Robust Range-Only Beacon Localization

Edwin Olson, John Leonard, and Seth Teller
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
eolson@mit.edu, jleonard@mit.edu, teller@csail.mit.edu

***Abstract*— We present a system capable of simultaneously estimating the position of an Autonomous Underwater Vehicle (AUV) and the positions of stationary range-only beacons. Notably, our system does not require beacon positions a priori, and our system performs well even when range measurements are severely degraded by noise and outliers. We present a powerful outlier rejection method that can identify groups of range measurements that are consistent with each other, and a method for initializing beacon positions in an EKF. We have successfully applied our algorithms to real-world data and have demonstrated a SLAM system whose navigation performance is comparable to that of systems that assume known beacon locations.**

***Index Terms*— Outlier Rejection, Simultaneous Localization and Mapping (SLAM), Extended Kalman Filter (EKF), Long Baseline (LBL) Navigation, clustering, active exploration**

I. Introduction

Stationary acoustic transponder beacons (also known as Long Baseline, or LBL, beacons) are commonly used as navigational aids in AUV systems (Fig. 1). The distance to a beacon can be measured by sending an acoustic signal and waiting for the beacon's response. In typical experiments, the locations of the beacons are carefully surveyed, allowing the position of the AUV to be easily determined by trilateration.

In this paper, we consider an AUV navigating in a field of beacons whose locations are initally unknown. There are a number of important applications:

- Unsurveyed beacons: In some applications, it is impractical or impossible to survey beacons. For example, the beacons could be deployed by other autonomous vehicles, or dropped by an airplane.
- Beacon movement detection and recalibration: Most navigation systems assume that each beacon remains at its surveyed location. It is important to be able to detect whether a beacon has become unanchored, and ideally, determine the beacon's new position.

We present a system that performs range-only Simultaneous Localization and Mapping (SLAM), i.e., depends only on range measurements to features in the environment. The chief contributions of this paper are an application of graph partitioning to range-measurement outlier rejection and a method for determining when to instantiate new features into a navigation filter.

We demonstrate how these methods can be combined into a complete navigational system based on the Extended Kalman Filter (EKF), and show navigational results from data collected by an Odyssey III AUV during the GOATS'02 experiment off the coast of Italy. In these experiments, four LBL beacons were deployed. The beacons were carefully surveyed, providing a reliable ground-truth.

Range-only measurements often lead to ambiguities that can only be resolved by manuevering the AUV. Given two candidate locations for a beacon, we show how to optimally disambiguate the two possibilities by modifying the AUV trajectory.

Fig. 1. Caribou AUV. Our experiments used an Odyssey-III class vehicle equipped with DVL/LBL/GPS and INS. The primary payload of Caribou was a synthetic aperture sonar (SAS), not pictured. While we did not use data from the SAS, our algorithms were designed to cope with the interference it caused.

II. Previous Work

The problem of navigating with range-only data has not been studied extensively, but several authors have approached the problem. Newman and Leonard approached range-only SLAM by casting the problem as a nonlinear optimization over a search space including not just the beacon locations, but also the AUV's position at each point in time [1]. As the authors acknowledge, the algorithm is prone to divergence due to the search's random initialization. Consequently, their approach did not robustly deal with ambiguities, such as those arising from baseline crossings.

Kantor [2] addresses the problem in a terrestrial setting, briefly discussing the case of unknown beacon locations. However, he assumes that the beacon locations are approximately known, and does not discuss the case when *no* prior is available.

Most systems incorporate some type of outlier rejection strategy. When priors on beacon locations are available, extremely unlikely measurements can be discarded. Other methods include searching for intervals of data that are relatively smooth and continuous (and thus presumably not caused by noise), and using these intervals to help interpret noisier intervals [1].

This work was funded in part by the Department of Commerce/NOAA Sea Grant College Program under grant number NA 16 RG 2255 and ONR awards N00014-02-C-0210 and N00014-03-1-0879.

Our outlier rejection method uses a form of spectral graph partitioning. Shi's Normalized Cuts [3] and Ding's MinMax-Cuts [4] are typical approaches. These approaches attempt to find balanced clusters, but the problem of outlier rejection is more akin to finding *one* cluster amidst noise. In this paper, we make use of Single Cluster Graph Partitioning (SCGP), a consistency-based clustering technique that is well-suited to outlier rejection. We developed SCGP to support a number of other clustering applications; the method is described in detail elsewhere [5].

Our method for initializing feature locations is reminiscent of Hough transforms. Hough transforms have previously been used to classify sonar returns from point and line features by Tardós [6]. Hough-style voting was also used by Wijk [7].

III. Noise in LBL Data

Most navigation systems use a Kalman filter for state estimation. Kalman filters produce optimal estimates when measurement noise is Gaussian and stationary, but their performance can be poor when the noise is more complex.

LBL data is corrupted by a number of non-Gaussian and non-stationary noise processes (see Fig. 2). After removing outliers, the inliers are often far more Gaussian in distribution, allowing better performance from a Kalman filter.

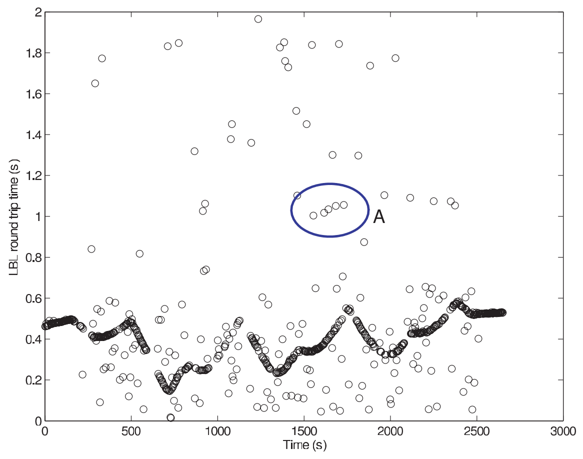

Fig. 2. LBL Range Data. The range data for each beacon is corrupted by large amounts of noise. Some outliers come in bursts (A) and have slowly varying range measurements that mimic valid data.

