Precision Localization in Monte Carlo Sensor Networks

Thomas C. Henderson
School of Computing
University of Utah
Salt Lake City, UT, USA
Email: tch@cs.utah.edu

Edward Grant, Kyle Luthy,
Leonardo Mattos, and Matt Craver
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, USA
Email:egrant@eos.ncsu.edu

***Abstract*— We have proposed Monte Carlo Sensor Networks as a method to solve certain sensor queries in the presence of noise and partial information. In that work we used very coarse position estimates for enemy agents. Here we propose methods to (1) improve the posterior probability estimates by using a more precise analysis of the sensor range geometry, and (2) help select advantageous locations to place the sensor nodes.**

I. Introduction

Biswas et al. [1] introduced a probabilistic approach to inference with limited information in sensor networks. They represented the sensor network as a Bayesian network and performed approximate inference using Markov Chain Monte Carlo (MCMC). The goal is to robustly answer queries even under noisy or partial information scenarios. We have proposed an alternative method based on simple Monte Carlo (MC) estimation[2]; our method allows a distributed algorithm and pre-computation of probabilities. In this paper, we examine the improvement gained by doing more precise analysis of the geometry in enemy location assignment as well as determining advantageous locations to put sensor nodes.

Many advances have been made in sensor network technology and algorithms in the last few years. See [3] for an overview of the state of the art. Work has been done on: architecture [4], systems and security [5], [6], [7], and applications [8]. Our own work has focused on the creation of an information field useful to mobile agents, human or machine, that accomplish tasks based on the information provided by the sensor network [9], [10], [11], [12], [13].

Some drawbacks of sensor networks include the need to conserve power and not run all the nodes all the time (partial data), and sensors are noisy (sometimes return the wrong value). This motivates a statistical approach to decision making. Biswas et al. [1] introduced an interesting problem in the context of sensor networks, as well as an MCMC solution. We present improvements to our Monte Carlo method.

A. The Problem

Suppose there is a 2D area of interest (we'll consider the unit square), and a friendly agent located at a fixed position, LF, in the area. In addition, there are m sensor nodes located throughout the area (these sensor nodes report the presence or absence of enemy agents within some fixed range). Finally, n enemy agents are placed in the area; each enemy agent's coordinates are chosen by independently sampling a uniform distribution for x and y (see Figure 1 for an example).

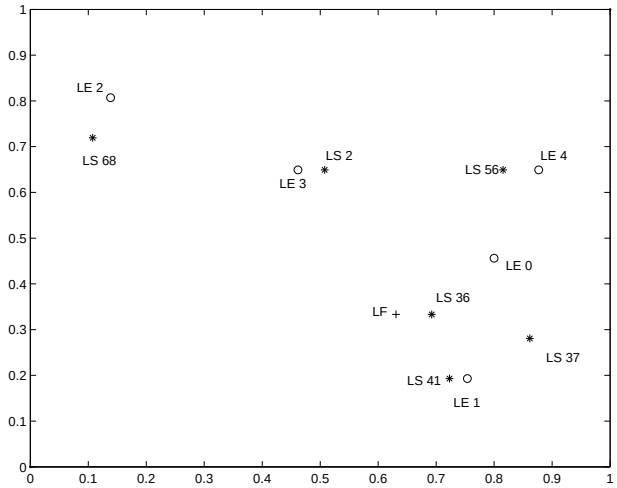

Fig. 1. Example Layout of friendly agent ($+$), enemy agents, LE ($\circ$) and sensors, LS ($*$) (from Biswas et al.)

Query: Given a set of sensor responses, R_i (each declares there is or is not an enemy within its range), and a set of probabilities of the correctness of the responses, what is the probability, $P(S \mid R_i)$, that the friendly agent is surrounded by the enemy agents

(surrounded means that it is within the convex hull of the enemy locations).

See Biswas et al.[1] and Henderson et al.[2] for details on the problem and methods to solve it. Both use approximation techniques since the combinatorial complexity is very high for the exact solution.

