A HIERARCHICAL MULTIPLE TARGET TRACKING ALGORITHM FOR SENSOR NETWORKS

by

Songhwai Oh and Shankar Sastry

Memorandum No. UCB/ERL M04/1

5 January 2004

A HIERARCHICAL MULTIPLE TARGET TRACKING ALGORITHM FOR SENSOR NETWORKS

by

Songhwai Oh and Shankar Sastry

Memorandum No. UCB/ERL M04/1

5 January 2004

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Hierarchical Multiple Target Tracking Algorithm for Sensor Networks

Songhwai Oh and Shankar Sastry

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, USA
{sho,sastry}@eecs.berkeley.edu

Abstract. In wireless ad-hoc sensor networks, many inexpensive and small sensor-rich devices are deployed to monitor and control our environment. Each device, called a sensor node, is capable of sensing, computation and communication. Sensor nodes form a wireless ad-hoc network for communication. The limited supply of power and other constraints limit the capabilities of each sensor node. For example, a typical sensor node has short communication and sensing ranges, a limited computational power and a limited amount of memory. However, the abundant number of spatially spread sensors will enable us to monitor changes in our environment accurately despite of inaccuracy of each sensor node. The constraints on a sensor node demand a new set of applications which is different from the traditional applications designed for centralized computations. An application for sensor networks needs to be scalable, flexible, and autonomous. With these requirements in mind, in this paper, we develop a scalable hierarchical multiple target tracking algorithm that is capable of initiating and terminating tracks and robust against transmission failures and communication delays.

1 Introduction

In wireless ad-hoc sensor networks, many inexpensive and small sensor-rich devices are deployed to monitor and control our environment. It is envisioned that the sensor networks will connect us to the physical world in a pervasive manner [1]. Each device, called a sensor node, is capable of sensing, computation and communication. Sensor nodes form a wireless ad-hoc network for communication. The limited supply of power and other constraints, such as manufacturing costs and limited package sizes, limit the capabilities of each sensor node. For example, a typical sensor node has short communication and sensing ranges, a limited computational power and a limited amount of memory. However, the abundant number of spatially spread sensors will enable us to monitor changes in our environment accurately despite of inaccuracy of each sensor node. The constraints on a sensor node demand a new set of applications which is different from the traditional applications designed for centralized computations. An application for sensor networks needs to be scalable, flexible, and autonomous. With these requirements in mind, in this paper, we develop a hierarchical multiple target tracking algorithm for sensor networks.

1.1 Multiple Target Tracking Algorithms

The multiple target tracking plays an important role in many areas of engineering such as surveillance, computer vision, and signal processing [2, 3]. Under the most general setup, a varying number of targets are moving around in a region with continuous motions and the positions of moving targets are sampled at random intervals. The measurements about the positions are noisy, with detection probability less than one, and there is a noise background of spurious position reports (false alarms). Targets arise at random in space and time. Each target persists independently for a random length of time and ceases to exist. A track of a target is defined as a path in space-time traveled by the target. The seminal paper by Sittler [4] introduced the major concepts about multiple target tracking and a method to evaluate tracks. He pointed out two major problems in multiple target tracking: data association and state estimation. The essence of the multiple target tracking problem is to find tracks from the noisy observations and it requires solutions to both data association and state estimation problems.

In [4], the data association problem in multiple target tracking is described as a problem of finding a partition of observations such that each element of a partition is a collection of observations generated by a single target or clutter. However, due to the noises in state transition and observation, we cannot expect to find the exact solution. Hence, Sittler developed a probabilistic surveillance model and searched for a partition of observations on which the likelihood function defined by the surveillance model is maximized. This data-oriented view of data association has been applied and extended by many authors [5–10]. The most successful multiple target tracking algorithm based on this view is the multiple hypothesis tracker (MHT) [7]. In MHT, each hypothesis associates past observations with a target such that an observation is not shared by more than one target. As a new set of observations arrives, a new set of hypotheses is formed from the previous hypotheses. The construction of new hypotheses requires the enumeration of all possibilities and the size of hypotheses grows exponentially. At each time step, each hypothesis is scored by the probability of having the hypothesis given the observations, i.e., the posterior of the hypothesis. The algorithm returns the hypothesis with the highest score as a solution. MHT is categorized as a deferred logic [10] in which the decision about forming a new track or removing an existing track is delayed until enough observations are collected. Hence, MHT is capable of initiating and terminating a varying number of targets and suitable for surveillance applications in which the tracker is required to initiate and terminate a varying number of tracks autonomously. However, the size of the hypotheses grows exponentially and the enumeration of all hypotheses is not practical. The initial implementation and later extensions proposed several heuristics, such as pruning, gating, clustering and N-scan-back logic, to reduce the complexity of the problem [7, 8]. However, the hueristics are used at the expense of the optimality and the algorithm can still suffer in a dense environment. Furthermore, the running time at each step of the algorithm cannot be bounded easily, making it difficult to be deployed in a real-time surveillance. As a method of pruning, an efficient method of finding k-best hypothesis is developed in [9].

As opposed to finding the optimal association, JPDAF computes the weights of all the possible associations from the *latest* set of observations to the known tracks and clutter [2]. Given an association, the state of a target is estimated by a filtering algorithm and this conditional expectation of state is weighted by the association weight. Then the state of a target is estimated by summing over the weighted conditional expectations. JPDAF is a sequential tracker in which the associations between the known targets and the latest observations are made sequentially and the associations made in the past are not reversible [10]. The sequential trackers are more efficient than deferred logic trackers such as MHT but they are prone to make erroneous associations [10]. At each stage, the association between previously estimated states of targets and the latest observations is optimal from the Bayesian point of view. But the algorithm is suboptimal since the states estimated at each step of algorithm can be different from the states estimated from all observations from the initial to current time. In addition, it is inferred that the exact calculation at each stage is NP-hard [11] since the related problem of finding the permanent of a 0-1 maxtix is #P-complete [12]. Since only the current set of observations are considered in JPDAF, it cannot initiate or terminate tracks. Also JPDAF assumes a fixed number of targets and requires a good initial state for each target. There are restricted extensions to JPDAF to allow the formation of a new track. See [3] for references. Instead of keeping a single Gaussian component for each target, the multisensor multitarget mixture reduction (MTMR) maintains a fixed number of Gaussian components for each target to prevent the loss of information when there are several significant and well spaced componenets [13]. It has been shown that MTMR performs better than JPDAF but at the expense of computation [13]. But MTMR also assumes a fixed number of targets and requires a good set of initial states. The probabilistic multi-hypothesis tracking (PMHT) uses probabilistic associations between observations and targets to avoid the maintenance of a hypothesis tree and the enumeration over all possible associations [14]. PMHT allows an association between a single track and an arbitrary number of observations. This assumption may not represent the physical reality in many cases but reduces the complexity of the problem [15]. However, a fixed known number of targets is assumed and the track initiation and termination are difficult under PMHT. A hypothesis test is presented in [15] as a method to allow a varying number of tracks but the paper also addresses the difficulty with estimating the initial states of new targets.

