Detecting and Tracking Level Sets of Scalar Fields using a Robotic Sensor Network

Karthik Dantu and Gaurav S. Sukhatme
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
Email: {dantu, gaurav}@usc.edu

***Abstract*— We introduce an algorithm which detects and traces a specified level set of a scalar field (a contour) on a plane. A network of static sensor nodes with limited communication and processing are deployed in a planar environment along with a mobile node which can both sense and move.As the mobile node moves through the environment, it computes the local spatial gradient of the field by communicating with its immediate neighbors in the static sensor network. The algorithm causes the mobile node to perform gradient descent on the scalar field till it arrives at a location on the desired contour. From this point onwards, the algorithm drives the mobile node to trace the desired contour without departing from it.Experiments in simulation indicate that the required contour is found with reasonable accuracy (between 80-90%) for networks with node degree of 6 or greater. Our results also indicate that the paths generated by our algorithm are near-optimal in terms of the distance traversed by the mobile node. Our preliminary experimental results with a physical robot show that our algorithm is feasible.**

I. Introduction

Sensor networks are quickly evolving into powerful tools for environment monitoring. Their use is [15] particularly apt for environments which are remote, hostile, or inaccessible. In such environments infrastructure does not exist, and is hard to build and maintain.

There are several key requirements that a sensor network needs to satisfy to be viable in unstructured environments. It needs to function unattended and conserve energy to extend lifetime. The applications developed for such a network need to scale to large sizes. This is particularly difficult since individual nodes of the network usually have meager computation, storage and communication capabilities [5].

To address scalability, a major area of research in sensor networks has focused on distributed algorithms [6] which rely on local communication among network nodes. To facilitate this, processing at nodes is designed to operate with local data input as much as possible since the energy cost of moving bits across the network is relatively high compared to the cost of computation [12].

It has been noted that spatio-temporal irregularity is one of the inherent characteristics of sensor networks [7]. There are two ways to approach this problem:

- The deployment could be assumed to be fixed and algorithms that incorporate the spatio-temporal irregularity [7] might be utilized.
- Adapt the deployment by relocating deployed sensors as and when required. This approach needs mobility.

For the latter case to be viable, the re-deployment needs to be autonomous in that the network nodes should be robotically mobile (not human portable). For true unattended functioning, the network of nodes needs to effectively function as a collection of autonomous robots, moving themselves in accordance with task requirements. In other words, the idea of adaptive fidelity [6] is to be extended to physical mobility of nodes.

Consider an example from environmental monitoring. We would like to detect the presence, and measure with relatively high accuracy the concentration of certain kinds of marine microorganisms in the ocean. We imagine that we have at our disposal nodes that can sample the water, detect and measure algal concentration, as well as measure temperature. The nodes can also communicate with each other using radio.

Fortuitously, certain algae are hypothesized to bloom near regions of sharp temperature gradients (thermoclines) underwater. Thermoclines occur at different depths and with varying profiles in the ocean. The problem for the network of sensors is to locate a thermocline and to sample the water for algae in the region near the thermocline. Given little prior information about the location of the thermocline, and the sheer size of such domains, over deployment of sampling nodes is difficult. We propose to use a strategy based on actuation [14].

The idea is to allow some of the nodes in the network to be autonomously mobile. We deploy static nodes at relatively low density over a wide area, and a (smaller) number of robotically mobile nodes that will spatially re-deploy as needed depending on where the thermocline is located. The robotic nodes will use information from the static network to facilitate their own re-deployment. One can think of similar problems for a variety of environmental phenomena, including finding the edge of a forest fire, the boundary of a chemical spill and so on.

The paper is structured as follows. Section II states the problem (contour finding) and our assumptions. Section III is a brief review of related work and Section IV provides the theoretical background for our work. Section V elaborates our approach. Section VI describes the simulation setup and Sections VII discusses simulation results. Section VIII describes our experimental setup, the validation experiments performed and their results. We conclude with a summary and a discussion of open issues.

II. Problem Formulation and Assumptions

We are interested in the detection of level sets of a scalar field being sensed (e.g.: isotherms if the sensed phenomenon is temperature) using one or more mobile sensor nodes and a network of stationary sensor nodes. Note that we use the terms *Level Set* and *Contour* interchangeably. Both refer to the set of points that have scalar field values equal to the desired level. We also make the assumption that the scalar field being detected has a measurable gradient in all regions (including those far from the contour under consideration). Note that certain classes of boundaries (e.g: step functions) do not possess this property. However many real physical phenomena (phenomena that can be modeled by diffusion processes) do have gradients that can be detected by sensors.