One significant source of error is the variable speed of sound in water. A measured range will exhibit error that is both a function of the environment (since sound speed varies with temperature and pressure along the acoustic path) and of the true range itself (since small variations in sound speed have a cumulative effect over the total acoustic path length).

Multipath is another significant source of non-Gaussian noise. In typical LBL operation, the acoustical energy from the transmitter travels in a straight line to the AUV. Multipath occurs when the receiver is triggered by an indirect path rather than the direct path. (For example, a pulse could travel from the beacon, reflect off the ocean surface or floor, then travel to the AUV.) Whether or not this happens is primarily a function of environmental conditions. These conditions change slowly since the AUV moves at a low speed. As a result, multipath errors can affect several successive measurements. These indirect range measurements may have very little variance, and their rate of change may correlate very well to the vehicle's estimated motion. Since multipath noise arises from an integer number of reflections from a small number of surfaces, the resulting noise is multimodal.

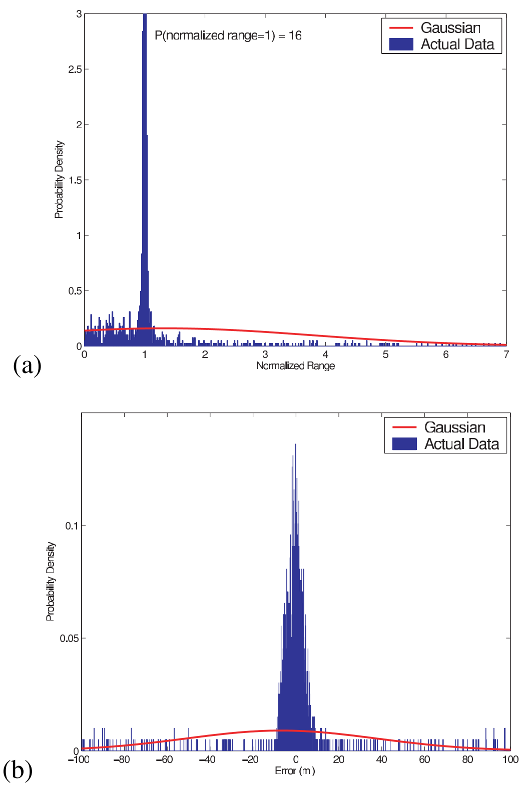

Fig. 3. LBL Range Error. (a) Range measurements normalized by the true range. (b) Absolute error. In both views of the data, a Gaussian noise model fits poorly due to the large number of outliers.

The AUV may also be operating in a noisy environment. If multiple vehicles are present, one might receive an LBL response caused by another vehicle's request. Finally, the AUV's payload can also interfere; a high intensity acoustic device like a Synthetic Aperture Sonar (SAS) can either mask or falsely trigger an LBL response. Closer examination of Fig. 2 shows that there is little noise until about 300 seconds into the mission, when the SAS was activated.

We characterized the noise of raw range measurements by collecting range data for an entire mission (Fig. 3). We modeled the error in two ways: as multiplicative noise and as additive noise. The multiplicative noise model is applicable to speed of sound errors, while the additive model better describes multipath and other noise sources. While the dominant mode of the noise distributions can be readily approximated by a Gaussian, there is considerable probability mass in the tails which cannot.

IV. Spectral Graph Partitioning for Outlier Rejection

Our approach for identifying outliers in range data is to represent a set of measurements as a graph, and apply graph partitioning algorithms to identify sets of consistent measurements. We associate each measurement with the vehicle's dead-reckoned position at the time of the measurement. If we consider only the planar motion of the AUV, each measurement can be drawn as a circle in the plane, centered at the vehicle's position with a radius equal to the measured range. The beacon is constrained to lie on a circle with this radius (see Fig. 4). In the general 3D case, the beacon lies on a sphere, and our methods extend naturally to this case, though we have implemented only the 2D case.

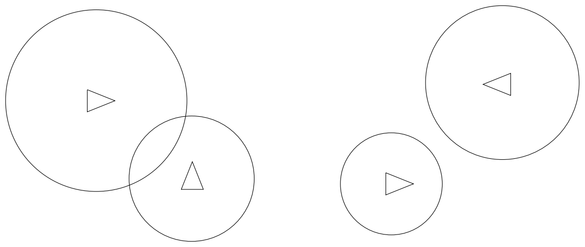

Fig. 4. Measurement consistency. Range measurements can be described as circles in the plane centered at the vehicle's position. For any given measurement, the beacon must be located on the circle. Measurements are consistent if the circles intersect (left). (The AUV's heading is not used.)

Consider a set of range measurements $M_i : 1 \leq i \leq N$. We say that two measurements are *consistent* if both can be explained by a beacon at some particular location. In other words, if the circles describing the measurements intersect (within some tolerance), they are consistent.

We can form an undirected graph from pairwise consistencies. Each measurement becomes a vertex in the graph; consistent measurements are connected by an edge (Fig. 5).

The problem of outlier rejection can then be posed as a graph partitioning problem: divide the graph into two sets of vertices by cutting edges such that inliers are in one partition, and outliers are in the other. Inliers will tend to be highly consistent with each other, whereas outliers will have only random consistency with other measurements.

A graph resulting from eight hypothetical measurements, including three outliers, is shown in Fig. 5. Note that only the connectivity of the graph is relevant; the position of the nodes relative to each other has no meaning. In the example, nodes 1-5 are well-connected to each other. This means that they are consistent with each other, and are therefore less likely to be the result of noise. Nodes 6-8, while connected to the other nodes, are less likely to be inliers. A good partition of this graph would be cut A; it separates the highly connected measurements from those that are poorly connected. Cut B is poor since it would divide a large number of consistent measurements. Cut C is poor since it would leave several unlikely measurements (6 and 8) classified as inliers.

