

Topology Inference for a Vision-Based Sensor Network

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

Abstract

In this paper we describe a technique to infer the topology and connectivity information of a network of cameras based on observed motion in the environment. While the technique can use labels from reliable camera systems, the algorithm is powerful enough to function using ambiguous tracking data. The method requires no prior knowledge of the relative locations of the cameras and operates under very weak environmental assumptions. Our approach stochastically samples plausible agent trajectories based on a delay model that allows for transitions to and from sources and sinks in the environment. The technique demonstrates considerable robustness both to sensor error and non-trivial patterns of agent motion. The output of the method is a Markov model describing the behavior of agents in the system and the underlying traffic patterns. The concept is demonstrated with simulation data and verified with experiments conducted on a six camera sensor network.

1. Introduction

In this paper we consider the problem of learning the connectivity information of a network of cameras with non-overlapping fields of view based on non-discriminating observations. Our purpose is to reconstruct the topology of the network. By ‘topology’ we are referring to the physical inter-sensor connectivity from the point of view of an agent navigating the environment (Figure 1).

In our approach, we attempt to recover correspondences between cameras by exploiting motion present in the environment. The method is based on reconstructing agent trajectories that best explain the observational data and using these trajectories to determine likely network parameters. We assume that the sensors are un-discriminating and report only that they have detected something but do not provide a description or signature. More detailed sensor information could be probabilistically incorporated into the algorithm

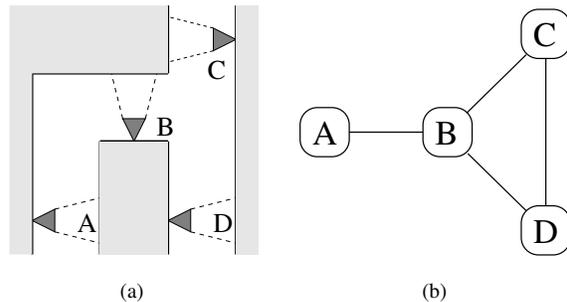


Figure 1. Example of a sensor network layout (a) and corresponding topology (b).

making the problem easier.

We demonstrate our approach experimentally with a six camera network. The system automatically calibrates itself based solely on motion detection. It is able to infer a connectivity graph of the environment and inter-vertex delay times both with a high degree of accuracy.

The ability of a surveillance or monitoring system to automatically determine the connectivity parameters describing its environment is useful for a number of reasons. Although the topology information can be manually entered during installation, more detailed parameters such as inter-camera delay distributions are difficult to determine, and a change in the environment or network would require recalibration. Once calibrated, the connectivity information could aid in conventional target tracking and additional monitoring activities. For example, by reconstructing trajectories, a vehicle monitoring network distributed about a city could help make decisions about road improvements which might best alleviate congestion. In addition, the topological information could be combined with relative localization techniques [11, 8] to recover a more complete representation of the environment.

2 Background

Motion in the environment can be exploited to calibrate a network of cameras. Most efforts using this technique have focused on sensor self-localization. Stein [13], for example, considered recovering a rough planar alignment of the location and orientation of the individual cameras. Using a least-median-of-squares technique, he determined the correspondence between moving objects in pairs of cameras. His approach, however, required some overlap in the field of view of the cameras.

Similarly, Fisher [4] looked at estimating the relative orientation and location of cameras but with non-overlapping fields of view by exploiting the motion of distant moving objects such as stars. The objects were assumed to have well-behaved linear or parabolic trajectories, and it was necessary that the observed objects could be uniquely identified across separate cameras.

In a more recent effort, Rahimi, Dunagan, and Darrell [10] described a simultaneous calibration and tracking algorithm that uses a velocity extrapolation technique to self-localize a network of non-overlapping cameras based on the motion of a single target. Their work avoided the difficult problem of associating observations with different targets by assuming only one source of motion.

Connectivity information or *network topology* can be recovered by exploiting motion in the environment. In contrast to this paper, current efforts either address a slightly different problem than the one we are interested in [5] or they employ considerably different methods [6, 3].

In order to track multiple agents across disjoint fields of view, Javed *et al.* [5] first calibrated the connectivity information of their surveillance system using observational data. To learn the probability of correspondence (transition probabilities) and inter-camera travel times (delay distributions), they assumed a training period in which the data association between observations and agents was known. Given this observation ownership information, they employed a Parzen window technique that looks for correspondences in agent velocity, inter-camera travel time, and the location of agent exit and entry in the fields of view of the camera.