The data association problem of multiple target tracking formulated under the data-oriented view is also known to be NP-hard [10]. It is a multidimensional assignment problem [10] and the multidimensional assignment problem is NP-hard since its special case 3-dimensional matching is NP-hard [16]. The multiple target tracking problem was first formulated as a combinatorial optimization problem in [4]. Later, heuristics are added to speed up the algorithm [7–9]. An optimization approach to the data association has been applied as a 0-1 integer programming problem [5] and as a multidimensional assignment problem [10]. In both cases one needs to find a feasible set of tracks from all possible tracks,

i.e., all possible partitions of observations, to prevent the exponential explosion and compute the cost of each feasible track, such as the negative log likelihood. Then the optimization routine finds a subset from the feasible tracks such that the combined costs are minimized while satisfying the constraints, i.e., each track has at most one observation at each time and no two tracks share the same observation. The gating method similar to the ones described in [6, 7] is used to find a feasible set of tracks. However, in a dense environment, the size of the feasible tracks can be very large and the complexity of the optimization routine increases dramatically, since the number of parameters in the optimization routine depends on the number of feasible tracks.

1.2 Multiple Target Tracking in Sensor Networks

The surveillance and multiple target tracking is a canonical application of sensor networks as it exhibits different aspects of sensor networks such as event detection, sensor information fusion, communication, sensor management, and decision making. In sensor networks, we seek an autonomous tracking algorithm which does not require a continuous monitoring by a human operator. This requirement excludes the algorithms such as JPDAF, MTMR and PMHT since they are not capable of initiating and terminating tracks. But the large computational cost and memory requirement also prevent us from using MHT on sensor networks. We also need to consider the following constraints on sensor networks. Due to the limited supply of power and a short communication range, the multihop wireless ad-hoc communication is used in sensor networks. In many cases, the communication bandwidth is low and the communication links are not reliable, causing transmission failures. In addition, due to the low communication bandwidth and limited amount of memory, communication delays can occur frequently when the communication load on the network is heavy. It is well known that the communication is costlier than computation in sensor networks in terms of power usage [17]. Hence it is essential to fuse local observations before the transmission. Since the data association problem is NP-hard, we cannot solve it only with local information. But, at the same time, we cannot afford to have a centralized algorithm since such solution cannot be scalable. In summary, we need a simple and efficient tracking algorithm that is capable of initiating and terminating tracks, uses less memory, combines local information to reduce the communication load, and scalable. Also it must be robust against transmission failures and communication delays. But at the same time we want an algorithm that can provide a good solution and improve its solution toward the optimal solution given enough computation time.

The multiple tracking algorithms have been applied to sensor networks. In [18], a distributed tracking algorithm based on MHT is developed for multiple sensors with wide sensing range and large computational power and memory. A method of fusing observations on a fixed network is described and a hierarchical approach to combine tracks is explored. But the approach is not suitable for sensor networks since it demands large computational power and large amount of memory on each sensor. However, if the sensing range of each sensor node

is large, we can apply the algorithm presented in this paper, instead of MHT, to the fusion techniques developed in [18] to achieve an efficient and accurate tracking algorithm. A geometric approach to track a half-plane shadow using sensor networks is developed in [19]. In [20], the distributed track initiation and maintenance methods are described. By electing a leader among the sensors by which a target is detected, unnecessary communications are reduced. But considering the complexity of the data association problem, the approach may suffer from incorrect associations when the false alarm rate is high or when there are many targets moving close to each other. We have adapted their simple local information fusion strategy in our work.

In [21], an efficient sampling-based data association algorithm, Markov chain Monte Carlo (MCMC) data association, is presented. It has been shown that the algorithm is computationally efficient compared to MHT and performs much better than MHT or greedy algorithm when the target detection probability is low [21]. It also performs well in a dense environment, i.e. when there are many targets moving close to each other, and when the false alarm rate is high. The algorithm performs the data association based the current and past observations, hence, it can easily adapt the delayed observations to improve the accuracy of the estimation. The algorithm seeks a maximum a posteriori solution to the data association problem and is capable of initiating and terminating tracks. Furthermore, the algorithm requires less memory as it only maintains the current hypothesis and the hypothesis with the highest posterior. So the enumeration and maintenance of all or some reduced number of hypothesis as in [7–9] are not required. In this paper, we extend the MCMC data association algorithm to sensor networks in a hierarchical manner so that the algorithm becomes scalable.

We consider a simple shortest-path routing scheme on a sensor network. The transmission failures and communication delays of the network are characterized probabilistically. We assume the availability of a small number of special nodes ("supernodes") that are more capable than the other nodes in terms of the computational power and communication range. Each node is assigned to the nearest supernode and nodes are grouped by supernodes. We call the group of sensor nodes formed around a supernode as a tracking group. When a node detects a possible target, it communicates with its neighbors and observations from the neighboring sensors are fused and sent to its supernode. Each supernode receives the fused observations from its tracking group and executes the tracking algorithm. Each supernode communicates with neighboring supernodes when a target moves away from its range. The performance of the algorithm is evaluated in simulations.

The remainder of this paper is structured as follows. The multiple target tracking problem is described in Section 2 and the MCMC data association algorithm for multiple target tracking is presented in Section 3. The sensor network model and the hierarchical tracking method is given in Section 4. The simulation results are shown in Section 5.

2 Multiple Target Tracking Problem

Let $T \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the unknown number of objects moving around the surveillance region $\mathcal{R}$ for some duration $[t_i^k, t_f^k] \subset [1, T]$ for $k = 1, \ldots, K$. Let V be the volume of $\mathcal{R}$. Each object arises at a random position in $\mathcal{R}$ at t_i^k, moves independently around $\mathcal{R}$ until t_f^k and disappears. The number of objects arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $(\lambda_b V)$ where λ_b is the birth rate of new objects per unit volume for the duration of surveillance. The initial position of a new object is uniformly distributed over $\mathcal{R}$.

Let $F^k : \mathbb{R}^d \to \mathbb{R}^d$ be the discrete-time dynamics of the object k, where d is the dimension of the state variable, and let $x_t^k \in \mathbb{R}^d$ be the state of the object k at time t for $k = 1, 2, \ldots, K$. The object k moves according to

$$x_{t+1}^k = F^k(x_t^k) + w_t^k \qquad \text{for } t = t_i^k, \ldots, t_f^k - 1, \tag{1}$$

where $w_t^k \in \mathbb{R}^d$ are white noise processes. Note that F^k can be the same for all or some of the objects. The noisy observation about the state of the object is measured with the detection probability p_d which is less than unity. There are also false alarms and the number of false alarms has a Poisson distribution with a parameter $(\lambda_f V)$ where λ_f is the false alarm rate per unit time, per unit volume. Let n_t be the number of observations at time t which includes both noisy observations and false alarms. Let $y_t^j \in \mathbb{R}^m$ be the j-th observation at time t for $j = 1, \ldots, n_t$, where m is the dimensionality of each observation vector. Each object generates a unique observation at each sampling time if it is detected. Let $H^j : \mathbb{R}^d \to \mathbb{R}^m$ be the observation model. Then the observations are generated as follows:

$$y_t^j = \begin{cases} H^j(x_t^k) + v_t^j \text{ if } j\text{-th observation is from } x_t^k \\ u_t \qquad\qquad\quad \text{otherwise,} \end{cases} \tag{2}$$

where $v_t^j \in \mathbb{R}^m$ are white noise processes and $u_t \sim \text{Unif}(\mathcal{R})$ is a noise process for false alarms. Notice the with probability $1 - p_d$ the object is not detected and we call this a missing observation.

