Probabilistic Self-Localization for Sensor Networks

Dimitri Marinakis, Gregory Dudek
Centre for Intelligent Machines, McGill University
3480 University St, Montreal, Quebec, Canada H3A 2A7
{dmarinak,dudek}@cim.mcgill.ca

Abstract

This paper describes a technique for the *probabilistic* self-localization of a sensor network based on noisy inter-sensor range data. Our method is based on a number of parallel instances of Markov Chain Monte Carlo (MCMC). By combining estimates drawn from these parallel chains, we build up a representation of the underlying probability distribution function (PDF) for the network pose. Our approach includes sensor data incrementally in order to avoid local minima and is shown to produce meaningful results efficiently. We return a distribution over sensor locations rather than a single maximum likelihood estimate. This can then be used for subsequent exploration and validation.

Introduction

Advances in computing hardware are making the deployment of sensor networks practical for a variety of control and information gathering purposes. The components of such networks can include emplaced motion sensors, emplaced cameras, robots, or even cell phones. For many sensor network applications, self-calibration is seen as a critical issue (Correal & Patwari 2001). The goal is that the network becomes capable of automatically configuring itself to its environment in order to carry out its assigned task.

In this paper we present an algorithm for inferring a *probabilistic* representation of a sensor network pose. This problem of self-localization in sensor networks is recognized as a key requirement for many network applications (Akyildiz *et al.* 2002) and can be considered an important step in the overall goal of developing self-adapting and self-configuring networks.

Our method uses an iterative Markov Chain Monte Carlo (MCMC) based algorithm to estimate a probability distribution function (PDF) for the network pose based on a measurement model for the collected range data. At each iteration, range data from those sensors best localized are incorporated into the algorithm. The final result of our algorithm is a particle representation of the PDF describing the position of each sensor. This stands in contrast to most previous self-localization work, which returns a single maximum likelihood estimation for the location of each sensor.

By determining the PDF, our method indicates the degree of certainty by which each sensor has been localized. This can be useful when dealing with the non-Gaussian and multi-modal distributions that can arise when only range data is available. The estimate of localization certainty can be used by a self-configuring system to determine how additional resources should be used in order to improve localization accuracy; *i.e.* through the use of actuators, mobile components, or the additional deployment of sensors. Even if adaptation is not the goal, the certainty information can aid higher level applications relying on the metric localization data. For example, a tracking application might take into account the certainty of each sensor's position when attempting to reconcile conflicting sensor data regarding a target location.

In addition, by providing a probabilistic estimate of sensor locations, our technique yields information that could be utilized in an effort to recover a more complete representation of the environment. For example, there are techniques for inferring the physical topology of a sensor network (Marinakis, Dudek, & Fleet 2005). This topology information identifies physical inter-sensor connectivity from the point of view of an agent navigating the environment (as opposed to a description of the network's wireless communication connectivity.) Our probabilistic self-localization technique could be used to complement a topology inference algorithm. By considering both the metric and connectivity information of the surrounding environment, further information regarding obstructions and motion corridors could be inferred. For example, two spatially proximal nodes that were not topologically adjacent would suggest a barrier at a particular location, perhaps an interior wall or a river (in an outdoor deployment).

In this work we assume the existence of a powerful network component that is capable of running a computationally sophisticated algorithm. It has been well established that the Gibbs and MCMC algorithms can be computed using distributed computational frameworks (Rosenthal 2000). Therefore, in this paper we restrict our attention to the underlying inference and algorithmic issues related to the geometric constraint problem and for future work leave the task of distributing the algorithm using established methods. Note that the assumption of a hierarchical arrangement of network components based on computational power holds true for

several real world sensor networks, especially in control and data collection systems (Mainwaring *et al.* 2002) (Wang *et al.* 2003). For example, a typical network might contain a number of resource-limited sensors that pass messages using a wireless multi-hop protocol to a more powerful single-board "gateway" computer which communicates to the outside world using a wireless Ethernet connection. In such an example, the gateway computer could periodically collect inter-sensor range data and update its network pose estimate using a version of our algorithm.

Related Work

The majority of self-localization efforts to date have focused on recovering the relative locations of the sensors in situations where GPS locators are too expensive, not available, or otherwise impractical (Capkun, Hamdi, & Hubaux 2001). Localization efforts are usually based on methods for estimating the distances between sensors. Common techniques include the use of received communication signal strength in radio networks (Bulusu, Heidemann, & Estrin 2000), or time-of-arrival ranging using ultrasound (Niculescu & Nath 2003). These techniques typically have limited accuracy and localization algorithms must be able to handle some degree of noise in the range data (Moore *et al.* 2004). Many approaches assume that several of the sensors in the network have a known location and act as beacons or anchor nodes for their neighbors (Patwari *et al.* 2003).