We construct an $N \times N$ adjacency matrix A by setting $A_{ij} = 1$ iff M_i and M_j are consistent.

$$A_{ij} = \begin{cases} 1 & i \neq j,\ M_i \text{ and } M_j \text{ are consistent} \\ k & i = j, \text{ with arbitrary constant } k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

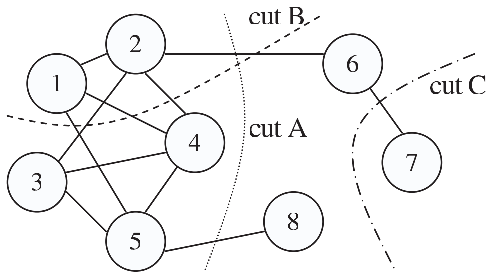

Fig. 5. Simplified Partitioning Problem. Each measurement is a node in the graph, and edges connect consistent measurements. The outlier rejection problem is to find a graph partition that separates well-connected vertices (inliers) from poorly-connected vertices (outliers).

The diagonal elements of A can be set to any constant k without changing the solution.

Our consistency metric for two measurements leads to a symmetric adjacency matrix A. For our notional problem, it is:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2}$$

Let u be an $N \times 1$ indicator vector with each element either 0 or 1; if $u_i = 1$ then measurement M_i is an inlier. In single-cluster graph partitioning [5], the quality of the cut is given by the scalar-valued function:

$$r(u) = \frac{u^T A u}{u^T u}. \tag{3}$$

The product $u^T A u$ is twice the number of edges within the inlier cluster. The denominator, $u^T u$, is simply the total number of vertices classified as inliers. Thus the metric $r(u)$ computes the average connectivity of the inliers; it will be large for sets of highly consistent measurements, and small for sets of inconsistent measurements. For our notional problem, we can compute the metric $r(u)$ for each cut in Fig. 5.

Cut A	**Cut B**	**Cut C**
1.6	0.5	1.4

Cut A, the intuitively correct cut, has the highest score. Cut B, which breaks a great deal of connectivity, has a very low score. Cut C, while not optimal, scores relatively high because it preserves almost all of the connectivity. Cut C does more poorly than A, however, because it includes nodes 6 and 8 as inliers, which have lower connectivity than nodes 1-5. This reduces the average connectivity.

Average inlier connectivity is a good metric for outlier rejection. Consider an incremental argument: given a set of inliers u, measurement M_i should be added if it is at least as well connected to the inliers as the inliers are connected to themselves. If this is true, adding M_i to u will increase $r(u)$.

The challenge is to find an indicator vector u that maximizes $r(u)$. This is a hard problem for discrete-valued u. However, if we allow the indicator value to be continuous-valued, we can readily compute the solution.

Consider the gradient of $r(u)$, remembering that A is symmetric:

$$\nabla r(u) = \frac{Auu^T u - u^T Auu}{(u^T u)^2} = \frac{Au - ru}{u^T u} \quad (4)$$

Setting the gradient to zero yields the extrema of $r(u)$:

$$Au = ru. \quad (5)$$

This is an eigenvector problem: the product Au must be in the same direction as u, scaled by a factor r. We know all of the solutions to equation 5; they are the eigenvalues/vectors of matrix A.

Our goal is to maximize r, so we pick r to be the largest eigenvalue and u to be the corresponding eigenvector. For the problem in Fig. 5, the eigenvector u is plotted in Fig. 6. The indicator values for the inlier measurements (1-5) are significantly larger than those for the outliers (6-8).

The metric $r(u)$ is also the Rayleigh quotient of matrix A, whose extrema values have previously been known [8].

Smaller eigenvalues of A correspond to alternative cuts with lower scores. Since A is symmetric, the cuts are orthogonal: they provide radically different interpretations of the data. If the first and second eigenvalue are similar in magnitude, it means that two orthogonal cuts are of similar quality. If the data is composed of a single set of interconnected inliers plus random outliers, this should not occur since two orthogonal cuts cannot similarly separate the inliers from outliers. Using this fact, we can perform a sanity check on our data; if the first and second eigenvalue are similar in magnitude, the data must be considered suspect.

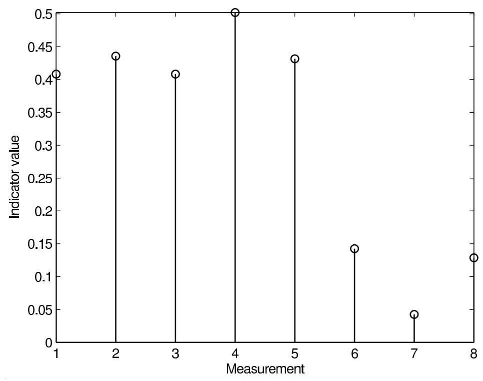

Fig. 6. First eigenvector of A, $u \in \Re^8$. The continuous indicator vector is the first eigenvector of A. Large values indicate that the corresponding measurement is an inlier; small values indicate outliers.

At this point, we have the optimal u that maximizes $r(u)$. The vector u, however, is continuous-valued, while the inlier/outlier classification problem is discrete-valued.

Consider the discrete-valued indicator vector $v(t)$, which is the vector u thresholded by the scalar t:

$$v_i(t) = \begin{cases} 1 & \text{if } u_i \geq t \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

We wish to find the optimal threshold, t_{opt}:

$$t_{opt} = \begin{array}{c} max \\ t \in u \end{array} \frac{v(t)^T u}{v(t)^T v(t)}. \quad (7)$$

Maximizing the dot product of $v(t)$ and u yields the vector $v(t)$ that is closest to the direction of u. This maximization problem can be solved in a $O(N \log N)$ time by evaluating $v(t)$ with t equal to each element of u. If the elements of u are sorted, each trial adds only one element to the inlier set, allowing $v(t)$ to be updated in $O(1)$ time; the cost of sorting the elements dominates.

The algorithm presented here does not guarantee that the final indicator vector $v(t_{opt})$ is globally optimal. This algorithm computes the optimal continuous indicator vector, and then computes the optimal discretization of *that* vector. It is possible that some other discrete vector, which does not correspond to any thresholding t of u, is actually optimal. This is more likely to occur when the inlier elements of u have widely varying values; this corresponds to the direction of u having varying magnitudes in each dimension. In this case, $v(t)$ will not be able to approximate u very well since each component of $v(t)$ must be either zero or one. In practice, however, not only do the inlier elements of u have similar magnitude (they typically intersect roughly the same number of other measurements), but the discrete vector $v(t)$ performs well.