Under the data-oriented approach, the multiple target tracking problem is to partition the observations such that the posterior is maximized. The posterior function is described in (6).

3 MCMC Data Association Algorithm

Let us first specify the dynamic and measurement models. Here we use the usual linear system model but the method can be easily extended to non-linear models coupled with a non-linear regression. If an object is observed k times at $t_1, t_2, \ldots, t_k$, its dynamic and measurement models can be expressed as:

$$\begin{aligned} x_{t_{i+1}} &= A(t_{i+1} - t_i)x_{t_i} + G(t_{i+1} - t_i)w_{t_i} \qquad \text{for } i = 1, \ldots, k-1 \\ y_{t_i} &= Cx_{t_i} + v_{t_i} \qquad \text{for } i = 1, \ldots, k, \end{aligned} \tag{3}$$

where w_{t_i} and v_{t_i} are white Gaussian noises with zero mean and covariance Q and R, respectively. $A(\cdot)$, $G(\cdot)$, and C are matrices with appropriate sizes. The entries of the matrix $A(t_{i+1} - t_i)$ and $G(t_{i+1} - t_i)$ are determined by the sampling interval $t_{i+1} - t_i$ for each i. For clarity, the subsequence notation for the time index is suppressed for now. Then applying the Kalman filter, the update equations are

$$\begin{aligned} \bar{x}_{t+1} &= A\hat{x}_t \\ \bar{P}_{t+1} &= A\hat{P}_t A^T + GQG^T \end{aligned} \tag{4}$$

and the measurement update equations are

$$\begin{aligned} \hat{x}_t &= \bar{x}_t + K_t(y_t - C\bar{x}_t) \\ \hat{P}_t &= \bar{P}_t - \bar{P}_t C^T B_t^{-1} C\bar{P}_t, \end{aligned} \tag{5}$$

where $K_t = \hat{P}_t C^T R^{-1}$ is the Kalman gain matrix and $B_t = C\bar{P}_t C^T + R$.

Let $y_t = \{y_t^j : j = 1, \ldots, n_t\}$ and $Y = \bigcup_{t\in\{1,\ldots,T\}} y_t$. Let Ω be a collection of partitions of Y such that, for $\omega \in \Omega$,

1. $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$;
2. $\bigcup_{k=0}^{K} \tau_k = Y$ and $\tau_i \cap \tau_j = \emptyset$ for $i \neq j$;
3. τ_0 is a set of false alarms;
4. $|\tau_k \cap y_t| \leq 1$ for $k = 1, \ldots, K$ and $t = 1, \ldots, T$; and
5. $|\tau_k| > 1$ for $k = 1, \ldots, K$.

Here K is the number of tracks for the given partition $\omega \in \Omega$. We call τ_k a track when there is no confusion although the actual track is the set of estimated states from observations τ_k. However, we assume there is a deterministic function that returns a set of estimated states given the set of observation, so no distinction is required. We denote by $\tau_k(t)$ the observation at time t that is assigned to the track τ_k. Notice that $\tau_k(t)$ can be empty if it is a missing observation or if t is before the appearance time or after the disappearance time of the associated target. The fourth requirement says that a track can have at most one observation at each time, but, in the case of multiple sensors, we can easily relax this requirement to allow multiple observations per track. A track is assumed to contain at least two observations since we cannot distinguish a track with a single observation from a false alarm. Let $n_{\mathrm{d}}(t)$ be the number of detected objects at time t, i.e., $n_{\mathrm{d}}(t) = |\{\tau_k(t) : 1 \leq k \leq K\}|$, and let $n_{\mathrm{d}} = \sum_{t=1}^{T} n_{\mathrm{d}}(t)$ be the total number of detections. Similarly, let $n_{\mathrm{f}}(t)$ be the number of false alarms at time t, i.e., $n_{\mathrm{f}}(t) = |\tau_0(t)|$, and let $n_{\mathrm{f}} = \sum_{t=1}^{T} n_{\mathrm{f}}(t)$ be the total number of false alarms.

In [22], the exchangeability of $\omega \in \Omega$ is assumed and a uniform prior on ω is used. However, it is no longer true when there are a varying number of objects with false alarms and missing observations. We instead assume the exchangeability among ω's with the same number of tracks, false alarms and missing observations. Once a partition $\omega \in \Omega$ is chosen, the tracks $\tau_1, \ldots, \tau_K \in \omega$ and a set of false alarms $\tau_0 \in \omega$ are completely determined. Hence, for each track, we can

estimate the state of the object independently since each object moves independently from other objects. For each track $\tau \in \omega$, we apply the Kalman filter (4,5) to estimate the states $\hat{x}_t(\tau)$ and covariances $B_t(\tau)$. Let $n_{\mathrm{u}} = \sum_{t=1}^{\mathrm{T}} K - n_{\mathrm{d}}(t)$ and $\mathcal{N}(\mu, \Sigma)$ be the Gaussian density function with mean μ and covariance matrix Σ. It can be shown that the posterior of ω is [21]:

$$P(\omega|Y) = \frac{1}{Z}\frac{1}{K!}p_{\mathrm{d}}^{n_{\mathrm{d}}}(1-p_{\mathrm{d}})^{n_{\mathrm{u}}}\lambda_{\mathrm{f}}^{n_{\mathrm{f}}}\lambda_{\mathrm{b}}^{K} \times \prod_{\tau\in\omega^1}\prod_{i=1}^{|\tau|-1}\mathcal{N}(\tau(t_{i+1}) - A(t_{i+1}-t_i)\hat{x}_{t_i}(\tau), B_{t_{i+1}}(\tau)), \tag{6}$$

where Z is a normalizing constant and $\hat{x}_{t_i}(\tau)$ denotes the estimated state of the track τ at time t_i and $B_{t_i}(\tau) = C\bar{P}_{t_i}(\tau)C^{\mathrm{T}} + R$ for the track τ. Now the goal is to find a partition of observations such that $P(\omega|Y)$ is maximized.

Now we describe the Markov chain Monte Carlo (MCMC) data association algorithm. The set Ω becomes a state space of the MCMC sampler and we sample from Ω such that its stationary distribution is the posterior $P(\omega|Y)$. If we are at state $\omega \in \Omega$, we propose $\omega' \in \Omega$ following the proposal distribution $q(\omega, \omega')$. The move is accepted with an acceptance probability $A(\omega, \omega')$ where

$$A(\omega, \omega') = \min\left(1, \frac{P(\omega'|Y)q(\omega', \omega)}{P(\omega|Y)q(\omega, \omega')}\right), \tag{7}$$

otherwise the sampler stays at ω, so that the detail balance is satisfied. If we make sure that the chain is irreducible and aperiodic, then the chain converges to its stationary distribution [23]. For more information about MCMC, see [23, 24]. In order to make the algorithm more efficient, we make two additional assumptions: (1) the directional spead of any target in $\mathcal{R}$ is less than $\bar{v}$; and (2) the number of consecutive missing observations of any track is less than $\bar{d}$. The first assumption is reasonable in a surveillance scenario since, in many cases, the maximal speed of a vehicle is generally known based on the vehicle type and terrain conditions. The second assumption is a user defined parameter and it can be used as one of the criteria to distinguish an event of a new object's appearance from an event of a continuation of an existing object. We will now assume that these two new conditions are added to the definition of Ω so that each element $\omega \in \Omega$ satisfies two assumptions above as well as the five conditions described earlier.