Static sensor nodes of limited communication range are assumed to be uniformly randomly deployed in the 2D space over which the field is defined. The values of the field could correspond to light, temperature etc. Each sensor node is assumed to possess appropriate sensing to measure the field along with computation and communication capabilities. Also, it is assumed that the nodes are localized. Several techniques exist in the literature to provide node localization; we use the one in [1] for our experimental validation. In addition to the static sensor nodes, it is assumed that one or more autonomously mobile nodes (e.g. [4]) are randomly deployed in the environment.

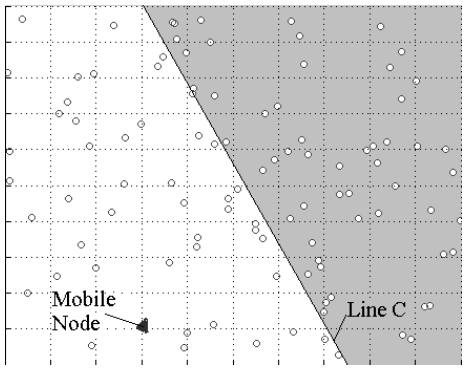

Fig. 1. Initial deployment and a particular level set to be traced by the mobile node

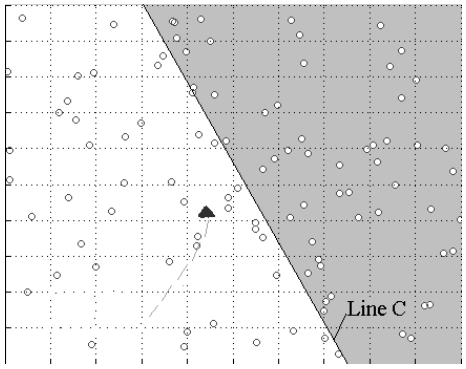

Fig. 2. The mobile node moves towards the level set

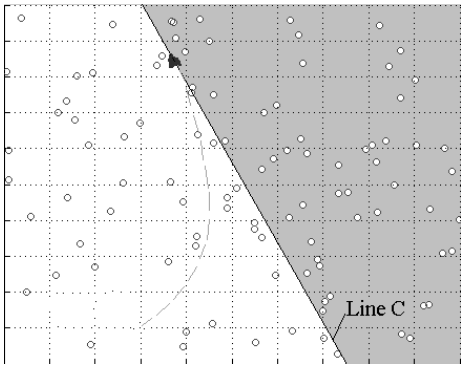

Fig. 3. The mobile node traces the locus of the contour

The objective is to detect and possibly trace a contour consisting of points in space where the scalar field values are equal to some prespecified desire level using the mobile node(s). Each mobile node is provided with the level that defines the contour to be traced. In Figure 1, 2, 3, the mobile node starts at an arbitrary position and is given the desired level of the scalar field that describes the desired contour. Its task is to locate the contour and to trace it. The other nodes in the picture can sense the value of the field but cannot move.

III. Related Work

A. Edge Detection

Boundary or edge-finding has received some attention in the sensor networking community. There are various schemes proposed using techniques from varied backgrounds e.g. computational geometry, image processing and statistics.

The dual-space approach to tracking boundaries [9] attempts to track/detect simple boundaries (approximated by straight lines) of physical phenomena by bounding them by certain nodes in the sensor network. The approach is to map the problem of detecting lines using points (sensor nodes) to a dual where lines are transformed into points and vice versa. A topological sweep is performed to detect the boundary nodes of the phenomenon and only those nodes are kept active. To move out of the region bounded by these nodes, it is shown that the boundary has to cross one or more of the active nodes. The algorithm in [9] monitors for such crossings and activates nodes appropriately. Thus, linear boundaries are localized in an energy-efficient manner.

Another paper on localized approaches to edge detection in sensor fields [2] introduces three algorithms to solve the problem of edge detection using static sensor nodes. The statistical approach tries to statistically estimate if a given sensor is an edge sensor by probing the neighborhood. There is a trade off between the amount of information communicated amongst nodes in a neighborhood to the certainty of the solution. The second technique in [2] is inspired by image processing. The idea is to use a high-pass filter to filter out noise and only retain the prominent differences (the edge) from the sensed data. Also, unlike images, since sampling is not possible at regular intervals, the authors provide weights

for the sensors based on a continuous version of the filter. The third technique discussed is a classifier-based approach inspired by pattern recognition. The idea is to classify sensor readings into two partitions. This scheme seems to provide the best results. All three schemes rely on what is called the *Probing Radius* which is the range around a sensor from which it can get other sensor information. Increase in probing radius increases the certainty of the decision of a sensor. However, the communication cost rises roughly as the square of the probing radius which constitutes an energy-quality trade off.

