterior using a (large) set of samples, or *particles*.

For practical purposes, when SLAM is being used to provide a pose for the rest of the vehicle control software, we usually want to turn the set particles into a single point estimate. If the posterior distribution is Gaussian, then the mean is a good estimator, but other estimators may be better if the distribution becomes non-Gaussian – which is indicated by high position variance.

The particle filter algorithm, based on the Sampling Importance Resampling Filter of [25], has the following steps:

1) **Initialize** The particles start with their poses s_0 initialized according to some initial distribution and their maps Θ (possibly) containing some prior information about the world. This is called the *prior distribution*, and it may be very large if the initial position is uncertain.
2) **Predict** The dead-reckoned position innovation u_t is computed using the navigation sensors (IMU, DVL and depth sensor). A new position s_t is predicted for each particle using the vehicle dead reckoning-based motion model:
$$s_t = h(s_{t-1}, u_t, N(0, \sigma_u)).$$
This new distribution of the particles is called the *proposal distribution*.
3) **Weight** The weight w for each particle is computed using the measurement model and the sonar range measurements (from the $\#_{son}$ different sonars):
$$w = \eta \prod_{n=1}^{\#_{son}} p(z_t | s_t, \Theta),$$
where η is some constant normalizing factor. In our implementation, the real range measurements z are compared to ray-traced ranges $\hat{z}$ using the particle pose

and map. We compare the simulated and real ranges using the measurement model

$$z = g(s_t, u_t, N(0, \sigma_z)),$$

which is assumed to have a normal noise model, so

$$p(z|s, \Theta) = \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{\frac{-(\hat{z}-z)^2}{2\sigma_z^2}}.$$

Substituting into the expression for particle weight and taking the logarithm of both sides shows that maximizing this weight metric is very close to minimizing the intuitive sum squared error metric:

$$\log w = C - \frac{1}{2\sigma^2} \sum_{i=1}^{\#_{son}} (\hat{z} - z)^2,$$

where $C = \#_{son} \times \log\left(\sqrt{2\pi\sigma^2}\right)$.

4) **Resample** The $O(n)$ algorithm described in [26] is used to resample the set of particles according to the weights w such that particles with low weights are likely to be discarded and particles with high weights are likely to be duplicated.
Frequently, a rule of thumb introduced by [27] based on a metric by [28] is used to decide whether or not to resample:

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

so that resampling is only performed when the number of effective particles N_{eff} falls below half the number of particles,$N/2$.
Another trick that is useful in slowing the convergence of the particle filter is to add noise to the resampled particle positions, effectively re-initializing the particle filter around the particles with high weights. The amount of resampling noise can be gradually reduced, as in simulated annealing. After resampling, the set of particles is a new estimate of the new SLAM posterior.
5) **Update** The measurements z are inserted into the particle maps $\Theta^{(m)}$ to update the map according to the sonar beam model of each measurement relative to the particle position. This is when maps must be copied and updated. We save duplicate insertions by inserting *before* copying successfully resampled particles. Note that in this paper, we are primarily testing localization, and do not update the map after it has been constructed.
6) **Estimate** Generate a position estimate from the particles.
7) **Repeat from Predict**

We now describe the vehicle model and sonar sensor model particular to the MBAUV, as well as the 3D map representation.

B. Vehicle Motion Model

The state vector s of the vehicle model is the static 6 DOF vehicle pose q, together with its first and second derivatives:

$$q = [\phi, \theta, \psi, x, y, z]$$
$$s = [q, \dot{q}, \ddot{q}]$$

As described above, this state vector is updated according to the vehicle model

$$s_t = h(s_{t-1}, u_t, N(0, \sigma_u))$$

where u is the control vector, in this case the measurements from the navigation sensors, and σ_u is the corresponding Gaussian noise for each measurement. For example, if a GPS position fix is available, then the x and y fields of u are set to the fix position, and the corresponding fields of σ_u are set to 15 m, or the standard deviation of the GPS fix.

During the prediction step the particle filter updates each particle position using the available measurements and sampling from the Gaussian noise model for each measurement. Under prediction alone, the particles will gradually disperse according to the navigation sensor error model, representing the gradual accumulation of dead reckoning error.

C. 3D Octree Evidence Grids

An evidence grid is a uniform discretization of space into cells in which the value indicates the probability or degree of belief in some property within that cell. In 3D, the cells are cubic blocks of volume, or voxels. The most common property is occupancy, so evidence grids are often also called occupancy grids [29]. The primary operations on a map are inserting new evidence, querying to simulate measurements, and copying the entire map, which is necessary for the update step of the particle filter. We call the process of updating all of the voxels which are affected by a particular measurement, effectively inserting information into the map, an *insertion*, and likewise the process of casting a ray within a map until intersects with an occupied voxel we call a *query*. Often, the log-odds value for each voxel θ

$$l(\theta) = \log\left(\frac{p(\theta)}{1 - p(\theta)}\right)$$

is stored in the map rather than the raw probabilities because it behaves better numerically, and because the Bayesian update rule for a particular voxel according to the sensor model (like a conic beam-pattern) for some measurement z becomes a simple addition [29]:

$$\log(\theta') = \overbrace{\log\left(\frac{p(\theta|z)}{1 - p(\theta|z)}\right)}^{\text{beam model}} + \overbrace{\log\left(\frac{1 - p(\theta)}{p(\theta)}\right)}^{\text{map prior}} + \log(\theta).$$

