A Practical Algorithm for Network Topology Inference

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

Abstract— **When a network of robots or static sensors is emplaced in an environment, the spatial relationships between the sensing units must be inferred or computed for most key applications. In this paper we present a Monte Carlo Expectation Maximization algorithm for recovering the connectivity information (*i.e.* topological map) of a network using only detection events from deployed sensors. The technique is based on stochastically reconstructing samples of plausible agent trajectories allowing for the possibility of transitions to and from sources and sinks in the environment. We demonstrate robustness to sensor error and non-trivial patterns of agent motion. The result of the algorithm is a probabilistic model of the sensor network connectivity graph and the underlying traffic trends. We conclude with results from numerical simulations and an experiment conducted with a heterogeneous sensor network.**

I. Introduction

We are interested in networks made up of static sensors, mobile robots, and sensors carried passively on other objects. In general, if a sensor can move actively there are several SLAM-like methods that can be used to estimate it's position. For a network made up only of passive sensors, the problem is more difficult. In this paper we address the problem of inferring the *topology*, or inter-node connectivity, of a sensor network given only unlabelled observations of activity in the environment (*i.e.* we make the pessimistic assumption that the objects being observed cannot be distinguished from one another). We wish to recover the physical connectivity of the sensors with respect to one another from the point of view of an agent navigating the environment. We assume only indistinguishable agents moving in the environment to allow us to examine the worst-case sensing scenario. This is the case, for example, if only motion sensors are used. If more reliable sensors are used (such as cameras that can identify specific people) then our solution can accommodate this and shows even better performance.

This topological information we seek differs from a metric representation which identifies the relative locations of the sensors but does not provide information about the layout of the region, or obstructing objects within it. We assume that we have no prior knowledge of the relative locations of the sensors and that we have only a limited knowledge of the type of activity present in the environment. We must use observational data returned from our sensors to understand the motion of agents present in the environment. By infering underlying patterns in their motions we can then recover the relationships between the sensors of our network.

Our approach is to divide the problem into two interdependent sub-problems: first, inferring the association between sensor observations and motion sources (agents) moving though the environment, and second, inferring the network connectivity parameters that best describe these inter-node transitions. We construct plausible trajectories of agent motions through the network and augment our observational data with this information. Since the trajectory information allows us to determine likely connectivity parameters and the connectivity parameters suggest possible trajectories, we can iteratively converge toward a final answer through statistical methods.

As sensor networks are established in more locations for monitoring and surveillance purposes, there will be a demand for algorithms and software approaches that can make inferences about the environment based on large quantities of highly distributed and possibly low quality sensing information. This is especially true in areas where we are unable to venture ourselves, or unwilling to venture for fear of influencing the data we are collecting. On Great Duck Island, Maine, for example, a sensor network was successfully employed to collect habitat data without disturbing wildlife with human presence [1], [2]. Another example is the proposed underwater observing system NEPTUNE [3], which plans to wire the Juan de Fuca tectonic plate off the coast of the North-West Pacific ocean. The underwater network will generate observational data from a variety of distributed sensors which could be used to infer additional information about the ocean environment that would be difficult to collect directly for logistical and financial reasons.

This paper addresses a single aspect of the more general problem of inferring information about the environment given distributed sensor data: recovering connectivity parameters. Monitoring projects that log data for offline analysis should be able to benefit from our technique. For example, a vehicle monitoring network distributed about a city could help make decisions about road improvements which might best alleviate congestion. Another motivation are applications using the connectivity information inferred by our technique for sensor network self-calibration efforts; *e.g.* the calibration of a surveillance system. With this work, we are addressing a type of problem that will grow in importance as distributed sensing becomes more prevalent.

II. Background

It is recognized that self-calibration and other more general self-configuration algorithms are important issues for both multi-robot systems and for sensor networks [4] [5]. The main point is that a network must operate autonomously in an dynamic environment. It should be capable of re-organizing itself to handle network changes such as individual node failures or changes in communication range.

A key requirement for many network applications is the ability to self-localize in the absence of GPS data [6], [7]; *i.e.* recovering the relative metric location of each node of the network in cases where GPS is too expensive, not available, or otherwise impractical. In general, localization efforts are based on methods for estimating the distances between sensors. Common techniques include the use of received communication signal strength in radio networks [8], or time-of-arrival ranging using ultrasound [9], [10].