The sampler consists of five types of moves. They are (1) a birth/death move pair; (2) a split/merge move pair; (3) an extension/reduction move pair; (4) a track update move; and (5) a track switch move. So there are a total of 8 moves. The paired move types are used to guarantee the reversibility of the chain. Let m be the move and it is chosen randomly from the distribution $\xi_K(m)$ where K is the number of tracks of the current partition ω. Assume that we index each move by an integer such that $m = 1$ for a birth move, $m = 2$ for a death move and so on. When there is no track, we can only propose a birth move, so we set $\xi_0(m = 1) = 1$ and 0 for all other moves. When there is only a single target, we cannot propose a merge or track switch move, so $\xi_1(m = 4) = \xi_1(m = 7) = 0$. If

it is desired, although not required, we can limit K and assign zero probability of proposing a birth or split move when we are at the limit of K. The algorithm is shown in Figure 1 and the details of each move are given in [21].

Algorithm 1 (MCMC Data Association) :
Input: $Y, n_{\text{mc}}, \omega_{\text{init}}$
Output: $\hat{\omega} = \arg\max_{n=1}^{n_{\text{mc}}} p(\omega(n)|Y)$

$\omega \leftarrow \omega_{\text{init}}$
for $n = 1$ to n_{mc}
  choose a move m according to the distribution ξ_K
  propose ω' based on the move m and current state ω
  sample U from Unif[0, 1]
  if $U < A(\omega, \omega')$
    $\omega \leftarrow \omega'$
  end
  $\omega(n) \leftarrow \omega$
end

Fig. 1. MCMC Data Association Algorithm

In Algorithm 1, the acceptance probability is defined in (7) in which the posterior (6) is used. Since, in the track update move, there is always a positive probability of staying at the current state, the chain is aperiodic. The chain is also irreducible since it has a positive probability of visiting every element of Ω. Since the transitions described in Algorithm 1 satisfy the detailed balance, by the ergodic theorem, the chain converges to its stationary distribution.

The extension of the MCMC tracker to an online, real-time tracking is a trivial task. It has been shown that the MCMC tracker works well even when T is small [21]. Hence we implement a sliding window of size w_s using Algorithm 1. Suppose that we are at time t and receiving new observations, we use the previous estimate to initialize the MCMC tracker and run the MCMC tracker on the observations from $t - w_s + 1$ to t, which combines both previous and new observations. This approach makes it easier to incorporate delayed observations in sensor networks.

4 Sensor Network Model

In this section, we describe the sensor network and sensor model used for simulations in Section 5. Let N_s be the number of sensor nodes, including both supernodes and regular nodes, deployed over the surveillance region $\mathcal{R} \subset \mathbb{R}^2$.

We assume that supernodes have more computational power and its communication range is long enough so it can communicate with neighboring supernodes. Let $s_i \in \mathcal{R}$ be the location of the i-th sensor node and let $S = \{s_i : 1 \le i \le N_s\}$. Let $R_t \in \mathbb{R}$ be the transmission range of regular sensors. A pair of sensor nodes i and j can communicate to each other if the Euclidean distance $\|s_i - s_j\| \le R_t$. Let $G = (S, E)$ be a communication graph such that $(s_i, s_j) \in E$ if and only if $\|s_i - s_j\| \le R_t$. Note that the communication links between supernodes and (possible) communication links from supernodes to regular nodes are not represented in this communication graph G. Let $N_{ss} \ll N_s$ be the number of supernodes and let $s_j^s \in S$ be the position of the j-th supernode, for $j = 1, \ldots, N_{ss}$. Let $g : \{1, \ldots, N_s\} \to \{1, \ldots, N_{ss}\}$ be the assignment of each sensor to its nearest supernode such that $g(i) = j$ if $\|s_i - s_j^s\| = \min_{k=1,\ldots,N_{ss}} \|s_i - s_k^s\|$. For a node i, if $g(i) = j$, then the shortest path from s_i to s_j^s in G is denoted by $sp(i)$.

Let $R_s \in \mathbb{R}$ be the sensing range. If there is an object at $x \in \mathcal{R}$, each sensor within radius R_s from x detects the presence of the object with the detection probability p_d. The detection of an object by the sensor i is recorded by the sensor sensibility z_i such that

$$z_i = \frac{\beta}{\|s_i - x\|^\alpha}(1 + .1e_i), \tag{8}$$

where β and α are constants specific to the sensor type and e_i is a Gaussian noise with zero mean and variance of 1. This sensor model is based on the general sensibility model used in [25] but we have added a Gaussian noise that is proportional to the signal strength. If $z_i \le 0$, we assume there is no detection. For each i, if z_i is positive, the node transmits z_i to its neighboring nodes, which are at most $2R_s$ away from s_i, and listens to incoming transmissions from its $2R_s$ neighborhood. Note that this approach is similar to the leader election scheme in [20]. However, this approach may cause some missing observations if there is more than one object in this disk of radius $2R_s$. A better approach to fuse local data is required and we will address this issue in our future work. For the node i, if z_i is the larger than all incoming transmissions, $z_{i_1}, \ldots, z_{i_{k-1}}$, and $z_{i_k} = z_i$, then the position of the object is estimated as

$$\hat{z}_i = \frac{\sum_{j=1}^k z_{i_j} s_{i_j}}{\sum_{j=1}^k z_{i_j}}. \tag{9}$$

Then $\hat{z}_i$ is transmitted to the supernode $g(i)$ via the shortest path $sp(i)$. If z_i is not the largest compared to the incoming transmissions, the node i does nothing and goes back to the sensing mode.

A transmission along the edge (s_i, s_j) fails independently with probability p_e^t, and the message never reaches a supernode. So we can consider the transmission failure as another form of the detection failure. Let $k_{\max} = \max_{1 \le i \le N_s} |sp(i)|$, where $|sp(i)|$ is the number of edges between s_i and $s_{g(i)}^s$, and $n_k = \#\{i : |sp(i)| = k, 1 \le i \le N_s\}$. Assume the sensors are uniformly distributed and let $\bar{n} = N_s \frac{\pi R_s^2}{V}$, the average number of sensors in a disk of radius R_s. Then the

effective probability of detection becomes

$$p_{\mathrm{d}}^{\mathrm{eff}} = p_{\mathrm{d},\bar{n}} \left(\sum_{k=0}^{k_{\max}} (1 - p_{\mathrm{e}}^{\mathrm{t}})^k \frac{n_k}{N_{\mathrm{s}}} \right), \tag{10}$$

where $p_{\mathrm{d},\bar{n}} = 1 - (1 - p_{\mathrm{d}})^{\bar{n}}$ is the probability that at least one of the sensors in a disk of radius R_{s} detects an object when the object is placed at the center of the disk.