Outlier rejection performance can be improved by incorporating *a priori* knowledge of the noise characteristics into the threshold. The continuous vector u provides a score for each measurement; if the number of inliers is known *a priori*, we can simply select the measurements with the largest scores. If the number of inliers was smaller than expected, then this procedure will permit outliers in the output. A more conservative approach is to use the discretized set determined by t_{opt} if it is smaller.

A typical result on 25 real range measurements is shown in Fig. 7. Several extreme outliers are outside of the plot area. In this case, over 25% of the measurements are outliers, and every measurement is correctly classified. Some of the outliers *are* consistent with inliers; however their connectivity is so low in the graph that they are assigned small indicator values. Also note that the algorithm works for the case of unconnected graphs.

As mentioned above, measurements are also considered consistent if they intersect within some tolerance. The tolerance is an important consideration for good outlier rejection. Given no noise and accurate vehicle positions, inliers always intersect. Now consider two measurements made in quick succession, with noise. If noise makes the first measurement range a bit too large and the second measurement a bit too small, it is possible that the circles will not intersect. The algorithm works best if inliers have as much connectivity to other inliers as possible; adding a small intersection tolerance helps improve their connectivity. Outliers are typically so different from inliers that this tolerance rarely changes the graph's connectivity.

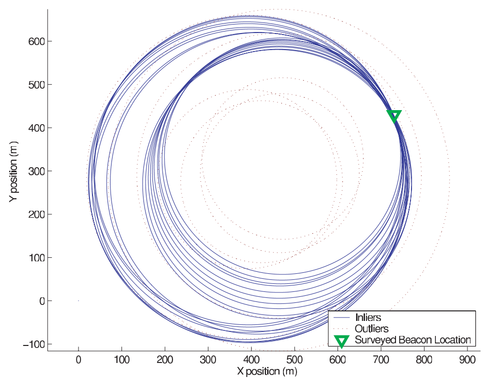

Fig. 7. Outlier Rejection Result. Twenty-five measurements are plotted, showing both inliers and outliers. Measurements that were consistent with many other measurements were classified as inliers.

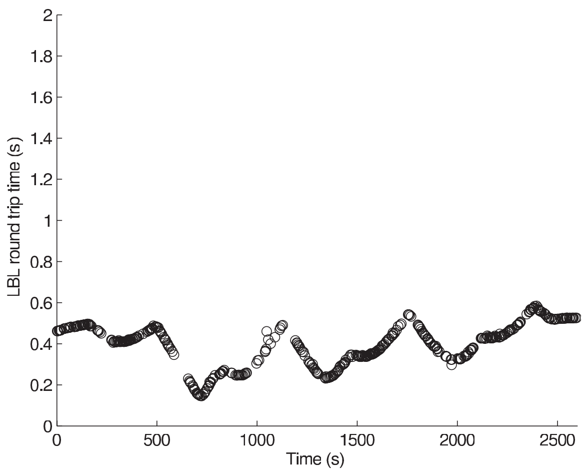

Fig. 8. Data from Fig. 2 After Outlier Rejection. The filtered data is dramatically cleaner than the input data, and no beacon location information was required. The raw measurements were filtered in blocks of ten measurements (about 30 seconds) each.

An important benefit of our outlier rejection algorithm is its ability to incorporate information about the vehicle's motion. For example, if the measured range to a beacon increased slowly over time, with very little apparent noise, most algorithms would classify those measurements as inliers. However, those measurements could be spurious; if the vehicle was stationary, for example, then the range *should* be constant. Our algorithm can reject this sort of noise, since it incorporates knowledge of the vehicle's trajectory.

A. Computation in Blocks

The heart of the outlier rejection algorithm is the adjacency matrix A. When testing two measurements for consistency, we determine whether two circles (centered at the vehicle's estimated locations) intersect. However, if the measurements in A span a large time interval, accumulation of navigational errors can cause measurements to appear consistent when they are not and vice versa.

The maximum acceptable time interval is dependent on the quality of navigation information available. With Doppler velocity logs (DVL) and a fluxgate compass, time windows of up to 10 minutes are practical. We have gotten good results using both DVL/compass and compass alone (using thrust control to estimate forward speed).

Of course, if high-precision inertial devices are used, or if the vehicle can use GPS, accumulation of navigation error is not a significant issue.

Processing the data from Fig. 2 in blocks of ten measurements produces the improved data in Fig. 8.

B. Comparison to other spectral partitioning methods

A number of similarly motivated partitioning methods exist ([3], [4]). These algorithms also compute a graph partition through essentially the same mechanism described here. However, they are fundamentally different in their formulation: they are designed to cluster data that contains two different sets of consistent data. This is a different problem than finding a single set of consistent data amidst noisy outliers. This fact accounts for the different objective function $r(u)$ used in this paper.

C. Computational Optimization

Ultimately, our method must compute the dominant eigenvector of a potentially large matrix. From an implementation standpoint, several optimizations can be employed to reduce the computational burden.

As already discussed, outlier rejection should be done on modestly-sized sets of measurements. Since the size of matrix A is determined by the number of measurements, controlling the block size has a direct impact on computational requirements.

A fortuitous advantage of the average inlier connectivity metric is that the solution is the *largest* eigenvalue. The largest eigenvalue can be found in a fast iterative manner via the Power Method [8]. For any vector x not perpendicular to the largest eigenvector u, the direction of $A^n x$ approaches u as $n \to \infty$. In practice, x converges to a sufficiently good approximation of u in a few iterations.

Other spectral partitioning algorithms have eigenvalue solutions as well, but their solution is found in different eigenvalues that are more costly to compute.