The problem of inferring the topology of a sensor network is closely related to that of metric self-localization. In self-localization, the goal is to recover the relative locations of the nodes independent of the layout of the space in which the network is embedded. Topology inference as we define it, however, must take into account the spatial constraints of the environment since they determine the inter-node connectivity parameters. These two tasks could complement one other. Information regarding the spatial locations of the nodes as well as their communication connectivity can make it easier to determine topologically adjacent nodes and *vica versa*, although, in many cases, the information can be misleading (Figure 1).

In this work, we attempt to solve for the topology of a network, accounting for spatial constraints, without relying on traditional self-localization techniques such as time-to-arrival and signal strength. However, along with target signatures, these methods could be incorporated into our approach as part of the probabilistic framework.

Although much of the research conducted in self-configuration efforts is based on developing distributed, computationally efficient algorithms appropriate for low-power sensor network platform, some recent work has looked at the self-calibration of multi-sensors networks by exploiting motion in the environment [11], [12], [13], [14], [15]. These efforts generally assume vision-based sensors and place less

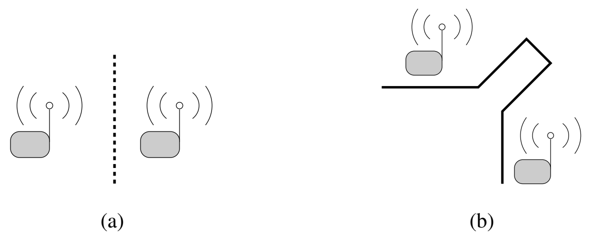


Fig. 1. Examples where communication signal strength is misleading: a) thin interior wall prevents passage but signal is strong b) blocking exterior wall prevents signal but nodes are topologically adjacent

emphasis on the traditional sensor network concerns of efficiency and distributed processing. Instead, they focus on research issues regarding the processing of observations collected from distributed sensors.

Of relevance is the work of Ellis, Makris, and Black [14] [15] on the topology inference of camera-networks. They described an approach in which they use temporal correlations in observations of agents' movements to identify links between network components. They used a thresholding heuristic to identify peaks in the temporal distribution of travel times between nodes. Like Ellis, Makris, and Black, we are interested in exploiting motion to recover information regarding our environment. However, we employ considerably different methods.

Some related work [16] has shown the validity of an MCEM-based algorithm for recovering the connectivity information of a network using only detection events from the deployed sensors. The technique was based on reconstructing plausible agents trajectories, but made some unrealistic assumptions about their motions. In this paper we present a new algorithm for network topology inference that is more robust to observational noise than previously developed approaches, hence making the technique more applicable to real world situations. We assess our new algorithm with simulations and experiments conducted on a heterogeneous sensor network.

III. Problem Description

We formalize the problem of topology inference in terms of the inference of a weighted directed graph which captures the connectivity relationships between the positions of the sensors' nodes. The motion of multiple agents moving asynchronously through a sensor network region can be modeled as a semi-Markov process. The network of sensors is described as a directed graph $G = (V, E)$, where the vertices $V = v_i$ represent the locations where sensors are deployed, and the edges $E = e_{i,j}$ represent the connectivity between them; an edge $e_{i,j}$ denotes a path from the position of sensor v_i to the position of sensor v_j. The motion of each agent in this graph can be described in terms of their transition probability across each of the edges $A_n = \{a_{ij}\}$, as well as a temporal distribution indicating the duration of each transition D_n. The input to the system are the observations $O = \{o_t\}$, which are a list of events detected at arbitrary times from the various vertices of the graph. Each event indicate the likely presence of an agent at that position at that time.

The goal of our work is to estimate the parameters describing this semi-Markov process. We assume that the agents' probabilistic behavior is homogeneous; *i.e.* the motion of all agents are described by the same A and D. In addition, we assume that the distribution of the inter-sensor (*i.e.* inter-vertex) transition times can be described by a windowed normal distribution. We will show later, however, that we can relax this assumption in some situations.

Given the observations O and the number of agents N, the problem is to estimate the network connectivity parameters A and D, subsequently referred to as θ.