The communication delay is modeled by the negative binomial distribution. We assume each node has the same probability $p_{\mathrm{e}}^{\mathrm{d}}$ of delaying a message. If d_i is the number of delays occurred on the message originating from the sensor i, d_i is distributed as

$$p(d_i = d) = \binom{|sp(i)| + d - 1}{d} (1 - p_{\mathrm{e}}^{\mathrm{d}})^{|sp(i)|} (p_{\mathrm{e}}^{\mathrm{d}})^d. \tag{11}$$

If the network is heavily loaded, the independence assumptions on the transmission failure and communication delay may not hold. Also the described model uses the shorted path routing and it does not capture the possible benefits from using the multipath routing such as the directed diffusion [26]. However, the model is realistic under the moderate conditions and we have chosen it for its simplicity.

The supernodes maintains a set of observations $Y = \{y_t^j : t_{\mathrm{curr}} - w_{\mathrm{s}} < t \leq t_{\mathrm{curr}}, 1 \leq j \leq n_t\}$, where t_{curr} is the current time and w_{s} is the window size. Each y_t^j is a fused observation $\hat{z}_i$ for some sensor i. At time $t+1$, the observations at time $t_{\mathrm{curr}} - w_{\mathrm{s}}$ are removed from Y and a new set of observations is appended to Y. Any delayed observations are appended to appropriate slots. Then each supernode initialize the Markov chain with the previously estimated tracks and executes Algorithm 1 on Y. Once tracks are found, the next state of each track is predicted. If the predicted next state belongs to the surveillance area of another supernode, the track information is passed to the corresponding supernode. The newly received tracks are incorporated into the initial state of the Markov chain for the next time step.

5 Simulation Results

For simulations below, we consider the surveillance over a rectangular region on a plane, $\mathcal{R} = [0, 200]^2$. The state vector is $x = [x, y, \dot{x}, \dot{y}]^T$ where (x, y) is a position in $\mathcal{R}$ along the usual x and y axes and $(\dot{x}, \dot{y})$ is the velocity vector. The linear system model (3) is used [27] where δ is an interval between observations and

$$A(\delta) = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad G(\delta) = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix} \qquad \text{and} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The covariance matrices are

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} \frac{R_s^2}{25} & 0 \\ 0 & \frac{R_s}{25} \end{bmatrix}.$$

In simulations, 1600 sensor nodes are distributed uniformly over $\mathcal{R}$ (see Figure 2). There are four supernodes. Imagine that $\mathcal{R}$ is divided into four equal size quadrants. We place each supernode close to the center of each quadrant of $\mathcal{R}$. Roughly 400 sensor nodes form a tracking group around each supernode. The transmission and sensing ranges are $R_t = 10$ and $R_s = 5$, respectively. For the sensor model, we use $\alpha = 3$ and $\beta = 1$ and the sensor readings are bounded above by 125. The exponent $\alpha = 3$ is a reasonable assumption for a magnetometer. The maximal directional speed is set to $\bar{v} = 20$. The maximal number of consecutive missing observations is set to $\bar{d} = 3$ for simulations in Section 5.1 and 5.2 and $\bar{d} = 5$ in Section 5.3. Observations with delay longer than 4 are treated as missing observations. We first study the effects of the transmission failures and communication delays on the performance of the tracking algorithm. Then an example of surveillance using sensor networks is demonstrated in Section 5.3.

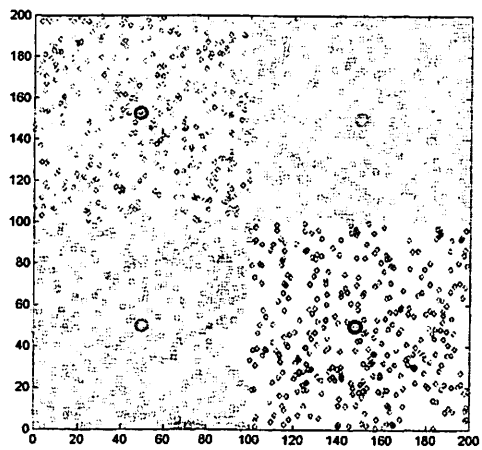


Fig. 2. Uniformly Distributed Sensor Network: regular sensor nodes (small squares and diamonds) and supernodes (large circles)

In all simulations, the online version of Algorithm 1 is used with $w_s = 10$ and the window is forward by a single step. Since we are working in a simulation environment, we measure the performance of the algorithm by comparing the tracks estimated by the algorithm against the tracks constructed from true associations for the corresponding window. The true associations are based on

the actual tracks of targets from which simulations are generated. We will call those tracks constructed from true associations as "best" tracks. Our experience shows that it requires at least three or four observations from a single target before a track is initiated. So when constructing the best tracks, we only consider tracks with more than three observations. The tracks are compared by the average estimation error, average number of tracks, and average log posterior and they are computed at each time on the window of size w_s. The computations of the average number of tracks and average log posterior are trivial. So we only describe how the average estimation error is computed. Since the number of tracks are not fixed and they have variable lengths, a direct computation of the estimation error is not possible. So we propose a different way to compute the estimation error. Suppose we are given a set of observations $Y = \{y_t^j : t_{\text{curr}} - w_s < t \leq t_{\text{curr}}, 1 \leq j \leq n_t\}$. Let ω^* be the true tracks from which simulations are generated. Let ω be either the tracks estimated by the algorithm or tracks constructed from true associations. For each track $\tau \in (\omega \setminus \tau_0)$, let $\epsilon(\tau)$ be the estimation error of the track τ and let $f = |\tau|$. Let $s_t(\tau)$ be the position, i.e., the first two componenets of the state parameter, of the track τ at time t. For $s_{t_1}(\tau)$, find $\tau^* \in (\omega^* \setminus \tau_0^*)$ such that $\|s_{t_1}(\tau^*) - s_{t_1}(\tau)\|$ is minimized. If such τ^* is not found, i.e., when there is no track in $\omega*$ at time t_1, $\epsilon(\tau) = c_\epsilon f$, where c_ϵ is a constant. Otherwise, the estimation error of track τ is computed as

$$\epsilon(\tau) = \sum_{j=1}^{f} \begin{cases} \|s_{t_j}(\tau^*) - s_{t_j}(\tau)\| & \text{if } t_j \leq t_{|\tau^*|} \\ c_\epsilon & \text{otherwise.} \end{cases}$$

Then the estimation error is defined as

$$\epsilon = \frac{1}{Z} \sum_{\tau \in (\omega \setminus \tau_0)} \epsilon(\tau),$$

where $Z = \sum_{\tau \in (\omega \setminus \tau_0)} |\tau|$. We used $c_\epsilon = 10$ in simulations below.