Considering the low rate at which an AUV receives LBL data (each beacon is queried roughly every five seconds), an AUV is typically starved for data and has CPU time to spare. Even though spectral outlier rejection is more expensive than other methods like gating, it makes sense to trade CPU time for higher-quality outlier rejection and thus higher quality navigation.

D. Extension to Multiple Vehicles

It is possible to extend the spectral outlier rejection to multiple vehicles. In fact, the algorithm makes no assumptions about how many vehicles are operating; it assumes only that approximate vehicle positions are available for each range

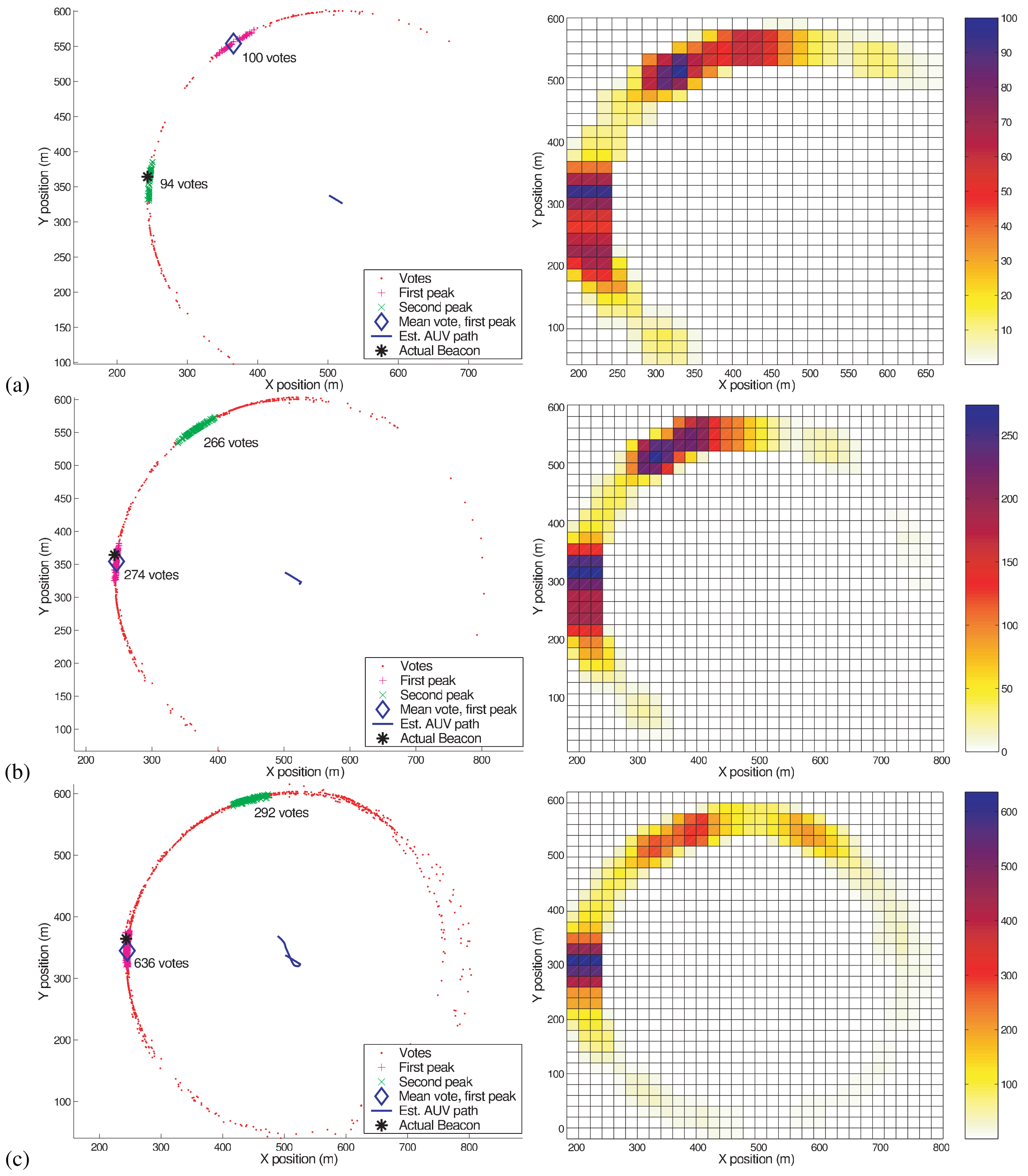

Fig. 9. Beacon Localization Using Voting. Using the inliers computed by our spectral outlier rejection algorithm, we compute possible beacon locations by finding the pairwise intersections of a set of measurements. Each of these intersections is a *vote* in a two-dimensional accumulator. The figure shows the votes in Euclidean space (left) and the vote density (right) at three different times during the vehicle's mission. The two most prominent peaks are extracted from the accumulator and are used to label the vote plot. In (a) and (b), the vehicle trajectory was essentially straight, leading to an even distribution of votes. In (c), the vehicle trajectory was no longer straight, leading to a dominant peak in the accumulator. This allowed the beacon to be initialized in the navigation filter.

measurement. A consistency matrix A can be computed regardless of how many vehicles were involved in data acquisition. The algorithm is entirely unchanged; it identifies a single set of inliers.

In order for multiple vehicles to make use of share data, they must have estimates of their positions in a common coordinate frame. If they know their starting positions, then they can share data immediately. Otherwise, the vehicles must independently localize at least two beacons in order to define a new common coordinate system. Once this is achieved, they can collaborate on localizing beacons.

It is not necessary to do outlier rejection on vehicles before trying to fuse the data. Groups of vehicles in very noisy environments might not be able to individually discern inliers from outliers, but taken collectively, their data might be usable. We hope to experimentally verify the performance of the algorithm on multiple vehicles.

An interesting scenario with multiple vehicles involves a command/control vehicle that autonomously deploys a beacon field. All of the vehicles use range measurements to establish a common coordinate system without the use of GPS or *a priori* beacon locations. Sensing platforms can identify targets of interest and relay the information to other vehicles. Since no surveying of the operational area is required, the vehicles could conceivably be deployed without any human support.