IV. Topology Inference Algorithm

In this section we will briefly describe the fundamental topology inference algorithm that takes non-discriminating observations and returns inferred network parameters. The technique assumes knowledge of the number of agents in the environment and attempts to augment the given observations with an additional data association that links each observation to an individual agent.

In the next section, we will present a new version of the algorithm that can operate under weaker assumptions regarding the motion of agents through the environment.

A. Monte Carlo Expectation Maximization

We use the EM algorithm [17]. to solve the connectivity problem by simultaneously converging toward both the correct observation data correspondences and the correct network parameters. We iterate over the following two steps:

1) *The E-Step:* which calculates the expected log likelihood of the complete data given the current parameter guess:

$$Q\big(\theta, \theta^{(i-1)}\big) = E\Big[\log p(O, Z|\theta)|O, \theta^{(i-1)}\Big]$$

where O is the vector of binary observations collected by each sensor, and Z represents the hidden variable that determines the data correspondence between the observations and agents moving throughout the system.

2) *The M-Step:* which then updates our current parameter guess with a value that maximizes the expected log likelihood:

$$\theta^{(i)} = \operatorname*{argmax}_{\theta} Q\big(\theta, \theta^{(i-1)}\big)$$

We employ MCEM [18] to calculate the E-Step because of the intractability of summing over the high dimensional data correspondences. We approximate $Q\big(\theta, \theta^{(i-1)}\big)$ by drawing M samples of an ownership vector $L^{(m)} = \{l_i^m\}$ which uniquely assigns the agent i to the observation o_i in sample m:

$$\theta^{(i)} = \operatorname*{argmax}_{\theta} \left[\frac{1}{M}\sum_{m=1}^{M} \log p(L^{(m)}, O|\theta)\right]$$

where $L^{(m)}$ is drawn using the previously estimated $\theta^{(i-1)}$ according to a Markov Chain Monte Carlo sampling technique, explained in the next section.

At every iteration we obtain M samples of the ownership vector L, which are then used to re-estimate the connectivity parameter θ (the M-Step). At every iteration of the algorithm the likelihood of the ownership vector increases, and the process is terminated when subsequent iterations result in very small changes to θ.

In general, we make the assumption that the inter-vertex delays are normally distributed and determine the maximum likelihood mean and variance for each of the inter-vertex distributions along with transition likelihoods. In a subsequent section, we will describe how we occasionally reject outlying low likelihood delay data and omit it from the parameter update stage.

B. Markov Chain Monte Carlo Sampling

We use Markov Chain Monte Carlo sampling to assign each of the observations to one of the agents, thereby breaking the multi-agent problem into multiple versions of a single-agent problem. In the single agent case, the observations O specify a single trajectory through the graph which can be used to obtain a maximum likelihood estimate for θ. Therefore, we look for a data association that breaks O into multiple single agent trajectories. We express this data association as an ownership vector L that assigns each of the observations to a particular agent.

Given some guess of the connectivity parameter θ, we can obtain a likely data association L using the Metropolis algorithm; an established method of MCMC sampling [19]. From our current state in the Markov Chain specified by our current observation assignment L, we propose a symmetric transition to a new state by reassigning a randomly selected observation to a new agent selected uniformly at random. This new data association L' is then accepted or rejected based on the following acceptance probability:

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

However, the acceptance probability α can be expressed in a simple form since the trajectories described by L'

differ from those in L by only a few edge transitions. Consider L as a collection of ordered non-intersecting sets containing the observations assigned to each agent $L = (T_1 \cup T_2 \cup \ldots \cup T_N), T_n = \{w_{jk}\}$ where w_{jk} refers to the edge traversal between vertices j and k. The probability of a single agent trajectory is then the product of all of its edge transitions. Therefore a proposed change that reassigns the observation o_n from agent y to agent x must remove an edge traversal w from T_y and add it to T_x. Only the change in the trajectories of these two agents need be considered since all other transitions remain unchanged.

In between each complete sample of the ownership vector L, each of the observations are tested for a potential transition to an alternative agent assignment. This testing is accomplished in random order and should provide a large enough spacing between realizations of the Markov Chain that we can assume some degree of independence in between samples. The resulting chain is ergodic and reversible and should thus produce samples representative of the underlying probability distribution.