The first term on the right-hand-side is the sensor model (discussed next), and the second is the map prior, or the expected occupancy of space before we have collected any measurements. If the prior $p(\theta) = 0.5$, the second term is zero and the initialization simply sets all voxels to zero. By

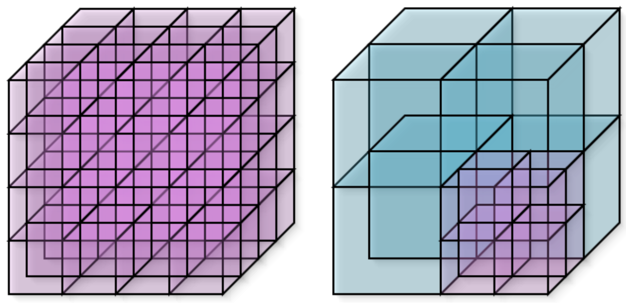
Fig. 3. On the left is a uniform volumetric grid, made up of cubic voxels. On the right is an octree: each level of an octree divides the remaining volume into eight octants, but the tree does not have to be fully expanded.

using log-odds, the update for each voxel can be reduced to simply summing the value given by the sonar model with the current voxel evidence.

The main difficulty with the 3D evidence grid approach arises from the cost of storing and manipulating large, high resolution maps. If the evidence log-odds are represented as single bytes (with values between -128 and 127), then an evidence grid 1024 cells on a side requires a megabyte of memory in 2D and a gigabyte in 3D.

We have developed the Deferred Reference Count Octree (DRCO) described in depth in [23], that offers a compact and efficient octree-based data structure that is optimized for use with a particle filter. An octree is a tree structure composed of a node, or *octnode*, which has eight children that equally subdivide the node's volume into *octants* (Figure 3). The children are octnodes in turn, which recursively divide the volume as far as necessary to represent the finest resolution required. The depth of the octree determines the resolution of the leaf nodes. The main advantage of an octree is that the tree does not need to be fully instantiated if pointers are used for the links between octnodes and their children. Large contiguous portions of an evidence grid are either empty, occupied, or unknown, and can be efficiently represented by a single octnode – truncating all the children which would have the same value. As evidence accumulates, the octree can compact homogeneous regions that emerge, such as the large empty volume inside a cavern.

The efficiency of the DRCO depends on the circumstances. It will be most efficient when the environment displays volumetric sparsity, since octrees compactly represent a map that is mostly empty or full. Most large-scale outdoor environments seem to be well suited for this type of exploitation.

Next, we discuss how a sonar sensor model is used to update and query the map.

D. Multibeam Sonar Model

The Reson 7125 multibeam sonar operates at 200 kHz and provides an array of 256 1° by 1° beams spread in a downward facing 150°swath perpendicular to the AUV's direction of travel [22]. Simplifying somewhat, the 1° degree beamwidth means that the range value returned by the sonar could have

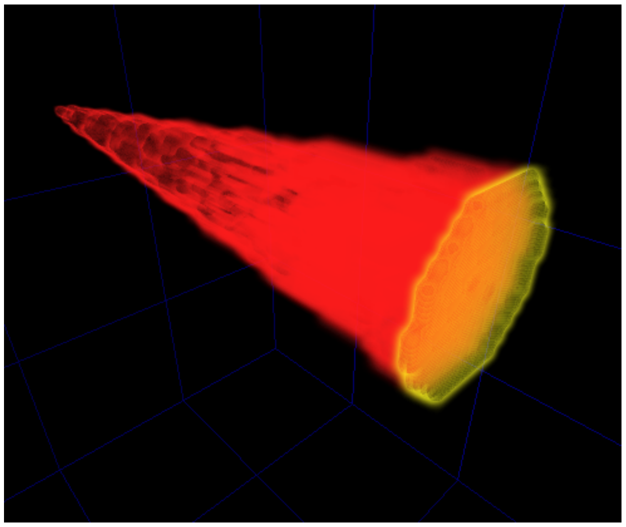
Fig. 4. A single 6° sonar beam as represented in an evidence grid – note that the actual beams are only 1°. Unknown regions are transparent, probably occupied regions are yellow, probably empty regions are red. In this visualization, very low and very high probability of occupancy regions have been clipped out, leaving isodox (equal occupancy belief) shells.

been caused by a reflection from anywhere within a cone projecting from the sonar transducer. At long distances, the sonar data is coarse, meaning that fine structure cannot be resolved perpendicular to the direction of the beam – but over time they do provide information about the environment around the vehicle. As measurements are collected, the evidence they provide about the occupancy of each voxel is entered into the evidence grid map. Given a particular range measurement, a sonar beam model specifies which voxels within the cone are probably empty and which voxels are probably occupied. There are several methods which can be used to construct a beam model, including deriving it from physical first principles or learning it [29]. We chose to use the simplest reasonable approximation – a cone with a cap that is loosely based on the beam-pattern of the sonar (Figure 4). The cone is drawn as a bundle of rays with constant negative value, with terminating voxels with constant positive values.

