Robust Range-Only Beacon Localization

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

***Abstract*— Most Autonomous Underwater Vehicle (AUV) systems rely on prior knowledge of beacon locations for localization. We present a system capable of navigating without prior beacon locations. Noise and outliers are major issues; we present a powerful outlier rejection method that imposes geometric constraints on measurements. We have successfully applied our algorithm to real-world data and have demonstrated navigation performance comparable to that of systems that assume known beacon locations.**

I. Introduction

Stationary acoustic transponder beacons (also known as Long Baseline, or LBL, beacons) are commonly used as navigational aids in AUV systems. AUVs can estimate the range to a beacon by sending a ping and measuring the length of time until a response is received. In most 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 beacon field when beacon locations are unknown. There are a number of important applications:

1) Unsurveyed beacons: There are a number of scenarios in which carefully surveying beacon locations is impractical, including autonomous or aerial deployment of beacon fields.
2) Detection of beacon movement: 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.

One major difficulty arises from the partial observability of beacon locations given range measurements. A single measurement does not contain enough information to determine the location of a beacon. Systems typically depend on estimates of both beacon and robot position, which allows an update to be performed on a single measurement by linearizing around the prior.

Without a prior, however, a linearization is not possible and single measurement cannot be used; many measurements must be fused into a single estimate. We describe a reliable method for estimating a beacon location given a number of range-only measurements.

Another major challenge to our system is the noisiness of LBL data. Use of a prior allows improbable measurements to be classified as outliers. However, when a prior is not known, outlier rejection is much more difficult. We present an outlier rejection method based on spectral graph partitioning that does not require a prior.

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.

An algorithm for finding likely beacon locations given a number of range-only measurements is also presented. We have named the combination of spectral outlier rejection and beacon localization the Range-Only Beacon Localization (ROBL) algorithm.

We demonstrate how these methods can be combined into a complete navigational system 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, allowing a direct comparison of the ROBL-based filter with a baseline filter that assumes beacon locations.

The path of the AUV has a profound impact on its ability to localize beacons. Range-only data often leads to multiple plausible locations for a beacon. We present an exploration strategy which optimally disambiguates multiple solutions.

We will consider the problem in two dimensions rather than three. The 2D case is easier to visualize and is more easily implemented. When AUVs operate in shallow depths, the 2D case closely resembles the 3D case. All of the algorithms we discuss have straightforward extensions to 3D.

II. Previous Work

The problem of navigating with range-only data has not been studied extensively, but a small number of important papers exist. One system uses only beacon range measurements, casting the problem as an enormous nonlinear optimization over a search space including not just the beacon locations, but also the robot's position at each point in time [1]. The system suffers from significant convergence problems and has difficulty handling baseline crossings.

A second paper [2] addresses the problem in a terrestrial system, briefly discussing the case of unknown beacon locations. However, the authors assume that the beacon locations are approximately known, and do not discuss the case when *no* prior is available.

Our outlier rejection method uses a form of spectral graph partitioning, which has been used in a number classification systems ([3], [4]). Much of the mathematical foundation for spectral analysis of adjacency matrices was established in [5], [6], [7].

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].

Hough transforms have previously been used to classify sonar returns from point and line features ([8]). Another recent work employing Hough-style voting is [9].

III. Noise in LBL Data

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

LBL data is corrupted by a number of non-Gaussian and non-stationary noise processes (see Figure 2). Removing outliers can greatly improve the statistical properties of the noise and improve the performance of the navigation filter.

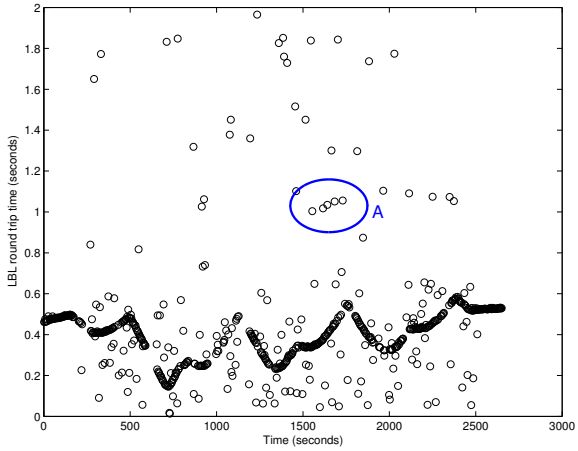

Fig. 2. LBL Range Data. The range data for each beacon is corrupted by large amounts of noise. Some outliers come in bursts and have slowly varying range measurements that mimic valid data (A).