C. Delay Model

In this section we present a re-working of the fundamental algorithm that allows for the transition of agents to and from sources and sinks and which therefore is more robust both to shifting numbers of agents in the environment and to agents that pause or delay their motion in between sensors. Additionally, assuming the existence of sources and sinks, we can recover their connectivity to each of the sensors in our network.

1) The Source/Sink Node: In addition to the maintaining a vertex that represents each sensor in our network, we introduce an additional vertex that represents the greater environment outside the monitored region: a *source/sink node*. We then modify our current trajectory sampling and parameter updating methodology to allow transitions to and from this additional node.

During the E-Step of our iterative EM process in which we are evaluating potential changes to agent trajectories, we replace our initial assumption of normally distributed inter-vertex delay times with a more sophisticated model that additionally allows for uniform but low probability jumps of almost arbitrary lengths. In other words, a mixture model is employed in which an inter-vertex delay time is assumed to arise from either a Gaussian distribution or from a uniform distribution of fixed likelihood (Figure 2).

During the M-Step, in which we use the data generated by M samples of the observation vector L to update our network parameters, the data assigned to the Gaussian distribution are assumed to be generated by "through-traffic" and are used to update our belief of the inter-node delay times and transition

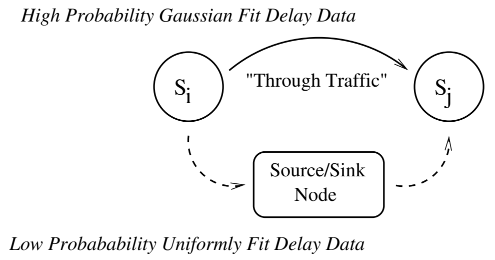

High Probability Gaussian Fit Delay Data
S_i
"Through Traffic"
S_j
Source/Sink Node
Low Probabability Uniformly Fit Delay Data

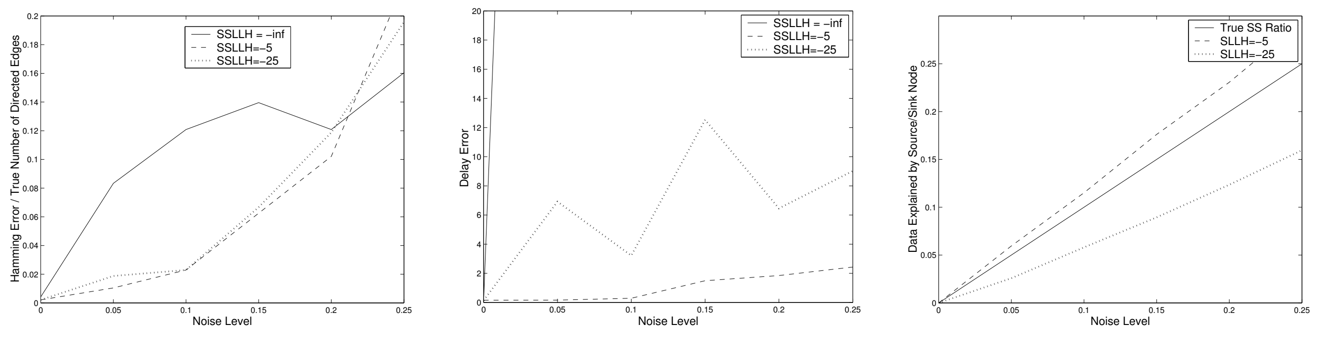
Fig. 2. Algorithm delay model.

likelihoods. However, the data fit to the uniform distribution are believed to be transitions from the first vertex into the sink/source node and then from the sink/source node to the second vertex. Therefore they are not used for updating inter-vertex delay parameters of the two nodes, but rather are used only for updating the belief of transitions to and from the source/sink node for the associated vertices.

While the Gaussian assigned delays are expected to be within a realistic temporal range for direct inter-vertex agent motion, the delay data fit to the uniform distribution is more loosely bounded. This gives the inference technique a manner of temporarily removing agents from the system by assigning them to long transitions and also can be used to explain events that would otherwise seem extremely unlikely such as the disappearance of an agent from one node and its almost immediate appearance at a second.