Likewise, the simplest method to query a sonar range from the 3D evidence grid is to cast a ray until some threshold (or other terminating condition) is met. Using matrix transformations for each voxel is too computationally expensive for operations such as filling in evidence cones or simulating ranges. These tasks can be decomposed into raster operations, which are performed with a 3D variant of the classic 2D Bresenham line drawing, or ray-casting, algorithm [30]. We have implemented these line-drawing operations on the octree-based DRCO that run within a (small) constant factor of the same operations on a uniform array.

It should be pointed out that while the 3D DRCO is a very general map representation, capable of representing arbitrary 3D geometry, it is more general than necessary for the specific task of localizing relative the ocean floor, which can almost always be represented more efficiently as a height map – which also has faster line-drawing operations.

E. Active Localization

Active localization gives the localization system the ability to influence the behavior of the vehicle. In particular, when the position estimate becomes uncertain, active localization can recommend actions that are expected to reduce this uncertainty. Since large regions of the ocean floor (and our dataset in particular) are basically flat and featureless, simply flying in a straight line may not be sufficient for quick and accurate position convergence – and active localization is extremely useful, if not essential.

As discussed above, many active localization methods use an information-theoretic framework, in which the entropy of the entire particle filter is estimated using a variety of heuristics [19]. By simulating the effect of executing several different candidate actions on the particle filter, the action that is expected to lead to the lowest entropy can be selected. But this is difficult: not only is the simulation computationally expensive, but it must be repeated for several different simulated datasets: ideally one per candidate action per particle. To avoid this, Stachniss et al. [31] only simulate the actions for a weighted subset of particles, while Kümmerle et al. [18] attempt to cluster the particles into groups and run one simulation per cluster.

But when we are just localizing, rather than updating maps in full SLAM, we just want to find an action that allows the filter to discriminate between particles, yielding a unimodal particle cloud around the true vehicle position. We would like to find this action without explicitly simulating the particle filter state and then evaluating its entropy. We can do this by examining the simulated datasets from a subset of particles: the most discriminative action is that which generates the simulated datasets that are the most different. Since the real dataset which results from taking that action will only be similar to a few of the particles' simulated datasets, we know that all of the other particles will be discarded. Simulating datasets is fast, only requiring ray-casting and a naive vehicle motion model, and these datasets can be quickly compared using, for example, sum-squared difference.

Thus, if A is the set of candidate actions, M is the set of all particles, and $m \subseteq M$ is a subset of particles generated by sampling according to the particle weights, then for each action $a \in A$ and particle $q \in m$, we simulate the range measurements z_a^q using the particle pose, the simulated vehicle trajectory as a result of taking the action, and the map. We then find the most discriminative action:

$$\arg\max_{a \in A} \sum_{i \in m} \sum_{j \in m} (z_a^i - z_a^j)^2$$

By periodically repeating this action selection process, the vehicle will select actions which are expected to allow it to best discriminate amongst its current set of particles, thus reducing the particle filter entropy – without ever explicitly estimating the entropy. Due to its simplicity, this approach can be run faster, for longer action horizons, than full filter simulation and entropy estimation methods.

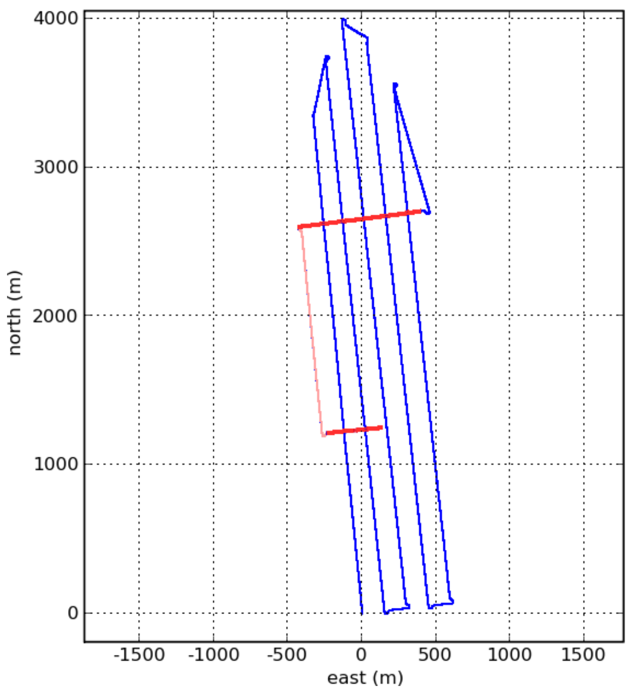

Fig. 5. Axial survey tracklines – the long tracklines were used to create the map and random segments of the cross-track lines were used to verify localization.

The set of candidate actions can be arbitrarily complicated, though the computational expense of evaluating them increases accordingly. In domains where the vehicle motion is constrained (by walls, for example), heuristics can applied to rapidly eliminate infeasible actions. In our experiments in the open underwater domain described below, we simply used straight-line motion for 30 m in one of the 8 cardinal and intercardinal directions as our set of candidate actions.