A third proposed solution to boundary estimation [11] attempts to use clusters and their hierarchical structure to minimize communication and thus reduce the overall energy consumption. The key idea is to devise a hierarchical processing strategy that enables nodes to collaboratively determine a non-uniform rectangular partition of the sensor domain that is adapted to the boundaries. This partition will have a fine resolution along the boundary and low resolution in homogeneous regions. The result is a *staircase-like* approximation to the boundary. The main motivation behind this work is that such algorithms have been explored in image processing and theoretical frameworks exist to analyze such strategies. Also, it provides a method of tuning the trade-off between accuracy of boundary estimate and energy consumption of the network.

All the above edge detection techniques are mathematically-inclined and adopt techniques from well-researched fields for edge detection. One of the main underlying motivations for these approaches is reduction in energy consumed. Our approach is inherently different in that we attempt to use mobile nodes and actively search for the required level set (using physical actuation). Our algorithm indirectly attempts to minimize energy consumption due to actuation by minimizing the distance traveled.

[10] studies distributed coverage control and proposes a distributed algorithm to redeploy mobile sensor nodes depending on the sensory function. This is the closest to our work. The authors propose a control law to move the robot to the centroid of the voronoi region formed based on the sensory function. This work is different from our work in that their objective is coverage control. Our objective is to detect a particular level set and quickly drive the mobile node to this level set.

IV. BACKGROUND

We are inspired by previous work in sensor-based planning [3]. For simplicity we consider only the situation in a plane. The Voronoi Graph of a planar environment consisting of obstacles and free space is the locus of points in free space that are equidistant from neighboring obstacles. This is schematically shown by the dotted line in Figure 4.

In [3] the authors propose and prove the convergence of a control law to drive a robot such that it traces the Voronoi Graph. We re quote the definition of the control law as applicable to two dimensions. Assume that the robot is at

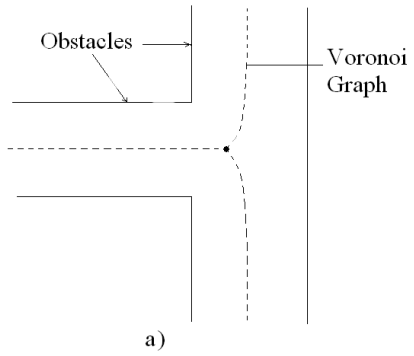

Fig. 4. The Voronoi Graph

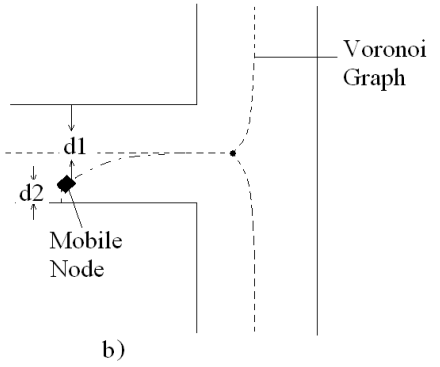

Fig. 5. The robot navigating towards the Voronoi Graph based on local sensor measurements

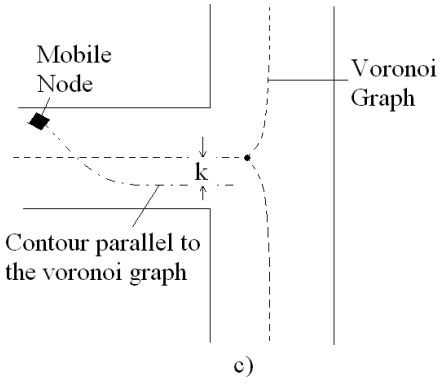

Fig. 6. Contour parallel to the Voronoi Graph.

location x in a workspace *W*, populated by convex obstacles C_1, .., C_n. The distance between a point x and an obstacle is defined as the distance between x and c_0, the nearest point on the obstacle to x.

This is represented by

$$d_i(x) = \min_{c_0 \in C_i} \| x - c_0 \| \tag{1}$$

The "gradient" of this function is defined as

$$\nabla d_i(x) = \frac{x - c_0}{\| x - c_0 \|} \tag{2}$$

Assuming a coordinate system (y, λ) such that λ points along the tangent of the Voronoi Graph edge and the y

coordinate spans Y, the hyperplane orthogonal to the VG, G(x) is defined as

$$G(x) = d_1(y, \lambda) - d_2(y, \lambda) \tag{3}$$

The control law ([3]) is given by:

$$\dot{x} = \alpha Null(\nabla G(x)) + \beta \nabla G(x)\dagger \tag{4}$$

where α, β are scalar gains, $Null(\nabla G(x))$ is the null space of $\nabla G(x)$, and $\nabla G(x)\dagger$ denotes the Penrose pseudo-inverse of $\nabla G(x)$.