While much of the research conducted on sensor networks is based on developing distributed, computationally efficient algorithms appropriate for networks of low-powered sensor platforms, recently there has been a shift towards more complex approaches incorporating advanced probabilistic techniques and graphical models (Ihler *et al.* 2005) (Paskin, Guestrin, & McFadden 2005). While practical implementation is a concern, many efforts employ computationally sophisticated techniques in the processing of distributed observations. This is especially true for sensor networks made up of vision-based sensors (Rahimi, Dunagan, & Darrell 2004) (Javed *et al.* 2003).

Many self-calibration problems in sensor networks have some relation to the simultaneous localization and mapping (SLAM) problem in mobile robotics. Generally, the solutions for SLAM and related problems employ a complex probabilistic framework and are computationally intensive. Examples of sensor network applications employing SLAM-like techniques include the work of Reklietis *et. al* (Rekleitis, Meger, & Dudek 2005) in their use of an extended Kalman filter for the self-calibration of a hybrid robot/camera-network system, and Coates, who employs a particle filter for distributed state estimation (Coates 2004).

Ihler *et al.* (Ihler *et al.* 2005) also consider the probabilistic sensor network self-localization problem, although employing a different approach than in our work. Conceptually, their method constructs a graphical model that captures the conditional independence of the various range estimates and then solves the graphical model using a stochastic version of belief propagation. Although the computational burden of this approach is distributed among the various network components, there is a heavy communication cost for this distribution in terms of overall bandwidth and message throughput. Additionally, the computational responsibility of each individual sensor is significant and could be challenging to implement without a floating point processor and only limited amounts of RAM. In short, it is unclear whether this technique is practical for certain sensor networks, such as those containing low-powered, resource-limited sensors.

Our self-localization technique is capable of extracting usable localization information from poor quality sensor data. we believe that there are network configurations in which the low bandwidth requirement, scalability, and computational efficiency of this technique will outweigh those of existing solutions.

Problem Description

The problem we are trying to solve is to determine a PDF for the location of each sensor i in a network, given N sensors (including a number of beacons of known position), inter-sensor range data R, and a measurement model L which characterizes the error in the distance estimates. The range data R consists of a (possibly incomplete) matrix where r_{ij} represents the distance estimate between node i and j as measured by node i.

Using the measurement model and the available inter-sensor range data, we can then construct a model that returns the likelihood $p(d_{ij}|R, L)$ for any distance d_{ij} between sensors i and j as determined by sensor i. In the case of a range data estimate taken by sensor i, this is simply $p(d_{ij}|r_{ij}, L)$. However, in the case of missing measurements due to limited communication range, obstacles in the environment, or various other problems, the likelihood can be replaced with a distribution based on prior assumptions. For example, if communication signal strength is used for range estimates, the absence of a signal suggests a distance greater than some minimum value. In this case, a uniform distribution over a range of distance values might be appropriate.

For any network pose X, where x_i gives the position of sensor i, we can evaluate its likelihood given our range data and measurement model:

$$p(X|R, L) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(d_{ij}|R, L)$$

where d_{ij} is the Euclidean distance between sensors i and j as determined by their locations x_i and x_j.

The problem is to provide a usable estimate of the PDF over all possible poses X. In traditional sensor-network self-localization, algorithms generally return an estimate of the maximum likelihood pose of each sensor. However, noise in the distance measurements dictates that any specific location estimate is actually a sample of a PDF, and the certainty of the measurement differs from node to node.

To build a PDF for the pose of a network we must search over a very high dimensional space for areas of high probability. In the case of accurate range data, the problem becomes that of embedding a weighted graph, which has been shown to be NP-hard (Saxe 1979). There can be multiple (or infinite) realizations explaining a specific set of edge lengths. In the version of the problem with non-accurate

edge data, these realizations correspond to different arrangements of the nodes that adequately explain the measurement data but result in substantially different pose estimates. Given perfect sensor fusion these cases should occur less often if there are large quantities of range data collected from a single region. However, in practice, alternate realizations of local groupings of sensors complicate the problem and lead to local minima.