F. Results

As described above, our test dataset is from Axial and Monterey Canyon (Figure 5), a 23.3 km survey in 5 legs of about 4 km, plus two cross-track legs, in about 4.4 hours (Figure 5). After examining the dataset, we discarded the last 27 out of 256 beams, which seemed to be very noisy. We divided the dataset into two pieces: the trackline legs and the cross-track legs. We then built a map with the trackline legs, and used the cross-track legs to test our localization methods.

1) Map construction: We used the onboard dead reckoning, together with the sonar range measurements, to construct an octree evidence grid map at 1 m resolution. The octmap dimensions were 8192^3, and although only a small fraction of this volume ended up being used, the large size meant that we didn't have to worry about the vehicle driving off the edge of the map – this scaleability is essential for a real-world system.

It took 22 minutes to construct the map on a 3 GHz P4 computer using the 20 km of trackline legs, a total of 6.5

million range measurements. Since this portion of the mission took 225 minutes to run, this is about 10 times realtime.

Un-compacted, the octree map size was 530 MB. After running lossy compaction, in which the octree map was compacted by thresholding into {empty, occupied} and homogeneous regions were consolidated, the map was just 78 MB. As an upper bound, a uniform array of 4096 x 4096 x 4096 would be 65536 MB. As a rough lower bound, a minimal 1 m resolution uniform array tightly fit to to the survey area would be about 4096 x 1024 x 20 = 100 MB; the compact octree efficiently represents the volume.

2) Localization: Although the dataset was collected at about 1500 m, we would like to demonstrate that our method can converge from the initial descent error that would result from a descent of 1000 and 6000 m. The vehicle travels at about 1.5 m/s and dives at about 30 degrees pitch, yielding a 0.75 m/s dive rate. Brokloff [2] reports performance of 0.16% of distance traveled with bottom tracking and 0.35% with only water tracking. Since the MBAUV has much better bottom tracking performance, 0.05% of distance traveled [22], we hope that assuming 0.35% during descent is a suitably conservative estimate. Thus, if the vehicle needs to dive 1000 m to bring the DVL into range of the bottom, the final position error on the bottom is 1000 m $\div$ 0.75 m/s $\times$ 0.35% = 4.7 m 1 σ. The same calculation for diving 6000 m yields a position error of 28 m 1 σ.

To evaluate our method, the particle filter used the map described above, and attempted to localize using data from the as-yet unused cross-track legs. This process is summarized in Figure 6. We used several different segments of the cross-track legs for testing. For a given segment of one of these legs, we first perturbed the initial position according to a 5 or 28 m 1 σ Gaussian (depending on whether we simulating a 1000 or 6000 m dive), and then distributed the particles around this perturbed mean according to a 10 or 35 m 1 σ Gaussian distribution. The distribution of the particles was larger than the dive-length derived position perturbation to ensure that there were particles near the true position. Repeating this entire process, we evaluated the particle filter performance over multiple runs according to how quickly and accurately it converged to the "ground truth" position. Convergence, indicated by low particle cloud variance, is an important metric because it indicates when the filter is confident in a unique position estimate. In the broader context of AUV operation, convergence indicates that the AUV is well localized and can begin to perform its other tasks on the sea floor.

The particle filter did not always converge, as indicated by low particle position variance: Figure 7 shows an example of successful convergence, while Figure 8 shows a failure to converge because there are multiple modes, one of which is the correct one, so the filter could converge given more data.

Convergence was consistent for the 1000 m test (Figure 9), but sometimes failed (detectably) for the 6000 m test (Figure 10). Since the variance remained high for these failed runs, the vehicle could choose to restart the entire localization process (with perhaps a different initial position bias), repeating the procedure until it successfully localized.

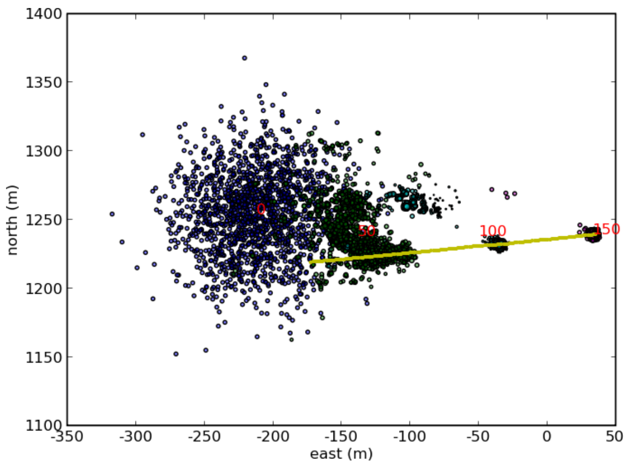

Fig. 7. Successful particle filter convergence: the particle cloud is unimodally distributed around the true position. Note that the mean of the initial particle distribution is significantly perturbed from the true initial position shown by the left end of the yellow line.