One source of error is the fluctuating speed of sound in water. The speed of sound varies with environmental conditions, making the error a function of the environment. The noise is also range dependent, since the total time of flight is multiplied by the speed of sound.

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 the environmental conditions. These conditions change slowly since the AUV moves at a low speed. As a result, an indirect path is often measured consistently over several successive measurements (Figure 2A). These indirect range measurements may have very little variance, and their rate of change may correlate very well to the vehicle's estimated motion. Multipath noise is non-uniformly distributed, since it corresponds to an integer number of reflections from discrete surfaces; thus different indirect paths generally result in a multimodal distribution.

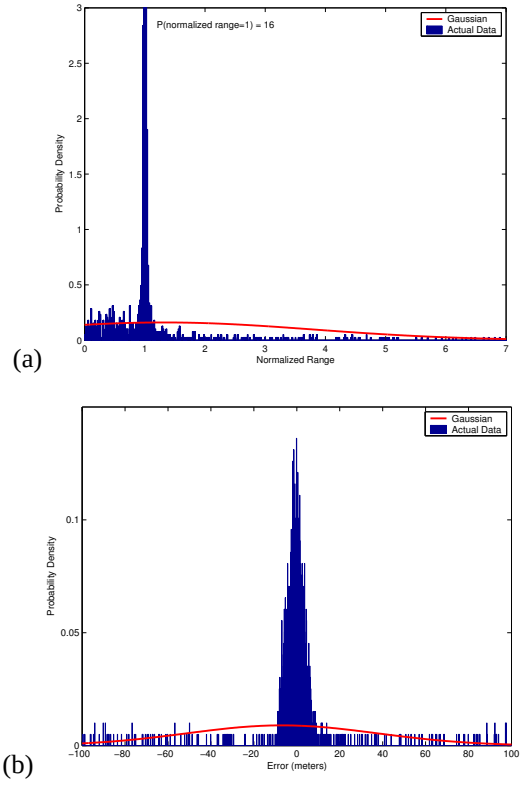

Fig. 3. LBL Range Error. (a) Range measurements normalized by the true range. (b) Absolute error. In both views of the data, a Gaussian noise model fits poorly due to the large number of outliers.

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 an SAS can either mask or falsely trigger an LBL response. Closer examination of Figure 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 (Figure 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. In both cases, the Gaussian fit to the data bears little resemblance to the actual data.

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. 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.

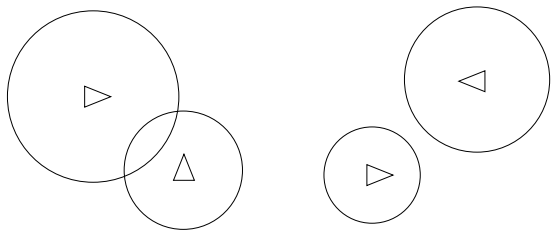

Fig. 4. Measurement consistency. Range measurements can be described as circles in the plane centered at the vehicle's position. For any given measurement, the beacon must be located on the circle. Measurements are consistent if the circles intersect (left). (The AUV's heading is not used.)

Consider a set of range measurements, $M_i : 1 \leq i \leq N$. Two measurements are *consistent* if they can both 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 (Figure 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 Figure 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 partitioning 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 divides a large number of consistent measurements. Cut C is poor since it leaves several unlikely measurements (6 and 8) classified as inliers.