This law basically solves the roots of the equation $\nabla G(x) = 0$. For the example shown in 5, this results in balancing vectors $d_1(x)$ and $d_2(x)$ thus resulting in a path that is equidistant from both the obstacles. Using this control law, a robot dropped anywhere in a 2D environment will move towards the nearest point on the Voronoi line as shown in Figure 5. Once on the Voronoi line it will stay on the line since the term orthogonal to the Voronoi line goes to zero.

We make the observation that this law allows a robot to trace arbitrary contours that are parallel to the VG by rewriting G as $G(x) = d_1(x) - d_2(x) - K$ where K is a constant that determines the distance of the new contour from the VG. In other words, by setting the appropriate value of K a robot can trace the contour shown in Figure 6. Any such contour which is the level set of a fixed offset from the VG can be traced.

The control law has interesting features:

- It is based on local sensor measurement
- It is memory less and does not need to much memory to implement
- It does not attempt to estimate the field it is measuring
- It is provably convergent
- Its path is critically damped i.e. the robot does not overshoot the VG contour and oscillate about it
- Lastly, it is designed to trace the contour of interest

V. Our Approach

We adapt the control law given in Equation 4 to perform contour tracing in a sensor field. A key difference between our setting and the environment described in the previous section, is that we deploy nodes at random discrete points. Although we do not have *walls* in our setting which the robot can range to, there is a continuous scalar field which can be sensed at discrete points where stationary nodes are deployed. This is the field whose level set we want the robot to find and trace.

We recast the function G from the previous section as follows:

G(x) = (sensor reading at the mobile node) - (threshold reading defining the contour to be traced)

This is appropriate since the control law tries to drive $G(x)$ to 0. When the algorithm converges, it results in equalizing the sensor reading at the mobile node and the desired level, which is our objective.

The mobile node collects the sensed values of the scalar field from all its neighbors within communication range along with their location. Using these data, it identifies two neighbors, one with the highest gain increase ratio and one with highest gain decrease ratio (with respect to distance from itself). Having identified these two nodes, two unit gradient vectors $\nabla d_1(x)$ and $\nabla d_2(x)$ are defined in the direction of each of the two chosen neighbors

$$\nabla G(x) = \nabla d_1(x) - \nabla d_2(x) \tag{5}$$

These two vectors are used to compute $G(x)$ (using equation 5) and $\nabla G(x)$(using equation 5). Both $G(x)$ and $\nabla G(x)$ are then used to compute $\dot{x}$(using equation 4. This is the velocity that is commanded to the actuators. In our system, it is assumed that there is a low-level controller on board the mobile node which is able to achieve the velocity commanded by the algorithm above. In this paper, we do not concern ourselves with the details of how that is done.

Algorithm 1 : Algorithm for the Mobile Node

loop
  for $i \in$ {Current neighbor list of mobile node} **do**
    $loc[i] \Leftarrow$ Location of i
    $sense[i] \Leftarrow$ Sensor reading of i
  end for
  $n1 \Leftarrow$ Node with best gain increase gradient {using $loc[]$ and $sense[]$}
  $n2 \Leftarrow$ Node with best gain decrease gradient {using $loc[]$ and $sense[]$}
  Compute $d_1(x)$ and $d_2(x)$
  Compute $\nabla G(x)$ and $G(x)$
  Compute $\dot{x}$
  Command $\dot{x}$ to actuators
end loop

As a baseline, we also implemented simple gradient descent based on local query. In this technique, the mobile node queries its neighbors for their locations and sensor values. It then chooses the maximum intensity gradient towards the boundary. It then compares this value with the previously obtained gradient (if any). It then moves one unit towards the contour in the direction of the node with the highest gradient. This process continues until the boundary is reached or the mobile node cannot move any further. We implemented this simple gradient descent as a baseline so we could compare our algorithm with it.

VI. Simulation

The simulation setup consists of a set of nodes placed uniformly randomly in an area of 100x100 units. The transmission range of all nodes is arbitrarily assumed to be 20 units. The mobile node is placed at a random location. We vary the number of stationary nodes placed to study the behavior of the algorithm for different average degree values. Prior research has shown that 6 is a good number for the average degree for the sensor network to be connected [16].

