Practical Mobile Robot Self-Localization

Jon Howell

Department of Computer Science
Dartmouth College
Hanover, NH 03755
jonh@cs.dartmouth.edu

Bruce Randall Donald*

Department of Computer Science
Dartmouth College
Hanover, NH 03755
brd@cs.dartmouth.edu

Abstract

A *mapmaking* robot integrates accumulated sensor data into a data structure that can be used for future localization or planning operations. *Localization* is the process of determining the robot's location within its environment. This paper describes experiments in which a robot simultaneously makes a map and localizes to that map. The map is a collection of tangent vectors constructed from stored sonar readings localized to a series of estimated poses. The vectors retain sensed surface normal information to improve accuracy. The localization scheme is a Hough transform into a space described by the robot's current sonar scan. The Hough transform finds a best fit in the presence of both sporadic sensor noise and discretization error.

1 Introduction

In their work on mapmaking and localization, Brown and Donald describe a localization algorithm based on *feasible poses* [3, 4]. A feasible pose is a pose consistent with available sensor and map information. For example, if the robot has a range sensor reading indicating an obstacle two meters due North, then any position two meters south of an obstacle edge in the map is a feasible pose. By intersecting the feasible poses for several sensor readings (for example, range data at different angles), the set of feasible poses becomes small, leaving only those poses that cannot be disambiguated by the available data. The availability of a prior pose estimate (perhaps informed by odometry data) allows the robot to select the most likely pose.

Sensor error may cause the intersection of feasible poses to be empty. We can account for sensor inaccuracy by constructing an error ball about the sensor values, or equivalently, blurring the map. Gross errors in sensor data can be absorbed by omitting some sensor readings; the final intersection must account for some threshold of the available sensor data, or perhaps as many sensor readings as can be used (a maximum vote) without producing an empty intersection of feasible poses.

Brown and Donald's paper describes a geometric approach to computing feasible poses, and analyzes its computational complexity. It also gives an approximate algorithm for the same problem using fast rasterized operations on a statistical occupancy grid. The work described herein extends upon the previous approximate algorithm in the following ways: First, we cast the feasible pose intersection operation as a Hough transform into a space modulo the outline of current range readings. This allows us to easily capture both the blurring and voting operations that account for different kinds of sensor error. Second, our map data includes surface normal information available using sonar sensors, which tightens the set of feasible poses induced by a given sensor reading. Third, we integrate mapmaking and localization: the map contributes to producing correct localization information, and the localization is used to position new sensor data on the map.

Our approach to mobile robot self-localization and map making, based on the algorithmic framework for self-localization in [4], contains no notion of explicit landmarks. This contrasts with Extended Kalman Filter (EKF) approaches (e.g., [24]) employed in earlier Simultaneous Localization and Map Making (SLAM) techniques [7, 16].

*This research is supported in part by the following grants to B.R.D. from the National Science Foundation: NSF 9802068, NSF CDA-9726389, NSF CISE/CDA-9805548, NSF EIA-9818299, NSF EIA-9901407, NSF IIS-9906790, NSF IRI-9530785, NSF IRI-9896020, and by an equipment grant from Microsoft Research.

2 The model

Many robots are either given a reliable map and asked to localize themselves, or given reliable position information and asked to construct a map. Our goal was to program a robot to perform both operations simultaneously. In our system, the robot begins by taking a sonar sounding, which produces a polar distance map of the robot's immediate neighborhood. The robot is assumed to be at the origin $(0, 0)$, and these initial soundings are taken to be the robot's initial map.

Then the robot moves in some direction, stops, and takes another sounding. This sounding is localized against the map, returning a most likely location of the robot relative to the origin. The soundings are then shifted to the robot's now-known position, and contributed to the map. This cycle repeats indefinitely as the robot explores; at each step, the soundings pointing "behind" the robot's path help it localize itself, and the soundings pointing "ahead" contribute new information to the map. Our current system does not yet address goal planning; but one could imagine specifying a goal as a geometric landmark ('pose') on the robot-made map.