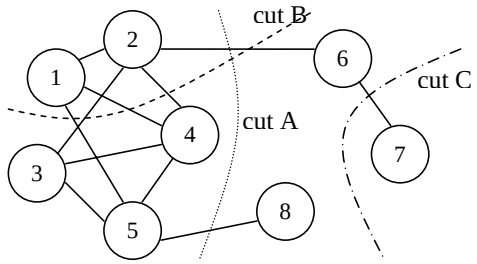

Fig. 5. Simplified Partitioning Problem. Each measurement is a node in the graph, and edges connect consistent measurements. The outlier rejection problem is to find a graph partitioning that separates well-connected vertices (inliers) from poorly-connected vertices (outliers).

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

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

Since the graph in Figure 5 undirected, the adjacency matrix A is symmetric. For our toy 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. We can measure the quality of a cut with $r(u)$, a scalar-valued function of u:

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

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. For our toy problem, we can compute the metric $r(u)$ for each of the cuts in Figure 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 which maximizes $r(u)$. However, 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:

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

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

$$Au = ru. \tag{5}$$

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 eigenvalue/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 toy problem in Figure 5, the eigenvector u is plotted in Figure 6. The indicator values for the inlier measurements (1-5) are significantly larger from the outliers (6-8).

The metric $r(u)$ is also the Rayleigh quotient of matrix A, whose extrema values have previously been known [10].

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.

At this point, we have the optimal u which 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:

$$v_i(t) = \begin{cases} 1 & \text{iff } u_i > t \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

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

$$t_{opt} = \underset{t \in u}{max} \frac{v(t)^T u}{v(t)^T v(t)}. \tag{7}$$

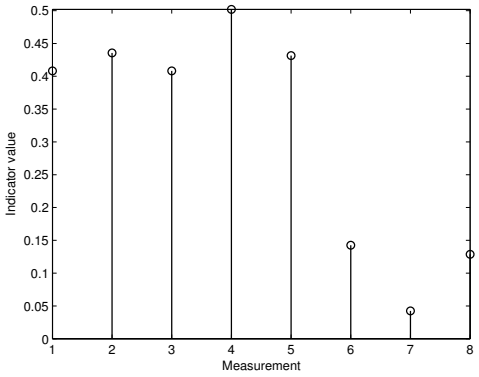

Fig. 6. First eigenvector of A, $u \in \Re^8$. The continuous indicator vector is the first eigenvector of A. Large values indicate that the corresponding measurement is an inlier; small values indicate outliers.

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 brute-force manner by trying every functionally distinct threshold value, i.e., try $t \in u$, computing $v(t)$ and the dot product (Equation 7) for each.

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 the components of $v(t)$ must be either zero or one. In practice, however, not only are the inlier elements of u of similar value (they typically intersect roughly the same number of other measurements), but the discrete vector $v(t)$ performs very well.

Outlier rejection performance can be improved by incorporating *a priori* knowledge of the noise characteristics into the threshold. For example, if consistently half of the measurements are outliers, then the algorithm should be biased towards rejecting half of the measurements. To optimally reject a fixed fraction of measurements, simply discard the measurements with the lowest magnitudes in u. If only half of the measurements are to be retained, then set $t = max(median(u), t_{opt})$. This guarantees that *at least* the expected number of samples is discarded *and* that the measurements were highly consistent with each other.

A typical result on 25 real range measurements is shown in Figure 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 given small indicator values. Also noteworthy is that one of the measurements is completely disconnected from the other measurements; the

algorithm works for the case of unconnected graphs.

As mentioned before, 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 their connectivity.

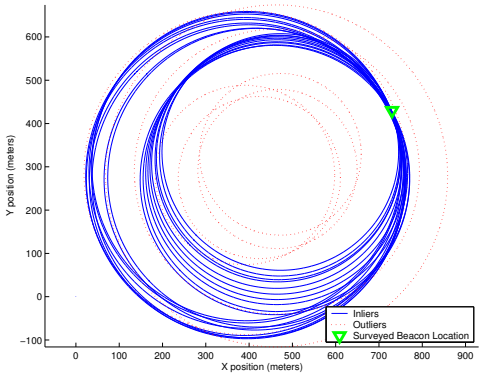

Fig. 7. Outlier Rejection Result. Twenty-five measurements are plotted, showing both inliers and outliers. Measurements that were consistent with many other measurements were classified as inliers.