Focusing on camera network calibration, Ellis, Makris and Black [6, 3] presented a technique for topology recovery based on event detection only. In their approach, they first identified entrance and exit points in camera fields of view and then attempted to find correspondences between these entrances and exits based on video data. Their technique relies on exploiting temporal correlation in observations of agent movements. The method employs a thresholding technique that looks for peaks in the temporal distribution of travel times between entrance-exit pairs; a clear peak suggesting that a correspondence exists. The tech-

nique gave promising results on experiments carried out on a six camera network. Although it requires a large number of observations, the method does not rely on object correlation across specific cameras. Thus, the approach can be used to efficiently produce an approximate network connectivity graph but when the network dynamics are complex or the traffic distribution exhibits substantial variation, it would appear the technique will have difficulty.

In previous work [7], we presented and verified, through numerical simulations, a network topology inference method based on constructing plausible agent trajectories. The technique employed a stochastic Expectation Maximization (EM) algorithm, an established statistical method for parameter estimation of incomplete data models [2] [15] that has been applied to many fields including multi-target tracking [9] and mapping in robotics [1, 12].

Although we verified the MCEM approach through numerical simulations, there were some difficulties in successfully applying the technique to real world situations. First, the fundamental algorithm made unrealistic assumptions regarding agent motion. Second, some significant systems-level infrastructure was needed in order to conduct experiments on a hardware.

In this paper, we present a new practical algorithm for topology inference that is robust to observational noise and non-trivial agent motion. This is achieved with a new delay model that allows motion to sources and sinks in the environment. We demonstrate the success of the approach both with simulations and with an experiment conducted on a six camera-based sensor network.

3 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 of the N agents 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 observations $O = \{o_t\}$ are a list of events detected at arbitrary times from the various vertices of the graph, which indicate the likely presence of one of the N agents 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 must make some assumptions about the distribution of the inter-sensor (*i.e.* inter-vertex) transition times. We make the assumption that the delays in moving between one sensor and another 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 .

4 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. (See Marinakis, Dudek, and Fleet [7].)

4.1 Monte Carlo Expectation Maximization

We use the EM algorithm [2]. 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:

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

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:

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

We employ MCEM [15] to calculate the E-Step because of the intractability of summing over the high dimensional data correspondences. We approximate $Q(\theta, \theta^{(i-1)})$ 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). We continue to iterate over the E-Step and the M-Step until we obtain a final estimate of θ . 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.

4.2 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 [14]. 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

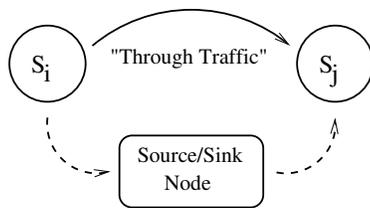
L as a collection of ordered non-intersecting sets containing the observations assigned to each agent $L = (T_1 \cup T_2 \cup \dots \cup T_N)$, $T_n = \{w_{jk}\}$ where w_{jk} refers to the edge traversal between vertices j and k . The probability of a single agent trajectory is then the product of all 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 true probability distribution.

5 Delay Model

In this section we present a re-working of the fundamental topology inference algorithm that allows for the transition of agents to and from sources and sinks in the environment. This makes the algorithm 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.

High Probability Gaussian Fit Delay Data



Low Probability Uniformly Fit Delay Data

Figure 2. Algorithm delay model.

In addition to 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*. A mixture model is employed during the E-Step of our iterative EM process in which we are evaluating potential changes to agent trajectories. An inter-vertex delay time is assumed to arise from either a Gaussian distribution or from a uniform distribution of fixed likelihood (Figure 2).

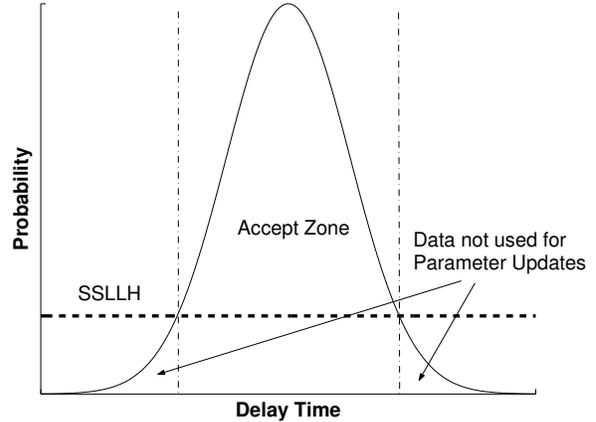


Figure 3. Graphical description of the SLLH Parameter.