Iterative MCMC

Our approach is to employ MCMC to build up an estimated PDF of the network pose. We run parallel instances of the Markov Chain, each instance, or *macro particle*, representing a single network pose estimate. We combine the estimates from each instance of the Markov Chain in a description of a PDF for the pose of the network.

We use an incremental approach of incorporating sensor information in order to avoid local minima as best as possible. The algorithm maintains a sub-group of nodes whose range data are used for localization. Sensors are incrementally added to the localizing sub-group based on the variance of their position estimates as maintained by each particle or instance of the Markov Chain.

For ease of implementation, we assume the existence of a number of beacon nodes at known locations. However, it is possible to compute the relative sensor positions using our technique without the use of beacon nodes by specifying a preferred reference frame. For example, without loss of generality, one sensor can be forced to the origin, another to the horizontal axis, and a third to the positive vertical direction.

The full description of the algorithm is as follows:

1. *Initialize Algorithm:* Initialize a localizing nodes sub-group $LocNodes$ to contain the beacon nodes of known position. Initialize a non-localizing nodes sub-group $NonLocNodes$ to include all the non-beacon nodes. Initialize M *particles* each maintaining a single estimate $X_m = \{x_{m1} \dots x_{mN}\}$ for each sensor in the network.
2. *Update Particles Using Localized Sensors:* For each particle's estimate of the network pose X_m update the position estimate using MCMC with only the range data collected from those sensors in $LocNodes$. Each particle initializes the Markov Chain with its previous belief of the network pose.
3. *Add to Localizing Sensors:* For each sensor i, compute the variance V_i of its position estimates $\{x_{1i} \dots x_{Mi}\}$ as maintained by each particle. Add the k sensors with the lowest V values to $LocNodes$
4. *Iterate Until Done:* Iterate over steps 2 to 3 until all sensors have been inserted into $LocNodes$. The resulting M network position samples are now used to represent a PDF describing the positions of the sensors.

At each iteration the algorithm maintains a reasonable representation of the PDF given the sensor information incorporated up to that point. The algorithm depends on the manner of drawing representative pose samples, and on the manner of analyzing the variance of those samples in order to assess their accuracy.

In order to draw representative samples of the sensor locations, we maintain a separate Markov Chain for each particle. We construct the Markov Chain using the Metropolis algorithm, a popular method of MCMC sampling. The number of particles used during the algorithm effects its ability to maintain a sufficient representation of the spatial probability distribution function for each sensor. Additional samples of the PDF can be drawn from the Markov Chain maintained by any single particle, however, these samples could share a common bias. Multiple particles, each randomly initialized, are necessary in order to represent the more complex distributions.

Given the current state in the Markov Chain X, specified by the combined location of each sensor in our network, we propose a symmetric transition to a new state X' by altering up to $Q \leq N$ of the sensor positions (in this work we set the value of Q to N). The new pose X' is a stochastic function based on the current pose X. A small amount of normally distributed noise is added to a subset of sensors. This new pose is then accepted or rejected based on the acceptance probability:

$$\alpha = \min\left(1, \frac{p(X'|R,L)}{p(X|R,L)}\right)$$

where R is the observed range data and L is the measurement model.

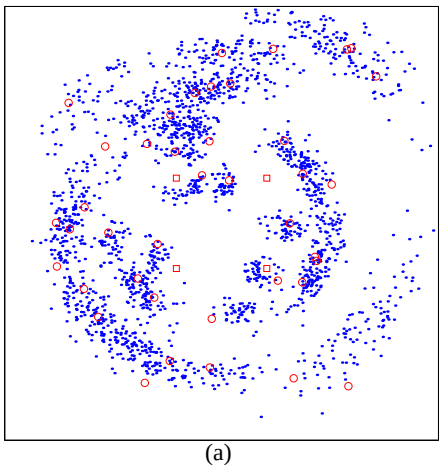
Although technically a new Markov Chain is employed during each iteration of the algorithm, the burn-in time is minimal since each particle maintains its position estimate from the last iteration and uses it to initialize the chain. The additional range data added during each iteration re-shapes the target distribution; in most cases this extra information should help to further concentrate the PDF. The old localization estimate should be close enough to the new peak or peaks in the probability landscape that the MCMC should quickly approach the new steady-state distribution and afterwards provide meaningful localization samples. In this work, we run the Markov-Chain for a fixed number of proposals during each iteration of the algorithm.

A variance metric is used to quantify the certainty of a sensor's position based on the variance of the M position samples provided by each of the particles. The average Euclidean distance of each position estimate from the mean of the distance estimates is used as a metric.