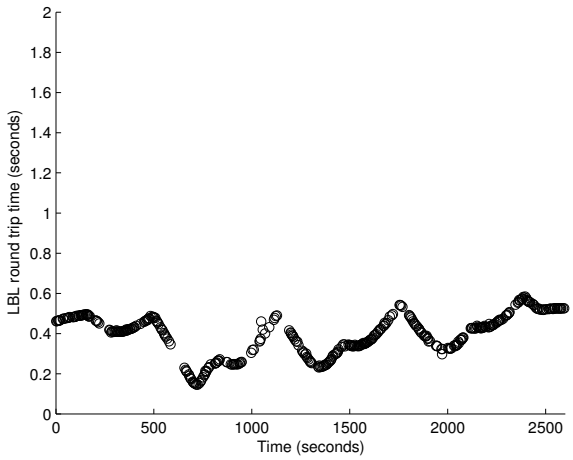

Fig. 8. Data from Figure 2 After Outlier Rejection. The filtered data is dramatically cleaner than the input data, and no beacon location information was required. The raw measurements were filtered in blocks of ten measurements (about 30 seconds) each.

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, the accumulation of navigational error 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.

B. Comparison to other spectral partitioning methods

A number of similarly motivated partitioning methods exist ([3], [4]). These algorithms also compute a graph partitioning 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, the largest eigenvalue of a potentially large matrix will need to be computed. 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 [10]. For any vector x not perpendicular to the largest eigenvector u, the direction of $A^n x$ approaches u as $n \rightarrow \infty$. In practice, x converges to a sufficiently good approximation of u in a few iterations.

Most other spectral partitioning algorithms have eigenvalue solutions as well, but their solution is found in different eigenvalues.

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, it makes sense to trade CPU time for 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 only assumes that approximate vehicle positions are available for each range measurement. A consistency matrix A can be computed regardless of how many vehicles were involved in data acquisition. The algorithm is entirely unchanged; a single set of inliers is identified.

In order for multiple vehicles to share data, they must have estimates of their positions in the same coordinate frame. If the robots know their starting positions, then they can share data immediately. Otherwise, the vehicles will need to 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 in the future.

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 known 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 location 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 [11]. Each consistent measurement pair "votes" for its two solutions. The physical world is discretized into a two-dimensional grid, with each grid cell corresponding to a rectangular area in the world. 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 in the solution: range uncertainty plus dead-reckoning uncertainty. Once all votes have been added to the accumulator, we can search the accumulator for the grid with the greatest number of votes.

If the 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 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 Figure 9. Empirically, we found a vote ratio of around 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. 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 robot's coordinate frame and the global frame will increase.

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 robot is located at (r_x, r_y), the beacon at (b_x, b_y) and the robot's orientation is r_t.