2) Parameter Tuning: The new algorithm works by discarding outliers in the delay data assigned to each pair of vertices and explaining their existence as transitions to and from a source/sink node. The key to this process is determining whether or not a delay value should be considered an outlier. This is implemented using a probability threshold called a Source Sink Log Likelihood (SSLLH) that determines when delay data should be incorporated into parameter updates (Figure 3). If the probability for an inter-vertex delay, as calculated given the current belief of the (Gaussian) delay distribution, is lower than the probability threshold specified by the SSLLH parameter, then this motion is interpreted as a transition made *via* the source/sink node. Transitions explained in this manner are not used to update the network parameters associated with the origin and destination vertices.

An effective value for the SSLLH parameter somewhat depends on the traffic patterns in the monitored environment and may have to be determined partially through experimental methods. Although a full discussion of the calibration technique is outside the scope of this paper, we consider the effect of different SSLLH parameter values in the next section.



(a) (b) (c)

Fig. 4. Average over 10 graphs using the simulator with 4 agents on 12 node, 48 edge graphs. The X axis indicates proportion of both white and systematic delay noise. Y axis shows: Hamming error per edge (a); delay error in seconds (b); and ratio of data explained by souce/sink node transitions(c).

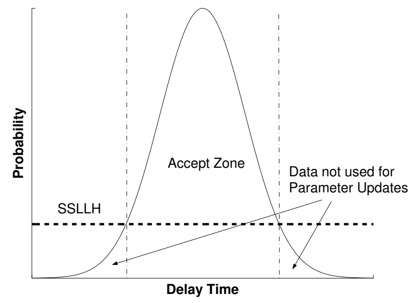


Fig. 3. Graphical description of the SSLLH Parameter

V. Simulation Results

A. The Simulator

We have developed a tool that simulates agent traffic through an environment represented as a planar graph. Our simulation tool takes as input the number of agents in the system and a weighted graph where the edge weights are proportional to mean transit times between the nodes. All connections are considered two ways; *i.e.* each connection is made up of two uni-directional edges. The output is a list of observations generated by randomly walking the agents through the environment. Inter-node transit times are determined based on a normal distribution with a standard deviation equal to the square root of the mean transit time.[1]

Two types of noise were modeled in order to assess performance using data that more closely reflects observations collected from realistic traffic patterns. First, a 'white' noise was generated by removing a percentage of correct observations and replacing them with randomly generated spurious observations. Second, a more systematic noise was generated by taking a percentage of inter-vertex transitions and increasing the Gaussian distributed delay time between them by an additional delay value selected uniformly at random. The hope is that small values of both these types of noise simulate both imperfect sensors and also the tendency for agents to stop occasionally in their trajectories; *e.g.* to talk, use the water fountain, or enter an office for an period.

A number of experiments were run using the simulator on randomly generated planar, connected graphs. The graphs were produced by selecting a sub-graph of the Delaunay triangulation of a set of randomly distributed points.

For each experiment, the results were obtained by comparing the final estimated transition matrix A' to the real transition matrix A. A graph of the inferred environment was obtained by thresholding A'. The Hamming error was then calculated by measuring the distance between the true and inferred graphs normalized by the number of directed edges m in the true graph:

$$HamErr_A = \left(\frac{1}{m}\right) \sum_{a_{ij} \in A, a'_{ij} \in A'} \left[thr(a_{ij}) - thr(a'_{ij})\right]^2$$

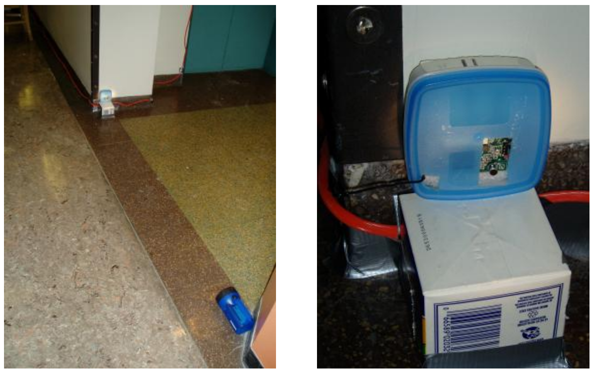
where $thr(a) = \lceil a_{ij} - \theta \rceil$.[2]

B. Performance Results

Our experiments show that in the absence of significant measurement noise, the network structure can be determined very reliably with a handful of agents and a sufficient number of observations (*e.g.* 4 or 10 agents). This appears to be true for various graph sizes, although for this low noise condition we have only tested graphs of limited size. For example, the topology of 95 *per cent* of 12 node graphs was perfectly inferred with zero Hamming error for 200 simulations with

[1]Negative transit times are rejected.