V. Estimating LBL Beacon Locations

Once beacon range measurements have been filtered, leaving only reasonably reliable data, the actual locations of the beacons must be determined. Three perfect measurements (made from non-collinear positions) uniquely determine a beacon location. However, the measurements are contaminated by noise, and three range measurements rarely intersect exactly.

Our approach, once again, is to consider pairs of measurements. A pair of measurements is not sufficient to constrain a beacon's location to a point; two circles have *two* point intersections and thus two possible solutions. However, provided the vehicle is not traveling in a straight line, some solutions will occur more often than others.

We use a voting scheme implemented with a two-dimensional accumulator similar to that used in a Hough transform [9]. The physical world is discretized into a two-dimensional grid, with each grid cell corresponding to a rectangular area in the world. Each consistent measurement pair "votes" for its two solutions. Ideally, solutions that are near each other should end up in the same cell, even in the presence of noise. This can be accomplished by choosing a grid size that matches the total uncertainty, range plus dead-reckoning uncertainty. Once all votes have been added to the accumulator, we can search the accumulator for the cell with the greatest number of votes.

If a beacon happens to lie near a cell boundary, it is possible that the votes for solutions around it will fall into *different* cells. In the worst case, the votes for similar solutions could be evenly split into four different cells, making it likely that none of the cells would be noticed when the accumulator is searched.

Fortunately, there is a simple solution: when voting, vote for a cell *and* all of its neighbors. At the expense of smearing the peak, this approach eliminates the risk of a peak being hidden due to the discretization of grid boundaries.

Our implementation finds the two largest peaks in the accumulator. After finding the first peak, all votes for that cell are removed from the accumulator so that the second peak can be found without influence from the first.

The number of votes for each peak serves as a confidence metric. If the ratio of votes between the first and second peak exceeds a threshold, then the first peak is declared to be the (approximate) beacon location. If the vote ratio is less than the threshold, then both peaks are still plausible solutions; no decision will be made until more range measurements are available. This process is illustrated in Fig. 9. Empirically, we found a vote ratio of about two to be sufficiently high to virtually guarantee that the correct peak is selected. Higher ratios increase confidence, of course, but they can unnecessarily delay making a decision.

Finding a solution as early as possible is highly advantageous for two reasons. First, it becomes difficult to reliably estimate measurement intersections for long time windows due to accumulating dead-reckoning error. The ability to make decisions based on smaller sets of data mitigates this problem. Second, until the AUV localizes a few beacons, it must rely on dead-reckoning for navigation. During this time, the global translational/rotational misalignment between the vehicle's coordinate frame and the global frame will increase.

In very large environments the memory requirements of a simple accumulator can be an impediment, especially if beacon positions in 3D must be determined. Examination of Fig. 9 shows that the accumulator is sparsely populated. Consequently, an H-tree (or octree in 3D) would greatly reduce the memory requirements.

VI. SLAM without Prior Beacon Locations

Once beacons have been *approximately* located, a conventional Extended Kalman Filter (EKF) can be used to jointly refine both vehicle position and beacon locations as additional measurements arrive. We have implemented our entire system on data gathered during the GOATS'02 experiment.

As opposed to systems in which beacon locations are known *a priori*, the beacon locations become part of the filter state and the covariance matrix is appropriately enlarged. A simple state vector incorporating one beacon location is shown below, where the AUV is located at (r_x, r_y), the beacon at (b_x, b_y) and the AUV's orientation is r_t.

$$x_n = \begin{pmatrix} r_x \\ r_y \\ r_t \\ b_x \\ b_y \end{pmatrix} \tag{8}$$

When we receive a new range measurement z_n to the beacon, we perform a Kalman update step. The first stage is to estimate the range to the beacon from our current state.

$$\hat{z}_n = \left[(r_x - b_x)^2 + (r_y - b_y)^2\right]^{\frac{1}{2}} \tag{9}$$

The EKF also requires the Jacobian $\mathcal{J}$ of z_n (the partial derivatives of z_n with respect to each state variable). After some algebra, we see that:

$$H_n = \mathcal{J}(z_n) = \begin{pmatrix} (r_x - b_x)/\hat{z}_n \\ (r_y - b_y)/\hat{z}_n \\ 0 \\ -(r_x - b_x)/\hat{z}_n \\ -(r_y - b_y)/\hat{z}_n \end{pmatrix}. \tag{10}$$

Given a model of the measurement noise variance R, the Kalman gain K_n can be computed:

$$K_n = P_n H_n^T (H_n P_n H_n^T + R)^{-1}. \tag{11}$$

The usual EKF time update steps are used unchanged. With x the state vector and P the covariance:

$$x_{n+1} = x_n + K_n(z_n - \hat{z}_n) \tag{12}$$

$$P_{n+1} = (I - K_n H_n)P_n \tag{13}$$

Unlike many navigation filters, we require the ability to dynamically increase the amount of state. When a new beacon is localized, the state vector x and covariance matrix P must be extended to incorporate information about the beacon. The initial estimate for the beacon's location is the output of the ROBL beacon localization algorithm (see Section V).

It might be tempting, if the *number* of beacons is known in advance, to "preallocate" room in the state and covariance matrix for them. When a beacon is first localized, the state could be initialized with a synthetic observation. This is a recipe for disaster. A Kalman update step must not be used to initialize a new feature since the Kalman update equations perform a linearization around the current state. In the case of a preallocated state, the "current state" is meaningless, and a linearization around it is unlikely to produce good results.