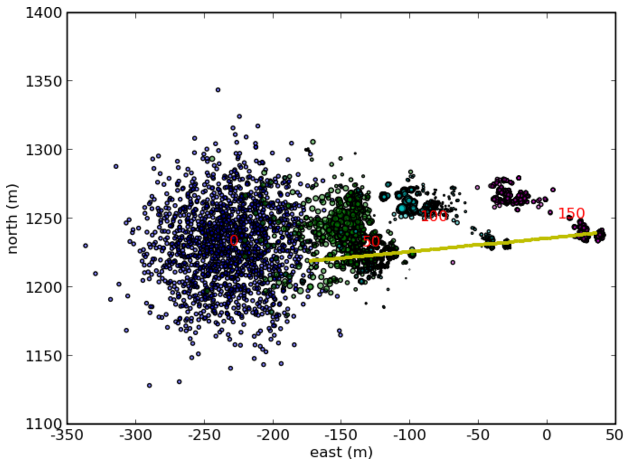

Fig. 8. Failed particle filter convergence: there are still multiple particle modes, one of which is the correct position, so the particle filter could still converge to the correct position given more data. Note that the mean of the initial particle distribution is significantly perturbed from the true initial position shown by the left end of the yellow line.

There are many variables which affect the number of particles that can be supported in real-time. One is the resolution of the map, which was 1 m for these experiments. Another is the number of range measurements which must be simulated during the weighting step of the particle filter. The multibeam collects 256 ranges at 2 Hz, but not all beams need be used for weighting. In particular, because all 256 1° beams are packed into a 150° swath, they overlap almost double. Thus, in realtime on a 3 GHz P4 computer, we can either support over 400 particles with all ranges, or over 800 particles if we decimate the multibeam fan by a factor of two. Likewise, it is reasonable to decimate the data along the vehicle's direction of travel, since it only moves about 0.75 m between pings. By adjusting these decimation factors, we sometimes ran with 1600 particles in real time: the benefits of additional particle

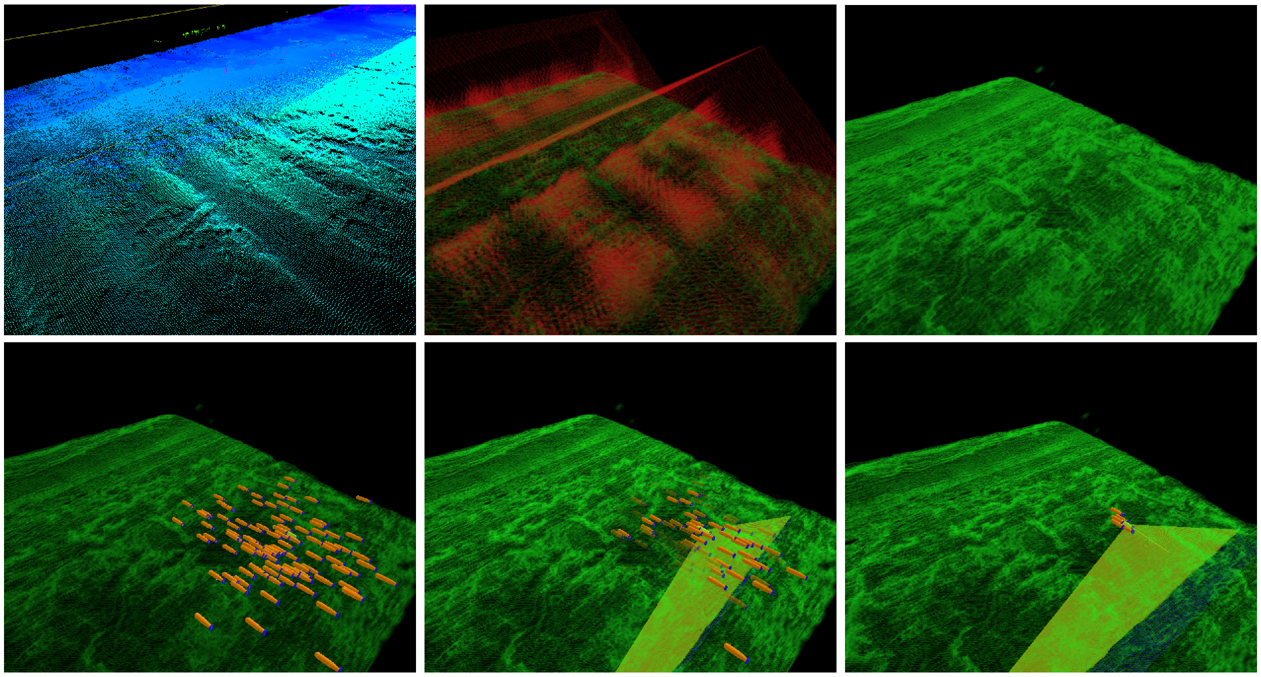

Fig. 6. Localization example showing 1) sonar pointcloud from two tracklines, 2) a portion of the evidence grid constructed from all the tracklines, red is known empty space and green is known occupied space, 3) the occupied portion of the evidence grid, 4) the true vehicle position (in yellow) at the beginning of a cross-track line (not used to construct the evidence grid) and the particle cloud of possible positions, 5 & 6) the iterative convergence of the particle cloud to the true position over the course of a few seconds.

density outweighed the loss of denser range data. However, for the results presented in this paper, we ran with 800 particles and a multibeam fan decimation factor of 2, which put us comfortably within realtime.