5.1 Transmission Failures

In order to study the effects of transmission failures alone, we assume that there are no delayed observations, no false alarms, and no missing detections. However, we use $\lambda_f = 1$ and, for each sensor, $p_d = .99$ for log posterior calculations. We first generated a random scenario with eight targets and the surveillance duration of $T = 20$. Then we applied different values of transmission failure rates, $0, .01, .02, .03, .04, .05, .1, .15$, and $.2$, and generated a total of nine test cases. The average effective detection proabilities at different values of transmission rates are shown in Figure 3, where $p_d = 1$ is assumed. The effective detection probability decays fast for a small increase in the transmission failure rate. For example, for $p_e^t = .15$, less than a half of observations are delivered to a supernode, making the tracking task very challenging. We ran Algorithm 1 with two different sample sizes, 1,000 and 5,000. For each sample size and test case, ten repeated runs are used for computing the average values. An example of a test

case is shown in Figure 4. The accumulated best tracks and tracks estimated from the algorithm are shown Figure 5. Figure 6 shows the average estimation error and the average number of detected targets at different values of transmission failure rates. The average number of targets is larger than 8 since a single track can be shared by more than one supernode. The average log posteriors at different values of transmission failure rates are shown in Figure 7. Figure 6(a) shows that the estimation error of the algorithm is close to the estimation error of the best tracks. Figure 7 shows that the log posteriors of the estimated tracks are very close to that of the best tracks. However, Figure 6(b) shows the number of targets decreases dramatically for $p_e^t > .1$ and even the best tracks perform badly. This is due to the low effective detection rate. But the average number of estimated tracks is very close to that of the best tracks. This simulation result indicates that tracking is not possible if the transmisstion rate is larger than .1. We can overcome this problem by transmitting redundant observations to a supernode where the number of redundant observations are based on the length of links from the sensing node to its supernode. However, we need to make sure that the redundancy does not overflow the communication load of the network.

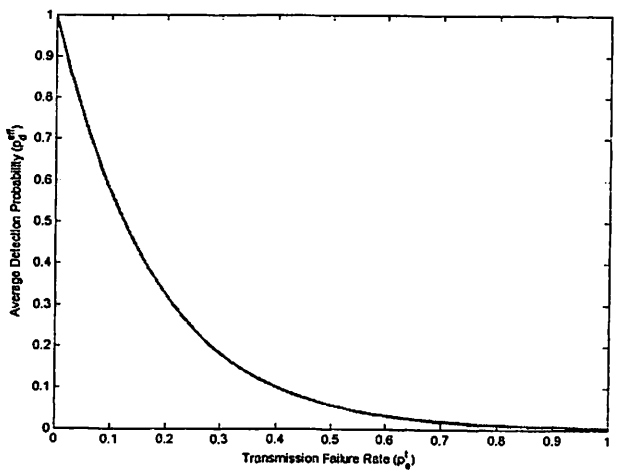


Fig. 3. Average effective detection probability vs. transmission failure rate ($p_d = 1$)

5.2 Communication Delays

To assess the effects of communication delays, we assume that there are no transmission failures, no false alarms, and no missing observations. We use $\lambda_f = 1$ and, for each sensor, $p_d = .99$ for log posterior calculations. Based on the

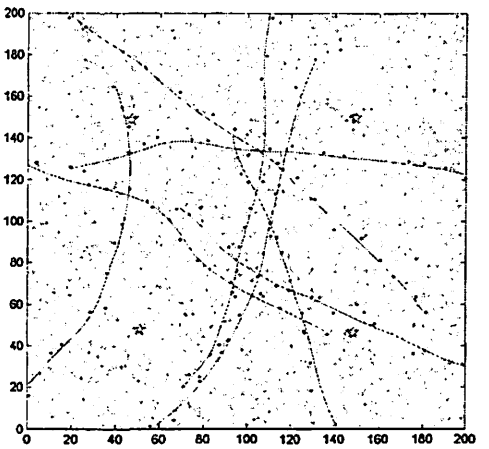
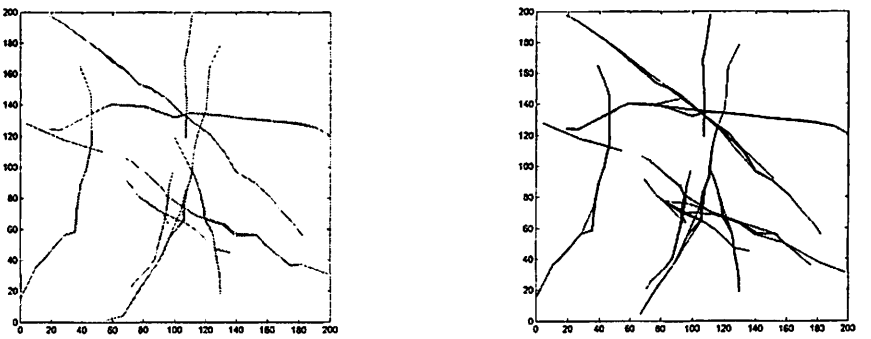


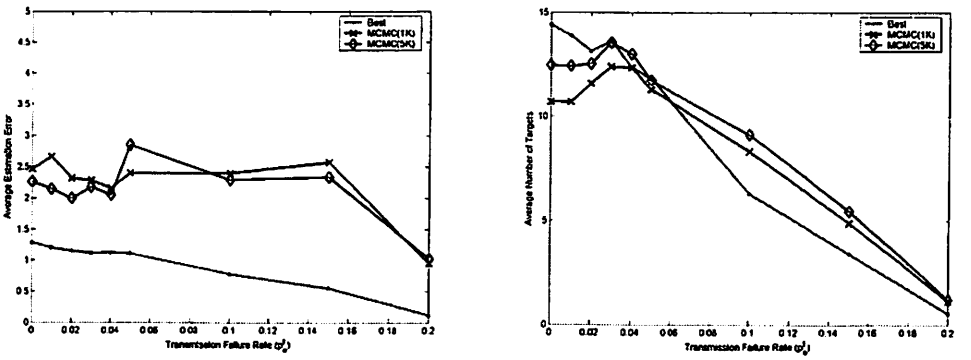
Fig. 4. A test case with $p_e^t = .05$, $p_e^d = 0$ with accumulated observations (dots), true tracks (solid line), sensor nodes (small circles)



(a) (b)

Fig. 5. A test case with $p_e^t = .05$, $p_e^d = 0$: (a) accumulated best tracks; (b) accumulated estimated tracks

same scenario used in Section 5.1, we applied different values of communication delay rates, 0, .05, .1, .15, .2, .25, and .3, and generated a total of seven test cases. Table 1 shows the distribution of delays at different communication delay rates. Note the number of delays are counted from observations after $t = 11$. Also notice that as we increase the delay rate, we lose some observations since observations with delays larger than 4 are treated as missing observations. We ran Algorithm 1 with two different sample sizes, 1,000 and 5,000. For each sample size and test



(a) (b)

Fig. 6. Transmission failure rates vs. (a) average estimation error; (b) average number of targets

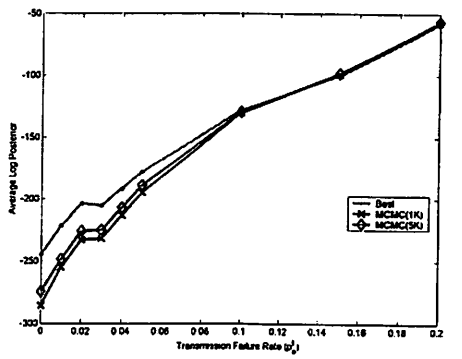


Fig. 7. Transmission failure rates vs. average log posterior

case, ten repeated runs are used for computing the average values. An example of a test case is shown in Figure 8. The accumulated best tracks and tracks estimated from the algorithm are shown Figure 9. Figure 10 shows the average estimation error and the average number of detected targets at different values of communication delay rates. The average log posteriors at different values of communication delay rates are shown in Figure 11. Due to the communication delay, the average number of estimated tracks is lower than the average number of the best tracks. However, the estimation error of the estimated tracks is close to that of the best tracks. It also shows more samples increase the average number of estimated tracks.