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". They are used to update our belief of the inter-node delay times and transition 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.

The delay model provides robustness to noise 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 through a tunable parameter, called Source Sink Log Likelihood (SLLH), that determines the threshold probability necessary for the delay data to be incorporated into parameter updates (Figure 3). The probability for an inter-vertex delay is first calculated given the current belief of the (Gaussian) delay distribution. If this probability is lower than the SLLH then this motion is interpreted as a transition made via the source/sink node. The delay is given a probability equal to the SLLH, and the transition is not used to update the network parameters associated with the origin and destination vertices.

The value assigned to the SLLH parameter determines how easily the algorithm discards outliers and, hence, provides a compromise between robustness to observational noise and a tendency to discard useful data.

6 Simulation Results

6.1 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 a 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:

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

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

6.2 Performance Results

When operating with noise-free data the results show that problems involving a limited number of agents were easy to solve given an adequate number of observations (Figure 4). For example, the topology of 95 *per cent* of

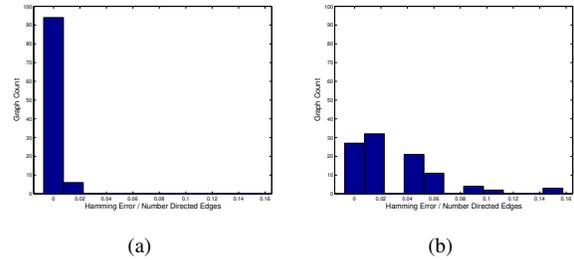


Figure 4. Histogram of Hamming error per edge using the simulator 100 randomly produced graphs with 12 nodes and 4 agents (a) and 12 nodes and 10 agents (b).

the generated 12 node graphs was perfectly inferred with zero Hamming error for simulations with 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.

Under noisy conditions (Figure 5), the performance of the algorithm depended somewhat on the value assigned to the SLLH parameter. When assigned a high SLLH 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 5(a)). The reduction in error for inferred mean delay times was especially dramatic (Figure 5(b)).

The performance of the algorithm under conditions of moderate error reflect the ability of the new algorithm to successfully identify and discard low probability transitions and explain them as transitions to the source/sink node (Figure 5(c)). Since the SLLH parameter can be tuned to the expected levels of noise in the environment, the new algorithm should be able to deliver better results than the fundamental topology algorithm.

7 Experimental Results

7.1 Experimental Setup

In order to test our technique under real-world conditions, we setup an experiment using a network of camera-based sensors and analyzed the results using our approach. The sensor nodes were built up of inexpensive PC hardware networked together over Ethernet using custom software. A single node consists of a 352x292 pixel resolution Labtech USB webcam connected to a Flexstar PEGASUS single board computer (Figure 6). The operating system

¹Negative transit times are rejected.

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

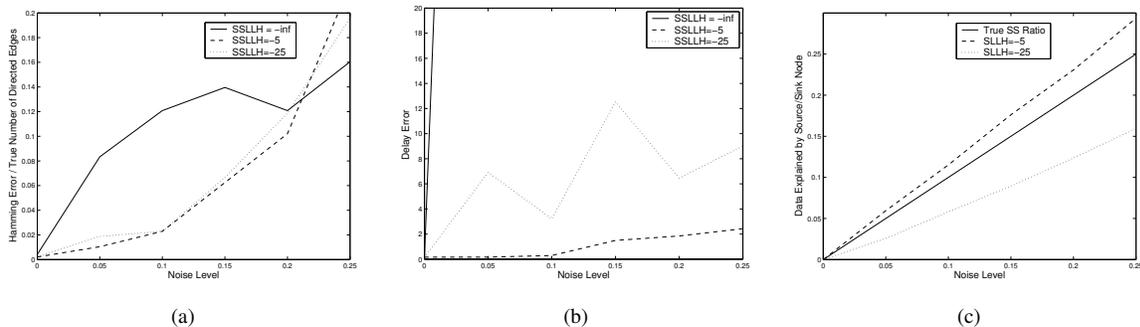


Figure 5. Average over 10 graphs using the simulator with 4 agents on 12 node, 48 edge graphs. The X axis indicates proportion of both white and systematic delay noise. Y axis shows: Hamming error per edge (a); delay error (b); and ratio of data explained by source/sink node transitions(c).