Consider the moment when a new beacon is first initialized. The error in the beacon estimate is strongly correlated with the error in the AUV's state since the AUV's state was used to estimate the beacon's position. Thus, the covariance for the beacon is initialized to be equal to that of the AUV. If an additional uncertainty N from the beacon localization phase is available (for example, the covariance of the points in the grid cell), it can be added to the covariance matrix as well. The result of initializing a new beacon b_2 is

$$x_n = \begin{pmatrix} r_x \\ r_y \\ r_t \\ b_{1x} \\ b_{1y} \end{pmatrix} \longrightarrow x_{n+1} = \begin{pmatrix} r_x \\ r_y \\ r_t \\ b_{1x} \\ b_{1y} \\ b_{2x} \\ b_{2y} \end{pmatrix} \tag{14}$$

$$P_n = \begin{pmatrix} C(r,r) & C(r,b_1) \\ C(b_1,r) & C(b_1,b_1) \end{pmatrix} \longrightarrow$$

$$P_{n+1} = \begin{pmatrix} C(r,r) & C(r,b_1) & C(r,r) \\ C(b_1,r) & C(b_1,b_1) & C(b_1,r) \\ C(r,r) & C(r,b_1) & C(r,r)+N \end{pmatrix} \tag{15}$$

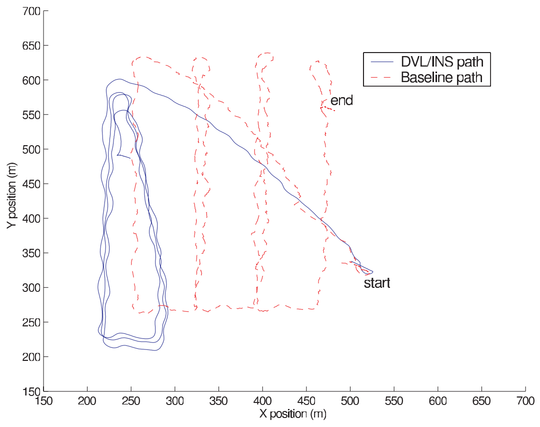

Fig. 10. Dead-Reckoned Path. The dead-reckoned path that we used to bootstrap our SLAM filter was relatively poor. It was constructed from DVL and an uncalibrated fluxgate compass. The baseline path was computed by a causal EKF filter using known beacon locations.

We have applied our algorithm to a dataset collected on an Odyssey III class vehicle during the GOATS'02 experiment. We used DVL/INS data for our dead-reckoned trajectory. The compass was poorly calibrated, resulting in a poor dead-reckoned path (see Fig. 10).

The path begins with a very long, almost straight segment. For beacons lying off the line of travel, this is a particularly difficult situation, since two solutions are always possible (one on either side of the vehicle). Due to this difficulty, it takes four minutes to localize the first beacon. All four beacons are localized eleven minutes into the mission.

Once several beacons are localized, the entire trajectory of the AUV, including the initial portion during which only dead-reckoning was possible, can be recomputed. The coordinate frame of the AUV will differ from the global coordinate frame by a rigid-body transformation, i.e., a simple translation and rotation. If the vehicle's initial position in the global frame is known, the magnitude of this translation and rotation is determined by the amount of dead-reckoning error accumulated prior to the localization of the beacons.

The results of the experiment are shown in Fig. 11. The global translation/rotation offset (which amounted to 17 meters and a few degrees of heading error) have been manually removed to allow the estimated path to be easily compared to the baseline path. The baseline path was generated by a causal EKF filter that used the surveyed beacon locations.

After global alignment, the beacon localization error was extremely low. One of the estimated beacon locations was used to determine the global translation error. The errors for the remaining beacons were:

Beacon 1	**Beacon 2**	**Beacon 3**
2.89 m	2.52 m	1.85 m

Since the global translational/rotational offset was estimated using ground-truth beacon positions, the beacon localization error is somewhat underestimated. However, due to the close agreement of the two vehicle tracks, the degree of over-fitting is arguably small.

Animations of both the beacon localization process

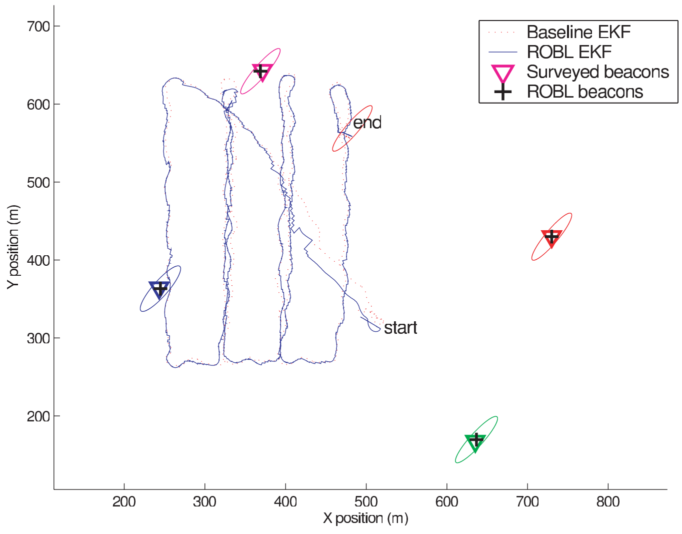

Fig. 11. SLAM filter results. Our SLAM filter's performance (labeled ROBL EKF) favorably compares to the performance of a baseline filter that uses prior beacon location information. A modest global translation/rotation error was manually corrected to allow a better comparison between the two computed trajectories. The error at the beginning of the path corresponds to dead-reckoning error accumulated before any beacons were localized.

and operation of the ROBL EKF filter are available at http://rvsn.csail.mit.edu/robl.

VII. Optimal Exploration

The number of measurements needed to localize a beacon is strongly dependent on the path of the AUV. Moving in a straight line does not provide sufficient information to localize a beacon: two possible solutions, one on each side of the vehicle, result. The vehicle must turn and travel in another direction to disambiguate the two solutions. In this section, we derive the AUV paths that, at any instant in time, optimally disambiguate the two possible solutions.

Suppose that an AUV makes a few observations of a new beacon while travelling in a roughly straight line, and that it has two possible solutions for the beacon's location, points A and B.

If the AUV were to continue travelling in the same direction (i.e., without turning), future range observations would be equally consistent with both point A and B, since both A and B will continue to be equally far away from the AUV. Instead, the AUV should move so that point A and point B are different distances from the AUV. To maximize the amount of knowledge about which point is more likely, the AUV should move so as to maximize the *absolute difference* in range to points A and B.