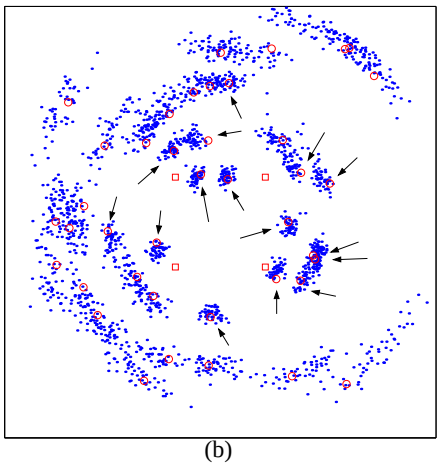
Our approach can be considered an enhanced Gibbs sampler. The distinguishing differences are the iterative inclusion of data, the parallel communicating instantiations of the Markov Chain, and the less restrictive proposals. All these enhancements are designed to help the process converge quickly while avoiding local minima in the domain of this particular geometric constraint problem. In the next section we will compare our approach to a standard Gibbs sampler and further investigate the performance of the algorithm.

Results from Simulation

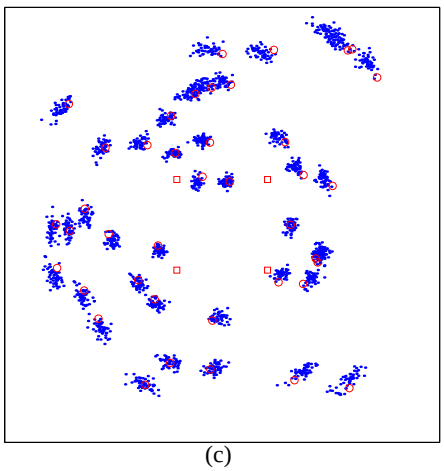
To evaluate the algorithm, a simulation was constructed based on a 2-D grid model of the environment. Sensors and



(a)



(b)



(c)

Figure 1: Example of localization results from a network of 40 sensors using 4 beacons and 50 particles. The squares indicate the beacon nodes and circles mark the true location of the sensors. a) The initial localization estimates using beacon data only. b) Intermediate results incorporating data from the sensors indicated by the arrows. c) The final estimates incorporating all sensor data.

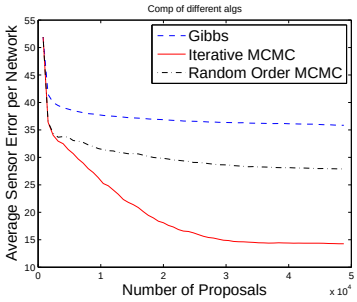


Figure 2: Plot of error as a function of Markov Chain proposals averaged over 20 trials on networks of 4 beacons and 40 randomly distributed sensors. A sensing range of 30 units was used in this comparison. The Random Order MCMC algorithm is the same as the Iterative MCMC algorithm, but includes sensor data in random order.

beacons were distributed on a 100x100 grid. Four beacon nodes were placed near the centre of the grid, one in each quadrant, and the remaining sensors were distributed randomly within the area. Each sensor collected distance estimates to all neighboring sensors within some finite sensing range. Inter-sensor range data were drawn from a normal distribution with a mean equal to the true distance and a standard deviation equal to the square-root of the true distance.

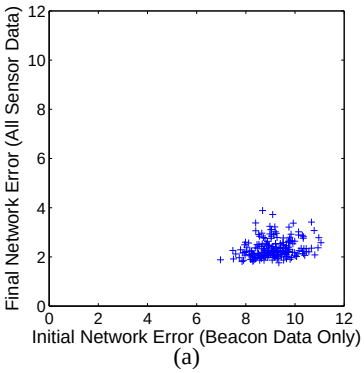
The algorithm was provided with the beacon locations, the beacon range data, the un-localized sensor range data, and an accurate measurement module. The resulting sensor localization estimates were assessed based on the mean Euclidean distance between the particle estimates and the true sensor location:

$$SensorErr_i = \frac{\sum_{m=1}^{M} d(x_{mi}, x_i')}{M}$$

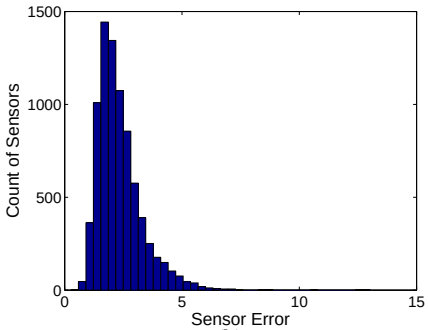
where x_i' is the true location of the sensor and $d(a, b)$ returns the Euclidean distance between a and b. Figure 1 shows an example of the localization algorithm on a 40 sensor network.