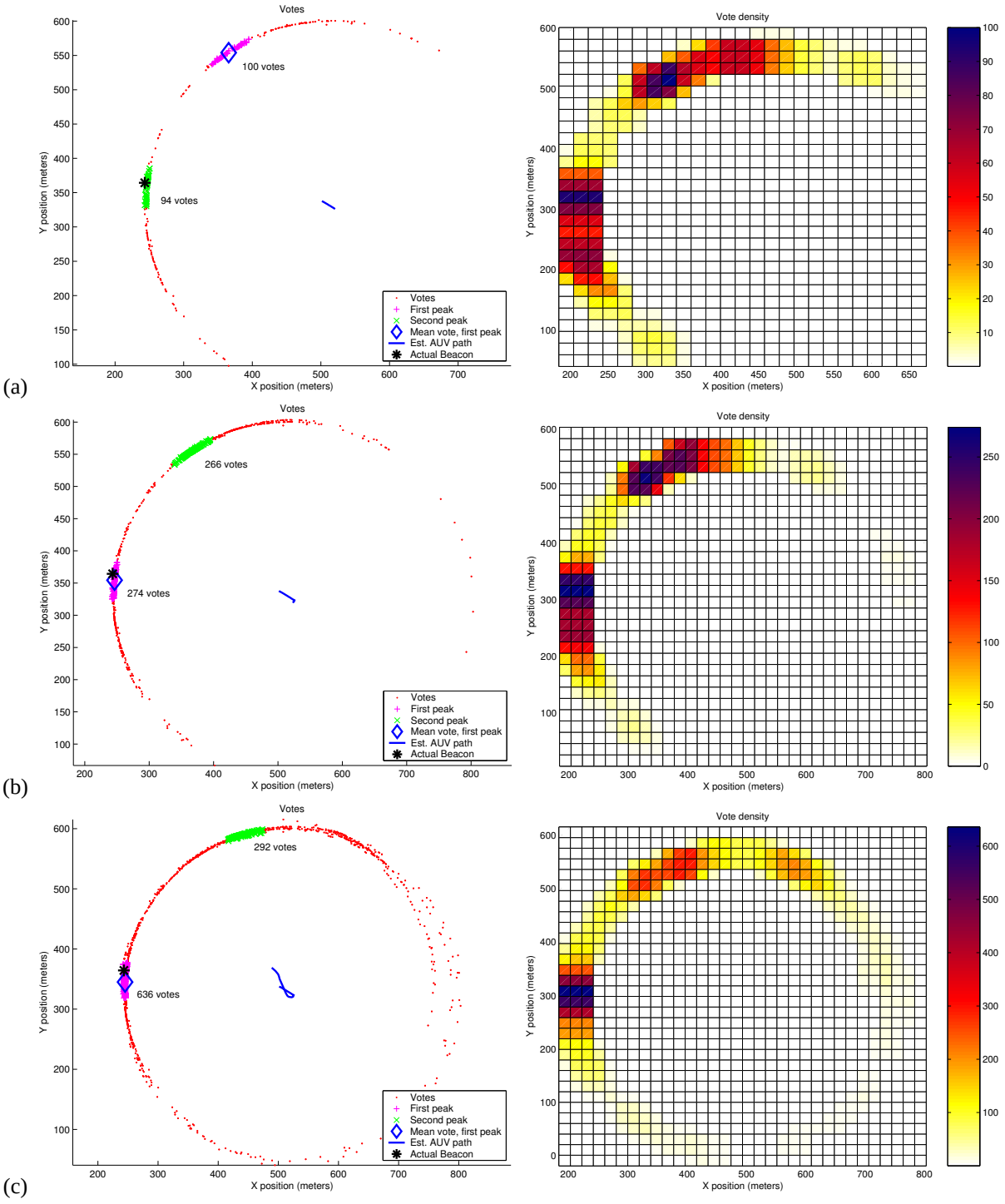

Fig. 9. Beacon Localization Using Voting. Using the inliers computed by our spectral outlier rejection algorithm, we compute possible beacon locations by finding the pairwise intersections of a set of measurements. Each of these intersections is a *vote* in a two-dimensional accumulator. The figure shows the votes in Euclidean space (left) and the vote density (right) in three successive iterations. At each iteration, the two largest peaks in the vote density are identified. If the largest peak is sufficiently larger than the second largest peak, the beacon's location is decided. Otherwise, initialization of the beacon is deferred until more data is available.

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

When we receive a new range measurement z_n to the beacon, we perform a Kalman update step. The first step 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 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, i.e.:

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

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

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 perfectly correlated with the error in the robot's state since the robot's state was used to estimate the beacon's position. Thus, the covariance data for the beacon is initialized to be equal to the robot's. 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 shown below.