Figure 6. A camera-based sensor.



(a) (b)

Figure 7. Examples of a background image (a) and a frame triggering an event detection (b).

used was Redhat linux based on kernel 2.4. The sensor nodes contain an Intel Celeron 500MHz CPU and 128 MB of RAM. They are disk-less and must netboot from a central server which they are connected to either *via* a wireless bridge or a standard Ethernet cable.

The software implements a standard client/server architecture over TCP/IP using linux sockets written in the C language. Each sensor runs an identical copy of the client program while a single copy of the server application runs on a central computer.

The client software implements a motion detector based on the Labtech webcam. During an initial period, a background image is captured from the camera and the method for triggering an event detection is calibrated. An intensity threshold is calibrated for each colour channel by calculating the standard deviation from the background based on a number of captured frames:

$$\theta_c = C * std\{Frm_0 - Bkgrd, \dots, Frm_n - Bkgrd\}$$

where C is a constant determining the sensitivity of the system. The sensor then enters an armed state in which captured frames are compared to the background image, and any difference exceeding the threshold triggers a detection

event (Figure 7). A frame rate of approximately 10Hz is obtained. Once triggered, the sensor re-arms itself after a couple of seconds of inactivity. The background is slowly updated to account for gradual changes in the scene; *e.g.* changes in lighting or a re-positioned object such as a door:

$$Bkgrd' = \alpha * Frm + (1 - \alpha) * Bkgrd$$

Events are transmitted over TCP/IP to a central server where they are time-stamped and logged for offline analysis. The server is multi-threaded and allows control of the system through a command line interface. In addition to detection events, the application allows a full resolution capture of an image or a low-resolution streaming of images from any sensor to the server.

The experiment was conducted in the hallways of one wing of an office building (Figure 8). The data were collected during a typical weekday for a period of five hours from 10:00am to 2:30 pm. In addition to the normal traffic

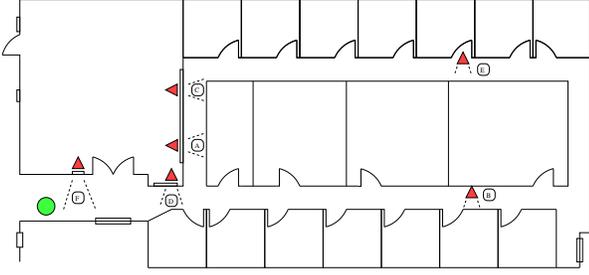


Figure 8. Six camera sensor network used for experiment. Triangles represent sensor positions; circle indicates the location of the central server.

Connection	Timed	Inferred
A,B	16	15 / 16
A,C	3	3 / 3
A,D	4	3 / 3
B,D	15	16 / 17
B,E	16	15 / 15
C,E	14	15 / 14
D,F	5	5 / 3

Table 1. Comparison of timed and inferred delay times (both ways) between sensors. All values rounded to nearest second.

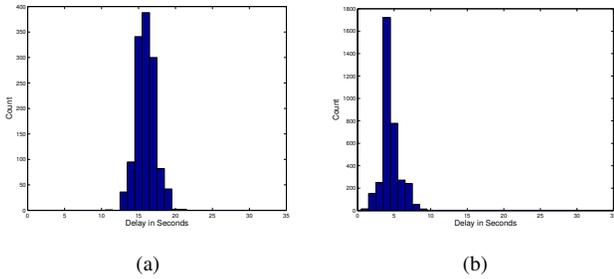


Figure 9. Examples of delay distributions for sensor A to sensor B (a) and sensor D to sensor F (b).

one or two subjects were encouraged to stroll about the region from time to time during the collection period in order to increase the density of observations. In total, about 1800 timestamped events were collected.

7.2 Assessment of Results

Ground truth values were calculated in order to assess the results inferred by the approach. A topological map of the environment (Figure 10(a)) was determined based on an analysis of the sensor network layout shown in Figure 8. Inter-vertex transition times for the connected sensors were recorded with a stopwatch for a typical subject walking at a normal speed (Table 1).