Assessment of Algorithm Performance

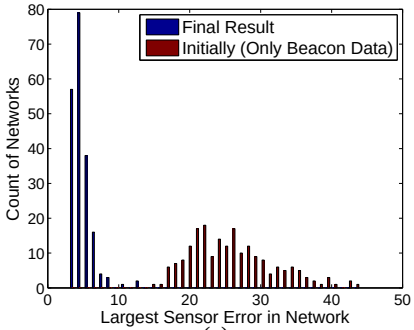
Our method compares favorably to other sampling based approaches (Figure 2). It both converges faster than a standard Gibbs sampler and returns a more accurate result. Performance of the algorithm depends on incorporating the sensor data in a manner that bootstraps each new Markov Chain with likely pose values. For example, consider the extreme case of including the data of an isolated sensor for which there exists no connecting range estimate to any member of the localizing group. Clearly this information is not useful in refining the position of this node relative the the localizing group. Additionally, this type of sensor data can cause the algorithm to become trapped at a local minimum since isolated groups of network components can develop that have settled into a high likelihood arrangement of arbitrary orientation and offset. Ultimately, it can be difficult for the algo-



(a)



(b)



(c)

Figure 3: Results from 200 trials of the algorithm on networks of 4 beacons and 40 randomly distributed sensors using an unlimited sensing range. a) Plot of final error output from the algorithm after including all sensor data as a function of the initial error calculated based on beacon data alone. b) Histogram of final sensor error across all networks (8000 sensors represented). c) Histogram comparing the error for the most poorly localized sensor in each network for the initial estimate using only beacon data and the final result. Vertical and horizontal axis for a) and the horizontal axis for b), and c) displays error in terms of distance units (simulation grid is of size 100x100).

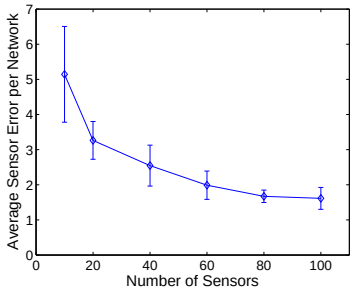


Figure 4: Average network error over 20 trials for different sized networks (confined to a 100x100 grid) with 4 beacon nodes. Error is displayed in terms of distance units and error bars show one standard deviation.

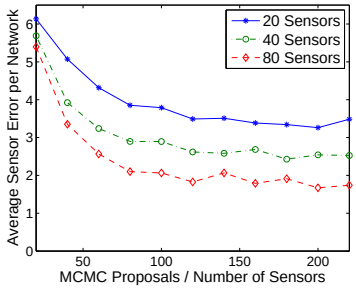


Figure 5: Final maximum sensor error averaged over 20 networks as a function of the total number of MCMC proposals divided by the number of sensors in the network.

rithm to reconcile such a group with the absolute positions of other sensors as dictated by the beacon nodes.

With a large sensing range, the algorithm was consistently able to utilize the noisy range data collected from the unlocalized sensors to improve the final network pose estimate over that of the initial estimate using beacon data alone (Figure 3(a)). For example, in 200 trials of the algorithm on networks of 4 beacons and 40 randomly distributed beacons, the final average Euclidean sensor error was reduced to a 0.31 proportion of the initial results obtained using beacon data only. The final average sensor error was 2.34 and the average particle variance was 2.83. Additionally, there was a high level of consistency in the performance of the algorithm (Figure 3(b)). The final Euclidean error of the most poorly located sensor in each network was less than 5 units on a 100x100 grid for 98 *per cent* of the networks (Figure 3(c)). Furthermore, since a separate location estimate is provided for each particle used, poorly localized sensors should be identifiable based on the variance of their position estimates.

The performance of the algorithm improves as the density of sensors in the region increases (Figure 4). The higher sensor density results in multiple range estimates to any one sensor which are combined to reduce bias. Additionally, in our model, range estimates are of higher quality at shorter range.

Performance appeared to level off when the total number of proposals used in the MCMC increased beyond a certain threshold. This threshold appears to be proportional to the number of sensors in the network over the trials we have evaluated (Figure 5). If this empirical observation holds under broader circumstances, then the rate of change in the variance between the macro-particles could be used as the terminating condition of the algorithm, and the computational time required would be:

$$T \propto (MN)C_{MCMC}$$

where N is the number of sensors, M is the number of particles, and C_{MCMC} is the constant computing power necessary to evaluate a single MCMC proposal.