By decimating almost all of the ranges, we can also simulate a vehicle with a simpler sonar system, although the full multibeam system is necessary to construct the high-resolution map. Individual pencil-beam sonars (or altimeters) are far simpler and cheaper than a multibeam system, and the DVL itself provides up to 4 beams. Figure 11 shows that convergence for the 1000 m test was slow, but reliable with just 4 beams. Anecdotally, the fewer the beams, the more sensitive convergence was to the amount of terrain variation.

Finally, the three tests, 6000 m, 1000 m, and 1000 m with 4 beams are compared in Figure 12.

3) Active Localization Results: Using the map constructed above, we simulated vehicle motion and range data so that we could allow the simulated vehicle to take different actions. We compared three methods for choosing the heading for the next 30 m leg of vehicle travel: a constant heading, a random heading, and an actively selected heading in which the vehicle chose between 8 different headings (at 45 degree increments) by finding the most discriminative action, as described above. Note that, as shown by the localization results with real data, sometimes traveling on a constant heading can yield excellent convergence: in order to distinguish whether active localization has an effect, we selected a starting location and direction in which constant heading was known to fail due to the lack of terrain variation. But even when constant heading fails we also needed to show that our action selection process was better than simply choosing a random heading. Localization convergence (Figure 13) shows that active heading selection significantly outperforms both constant heading and random heading selection.

V. Conclusion

We have demonstrated robust real-time localization with multibeam sonar data. Using conservative models of the vehicle's position error after 1000 m and 6000 m dives, we ran large numbers of randomized tests on multiple segments of real data. Although localization with 800 particles does not always converge for the 6000 m tests, the failure to converge is detectable and can be addressed either by restarting the localization procedure, or by applying active localization. When the filter converged, it did so within 200 s (half that for the 1000 m dive tests), yielding a single position estimate within 2 m of ground truth, which is competitive with the best LBL/USBL localization results. We have shown that by selecting the most discriminative action, active localization can yield accurate convergence in situations where the terrain variation is sparse and convergence does not necessary occur at all. We have also shown that our localization method converges with just 4 sonar beams, meaning that it could be applied to vehicles with less expensive sonar systems.

a) Future work: While the octree efficiently stores very large maps, it is limited by available memory. By segmenting the world into submaps, perhaps by only loading portions of the octree, we can improve the scaleability. Also, our method

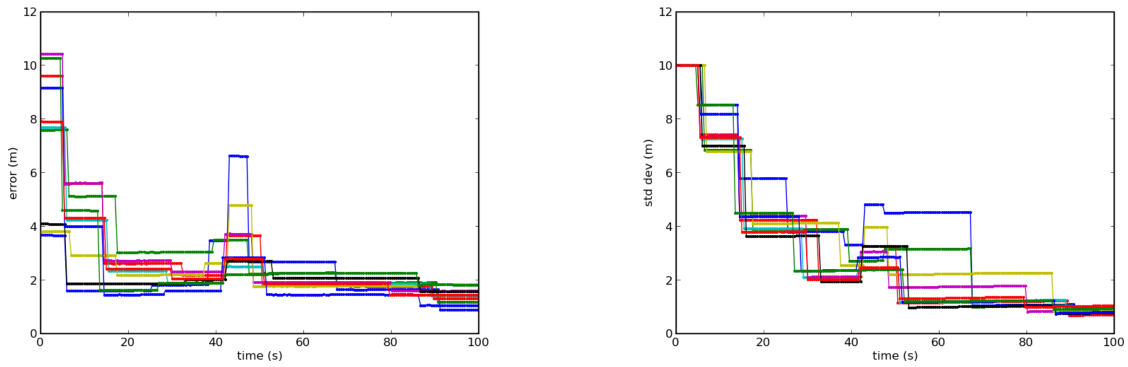

Fig. 9. Convergence plot for 10 different runs of the 1000 m test: on the left is the position error over time, and on the right is the standard deviation of the particle positions over time. The fact that all the particles converge to a position error of around 1.5 may indicate that our ground truth is slightly incorrect.