Table 1. Delay counts at different values of p_e^d

Delay rate	Number of observations with delay(s)				
p_e^d	0	1	2	3	4
0.00	16	0	0	0	0
0.05	12	3	0	0	0
0.10	9	3	2	1	1
0.15	6	4	3	2	0
0.20	3	6	2	3	1
0.25	1	1	4	5	2
0.30	4	2	2	2	4

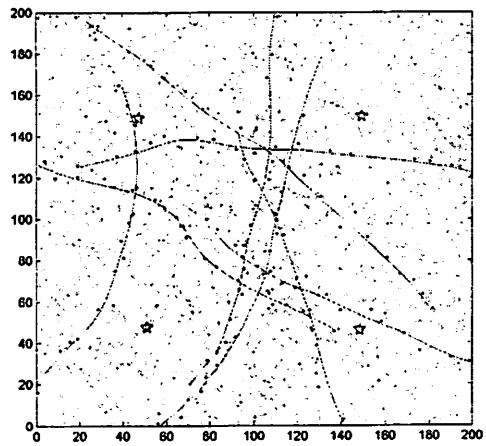
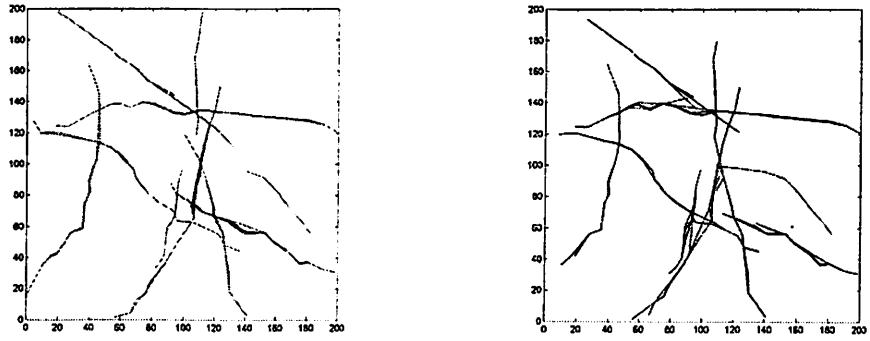


Fig. 8. A test case with $p_e^t = 0$, $p_e^d = .2$ with accumulated observations (dots), true tracks (solid line), sensor nodes (small circles)

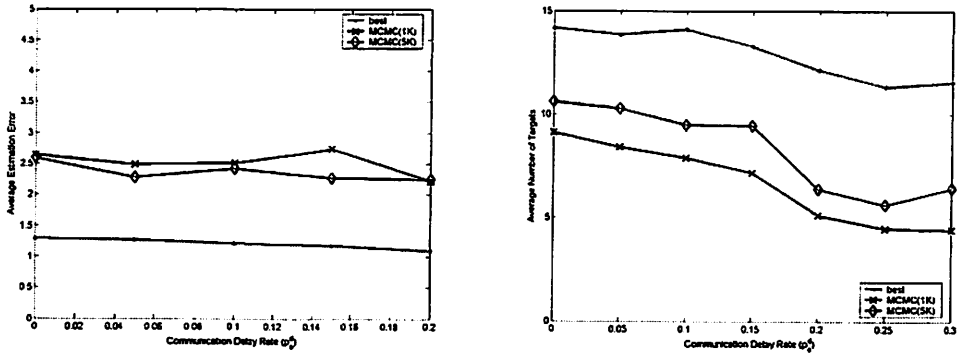
5.3 An Example of Surveillance with Sensor Networks

The same sensor network as in Section 5.1 and 5.2 is used, except $p_e^t = .05$, $p_e^d = .1$, $\lambda_f = 5$, and, for each sensor, $p_d = .9$. The surveillance duration is increased to $T = 100$. We ran Algorithm 1 with three different sample sizes, 1,000, 5,000, and 10,000. For each sample size, ten repeated runs are used for computing the average values. The accumulated best tracks and tracks estimated from the algorithm are shown in Figure 12. Figure 13 shows the average estimation error and the average number of detected targets at each time step. The average log posteriors at each time step are shown in Figure 14. The average number of tracks estimated from the algorithm is very close to the average number of the best tracks. It shows that the log posteriors of the estimated tracks and the best tracks are almost identical. The algorithm is written in Matlab and run on PC



(a) (b)

Fig. 9. A test case with $p_e^t = 0$, $p_e^d = .2$: (a) accumulated best track; (b) accumulated estimated track



(a) (b)

Fig. 10. Communication delay rate vs. (a) average estimation error; (b) average number of targets

with a 2-GHz Intel Pentium 4 processor. It took 1.21 seconds per supernode, per simulation step for 1,000 samples, 6.00 seconds for 5,000 samples, and 12.01 seconds for 10,000 samples.

6 Conclusions

In this paper, a scalable hierarchical multiple target tracking algorithm for sensor networks is presented. The algorithm is based on the efficient MCMC data association algorithm and it is suitable for autonomous surveillance in sensor networks since the algorithm can initiate and terminate tracks, handle delayed

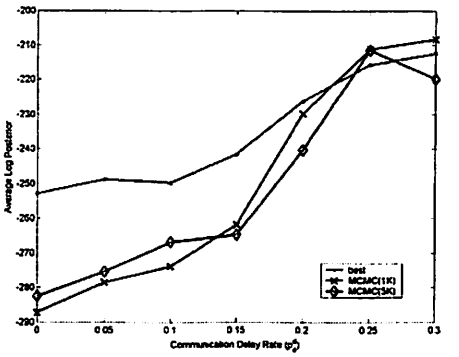
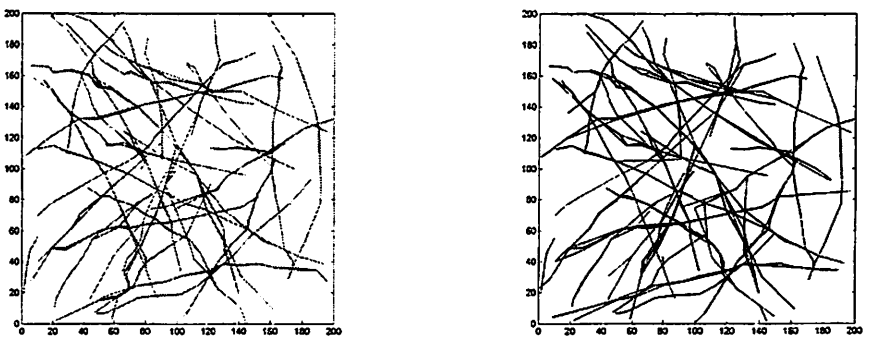