In the worst case, C_{MCMC} is proportional to QN, which corresponds to each sensor having a valid range estimate to every other sensor. Recall that Q specifies the maximum number of sensor locations that can be shifted in a new proposal, (in standard Gibbs Q is always one). However, under realistic conditions, each sensor will have some range beyond which distance estimates to other sensors are impossible or meaningless. Therefore, under ranged conditions a single MCMC proposal evaluation will be proportional to some range constant based on the ranging method employed, the environment, and the density of sensors:

$$T \propto KMQN$$

where K is the range constant.

Conclusion and Future Work

In this paper we have demonstrated and verified, through numerical simulations, an algorithm for the self-localization of a sensor network based on noisy inter-sensor range data. Unlike most previous related work, our method returns a representation of the PDF describing the position of each sensor in the network. This information indicates both probable locations for the sensor and the degree of certainty by which it has been localized.

Future work will look at improving the practicality of the algorithm. We would like to explore run time optimization techniques, such as dividing larger networks into multiple regions, evaluating each region separately and then merging the final result. Also of interest is determining when a network has been localized to some degree based on an analysis of the final pose samples returned by the algorithm. This could allow the algorithm to terminate when a network has been adequately localized, or could be used to flag localization problems.

References

Akyildiz, I. F.; Su, W.; Sankarasubramaniam, Y.; and Cayirci, E. A. 2002. A survey on sensor networks. *IEEE Communications Magazine* 40(8):102–114.

Bulusu, N.; Heidemann, J.; and Estrin, D. 2000. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine* 7(5):28–34.

Capkun, S.; Hamdi, M.; and Hubaux, J.-P. 2001. GPS-free positioning in mobile ad-hoc networks. In *HICSS*.

Coates, M. J. 2004. Distributed particle filtering for sensor networks. In *Int. Symp. Information Processing in Sensor Networks*.

Correal, N., and Patwari, N. 2001. Wireless sensor networks: Challenges and opportunities. In *MPRG/Virgina Tech Wireless Symposium*.

Ihler, A. T.; Fisher III, J. W.; Moses, R. L.; and Willsky, A. S. 2005. Nonparametric belief propagation for self-calibration in sensor networks. *IEEE Journal of Selected Areas in Communication*.

Javed, O.; Rasheed, Z.; Shafique, K.; and Shan, M. 2003. Tracking across multiple cameras with disjoint views. In *The Ninth IEEE International Conference on Computer Vision*.

Mainwaring, A.; Polastre, J.; Szewczyk, R.; Culler, D.; and Anderson, J. 2002. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*.

Marinakis, D.; Dudek, G.; and Fleet, D. 2005. Learning sensor network topology through monte carlo expectation maximization. In *IEEE Intl. Conf. on Robotics and Automation*.

Moore, D.; Leonard, J.; Rus, D.; and Teller, S. 2004. Robust distributed network localization with noisy range measurements. In *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys '04)*.

Niculescu, D., and Nath, B. 2003. Ad hoc positioning system (APS) using AoA. In *Proc. of INFOCOM*.

Paskin, M. A.; Guestrin, C. E.; and McFadden, J. 2005. A robust architecture for inference in sensor networks. In *In Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks 2005 (IPSN-05)*.

Patwari, N.; Hero, A.; Perkins, M.; Correal, N.; and O'Dea, R. 2003. Relative location estimation in wireless sensor networks. In *IEEE Transactions on Signal Processing*, volume 51 of *8*, 2137–2148.

Rahimi, A.; Dunagan, B.; and Darrell, T. 2004. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *CVPR 2004*, volume 1, 187–194.

Rekleitis, I.; Meger, D.; and Dudek, G. 2005. Simultaneous planning localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems (RAS) Journal, special issue on Planning and Uncertainty in Robotics*.

Rosenthal, J. S. 2000. Parallel computing and monte carlo algorithms. *Far East Journal of Theoretical Statistics* 4:207–236.

Saxe, J. B. 1979. Embeddability of weighted graphs in k-space is strongly np-hard. In *In Proc. 17th Alleron Conf. Commun. Control Comput.*, 480–489.

Wang, H.; Elson, J.; Girod, L.; Estrin, D.; and Yao, K. 2003. Target classification and localization in a habitat monitoring application. In *In Proc. of the IEEE ICASSP*.