II. Monte Carlo Sensor Networks

A. Sensor Model

Let the ground truth be defined as:

$$\hat{d}_i = \begin{cases} 0 & \text{if there's no enemy in range of sensor i} \\ 1 & \text{otherwise.} \end{cases}$$

Then, let:

$$d_i = \begin{cases} \hat{d}_i & \text{with probability } \rho \\ \neg\hat{d}_i & \text{with probability } 1-\rho \end{cases}$$

B. m-Sensor Case

There are 2^m terms in the probability calculation. Let $\mathbf{d} = [d_1, d_1, \ldots, d_m]$ be the set of sensor node assertions about the presence of enemies. We must consider all combinations of these being true or false. This is characterized by a vector $\mathbf{b} = [b_1, b_1, \ldots, b_m]$, where $b_i = 1$ means that d_i is true, else false. and the contribution of each to the total probability of being surrounded:

$$Prob(S \mid R) = \sum_{\mathbf{b} \in \mathcal{P}(\{0,1\}^m)} Prob(\mathbf{b}) Prob(S \mid \mathbf{b})$$

For example, if we assume that all the sensor nodes report that there is an enemy present, and the first $m-1$ sensor responses are correct but that sensor node m is wrong, then we compute that term in the above summation as:

$$\rho^{m-1}(1-\rho) Prob(S|Positions)$$

where $Positions$ indicates that Monte Carlo is run assuming enemies in positions indicated by the d_i's.

III. Precision Localization

There are a couple of issues we address in this paper: (1) enemy placement for posterior probability of being surrounded calculation, and (2) sensor node placement to increase the probability of making a correct assessment. First, if a sensor reports an enemy in its range, the enemy is assumed to be at the sensor location in order to make the Monte Carlo simulation; more generally, if multiple sensors with non-empty common overlapping range report an enemy, one location in that intersection area is used. We propose to refine that by sampling from the entire intersection area.

Second, in the original work, sensor placement was given as part of the statement of the problem. Here, we propose to compare sensor placement using two techniques: random, and sensor placement that puts sensor nodes in the highest impact region first.

A. Enemy Placement

The Monte Carlo simulation incorporates the sensor reports by assigning a location to a reported enemy. We propose two methods to determine this location:

1) Find the mean point of a certain polygon in the intersection of the circles.
2) Randomly sample (uniformly) locations in the intersection of the sensor clique reporting the enemy. This is important because some regions of the intersection area may not result in surrounding the friendly agent while others do. A large enough sample should capture this effect, not only for individual cliques, but for all combinations of them.

A point in the intersection of a set sensor ranges is found as follows (see Figure 2). First, each sensor's range is represented by a circle of radius r. We assume that all sensors have the same range. Cliques of circles are found where each clique is a maximal set of circles with non-empty intersection. Next, the set of intersection points of all the circles taken pairwise is determined (smaller empty and filled circles). Any point farther than r from any circle center in the clique is then deleted (smaller empty circles). The remaining set of points (P_1, P_2, P_3) are formed into

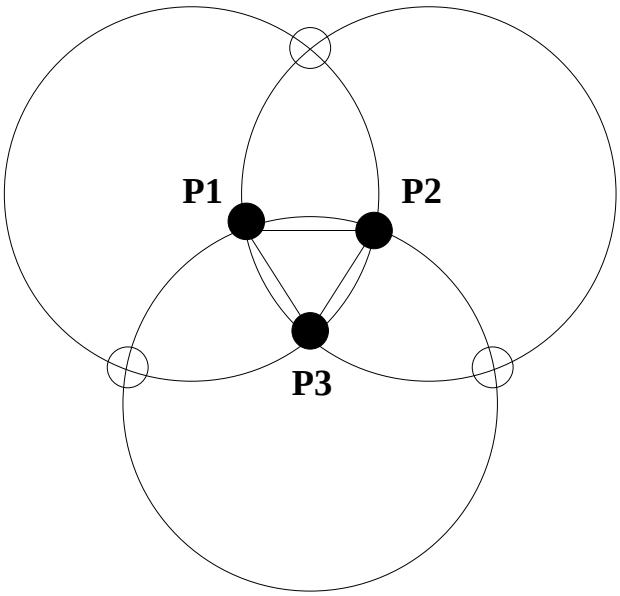

Fig. 2. Geometry of Sensor Range Intersection.

a convex polygon (e.g., using convhull in Matlab!); this polygon lies within the intersection area of the circles. The mean of the points (PM) on the polygon lies in the interior of the polygon and is used as the representative point.