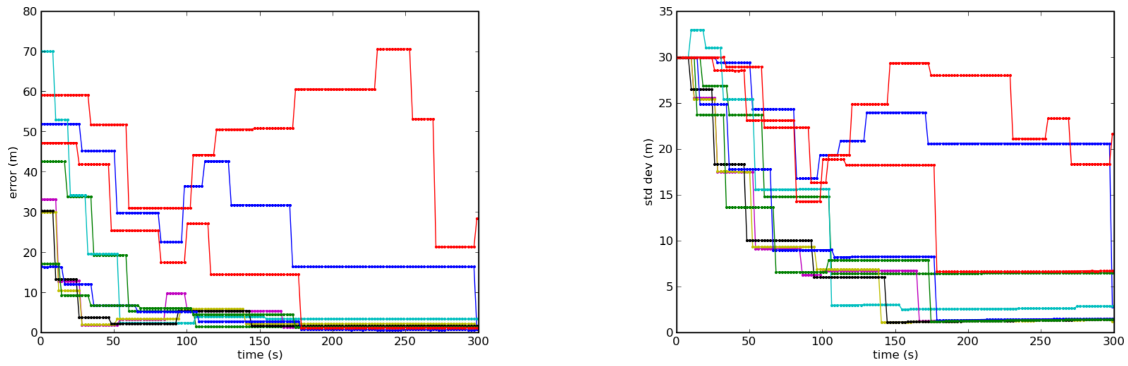

Fig. 10. Convergence plot for 10 different runs of the 6000 m test: on the left is the position error over time, and on the right is the standard deviation of the particle positions over time. Note that several runs failed to converge, but this can be detected by observing that the standard deviation stays high (indicating a disperse or multimodal particle distribution).

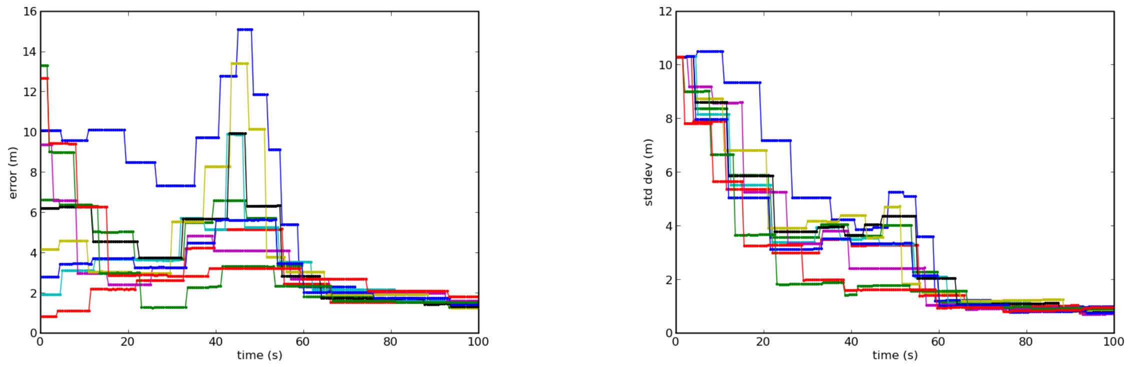

Fig. 11. Convergence 10 different runs of the 1000 m test with just 4 sonar beams: on the left is the position error over time, and on the right is the standard deviation of the particle positions over time.

for selecting candidate actions is intuitively satisfactory, but we would like to pursue a more rigorous foundation. It is clear that our approach would benefit from more particles, particularly for the 6000 m test. We may optimize our code, or we may wait for Moore's law to solve the problem. Finally, for the sake of simplicity the method described in this paper involved two distinct phases: map construction and localization. These phases can clearly be unified into a single SLAM system that would allow the vehicle to take advantage of newly mapped regions without returning to the surface.

Acknowledgment

The authors would like to thank David Caress, Charles Paull, and David Clague of the Monterey Bay Aquarium Research Institute for providing us with the dataset.

References

[1] M. B. Larsen, "High performance doppler-inertial navigation experimental results," in *Proc. of IEEE/MTS OCEANS*, 2000, pp. 1449–1456.

[2] N. Brokloff, "Dead reckoning with an adcp," *Sea Technology*, pp. 72–75, Dec 1998.

[3] L. Whitcomb, D. R. Yoerger, and H. Singh, "Combined doppler/LBL based navigation of underwater vehicles," in *Proc. of the 11th Intl. Symposium on Unmanned Untethered Submersible Technology*, August 1999.

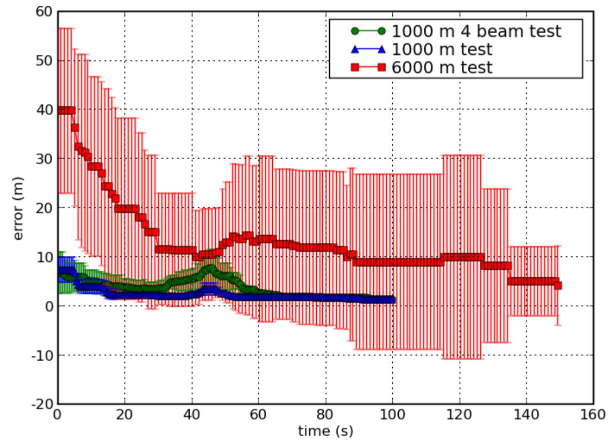

Fig. 12. Comparison of the average performance, with error bars, of 10 runs of the three types of tests. Note that the final error of the 6000 m test is so poor because several runs (detectably) failed to converge (see Figures 9 10 11).