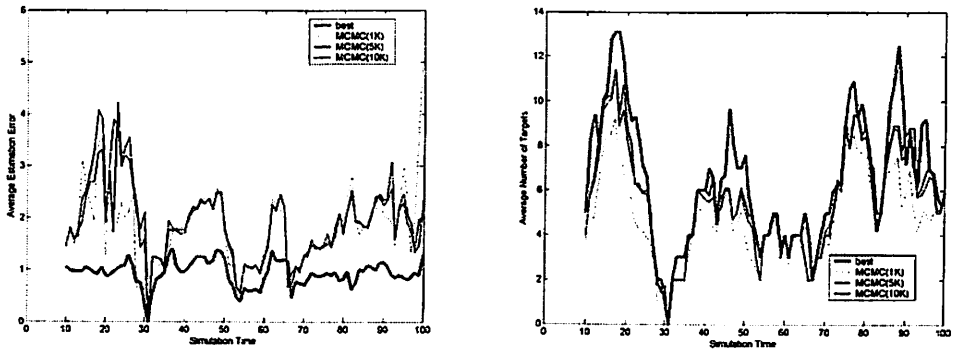
Fig. 11. Communication delay rate vs. average log posterior



(a) (b)

Fig. 12. The accumulated best and estimated tracks: (a) accumulated best track; (b) accumulated estimated track

observations, and require a small amount of memory. The algorithm is also robust against transmission failures. We consider a simple shortest-path routing scheme on a sensor network. The transmission failures and communication delays of the network are characterized probabilistically. A number of sensors form a tracking group around a supernode and tracking is performed on the observation received from the same tracking group or passed from neighboring supernodes. In order to reduce the communication overhead, the observations are first locally fused and then transmitted to its supernode. This hierarchical approach allows us to solve the global data association problem while keeping the approach scalable for a larger sensor network. It has been shown that tracking is not possible for the naive setting used in out simulations if the transmission failure rate is larger than .1. The transmission failure poses a more challenging problem than the



(a) (b)

Fig. 13. (a) Average estimation error; (b) average number of targets

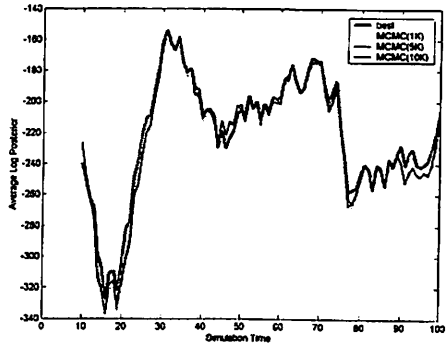


Fig. 14. Average log posterior

communication delay. However, this difficulty with transmission failures can be overcomed by transmitting redundant observations. But we need to make sure that such strategy does not exceed the capacity of the network.

References

1. Estrin, D., Culler, D., Pister, K., Sukhatme, G.: Connecting the physical world with pervasive networks. In: IEEE Pervasive Computing. Volume 1(1). (2002) 59–69
2. Bar-Shalom, Y., Fortmann, T.: Tracking and Data Association. Mathematics in Science and Engineering Series 179 Academic Press, San Diego, CA (1988)
3. Cox, I.: A review of statistical data association techniques for motion correspondence. In: International Journal of Computer Vision. Volume 10(1). (1993) 53–66

4. Sittler, R.: An optimal data association problem on surveillance theory. In: IEEE Trans. on Military Electronics. Volume MIL-8 (Apr). (1964) 125–139
5. Morefield, C.L.: Application of 0-1 integer programming to multitarget tracking problems. In: IEEE Trans. on Automatic Control. Volume 22. (1971) 3:302–312
6. Stein, J., Blackman, S.: Generalized correlation of multi-target track data. In: IEEE Trans. Aerosp. Electron. Syst. Volume AES-11(6). (1975) 1207–1217
7. Reid, D.: An algorithm for tracking multiple targets. In: IEEE Transaction on Automatic Control. Volume 24(6). (1979) 843–854
8. Kurien, T.: Issues in the design of practical multitarget tracking algorithms. In Bar-Shalom, Y., ed.: Multitarget-Multisensor Tracking: Advanced Applications. Artech House: Norwood, MA (1990)
9. Cox, I., Hingorani, S.: An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. In: International Conf. on Pattern Recognition. (1994) 437–443
10. Poore, A.: Multidimensional assignment and multitarget tracking. In: Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Volume 19. (1995) 169–196
11. Collins, J., Uhlmann, J.: Efficient gating in data association with multivariate distributed states. In: IEEE Trans. Aerospace and Electronic Systems. Volume 28(3). (1992)
12. Valiant, L.: The complexity of computing the permanent. In: Theoretical Computer Science. Volume 8. (1979) 189–201
13. Pao, L.: Multisensor multitarget mixture reduction algorithms for tracking. In: Proc. AIAA Guidance, Navigation, and Control Conf., Monterey, CA (1993) 28–37
14. Streit, R., Luginbuhl, T.: Maximum likelihood method for probabilistic multihypothesis tracking. In: Proc. SPIE. Volume 2235. (1994) 394–405
15. Hue, C., Le Cadre, J.P., Perez, P.: Sequential monte carlo methods for multiple target tracking and data fusion. In: IEEE Trans. on Signal Processing. Volume 50(2). (2002) 309–325
16. Papadimitriou, C., Steiglitz, I.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ (1982)
17. Doherty, L., Warneke, B.A., Boser, B., Pister, K.S.J.: Energy and performance considerations for smart dust. In: International Journal of Parallel and Distributed Sensor Networks. (2001)
18. Chong, C., Mori, S., Chang, K.: Multitarget multisensor tracking. In Bar-Shalom, Y., ed.: Multitarget-Multisensor Tracking: Advanced Applications. Artech House: Norwood, MA (1990) 247–295
19. Liu, J., Cheung, P., Guibas, L., Zhao, F.: A dual-space approach to tracking and sensor management in wireless sensor networks. In: ACM International Workshop on Wireless Sensor Networks and Applications Workshop, Atlanta (2002)
20. Liu, J., Liu, J., Reich, J., Cheung, P., Zhao, F.: Distributed group management for track initiation and maintenance in target localization applications. In: Proc. of 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03). (2003)
21. Oh, S., Sastry, S.: Mcmc data association for multiple target tracking. In: UCB Technical Report. (2003) (in preparation)
22. Pasula, H., Russell, S.J., Ostland, M., Ritov, Y.: Tracking many objects with many sensors. In: IJCAI-99, Stockholm (1999)
23. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to mcmc for machine learning, machine learning. In: Machine Learning. Volume 50. (2003) 5–43

24. Beichl, I., Sullivan, F.: The metropolis algorithm. In: Computing in Science and Engineering. Volume 2(1). (2000) 65–69
25. Meguerdichian, S., Koushanfar, F., Qu, G., Potkonjak, M.: Exposure in wireless ad hoc sensor networks. In: Procs. of 7th Annual International Conference on Mobile Computing and Networking. (2001) 139–150
26. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: In Proc. of 6th Annual International Conference on Mobile Computing and Networks. (2000)
27. Li, X.R., Jilkov, V.P.: A survey of maneuvering target tracking. In: In SPIE: Signal and Data Processing of Small Targets. Volume 4048-22. (2000)