The other alternative is to randomly generate points until one falls within distance r of all the circles in the clique. This adds to the complexity, but gives better results since the deterministic method may produce a point that always contributes to surrounding the friendly agent, or which never does.

Another interesting measure to have is the area of the intersection of the circle clique. This may be used to determine various likelihoods; e.g., if this region is small enough, then the computed polygon vertex mean will do as the enemy location.

This area is computed by adding the area of the polygon to the areas of all the regions lying between the polygon edges and the circumscribing circle of that edge (see Figure 3). This is found by subtracting

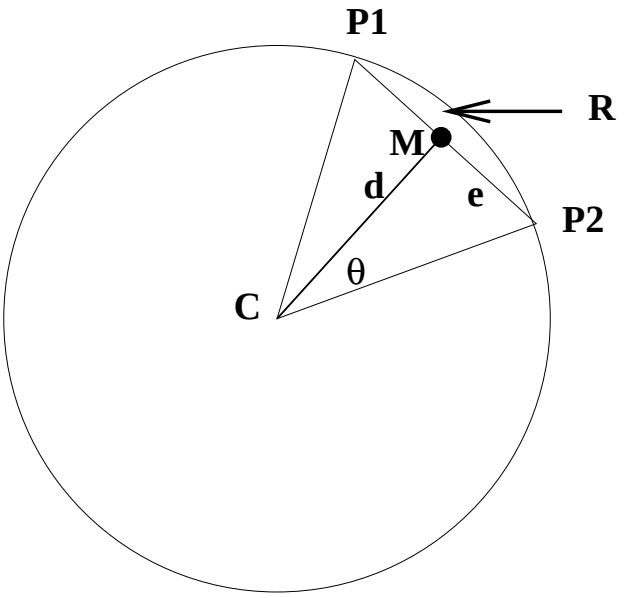

Fig. 3. Area of Circles' Intersection.

the area of the triangle formed by the center of the circle and the two polygon vertexes arising from that circle from the area of that circle's wedge, i.e., $(\theta/2) * r^2 - e * d$ where θ is the angle made from point 1, P_1, to point 2, P_2, on the circle's circumference, C is the center of the circle, and M is the midpoint between P_1 and P_2.

B. Optimal Sensor Placement

In this problem, we are given a sensor budget, and allowed to place them as we see fit. What is the best placement strategy to improve the posterior probability calculation? We compare two strategies for sensor placement: (1) random placement (uniform distribution), and (2) highest impact region first.

1) Random Sensor Placement: The experimental layout is as follows: a friendly agent is placed at 0.75,0.25 in the unit square. We assume five enemy agents, and two scenarios are explored: the friendly agent is surrounded (S), and it is not surrounded (NS). Figures 4 and 5 show the locations of the enemies used in the simulations for cases, S, and NS, respectively.

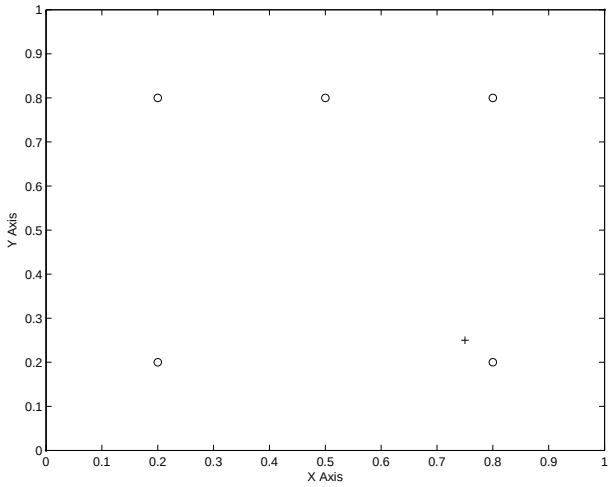

Fig. 4. Layout for NS (non-surrounded case).