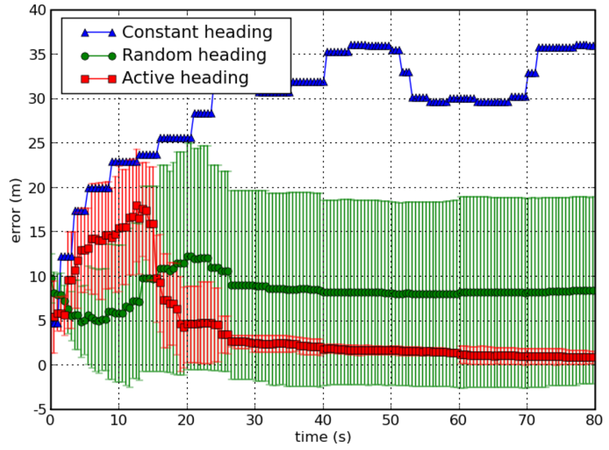

Fig. 13. Comparison if the average performance, with error bars, of several action selection strategies. This plot shows that actively selecting the next vehicle heading, as described in the text, reliably converges better than either keeping a constant heading or randomly selecting the next heading. The error bars indicate the standard deviation of the position error across 5 runs (except for constant heading which always has nearly the same result).

[4] J.-P. Peyronnet, R. Person, and F. Rybicki, "Posidonia 6000: a new long range highly accurate ultra short base line positioning system," in *OCEANS '98 Conference Proceedings*, vol. 3, Sep-1 Oct 1998, pp. 1721–1727 vol.3.

[5] B. Jalving and K. Gade, "Positioning accuracy for the hugin detailed seabed mapping uuv," in *OCEANS '98 Conference Proceedings*, vol. 1, Sep-1 Oct 1998, pp. 108–112 vol.1.

[6] B. Jalving, K. Gade, O. Hagen, and K. Vestgard, "A toolbox of aiding techniques for the hugin auv integrated inertial navigation system," *OCEANS 2003. Proceedings*, vol. 2, pp. 1146–1153 Vol.2, Sept. 2003.

[7] E. Olson, J. Leonard, and S. Teller, "Robust range-only beacon localization," in *Proceedings of Autonomous Underwater Vehicles, 2004*, 2004.

[8] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990.

[9] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization by entropy minimization," in *Advanced Mobile Robots, 1997. Proceedings., Second EUROMICRO workshop on*, Oct 1997, pp. 155–162.

[10] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, Lisbon, Portugal, September 2006, invited paper.

[11] C. Roman and H. Singh, "Improved vehicle based multibeam bathymetry using sub-maps and slam," *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3662–3669, Aug. 2005.

[12] T. Ura, T. Nakatani, and Y. Nose, "Terrain based localization method for wreck observation auv," in *OCEANS 2006*, Sept. 2006, pp. 1–6.

[13] A. Sarma, "Maximum likelihood estimates and cramer-rao bounds for map-matching based self-localization," *Oceans 2007*, pp. 1–10, 29 2007-Oct. 4 2007.

[14] S. Thrun, "Exploration and model building in mobile robot domains," in *In Proceedings of the IEEE International Conference on Neural Networks*, 1993, iEEE Neural Network Council.

[15] S. Koenig and R. Simmons, "Exploration with and without a map," in *Proc. of the Workshop on Learning Action Models, National Conference on Artificial Intelligence*, Washington, DC, July 1993.

[16] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1080–1087.

[17] A. Cassandra, L. Kaelbling, and J. Kurien, "Acting under uncertainty: discrete bayesian models for mobile-robot navigation," *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, vol. 2, pp. 963–972 vol.2, Nov 1996.

[18] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Active monte carlo localization in outdoor terrains using multi-level surface maps," in *Proc. of Field and Service Robotics*, Chamonix, France, 2007.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[20] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Processing Systems 12*, vol. 12, 1999, pp. 1043–1049.

[21] W. Kirkwood, D. Caress, H. Thomas, M. Sibenac, R. McEwen, F. Shane, R. Henthorn, and P. McGill, "Mapping payload development for mbari's dorado-class auvs," in *Proc of MTS/IEEE OCEANS*, vol. 3, Nov. 2004, pp. 1580–1585 Vol.3.

[22] R. Henthorn, D. Caress, H. Thomas, W. Kirkwood, R. McEwen, C. Paull, and R. Keaten, "High-resolution multibeam and subbottom surveys of submarine canyons and gas seeps using the mbari mapping auv," in *Proc of MTS/IEEE Oceans*, Boston, Mass., September 2006.

[23] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time slam with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, 2007.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. of the AAAI National Conference on Artificial Intelligence*, 2002, pp. 593–598.

[25] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *Proc. Inst. Elect. Eng. F*, vol. 140, April 1993, pp. 107–113.

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

[27] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proc. of the Sixteenth Conf. on Uncertainty in AI*, 2000, pp. 176–183.

[28] J. S. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Statistics and Computing*, vol. 6, no. 2, pp. 113–119, June 1996.

[29] M. Martin and H. Moravec, "Robot evidence grids," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-06, March 1996.

[30] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[31] C. Stachniss, G. Grisetti, D. Hähnel, and W. Burgard, "Improved rao-blackwellized mapping by adaptive sampling and active loop-closure," in *Proc. of the Workshop on Self-Organization of AdaptiVE behavior*, 2004, pp. 1–15.