The results obtained by running the new topology inference algorithm on the experimental data correspond closely to the ground truth values. Figure 10(b) shows the topological map obtained by thresholding the inferred transition matrix.³

³The number of agents was selected to be an estimate of the most likely

Disregarding self-connections, the difference between the inferred and deduced matrices amounts to a Hamming error of 1. The inferred connection from D to B was not given a transition probability large enough to be detected based on our thresholding technique. However, the opposite edge from B to D was correctly inferred. Of course, it would be easy to build into the algorithm the assumption that all edges must be two ways. A strong belief in an edge in one direction would dictate that the opposite edge must also exist.

The mean transition times produced by the algorithm are also consistent to those determined by stopwatch (Table 1). Some examples of inferred delay distributions are shown in Figure 9.

Sensor F marks the only heavily used entrance and exit to the region monitored by the network. The self-connection inferred to this node is due to a detected correlation in the delay between exit times and subsequent re-entry times for agent motion. In fact, this correlation is due to the tendency of subjects to re-enter the system after roughly the same time period (*e.g.* to use the washroom or photocopier). Therefore, the detection of this connection was actually a correct inference on the part of the algorithm.

It is interesting to note that two-way connections were inferred to the source/sink node from both sensors D and F (Figure 10(c)). It was possible for subjects to pass by either of these sensors on their way into or out of the monitored region. (The exit to the far right of the area, shown in Figure 8, was little used.) This demonstrates the function of the source/sink node as a method for the algorithm to explain sudden appearances and disappearance of agents in the system.

number of people in the system at any time (3), the SLLH parameter was set to -5 , and a threshold value of $\theta = 0.1$ was used to obtain the topological map

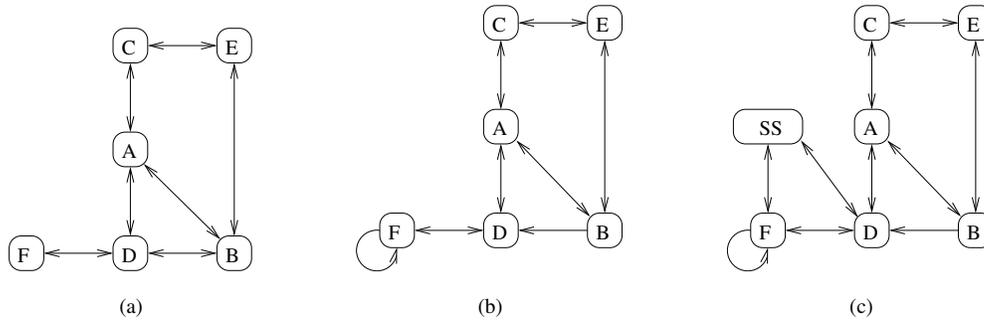


Figure 10. Analytical (a), inferred (b) and inferred with source/sink node (c) topological maps.

8 Conclusions and Future Work

In this paper, we have presented an algorithm for learning the connectivity information of a sensor network based on a stochastic trajectory sampling. The technique employs a realistic model of inter-sensor delay distributions that makes it robust to realistic variations in traffic patterns and observational noise in general. The approach was demonstrated with simulation data and verified with experiments conducted on a vision-based sensor network.

Future work will look at developing a more sophisticated vision system which produces probabilistically labeled tracking data. This additional information could be readily incorporated into the approach and would lead to more rapid convergence.

Acknowledgements:

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

References

- [1] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun. Sonar-based mapping with mobile robots using EM. In *Proc. 16th International Conf. on Machine Learning*, pages 67–76. Morgan Kaufmann, San Francisco, CA, 1999.
- [2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.
- [3] 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*, pages 165–171, Nice, France, October 2003.
- [4] R. B. Fisher. Self-organization of randomly placed sensors. In *Eur. Conf. on Computer Vision*, pages 146–160, Copenhagen, May 2002.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys '04)*, Baltimore, November 2004.
- [9] H. Pasula, S. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *IJCAI-99*, Stockholm, 1999.
- [10] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *CVPR 2004*, volume 1, pages 187–194, June 2004.
- [11] 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*, pages 166–179, Rome, Italy, 2001.
- [12] H. Shatkey and L. P. Kaelbling. Learning topological maps with weak local odometric information. In *IJCAI (2)*, pages 920–929, 1997.
- [13] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages 521–527, June 1999.
- [14] M. Tanner. *Tools for Statistical Inference*. Springer Verlag, New York, 3 edition, 1996.
- [15] 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*, 85(411):699–704, 1990.