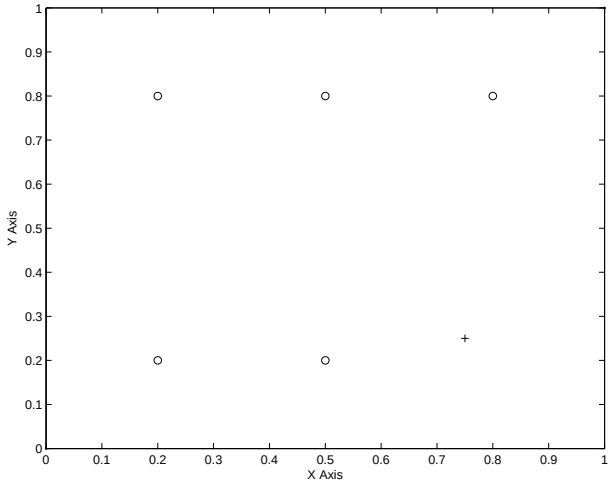

Fig. 5. Layout for NS (non-surrounded case).

For each of S and NS, 10 random sensor locations (for m sensors) are selected. Then, the MCSN method is used to determine a probability that the friendly agent is surrounded.

2) Highest Impact Region First: This approach divides the unit square into four regions with respect

to the friendly agent: up and left (UL), up and right (UR), down and left (DL), and down and right (DR). We select sensor placement locations by choosing in each region $\frac{(1-area(region)}{3}$ percent of the time. For example, if $LF = [0.75; 0.25]$, then $area(UL) = \frac{9}{16}$, $area(UR) = area(DL) = \frac{3}{16}$, and $area(DR) = \frac{1}{16}$. Therefore, the sensor locations are selected with probabilities: $prob(UL) = 0.1458$, $prob(UR) = prob(DL) = 0.2708$, and $prob(DR) = 0.3125$.

The simulation is performed as follows: a friendly agent is placed at 0.75,0.25 in the unit square. We assume five enemy agents, and two scenarios are explored: the friendly agent is surrounded (S), and it is not surrounded (NS). For each of S and NS cases, 10 sensor locations (for m sensors) are selected as described by the probability of selecting in each region. Then, the MCSN method is used to determine a probability that the friendly agent is surrounded.

Table 1 gives the mean posterior found for the S case random sensor placement and two versions of HIRF (all variances are less than 10^{-4}; in one, sensors are placed randomly in the selected areas, while in the other, sensors are placed so as to maximize the area covered in the rectangle where they are placed.

m	Random	HIRF, random	HIRF, max
0	0.27	same	same
1	0.25	0.48	0.51
2	0.31	0.51	0.50
3	0.30	0.53	0.52
4	0.33	0.58	0.67
5	0.22	0.68	0.70

Table 1. Random vs. HIRF Placement Surrounded Case

Table 2 gives the mean posterior found for the NS case random sensor placement and two versions of HIRF (all variances are less than 10^{-4}); in one, sensors are placed randomly in the selected areas, while in the other, sensors are placed so as to maximize the area covered in the rectangle where they are placed.

m	Random	HIRF, random	HIRF, max
0	0.27	same	same
1	0.27	0.36	0.39
2	0.28	0.40	0.39
3	0.22	0.39	0.38
4	0.26	0.57	0.54
5	0.26	0.58	0.50

Table 2. Random vs. HIRF Placement Not Surrounded Case

IV. Physical Experiments

The original set of Monte Carlo network experiments was carried out to better understand the physical nature of acoustic detection systems, and to derive metrics related to their probability; i.e., could they produce a correct report of enemy presence[2]. That set of experiments concluded that the correctness probability was approximately 70%. The physical experimentation indicated that there is uncertainty at the range boundary of acoustic sensors. Since range is directly related to a set sound level threshold, variations are due to sensor and emitter circuitry. However, these inconsistencies were found useful to test prediction in the presence of noise in the systems.