3 Related Work

There is a large literature on localization and pose-estimation. Guibas, Motwani and Raghavan present an $O(m + \log n + A)$ algorithm for localization, where m and n are the number of vertices in the visibility polygon and the world polygon, respectively, and A is the size of the output. The authors do not address uncertainty [11]. Kleinberg provides an online search algorithm by which a mobile robot moving within its environment can uniquely determine its location without access to an *a priori* map [13]. Localization work using *beacons* or *landmarks* includes [18, 12, 15, 14]. Several attempts have been made to use computer vision to detect and locate beacons (see [5, 1, 22], for example).

Drumheller [6] used the *interpretation-tree*-based two-dimensional matching algorithm of Lozano-Pérez and Grimson [10] and an *a priori* line-drawn map of the environment to produce *interpretations* (lists of possible (sonar segment/wall) pairings). This is similar to several matching algorithms based on the Hough Transform. See, for example, [23]. Beveridge and Riseman [2] use a 3D, perspective-based computer vision matching algorithm to track a robot's progress as it navigates in hallways. [17] present a sonar sensor which is capable of tracking environment features. Moravec and Elfes introduced the *statistical occupancy grid* method of map-making in [8, 19, 20] as a way of making detailed geometric maps using noisy sensors. Takeda and Latombe [25] approach the problem of motion planning with uncertainty by computing a *sensory uncertainty field* for each configuration in the robot's configuration space. Ratering and Gini [21] describe a *hybrid potential field*, which combines a global, *a priori* potential field, generated from the robot's map with a local potential field generated from instantaneous sensory data. Thrun, Fox and co-workers have employed Monte Carlo approximations for localization and map making [9].

4 Experimental setup

The data used in these experiments were collected from the RWI B14 robot named BONNIE that resides in the Dartmouth Robotics Lab. The environment consists of the assorted desks, chairs, robots, and cardboard boxes strewn about the lab. The robot has sixteen sonar transducers evenly spaced 22.5° apart. By rotating the robot's body in small increments and taking soundings from every sensor, we simulated a ring of 128 sensors at 2.8° spacing.

A sonar sensor measures the time-of-flight of a burst of sound. Sonar can see a "cone" of about 22° arc, so it blurs the boundaries of the objects it sees. The return value is a single number representing the first echo above some threshold amplitude, so the blur is a box convolution in polar coordinates. If sonar hits a surface at a wide angle from the surface normal, the signal may not return, or it may bounce off a second surface and return a measurement along some bent path, useless for our purposes.

In a typical run, the robot collects a set of sonar samples, estimates its pose, updates its map, plans its next motion, moves, and the cycle repeats. Because we did not address planning, a human simulates a software planner by viewing the map and entering a new destination into the control program. The online algorithm lends itself to repeatable offline simulation since we can replace real online input with input recorded on a previous run. This property facilitates repeatable experimental results and easy validation of proposed improvements to the algorithm.

When the map exceeds a specified size (in our experiments, 1500 samples), samples are randomly culled to reduce the map size. Culling provides two benefits: it bounds the computation time of the algorithm, and it "ages" old information. If a chair moves in the world, the representation of the chair on the robot's map will fade with time. Walls and other static features are constantly reinforced by new measurements. By varying the parameters of this process we can trade off between agility and stability in the map.

We calibrated the sonar by placing a box at known distances from 1 m to 4 m from the robot, and reading the sonar value. The returned values and the known values had a very close to linear relationship, which is not surprising: sound travels at a constant speed, and time is one quantity we can "sense" with high precision. All of the sonar soundings were mapped through the measured linear function to produce a distance value in millimeters. We also produced a histogram of the entire collection of 7,616 sonar readings, shown in Figure 1. Low values (less than 200) are reflections from the cables hanging off the robot; high values are always 16384, a sentinel meaning "no echo received." These thresholds were used to remove useless values from the input data.