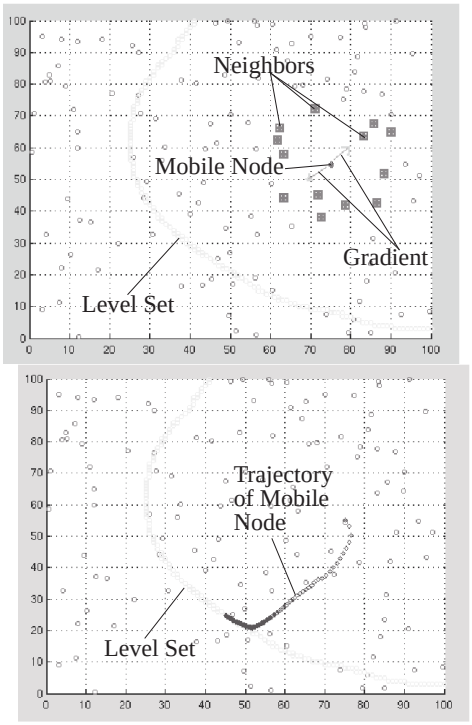

Fig. 7. Mobile Node communicating with two neighbors with best gradients. The mobile node is the filled circle, the immediate neighbors are filled squares and the circles are the stationary nodes.

We study the behavior of our algorithm for networks varying the average degree from 6 to 12.

The simulation setup allows us to randomly instantiate networks of varying densities. It also allows us to study the performance of the algorithm when error exists in various parameters - sensor measurement error, localization error and actuation error at the mobile node. A trial consists of initializing a particular static network, an arbitrarily initialized mobile node, a particular setting for sensor noise, and the execution of the control law by the mobile node. For a particular deployment we perform 100 trials differing only in the initial placement of the mobile node and the stochastic noise. We experimented with a number deployments which differed in the density of the network (as measure by the node degree). Each deployment was tested with and without sensor noise. We also varied the behavior of the phenomenon being sensed. We attempted to capture any variation in the behavior of the algorithm for gradients whose fields had different types of decay. Experiments were performed with five different types: Linear, Square, Logarithmic, Square Root and Gaussian.

VII. Simulation Results

We measure the performance of our algorithm using two metrics. The first is the *percentage of success*. This is defined as the percentage of the trials in which the node successfully found the contour. This metric is a rough guide to the feasibility of our approach. We would like to note that most failure was due to boundary conditions and lack of static neighbors.

The second metric is indicative of the quality of the solution. For each successful trial (where the contour is found), we measure the *ratio of the distance traveled by the mobile node to the minimum straight line distance between the mobile node's initial location and the contour*. This metric has a lower-bound of 1. Values close to 1 are good, and values significantly higher than 1 indicate poor performance (i.e. long paths).

Using these metrics we examine the impact of two parameters on our algorithm. These are the independent variables in the experimental trials. The first is *static network density* (as measured by the number of neighbors of each node), and the second is *error*.

A. Success Percentage

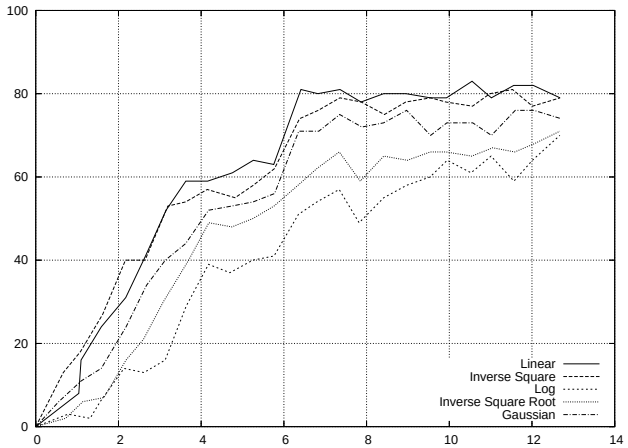

Fig. 8. Percentage of completion vs. Node Degree (for the simple gradient descent technique)

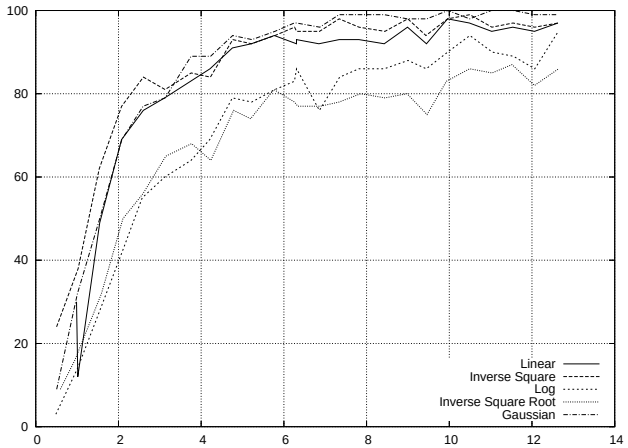

Fig. 9. Percentage of Completion vs. Node Degree (Our Algorithm)