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

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

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

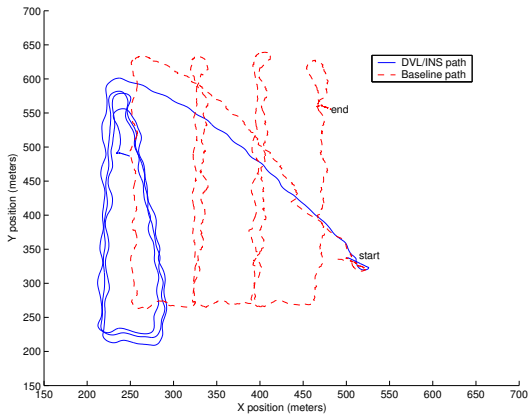

Fig. 10. Dead-Reckoned Path. The dead-reckoned path that we used to bootstrap our SLAM filter was relatively poor. It was constructed from DVL and an uncalibrated fluxgate compass.

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 Figure 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 robot, including the initial portion during which only dead-reckoning was possible, can be recomputed. The coordinate frame of the robot will differ from the global coordinate frame by 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 Figure 11. The global translation/rotation error (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 error was estimated by aligning the beacons, the beacon localization error is probably somewhat over-fit. However, due to the close agreement of the the two vehicle tracks, the degree of over-fitting is arguably small.

Animations of both the beacon localization process and operation of the ROBL SLAM filter are available at http://cgr.csail.mit.edu/robl.

VII. Optimal Exploration

The number of measurements needed to localize a beacon is strongly dependent on the path of the robot. Straight trajectories are the worst possible case; an AUV can travel in a straight line forever, and will not be able to determine whether the beacon is to the right or left.

Suppose a beacon's location has been reduced to two possible points, A or B. This will be the case after two range measurements from slightly different positions.

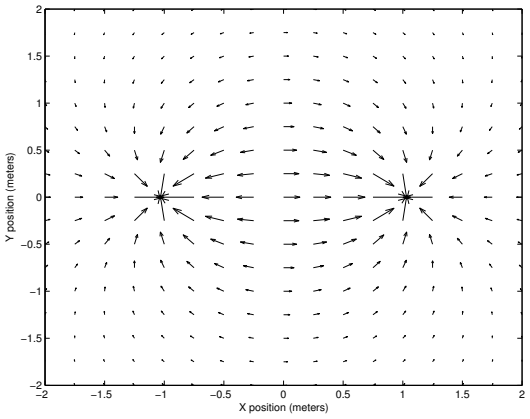

Fig. 12. Exploration Gradient. If a beacon's location is believed to be either (-1,0) or (1,0), then the best disambiguating motion is a function of the AUV's position. The vehicle maximizes the difference between the range measurements by traveling along the arrows. The length of the arrows indicates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

If the range to points A and B is the same, then a range measurement tells you nothing in terms of which point is more likely. To maximize the amount of knowledge about which point is more likely, the robot should move so that the *absolute difference* in range to points A and B is as large as possible.

Given points A, B, and the robot'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.

$$\nabla(r) = \left(\frac{B_x - R_x}{|B - R|} - \frac{A_x - R_x}{|A - R|} \right) \hat{x} + \left(\frac{B_y - R_y}{|B - R|} - \frac{A_y - R_y}{|A - R|} \right) \hat{y} \quad (17)$$

An example gradient field, with $A = (-1, 0)$ and $B = (1, 0)$ is plotted in Figure 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 robot moves.

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 quickly reduces 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 carefully surveyed beacon locations. Filtering noise in LBL data was 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. The combination of strong outlier rejection and beacon position estimation comprise our Range-Only Beacon Localization (ROBL) algorithm.

Using the ROBL algorithm, 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, the ROBL/SLAM filter 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 robot's path should be chosen to optimally resolve ambiguous data.

Future work includes reducing the dependence on dead-reckoning, and using particle filters to allow beacon positions to be estimated earlier.