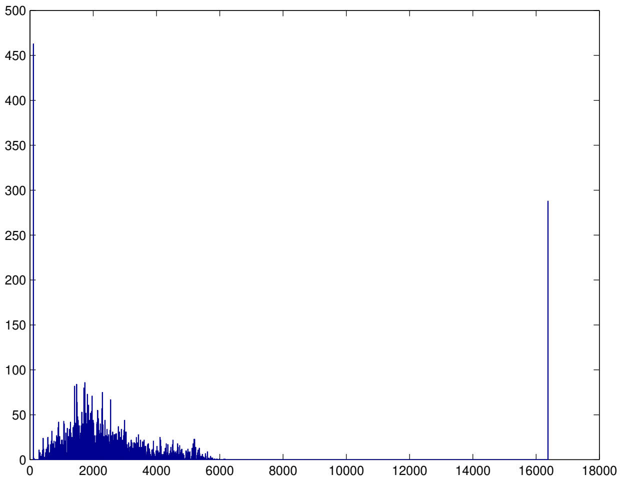

Figure 1: A histogram of 7168 sonar readings taken around the room, showing low and high garbage values.

We performed three experiments. The first determined how well the localization part of the algorithm performs given an accurate map. The second extended the algorithm to include orientation localization, and measured the accuracy of the orientation estimated by the algorithm. The third evaluated the entire algorithm, wherein the robot alternately localized its position using its existing map, then extended its map using the newly-localized sonar samples.

5 Localization Experiment

The localization experiment was designed to determine how well the robot could find itself given an existing map, a set of sonar readings, and a known orientation. To measure the positional accuracy of the algorithm, we manually placed the robot at twenty-eight different locations in the lab, and took soundings of the sonar surface around the robot at each location. In each case, we positioned the robot over the intersection of the steel floor tiles rather than using the robot's own locomotion, so that we could measure the robot's pose with some accuracy (position to within about a centimeter, and orientation to within a few degrees). See Figure 2.

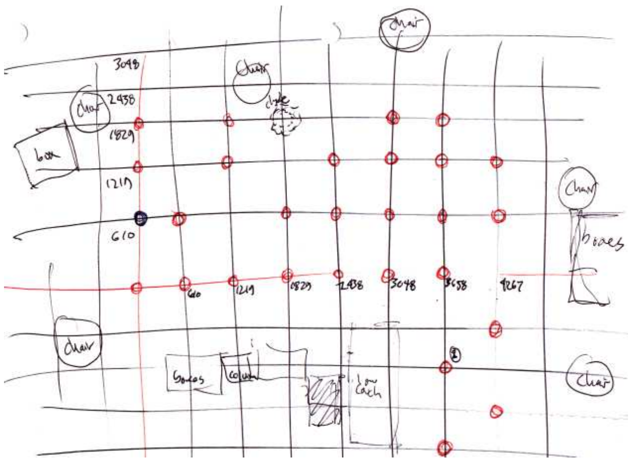

Figure 2: A hand-drawn map of the environment. Circles at intersections of the two-foot tiles show the locations where soundings were taken.

The experiment consisted of 28 trials. In each trial, the sonar readings from one of the 28 positions was removed into a test set. The remaining readings were combined with the known location of the robot when each was taken to create the "known map" (see Figure 3). Some fraction (frequently 13/16) of all sonar readings was discarded to make the algorithm run more quickly.

The test set of sonar readings consists of angles and distances from the robot's center, but the robot's location relative to the map is unknown. Hence, we compare each test sonar value with every map vector. The map vectors represent the position and the very approximate normal of some observed surface in the environment. Therefore, those pairs of $\langle \textit{test vector}, \textit{map vector} \rangle$ that did not agree to within some threshold were discarded. (We chose a threshold

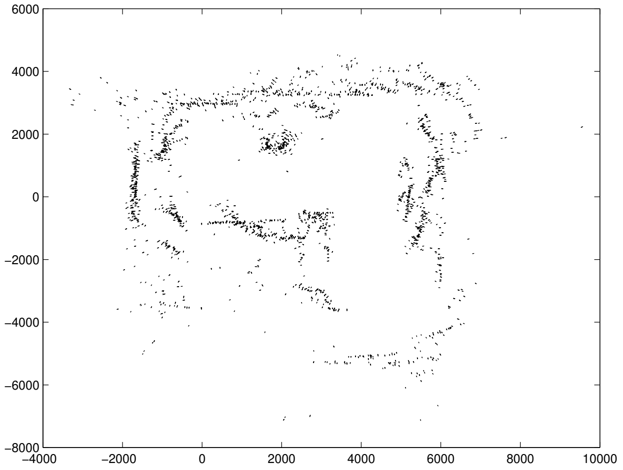

Figure 3: The known map: all 6414 valid sonar readings. Each vector points back at the position of the robot when that reading was taken; the vector's distance from the robot's position is equal to the distance value sensed by sonar.