Figure 9 shows the performance of our algorithm by plotting percentage of successful trials as a function of network density for varying levels of sensor noise. Figure 8 measures the performance of the simple gradient descent algorithm using the same metric. As mentioned earlier, we studied five different types of decay functions for the fading of the

scalar phenomenon - Linear, Inverse Square, Log, Inverse square root and Gaussian. The performance did not vary significantly across the different functions. Hence, for future results we pick inverse square decay as a representative function. It is to be noted that our algorithm has more than 80% success rate for network densities greater than 6 and gets better as the network density goes higher.

B. Effect of sensor error on algorithm success

Fig. 10 shows the effect of sensor noise on the baseline algorithm. From Fig. 10 it is clear that the algorithm's performance deteriorates rapidly when the sensors are erroneous. The error introduced is zero mean gaussian noise.

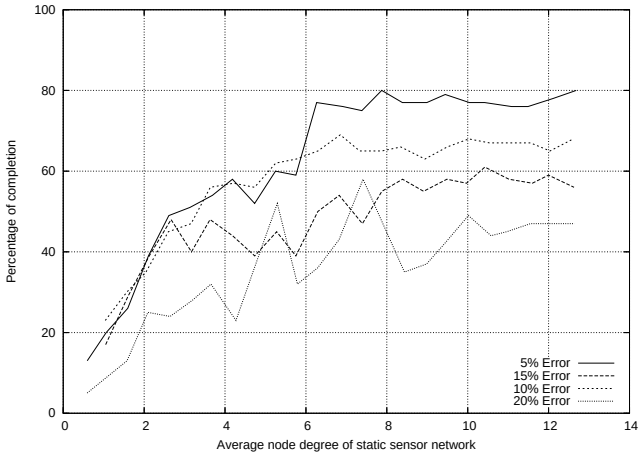

Fig. 10. Effect of sensor error on success percentage (Simple gradient descent algorithm)

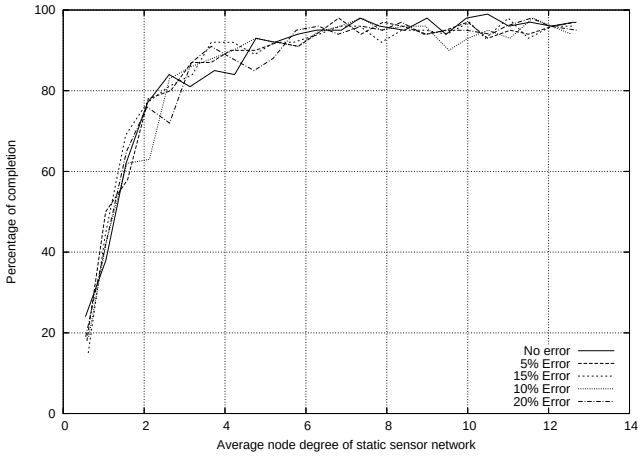

Fig. 11. Effect of sensor error on success percentage(Our algorithm)

However, our algorithm is fairly resilient to sensor noise. Even in the presence of the same levels of noise, its performance is comparable to the noiseless case (Fig. 11).

C. Optimality of successful routes

Our second metric of performance is the optimality of route taken to the contour.

The performance of the simple gradient descent algorithm is suboptimal in comparison with our algorithm (Fig. 12 13).

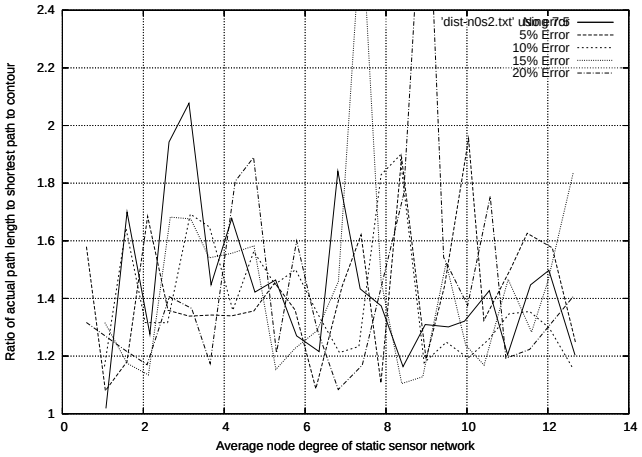

Fig. 12. Optimality of route for trials with sensor error (simple gradient descent algorithm)

There is a lot of variance in the performance of the baseline algorithm. However,our algorithm nicely converges to optimality as the static network density increases (Fig. 13).