Physical testing was carried out in the autonomous mobile robot test-bed in the Center for Robotics and Intelligent Machines (CRIM) at North Carolina State University [14]. The autonomous mobile robot used, the EvBots, are a generic platform, one that is used widely for evolutionary algorithm and distributed sensor experimentation. To test the Monte Carlo sensor network at the heart of this paper a new microphone circuit was constructed. This new acoustic sensor; provides data for hit or miss decision making within a specified range. For this set of Monte Carlo network experiments the EvBots served as stationary sensor nodes. The enemies in this case are PC speakers that emit a 1 KHz tone.

The physical experiments were carried out to verify the simulated sensor placement strategy described earlier in this paper, by scaling the environment to the unit square. The enemy nodes were initially placed as shown in Figure 5 (NS) and then later as shown in Figure 4 (S). For each placement the 4 sensors were incrementally placed within the regions detailed in the simulation (e.g., 1 sensor in DR, then 1 in DR and 1 in UR, etc.). This was done for both the case where the sensor was centered in the region, and later for the case where the sensor was placed randomly within the region being tested. This is shown for the centered case in Figure 6 with the physical setup shown in Figure 7.

In a similar manner to that of the simulation, the range of the sensors was set to a radius of 0.2, which translates into an expected 70% correctness factor. From this, a fringe threshold of 320 was empirically derived from the sensor's measured analog to digital values. Upon performing the experiment with 5 enemies and 4 sensors it was found that out of the 400 measurements recorded, only 4 fell below the 320 level, indicating that no enemy was present; when in

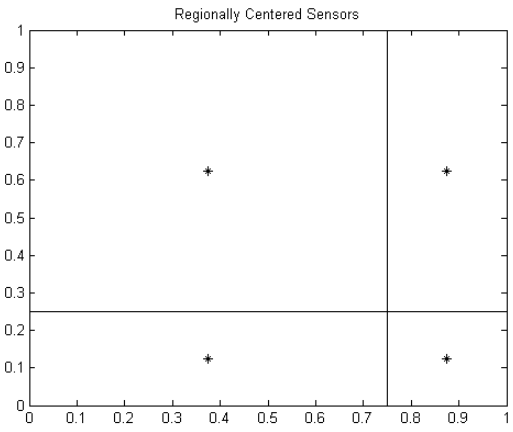

Fig. 6. Regionally Centered Sensor Nodes (friendly located at intersection of axes).

fact there was. These results imply that the friendly was always surrounded.

These preliminary Monte Carlo network tests highlight issues relating to moving between simulated worlds and real worlds, i.e., determining how to accurately set a sensor's range. The range can be set in one of three ways, although each presents its own problem.

1) Setting the sensor threshold given a sound source of a pre-determined intensity level. This was the technique used in these preliminary experiments. However, the derived sensor threshold does not prevent the sound from propagating past the desired range. Where multiple enemies are present there is an increase in the perceived sound level at a given point, due to the additive effects of noise.
2) Adjusting the gain on the sensor's amplifier. Increasing the gain of the amplifier also amplified ambient noise which, if high enough, resulted in false readings. Filtering was carried out to reduce these effects.
3) Adjusting the sound level emitted by the enemy. This adjustment was analogous to adjusting the sensor gain, but performed on the enemy rather than the sensor. Neither of these methods (2) or (3) adequately addressed the additive effects produced by multiple enemy agents.

In order to establish a base line for noise levels within the test-bed sound levels were sampled using varying numbers of enemies, see Figure 8. Sound samples were recorded using a B&K sound level meter, with the sound level from a single enemy being measured at 63.9dBC (ambient noise is 57.1dBC).

Fig. 7. Test Environment with the friendly wolf, sensor nodes, and speaker enemies.

Figure 8 shows the sound levels measured throughout the environment for the NS case. As can be seen from Figure 8 there are few sites within the test-bed where the sound level dips below this 63.9dBC.

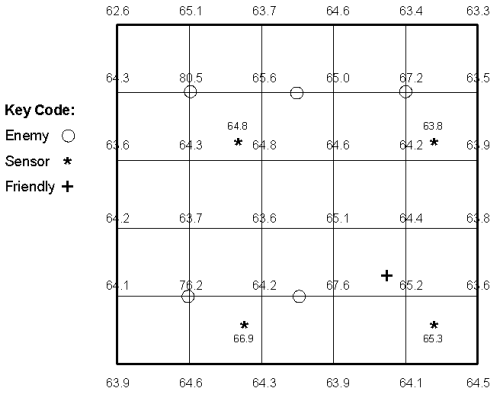

Fig. 8. Sound Level Mapping in the Test-Bed.