of 30° experimentally.) The idea is that a map vector represents an "outside" surface of some object; it is unlikely that a test vector "sees" that surface from behind or from a shallow angle.

The remaining pairs "vote" for the most feasible pose of the robot. The position of the map vector is added to the negative of the test vector to locate where the robot would have to have been for the test vector to represent a reflection off of the same surface responsible for the map vector. Each vote contributes a small value to a raster image. The location with the most votes is taken to be the most likely location of the robot. See Figure 4.

This transformation is a specialized Hough transform. A Hough transform maps points from a Cartesian space into a space representing the degrees of freedom of some geometric figure. For example, a linear Hough space might have two axes, one representing the angle of a line and the other representing the offset of the line from the center of the Cartesian space. When one transforms an image from Cartesian space, each point in the image contributes a vote for every line in the Hough space that passes through that point in Cartesian space. Wherever a line appears in the original image, votes for that line pile up in the Hough transform. The highest votes can be read directly out of the Hough space as equations for dominant lines in the source image.

Our algorithm is very similar, but instead of transforming to the Hough space of geometric lines or circles, we transform to a Hough space representing the surface of range measurements formed by the test vectors. The axes of our Hough space are simply the x and y coordinates of the origin the test vectors. Just as in the linear case, the voting process causes the "dark spots" in the Hough space to represent equations (locations) of the dominant occurrences of the test surface in the map.

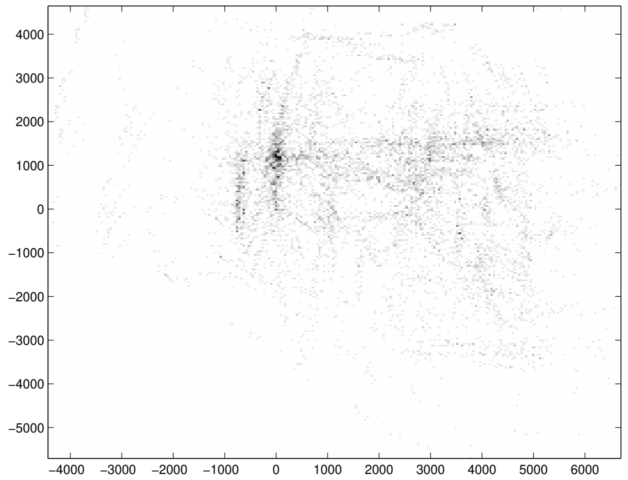

Figure 4: A collection of votes. Darker spots received more votes, and thus represent "more feasible" poses.

The surface of votes is very spiky, due to the sporadic nature of sonar samples of the environment. In fact, the map vectors can only be considered a guess at the location of the reflecting surface, because of the signal blur mentioned earlier. Therefore, we convolve the vote raster with a two-dimensional Gaussian surface to smooth it. Because of the choice of axes in our Hough transform, this is equivalent to but faster than blurring the sonar samples on the map directly. Figure 5 shows the smoothed vote surface superimposed with the known map (short vectors) and the star-shaped set of test vectors translated to the location of the maximum value on the vote surface.

To quantify the reliability of this algorithm, we ran it for all 28 test vectors, and repeated each run with 1/16, 2/16, and 3/16 of the map data. The results are displayed in Table 1.