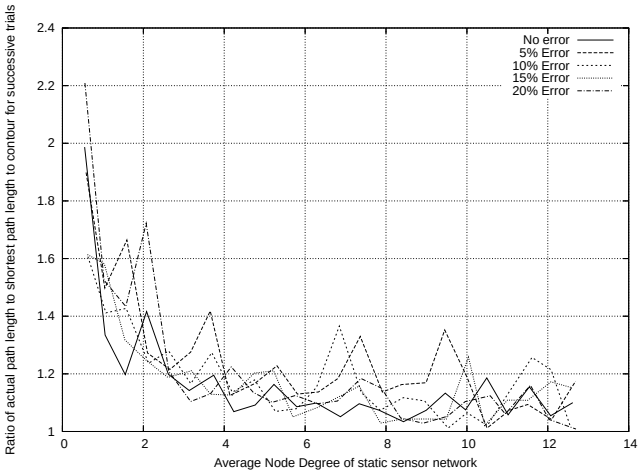

Fig. 13. Optimality of route for trials with sensor error(Our algorithm)

In conclusion, our algorithm is fairly resilient against sensor error and has a graceful loss of success with increase in error.

Based on our extensive simulations, we draw the following empirical conclusions about our algorithm:

1) **Saturation:** The general behavior of our algorithm is such that the percentage of successful trials increases and then saturates with increased network density irrespective of sensor noise. This saturation occurs at relatively low network densities (node degree approximately 6-8). For a network of n nodes, a node degree of 6 is needed to preserve connectivity [16]. For the network sizes we experimented with, the saturation we observe is at values less than or equal to those needed for connectivity. Given that without assured connectivity, the sensor network is not of much use for other applications, this seems reasonably practical.
2) **Sensitivity to Noise:** The simple gradient descent algorithm is extremely sensitive to noise (Fig-

ures[8](a)[8](b)). This is because the mobile node bases its decisions on the readings of one sensor. Sensing error thus dramatically skews the boundary detection mechanism. Our algorithm is nowhere near as sensitive to noise (Figure 11).

We measured the effectiveness of our algorithm when it terminates successfully. We measure the ratio of the distance traveled by the mobile node to the straight line distance from the start point to the nearest point on the boundary. The optimality gradually increases with increase in network density(Fig. 13.

3) **Type of Gradient:** We observe that the algorithm is independent of the type of decay experienced by the scalar field (Fig. 9). This advocates in favor of the algorithm as it is very hard to determine the exact model of decay of the scalar field in advance.

The results regarding solution quality (Fig. 13) follow a similar trend to the percentage completion graphs discussed earlier. However, they seem to follow two distinct but expected trends.

1) **Dependency on node density**: For node degrees greater than six, the ratio of actual distance traveled to straight line distance saturates to approximately 1.1, which is very close to the best possible value (1). Thus our approach is near optimal if the sensor network has a reasonable average node degree.
2) **Dependency on sensor noise:** As the percentage of sensor noise increases in the network, the ratio of distance traveled also increases. This is coupled with an increase in the variance of this ratio. This indicates that in high noise scenarios our scheme might perform sub optimally, causing the mobile node to travel a longer distance than needed to reach the boundary.

VIII. Experimental Results

Our experimental setup uses the Robomote [4](Figure 14) on a table-top testbed [13](Figure 14). The testbed is a 4ft by 10ft table. The Robomote is a small mobile robot which interfaces with the mote [8]. Components have been written in TinyOS such that only the mote needs to be programmed to control the Robomote. We use a laptop connected to a mote via serial port as a base-station. We interface through java and matlab to simulate parts of the experiment.

The first hardware experiment tests the validity of our algorithm in the presence of odometry error. We simulated the static node deployment and the scalar field in matlab. We simulated 20 nodes deployed in 4ft by 8 ft area.
Based on the current robomote location, its local sensors are queried for their values(assuming a radio range of 2 ft). These readings are then used along with the locations of the sensors to compute the control law. This determines the next destination for the robomote. It is observed that the robomote converges to the boundary with good consistency. Listed below are the distance traveled and the optimal distance of travel to the boundary for five cases when the robomote converged to the boundary. Although the ratio of distance traveled to the optimal distance is higher than that obtained by simulation, the feasibility of our algorithm is evident.