It was apparent when examining the recorded data that there were distinct variations in readings between the surrounded and not surrounded cases. A new threshold can be derived to fit this data, but will serve as a poor metric as it would only be applicable to this particular test scenario. Different configurations will

require new thresholds. Once again, this is primarily a problem with the effective range of the sensors and the intensity of the sound sources.

For future experimentation, the additive sound level problem will be addressed. A basic test can be arranged by placing enemies in a pattern similar to the experiments conducted here. Lowering the sound intensity (and thus the range) is akin to placing the enemies further apart. Doing this will reduce the additive effects derived from speaker emissions. More sensors will be required in order to detect enemies.

For more complex cases, where enemies are closer together, a more complex reasoning strategy will be required for the sensor(s). The sensor has a set range based on a single enemy, but it may in fact be hearing two. In this specific case the sensor's range can no longer be assumed static. This condition will lead to either or both sensors lying outside the defined range, yet still alerting the sensor to their presence.

V. Discussion and Conclusions

We have shown that both precision localization and sensor placement lead to improved estimates of the posterior probability of being surrounded.

In future work, we intend to explore these issues in a 100-node sensor network testbed now under construction; the EvBots will provide the mobile agent platforms for use as friendly and enemy agents. Moreover, we are exploring the application of sensor networks to snow and avalanche monitoring[15], and in this case the goal might be to surround (e.g., the signal of a buried person) with mobile agents or people – thus, sometimes it may be of interest to ensure that an object of interest is surrounded.

References

[1] R. Biswas, S. Thrun, and L. Guibas, "A probabilistic approach to inference with limited information in sensor networks," in *LCSN*, (Berkeley, CA), April 2004.

[2] T. C. Henderson, B. Erickson, T. Longoria, E. Grant, K. Luthy, L. Mattos, and M. Craver, "Monte carlo sensor networks," uucs-05-001, University of Utah, January 2005.

[3] F. Zhao and L. Guibas, "Preface," in *Proc of IPSN 2003*, (Palo Alto, CA), pp. v–vi, LNCS, April 2003.

[4] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," ece, UC Berkeley, October 2002.

[5] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly resilient, energy effi cient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review*, vol. 1, no. 2, 2002.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks*, vol. 8, pp. 521–534, Sept 2002.

[7] L. Zhang, "Simple protocols, complex behavior," in *Proc. IPAM Large-Scale Communication Networks Workshop*, March 2002.

[8] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor netwroks for habitat monitoring," in *WSNA 2002*, (Atlanta, GA), September 2002.

[9] T. C. Henderson, M. Dekhil, S. Morris, Y. Chen, and W. B. Thompson, "Smart sensor snow," *IEEE Conference on Intelligent Robots and Intelligent Systems*, October 1998.

[10] Y. Chen, "Snets: Smart sensor networks," Master's thesis, University of Utah, Salt Lake City, Utah, December 2000.

[11] Y. Chen and T. C. Henderson, "S-nets: Smart sensor networks," in *Proc International Symp on Experimental Robotics*, (Hawaii), pp. 85–94, Dec 2000.

[12] T. C. Henderson, "Leadership protocol for s-nets," in *Proc Multisensor Fusion and Integration*, (Baden-Baden, Germany), pp. 289–292, August 2001.

[13] T. C. Henderson, J.-C. Park, N. Smith, and R. Wright, "From motes to java stamps: Smart sensor network testbeds," in *Proc of IROS 2003*, (Las Vegas, NV), IEEE, October 2003.

[14] L. Mattos, "The EvBot-II: An enhanced evolutionary robotics platform equipped with integrated sensing for control," Master's thesis, North Carolina State University, Raleigh, NC, May 2003.

[15] T. C. Henderson, E. Grant, K. Luthy, and J. Cintron, "Snow monitoring with sensor networks," in *Proceedings of EMNETS Workshop*, (Tampa, FL), pp. 558–559, November 2004.