A "wild error" is defined as an error more than half a meter away from the known location where the sample was originally taken. Generally, these occurred when the sample was taken in a sparsely-populated corner of the map (the lower-right leg in the L-shaped region visible in Figure 3), and an approximate match in a heavily-populated section was able to outvote the

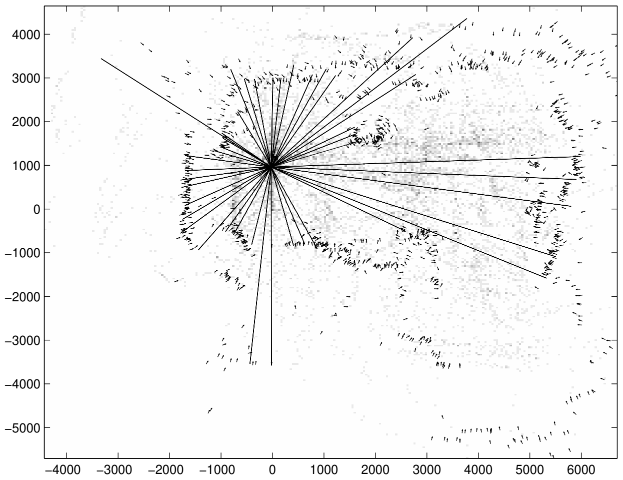

Figure 5: A successful localization. The star of vectors represent the sensed range data, sprouting from the computed pose of the robot on the map.

fraction of sonar samples	*raster res.*	*wild errors*	*tame errors mean*	*tame errors σ*
1/16		10.7%	91 mm	48 mm
2/16	47 mm	3.6%	54 mm	28 mm
3/16		7.1%	47 mm	20 mm
1/16		10.7%	86 mm	49 mm
2/16	24 mm	3.6%	46 mm	25 mm
3/16		3.6%	34 mm	19 mm

Table 1: Rate of erroneously localized test samples as more sonar samples are made available.

correct position. These gross errors could be easily detected and eliminated if an estimated pose were available, for with a prior localization plus the results of odometry over a short distance, the robot could estimate its pose accurately enough to discard wildly distant poses as infeasible.

The means of the remaining errors, denoted "tame errors," are far less than the odometry errors for this robot system; we measured them at 50 to 90 millimeters, depending on how much sonar data was used in the map. Increasing the resolution of the raster gave a small improvement in the accuracy of the algorithm. Doubling the resolution quadruples the size of the raster, which slows the convolution and maximum-finding phases of the algorithm; however, with the parameters presented here, the vote-collecting phase dominates the total run time.

6 Orientation Experiment

There is no reason the Hough transform described in Section 5 should be constrained to two dimensions. We extended it to include a third dimension, orientation. To reduce computation time, we exploited the odometry system by assuming incremental orientation errors were small, so that we needed search only a small slice of the orientation dimension centered on the commanded orientation. The computed pose is again the pose with the most votes, now along the three dimensions of x, y, and θ. When the sonar samples are inserted into the map, they are both shifted to the computed position and rotated by the computed orientation.

To test the accuracy of the orientation localization, we built a baseline map by running the robot in online mode and commanding it to move to six locations in a 1.5 m square. By positioning the robot at an intersection of floor tiles and attaching a 60 cm steel bar, we measured the orientation of the robot to 0.5° precision. The angle of the bar relative to the axis of the sonar array was known with somewhat lower accuracy. We commanded the robot to rotate to a series of orientations in a 17° range, measured the robot's orientation, then instructed the robot to take a sonar sounding and localize itself.

The result was a 1.27° mean error with a 0.72° standard deviation. The mean likely reflects the poor accuracy to which we could measure the absolute orientation of the robot; the low standard deviation is much more interesting because it indicates that the algorithm detected the robot's position with high repeatability. The 0.72° standard deviation of the results compares favorably with the 2.8° resolution of the sonar data and the 1.1° raster resolution used in the experiment.