Given points A, B, and the AUV's position R, the magnitude of the difference in range between R and A and R and B is given by:

$$r = \left| \left[(A_x - R_x)^2 + (A_y - R_y)^2\right]^{\frac{1}{2}} - \left[(B_x - R_x)^2 + (B_y - R_y)^2\right]^{\frac{1}{2}} \right| \quad (16)$$

The gradient $\nabla(r)$ is the direction of greatest increase.

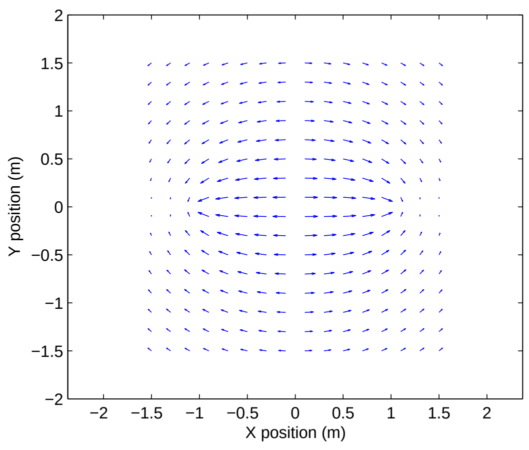

Fig. 12. Exploration Gradient. If a beacon's location is believed to be either (-1,0) or (1,0), then the best disambiguating motion is a function of the AUV's position. The vehicle maximizes the difference between the range measurements by traveling along the arrows. The length of the arrows indicates how rapidly the difference in range changes.

$$\begin{aligned} \nabla(r) &= \alpha \left(\frac{B_x - R_x}{|B-R|} - \frac{A_x - R_x}{|A-R|} \right) \hat{x} + \\ &\quad \alpha \left(\frac{B_y - R_y}{|B-R|} - \frac{A_y - R_y}{|A-R|} \right) \hat{y} \\ \alpha &= 2\, sign(|A-R| - |B-R|) \end{aligned} \quad (17)$$

An example gradient field with $A = (-1, 0)$ and $B = (1, 0)$ is plotted in Fig. 12. The arrows indicate the direction in which the vehicle should travel to maximize r. The length of the arrows indicate how rapidly r changes as the AUV moves.

Choosing an exploration strategy becomes particularly interesting when several beacons are simultaneously localized. Combining the gradient fields for each beacon would allow better path planning.

Better performance can be expected by employing an active exploration algorithm. Fewer measurements will be needed to find a solution, which means that lower-quality dead reckoning can be used. In addition, locating beacons earlier would reduce the magnitude of the global translation/rotation error resulting from navigating without fixed reference points.

VIII. Conclusion

We have described a system capable of performing localization without relying on surveyed beacon locations. Filtering noise in LBL data can be a major challenge, prompting our development of a outlier rejection algorithm based on spectral graph partitioning.

We showed how to compute likely beacon locations. An important feature of our method is its ability to discern whether more than one location is likely.

Using our outlier rejection and feature initialization algorithms, we implemented a SLAM filter and demonstrated it on data collected during the GOATS'02 experiment. The estimated vehicle path is close to the baseline path, which was computed using known beacon locations. In addition, we localized all four beacons to within a few meters of their surveyed positions.

The ability to localize a beacon is tightly coupled to the path traveled by the AUV. We showed how the AUV's path should be chosen to optimally resolve ambiguous data.

In the future, we plan to deploy our exploration ideas on a real vehicle. We are also interested in replacing the Extended Kalman Filter (EKF) with an Unscented Kalman filter [10] or particle filter which should be less vulnerable to the effects of linearization.

Acknowledgements

We would like to thank the MIT GOATS 2002 team, particularly Paul Newman, for providing us with the data used in this paper.

References

[1] P. M. Newman and J. Leonard, "Pure range-only sub-sea SLAM," in *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '02)*, 2003.

[2] G. A. Kantor and S. Singh, "Preliminary results in range-only localization and mapping," in *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '02)*, May 2002.

[3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 888–905, August 2000.

[4] C. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A minmaxcut spectral method for data clustering and graph partitioning," Lawrence Berkeley National Laboratory, Tech. Rep. 54111, 2003.

[5] E. Olson, M. Walter, J. Leonard, and S. Teller, "Single cluster graph partitioning for robotics applications," in *Proceedings of Robotics Science and Systems*, 2005, pp. 265–272.

[6] J. Tardós, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *Int. J. Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[7] O. Wijk, "Triangulation based fusion of sonar data with application in mobile robot mapping and localization," Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 2001.

[8] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993.

[9] P. Hough, "Machine analysis of bubble chamber pictures," *International Conference on High Energy Accelerators and Instrumentation*, 1953.

[10] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

Author Biographies

Edwin Olson is a PhD candidate at MIT's Computer Science and Intelligence Laboratory (CSAIL). His research focuses on scalable SLAM algorithms and SLAM-aware exploration. He holds Masters (2001) and B.S. (2000) degrees in Computer Science and Electrical Engineering from MIT.

John J. Leonard is Associate Professor of Mechanical and Ocean Engineering in the MIT Department of Mechanical Engineering. He is also a member of the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). His research addresses the problems of navigation, mapping, and persistent autonomy for autonomous mobile robots operating in unstructured environments. He holds the degrees of B.S.E. in Electrical Engineering and Science from the University of Pennsylvania (1987) and D.Phil. in Engineering Science from the University of Oxford (1994).

Seth Teller's research focuses on the development of efficient data structures and algorithms for large-scale outdoor and indoor mapping, wide-area localization, robust deployment and autoconfiguration of sensor networks, and natural robot command interfaces. He is also a principal developer of a new two-term robotics subject for undergraduates, "Robotics: Science and Systems," in the Electrical Engineering and Computer Science Department at MIT, where he is an Associate Professor of Computer Science and Engineering and a member of the Computer Science and Artificial Intelligence Laboratory. He holds the degrees of B.A. (1985) in Physics from Wesleyan University, and M.Sc. (1990) and Ph.D. (1992) in Computer Science from the University of California, Berkeley.