Acknowledgements

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

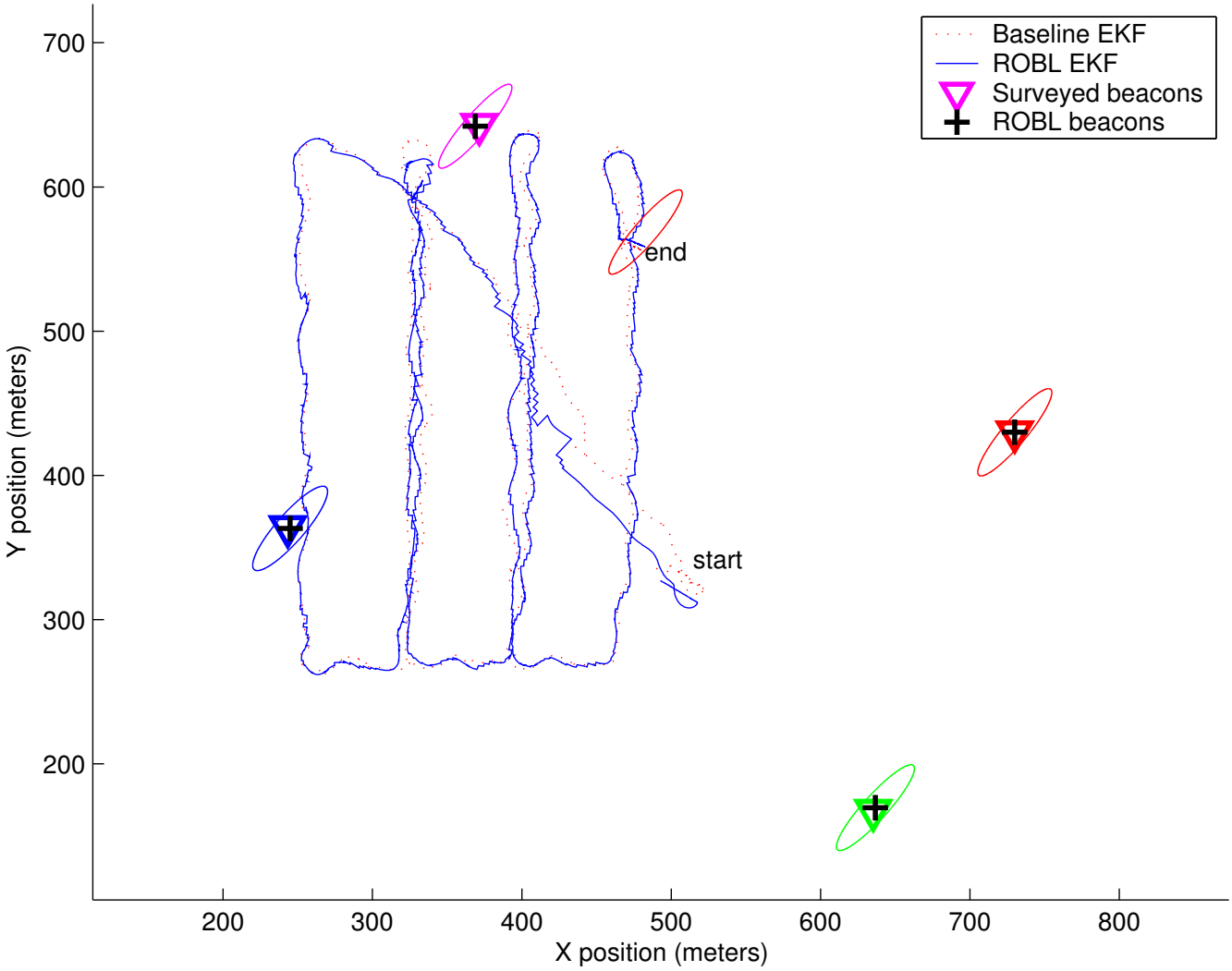

Fig. 11. SLAM filter results. Our SLAM filter's performance (labeled ROBL EKF) favorably compares to the performance of a baseline filter that uses prior beacon location information. A modest global translation/rotation error was manually corrected to allow a better comparison between the two computed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

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] W. Donath and A. Hoffman, "Lower bounds for the partitioning of graphs," *IBM J. of Research and Developement*, vol. 17, pp. 420–425, 1973.

[6] M. Fiedler, "Algebraic connectivity of graphs," *Czecheslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[7] ——, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czecheslovak Mathematical Journal*, vol. 25, no. 100, pp. 619–633, 1975.

[8] 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.

[9] 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.

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

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