7 Combined Mapmaking and Localization Experiment

The localization experiment in Section 5 introduced the voting algorithm for localization, and demonstrated that it is practical. It assumed, however, the availability of a rich, accurate map. An online algorithm would need to be able to construct a map from successive localization/map-making cycles. To explore this mode, we ran a combined mapmaking and localization experiment. Only soundings from successive positions of the robot are considered in the algorithm. The computed position of the robot is used to locate the robot in the map and contribute the new soundings

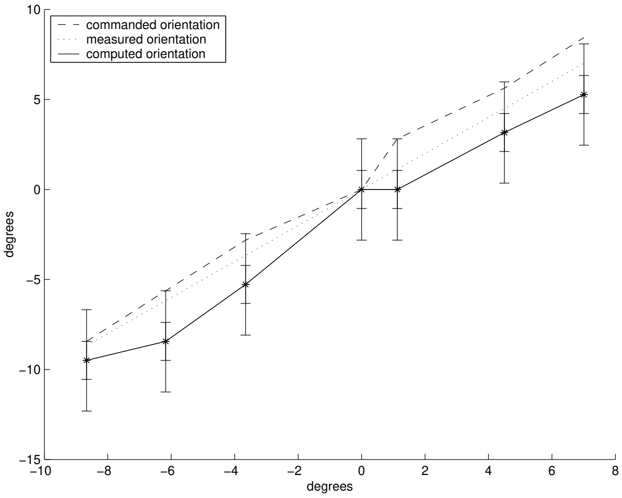

Figure 6: Orientation localization results. The dotted line is the measured orientation of the robot. The solid line is the computed location using the localization algorithm. The outer error bars depict the resolution of the sonar samples (2.8°), and the inner bars depict the angular resolution of the raster used in the localization computation (1.1°). The dashed line shows the commanded orientation, providing an idea of how the localization error compares in magnitude to the mechanical accuracy of the robot.

to the map.

We ran this experiment first by simulating an online path using our data from the first experiment that included measurements. The robot's path was formed by sorting the sample locations into a a space-filling order, simulating the path of a robot controlled by a planner that grows its map slowly by pushing the envelope of the well-mapped area. Figure 7 shows snapshots of the map at various stages of growth. (An animated version showing all of the steps is available on the web; the reference is given below.)

We also ran the same experiment many times entirely in online mode. After the robot localized itself in both position and orientation, it updated its map and presented the map to a human operator. The operator selected a new nearby destination on the map, then the robot drove to that destination using odometry and repeated the localization cycle. We had great success in driving the robot around in this manner, even outside the lab and out of sight. The online algorithm does a satisfying job of accumulating a new map that corresponds well with hand-drawn maps and our knowledge of the environment.

8 Conclusions

By casting the localization problem as a Hough transform for an environment-specific surface, we simplified Brown and Donald's rasterized localization algorithm. An attractive feature of this approach to localization is that it does not require landmarks: unfiltered range data is sufficient to localize the robot and build a map. Our algorithm also exploits approximate knowledge of surface normals to prune unlikely feasible poses. Finally, we integrated mapmaking with localization, showing how the algorithm could be used by an autonomous robot to dynamically generate and maintain a map from sensor data.

Because it exploits the long reach of a ranging sensor, the approach is less inclined to "drift" than odometry, which is based on local, relative sensing. The algorithm lets us trade off accuracy for speed, either by reducing the number of map samples considered, or by using a coarser grid to collect votes. The limitations of the algorithm are its scalability to maps with many samples and its "wild" errors. We discuss proposed solutions to both problems in the next section.

9 Extensions

We are extending this work to enhance its performance to scale to large maps. The algorithm slows linearly in the number of vectors in the map. We mentioned in Section 4 how aging can reduce the number of map vectors inspected. We propose two alternate solutions: restricting our inspection of the map to the immediate neighborhood of the robot and thinning the map.

Neighborhood reduction. If the robot has not traveled far from a previously localized position, there is no need to consider map vectors that cannot be reached by any of the test vectors. This pose estimate can be even further refined using even gross information from the robot's odometry relative to the previous localization. Odometry information can be incorporated by attenuating the Hough vote raster as a function of distance from the pose predicted by odometry. Both steps reduce the opportunities for wild errors. Attenuation based on odometry is likely to transfer any bias in the odometry error into the localization error.