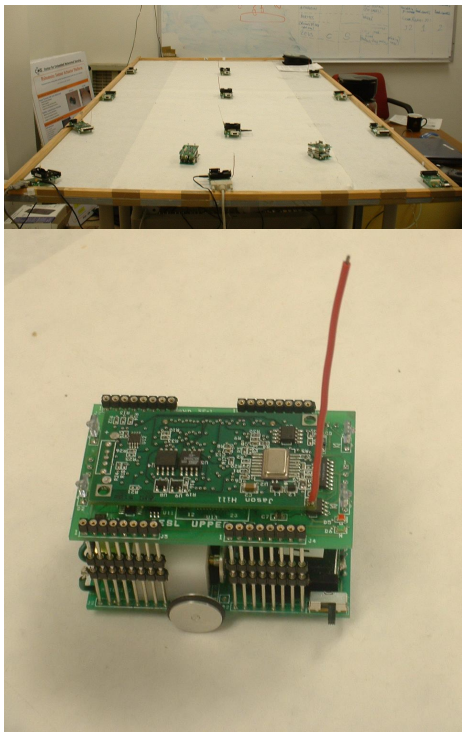

Fig. 14. (a)The Testbed (b) The Robomote

Optimal Distance	*Traveled Distance*	*Ratio*
5.5	8.3	1.509
5	6.1	1.22
5	7.6	1.52
3	5.5	1.83
5	7.8	1.56

We are in the process of performing experiments with sensor boards and using light as the scalar field.

IX. Conclusions

We described an algorithm which detects and traces a contour of a scalar field. A network of sensor nodes is deployed in a planar environment along with a mobile node which can both sense and move. Our algorithm causes the robotic node to move in a way which positions it on the desired contour, and keeps it there. The algorithm uses local communication between the robotic node and its immediate neighbors. Simulation results indicate that the paths generated by our algorithm are near-optimal in terms of their lengths. Simulation results also indicate that the contour is found reliably. Comparison with a simple gradient following algorithm indicates that our algorithm is significantly more robust to sensing noise. The algorithm is demonstrated to work reliably at network densities where each node has 6 or more neighbors.

Preliminary experimental results show the feasibility of the algorithm. Future work will include an analysis of the energy consumption and study of convergence properties

of the modified control law. One of the assumptions of our algorithm is that the decay function is monotonically decreasing. However, this might not be the case in reality. This introduces the possibility of local minima in the sensing which might prevent the mobile node from driving to the contour. It is also possible for the mobile node to drive to a location with no neighbors. This would result in breaking of the algorithm.

We are working on improving our algorithm to avoid both the above cases.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants CNS-0540420, CNS-0520305, and CCF-0120778.

REFERENCES

[1] N. Bulusu, J. Heidemann, D. Estrin, and T. Tran. Self-configuring localization systems: Design and experimental evaluation. In *ACM Transactions on Embedded Computing Systems (ACM TECS), Special issue on networked embedded systems*, 2003.

[2] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. In *IEEE Sensor Network Protocols and Applications Workshop*, May 2003.

[3] H. Choset, I. Konukseven, and A. Rizzi. Sensor based planing: A control law for generating the generalized voronoi graph. In *IEEE International Conference in Advanced Robotics*, 1997.

[4] K. Dantu, M. H. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme. Robomote: Enabling mobility in sensor networks. In *IEEE/ACM Fourth International Conference on Information Processing in Sensor Networks (IPSN-SPOTS)*, pages 404–409. IEEE, Apr 2005.

[5] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. In *IEEE Pervasive Computing*, pages 1(1), pp. 59–69, 2002.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next generation challenges: Scalable coordination in sensor networks. In *ACM Mobicom*, 1999.

[7] D. Ganesan, S. Ratnaswamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. In *2nd workshop on Hot Topics in Networks (HotNets-II)*, 2003.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.

[9] J. Liu, P. Cheung, L. Guibas, and F. Zhao. A dual-space approach to tracking and sensor management in wireless sensor networks. In *ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[10] D. R. M. Schwager, J. McLurkin. Distributed coverage control with sensory feedback for networked robots. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[11] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *Second International Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[12] G. Pottie and W. Kaiser. Embedding the internet: wireless integrated network sensors. In *Communications of the ACM*, pages 43(5):51–51, May 2000.

[13] M. Rahimi, R. Mediratta, K. Dantu, and G. Sukhatme. A testbed for experiments with sensor/actuator networks. In *USC/IRIS Tech Report IRIS-02-417*, 2002.

[14] G. S. Sukhatme, D. Estrin, D. Caron, M. J. Mataric, and A. Requicha. Proposed approach for combining distributed sensing, robotic sampling, and offline analysis for in situ marine monitoring. In *Proceedings of Advanced Environmental and Chemical Sensing Technology - SPIE*, November 2000.

[15] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM Press.

[16] F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. In *Wireless Networks*, 2003.