[2]A threshold value of $\theta = 0.1$ was selected for our experiments.



(a) (b)

Fig. 5. a) Complete setup and, b) close up of a deployed photocell-based sensor constructed out of a flashlight and a Crossbow wireless sensor. (Plastic containers were used as protective covering during experiments.)

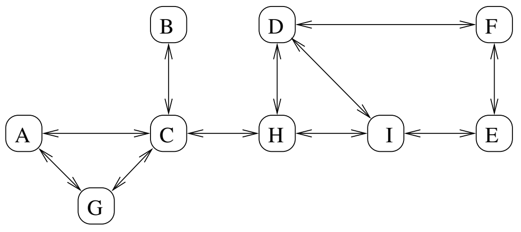
4 agents. Generally, the algorithm converged quickly, finding most of the coarse structure in the first few iterations and making incrementally smaller changes until convergence.

In the presence of sensor noise (Figure 4) the performance of the algorithm depends, as expected, on the SSLLH threshold. Recall that this parameter controls the model's ability to discard inconsistent data. When used with a high SSLLH value, the mixture approach for modeling delays was very successful at minimizing the effects of noise. Even when 10 *per cent* of the delay times were uniformly increased, the Hamming error of the inferred transition matrix was still quite low (Figure 4(a)). The reduction in error for inferred mean delay times was especially dramatic (Figure 4(b)). However, under conditions of extreme noice the advantage of our new delay model became less apparent.

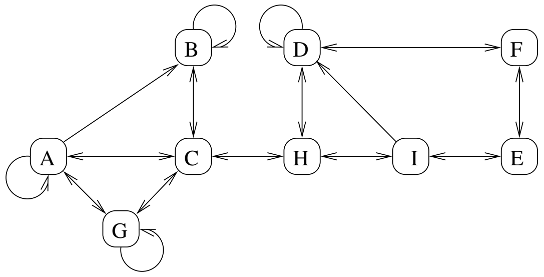
The performance of the algorithm under conditions of moderate error reflect the ability of the new delay model to successfully identify and discard low probability transitions and explain them as transitions to the source/sink node (Figure 4(c)). Since the SSLLH parameter can be tuned to expected levels of noise in the environment, the algorithm should perform well under real world conditions.

VI. Experimental Results

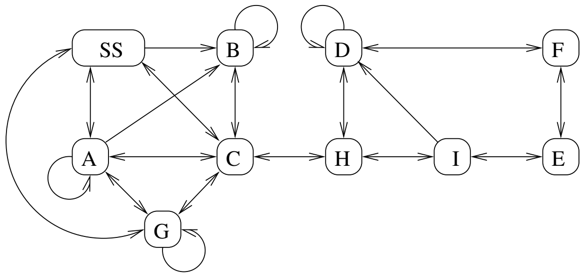
In order to test our technique under real-world conditions, we setup an experiment using a real sensor network of nine nodes and analyzed the results using our approach. The sensor nodes were built up of photocell-based sensors running on low-powered commercial devices and vision-based sensors running on single board computers (Figure 5). Both types of sensors were programmed to act as simple



(a)



(b)



(c)

Fig. 7. Topological maps of the environment that were: a) analytically determined based on the layout; b) inferred by the algorithm; c) inferred by the algorithm including the source/sink node.

motion detectors sending event messages to a central server, which logged the origin and time of the activity.

The experiment was conducted in the hallways of one wing of an office building (Figure 6). The data were collected during a six and a half hour period from 10:00am to 4:30 pm on a weekday. In total, approximately 4700 timestamped events were collected.