Density reduction. In Section 5, our algorithm discards a fraction of the map data to improve performance. One might set a limit on the number of map vectors to be inspected (the number needed to reasonably represent the surfaces in a neighborhood as

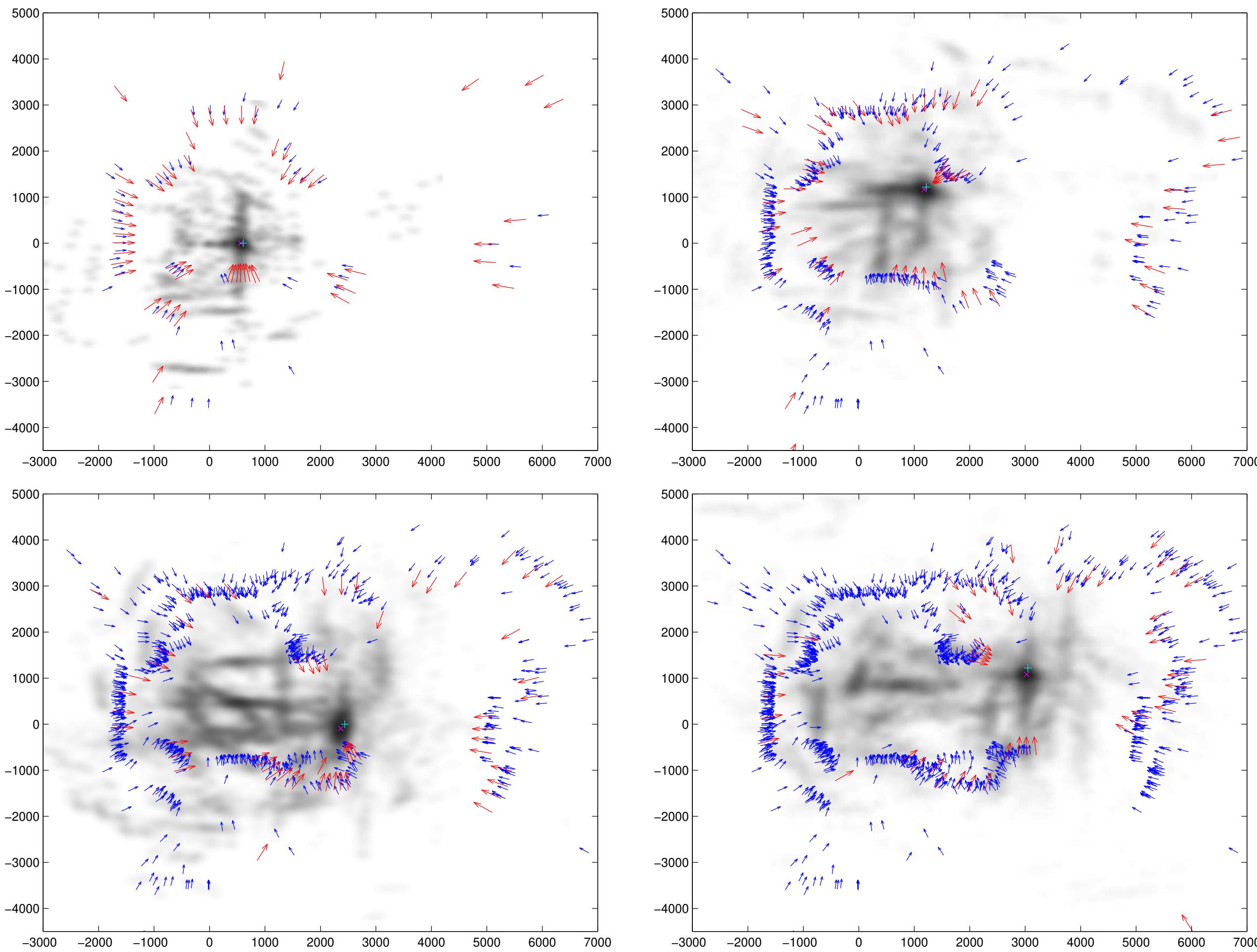

Figure 7: An online mapping sequence. The short dark vectors are the map vectors used in a round of the localization algorithm; the long grey vectors are the test set, translated to reflect the result of localization. In the next round of the algorithm, the test vectors join the map, the robot moves (simulated by selecting a new group of test vectors sampled from a nearby location in the room), and the algorithm repeats.

described previously), and then select that many vectors at random from the neighborhood to participate in the voting.