Ground truth values were calculated in order to assess the results inferred by the approach. A topological map of the environment was determined (Figure 7(a)) based on an analysis of the sensor network layout. In addition, inter-vertex transitions times for the connected sensors were recorded with a stopwatch for a typical subject walking at a normal

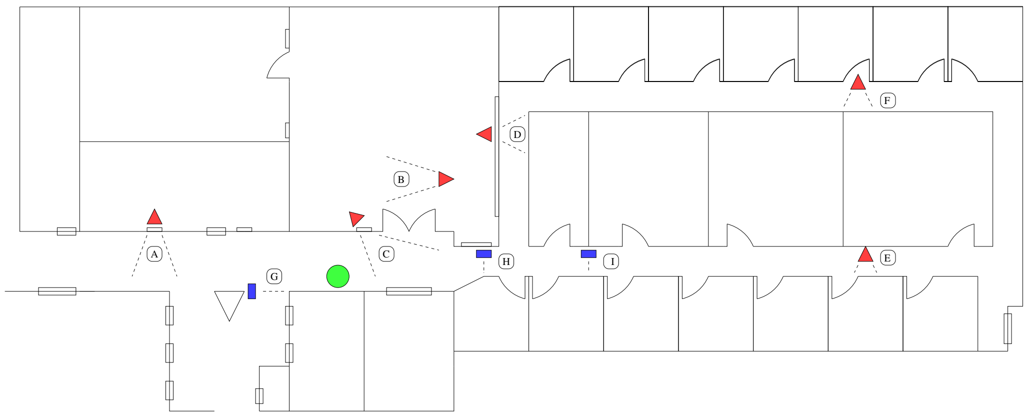
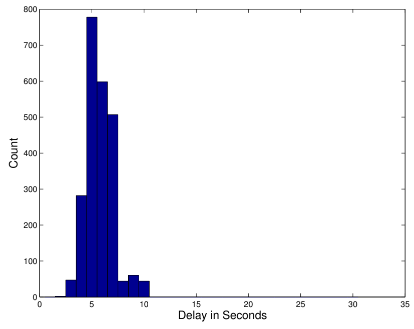
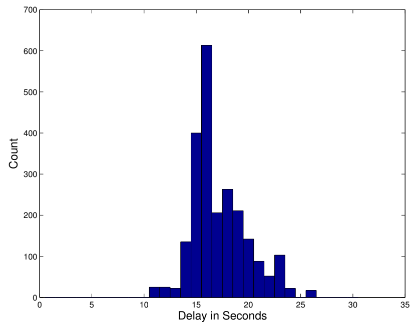


Fig. 6. The layout of the nine senor (heterogeneous) network used for the experiment. Labeled triangles represent vision-based sensor positions (A-F) and labeled rectangles represent low-powered photo-based sensors (G-I). The circle represents the location of the central server.



(a)



(b)

Fig. 8. Examples of delay distributions inferred for: a) sensor D to sensor H; b) sensor F to sensor D.

Connection	Timed	Inferred
A,G	6	8 / 11
A,C	9	12 / 10
B,C	5	6 / 8
C,G	5	5 / 5
C,H	5	6 / 6
D,F	14	15 / 17
D,H	5	5 / 6
D,I	6	7 / 7
E,F	13	13 / 13
E,I	13	15 / 14
H,I	4	4 / 4

TABLE I
A comparison of timed and inferred delay times (both ways) between sensors. All values are rounded to the nearest second.

speed (Table I).

Except for a few small differences, the network parameters inferred by our topology inference algorithm closely corresponded to the ground truth values. Figure 7 compares the analytically determined and inferred topological maps. Disregarding reflexive links, the difference between the inferred and 'ground truth' results amounted to a Hamming error of 2. The two significant errors are: an extra edge found between sensors A and B; and a missing one-way edge from sensor D to I. The missing edge from D to I is likely due to the tendency of people to go straight rather than turn right when navigating the corridor on the bottom right of the region (heading left) as shown in Figure 6 while the extra edge found leading from sensor A to sensor B is likely due to a correlation in the detection intervals between these two nodes.

The mean transition times produced by the algorithm were consistent to those determined by stopwatch (Table I, Figure 8). Additionally, the number of reflexive links or self-connections inferred by the algorithm also seem consistent with expected results. Except for node D, the other reflexive links all occur on sensors that are on the boundary of the monitored region (Figure 7(b)). Traffic passing node G, for example, might be correlated to the arrival of the elevator. (The elevators are located to the right, immediately below sensor G. See Figure 6.)