Raster resolution reduction. The algorithm also slows quadratically (or cubically when the orientation axis is used) in the resolution of the Hough image. One way to limit this problem would be to find a best match at a low resolution, then increase the resolution while limiting the scope of the new search to the neighborhood of the low-resolution match. Alternatively, we could use a gradient-ascent algorithm to explore the Hough vote surface quickly from an initial estimate; the votes are generally organized into ridges that culminate in a peak at the estimated pose, forming a structure quite amenable to hill-climbing.

Images

This paper is available on the web at http://www.cs.dartmouth.edu/~jonh/robots/. It is accompanied by color graphics and animations which complement the monochrome stills in the printed version of the paper.

Acknowledgements

We would like to thank Russell Brown for helpful discussions; Brian Brewington for help with Matlab; Jeff Oldham, who suggested aging the map vectors; and Daniela Rus for providing the great infrastructure of the Dartmouth Robotics Lab.

References

[1] S. Atiya and G.D. Hager. Real-time vision-based robot localization. *IEEE Transactions on Robotics and Automation*, 9(6):785–800, December 1993.

[2] J. Ross Beveridge and E. M. Riseman. Hallway navigation in perspective. In *Proceedings of the AAAI Fall Symposium, Sensory Aspects of Robotic Intelligence*, pages 125–132, November 1991.

[3] Russell G. Brown. *Algorithms for mobile robot localization and building flexible, robust, easy to use mobile robots*. PhD thesis, Cornell University, Ithaca, NY, May 1995.

[4] Russell G. Brown and Bruce R. Donald. Mobile robot self-localization without explicit landmarks. *Algorithmica*, 26(3/4):515–559, March/April 2000.

[5] F. Chenavier and J. L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2588–2593, 1992.

[6] M. Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 325–332, 1987.

[7] H. F. Durrant-Whyte, M. W. M. G. Dissanayake, and P. W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (SLAM) systems. In *Proceedings of the Ninth International Symposium on Robotics Research*, October 1999.

[8] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[9] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. Efficient multi-robot localization based on monte carlo approximation. In *Proceedings of the Ninth International Symposium on Robotics Research*, October 1999.

[10] W. Eric L. Grimson and T. Lozano-Pérez. Recognition and localization of overlapping parts from sparse data in two and three dimensions. In *1985 IEEE International Conference on Robotics and Automation*, pages 61–66, 1984.

[11] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem in two dimensions. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–268, January 1992.

[12] L. Kleeman. Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning. In *Proceedings of the 1992 IEEE International Conference on Robotics And Automation*, pages 2582–2587, 1992.

[13] J.M. Kleinberg. The localization problem for mobile robots. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 521–531, November 1994.

[14] I.S. Kweon and T. Kanade. Extracting topographic features for outdoor mobile robots. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1992–1997, 1991.

[15] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, August 1992.

[16] John J. Leonard and Hans Jacob S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proceedings of the Ninth International Symposium on Robotics Research*, October 1999.

[17] J.M. Manyika and H.F. Durrant-Whyte. A tracking sonar sensor for vehicle guidance. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pages 424–429, 1993.

[18] C.D. McGillem and T.S. Rappaport. Infra-red location system for navigation of autonomous vehicles. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1236–1238, 1988.

[19] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[20] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings, IEEE International Conference on Robotics and Automation*, pages 116–121, March 1985.

[21] S. Ratering and M. Gini. Robot navigation in a known environment with unknown moving obstacles. In *Proceedings of 1993 IEEE International Conference on Robotics and Automation*, pages 25–30, 1993.

[22] Y. Roth, A.S. Win, R.H. Arpaci, T. Weymouth, and R. Jain. Model-driven pose correction. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2625–2630, 1992.

[23] T.M. Silberberg, D.A. Harwood, and L.S. Davis. Object recognition using oriented model points. *Computer Vision, Graphics, and Image Processing*, 35(1):47–71, 1986.

[24] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[25] H. Takeda, C. Facchinetti, and J.-C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1002–1017, October 1994.