Note that the connections to the source/sink node also occur only for boundary nodes (Figure 7(c)) and are therefore consistent with an analytical assessment of the traffic patterns. Since traffic commonly enters and exits the monitored region *via* one of the boundary nodes, the inference algorithm should commonly employ the source/sink node in order bring the agent back into the system. One might also expect a connection to the source/sink node for nodes such as F which are near offices which could function as a sink or source of agent motion. During our experiment, however, this type of activity was presumably overshadowed by "through" traffic in that region.

VII. Conclusion and Future Work

We have developed a practical algorithm for learning the connectivity information of a sensor network based on a stochastic trajectory sampling technique that incorporates a realistic model of inter-sensor delay distributions. This algorithms was developed and refined using a simulator and then implemented and tested on a sensor network emplaced in an office environment, using a mixture of heterogeneous sensors. An important component of the model that distinguishes it from related work is the presence of an explicit process for handling "noise" in the system, including agents that become undetectable (*e.g.* people that go into offices). Results from an experiment conducted on an emplaced sensor network as well as supporting simulation data demonstrate the robustness of the technique to realistic variations in traffic patterns and observational noise in general.

Future work will look at removing the current dependence of the algorithm on *a priori* knowledge such as the number of agents in the environment and assumptions regarding traffic patterns (the SSLLH parameter). We have had some success by considering an 'Occam's Razor' type of approach that favors assumptions about input parameters that both lead to simplistic solutions and explain the majority of the observational data.

VIII. Acknowledgments

We would like to thank Ionnis Rekleitis, Philippe Giguere, and others of the Mobile Robotics lab, along with the CIM administration for their technical help and good ideas. Thank-you in addition to to Michelle Theberge for the photo, proof reading, and valuable assistance during the experiment.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[3] C. R. Barnes, J. R. Delaney, B. M. Howe, and N. Penrose, "Neptune: A regional cabled observatory in the northeast pacific," in *White paper for Ocean Research Interactive Observatory Networks (ORION) meeting*, January 2004.

[4] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: self-configuring localization systems," in *Sixth International Symposium on Communication Theory and Applications (ISCTA-01)*, Ambleside, Lake District, UK, July 2001.

[5] N. Correal and N. Patwari, "Wireless sensor networks: Challenges and opportunities," in *MPRG/Virgina Tech Wireless Symposium*, 2001.

[6] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *HICSS*, 2001.

[7] A. Savvides, C. Han, and M. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *7th annual international conference on Mobile computing and networking*, Rome, Italy, 2001, pp. 166–179.

[8] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, October 2000.

[9] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Mobile Computing and Networking*, 2000, pp. 32–43.

[10] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AoA," in *Proc. of INFOCOM*, San Francisco, CA., 2003.

[11] C. Stauffer and K. Tieu, "Automated multi-camera planar tracking correspondence modeling," in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, vol. 1, July 2003, pp. 259–266.

[12] A. Rahimi, B. Dunagan, and T. Darrell, "Simultaneous calibration and tracking with a network of non-overlapping sensors," in *CVPR 2004*, vol. 1, June 2004, pp. 187–194.

[13] O. Javed, Z. Rasheed, K. Shafique, and M. Shan, "Tracking across multiple cameras with disjoint views," in *The Ninth IEEE International Conference on Computer Vision*, Nice, France, 2003.

[14] T. Ellis, D. Makris, and J. Black, "Learning a multicamera topology," in *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Nice, France, October 2003, pp. 165–171.

[15] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2004*, Washington DC, June 2004.

[16] D. Marinakis, G. Dudek, and D. Fleet, "Learning sensor network topology through monte carlo expectation maximization," in *IEEE Intl. Conf. on Robotics and Automation*, Barcelona, Spain, April 2005.

[17] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[18] G. Wei and M. Tanner, "A monte-carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, vol. 85(411), pp. 699–704, 1990.

[19] M. Tanner, *Tools for Statistical Inference*, 3rd ed. New York: Springer Verlag, 1996.