Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems

Songhwai Oh, Stuart Russell, and Shankar Sastry

Abstract— **In this paper, we consider the general multiple-target tracking problem in which an unknown number of targets appears and disappears at random times and the goal is to find the tracks of targets from noisy observations. We propose an efficient real-time algorithm that solves the data association problem and is capable of initiating and terminating a varying number of tracks. We take the data-oriented, combinatorial optimization approach to the data association problem but avoid the enumeration of tracks by applying a sampling method called Markov chain Monte Carlo (MCMC). The MCMC data association algorithm can be viewed as a "deferred logic" method since its decision about forming a track is based on both current and past observations. At the same time, it can be viewed as an approximation to the optimal Bayesian filter. The algorithm shows remarkable performance compared to the greedy algorithm and the multiple hypothesis tracker (MHT) under extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates.**

I. INTRODUCTION

Multiple-target tracking plays an important role in many areas of engineering such as surveillance, computer vision, and signal processing [1], [5]. Under the most general setup, a varying number of indistinguishable targets is moving continuously in a given region and the positions of moving targets are sampled at random intervals. The measurements of the positions are noisy, with detection probability less than one, and there is a noise background of spurious position reports, i.e., false alarms. Targets arise at random in space and time. Each target persists independently for a random length of time and then ceases to exist. A track is defined as a path in space-time traveled by a target. The essence of the multiple-target tracking problem is to find tracks from the noisy observations; this requires solutions to both data association and state estimation problems [18].

The data association problem in multiple-target tracking is the 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 [18]. However, due to the noise in state transitions and observations, we cannot expect to find the exact solution. The most successful algorithm based on this data-oriented view is the multiple hypothesis tracker (MHT) [17]. In MHT, each hypothesis associates past observations with a target and, as a new set of observations arrives, a new set of hypotheses is formed from the previous hypotheses. The algorithm returns a hypothesis with the highest posterior as a solution. MHT is categorized as a "deferred logic" method [16] 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 tracks and suitable for autonomous surveillance applications. The main disadvantage of MHT is its computational complexity since the number of hypotheses grows exponentially over time. Various heuristics are developed to overcome this complexity, such as pruning, gating, clustering, N-scan-back logic, [17], [10] and k-best hypotheses [6] using Murty's algorithm [12]. But the heuristics are used at the expense of 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 deploy in a real-time surveillance system.

A different approach to the data association problem is the joint probabilistic data association filter (JPDAF) [1]. JPDAF is a suboptimal single-stage approximation to the optimal Bayesian filter. JPDAF is a sequential tracker in which the associations between the "known" targets and the latest observations are made sequentially. JPDAF assumes a fixed number of targets and cannot initiate or terminate tracks since only the current set of observations is considered. There are restricted extensions to JPDAF to allow the formation of a new track (see [5] and references within). Other multiple-target tracking algorithms, such as the multisensor multitarget mixture reduction (MTMR) [14] and the probabilistic multi-hypothesis tracker (PMHT) [19], also assume a fixed number of targets and cannot initiate or terminate tracks. Recently, a Bayesian model-based approach to track a varying number of targets which can initiate and terminate tracks was presented in [13].

Sequential trackers are typically more efficient than deferred-logic trackers such as MHT but they are prone to make erroneous associations since the associations made in the past are not reversible [16]. In addition, the exact calculation of association probabilities in JPDAF at each stage is NP-hard [4] since the related problem of finding the permanent of a 0-1 matrix is #P-complete [20]. In [8], a single-stage data association problem is considered and a leave-one-out heuristic is developed to avoid the enumeration of all possible associations.

The data association problem formulated under the data-oriented view such as MHT is also known to be NP-hard [16]. Hence, we cannot expect to find an optimal solution

This work was supported by DARPA F30602-00-2-0538 and ARO MURI DAAD 19-OZ-1-0383 and partially supported by DARPA-USAF FA8750-04-2-0222.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, {sho,russell,sastry}@eecs.berkeley.edu.

in polynomial time unless $P = NP$. An optimization approach to data association has been applied as a 0-1 integer programming problem [11] and as a multidimensional assignment problem [16]. In both cases, one first finds feasible tracks using the gating method and compute the cost of each feasible track. Then the optimization routine finds a subset of 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. However, in a dense environment, the number of feasible tracks can be large and the complexity of the algorithm increases dramatically.

The main contribution of this paper is the development of an efficient real-time algorithm that solves the data association problem and is capable of initiating and terminating a varying number of tracks. We take the data-oriented, combinatorial optimization approach to the data association problem but avoid the enumeration of tracks by applying a sampling method called Markov chain Monte Carlo (MCMC). Examples of MCMC are Metropolis-Hastings and Gibbs sampling [7]. The MCMC data association (MCMCDA) algorithm can be viewed as a deferred-logic method since its decision about forming a track is based on both current and past observations. At the same time, it can be considered as an approximation to the optimal Bayesian filter if it is used to approximate the association probabilities or expectations such as the average link travel time as done in [15]. MCMCDA shows remarkable performance compared to the greedy algorithm and MHT under extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates. The MCMC method has been applied to data association problems before. In [3], the Gibbs sampling method is used to track a single target using measurements from a finite number of linear models, where the measurement to model association is unknown. In [15], a combination of MCMC and expectation-maximization (EM) is used to simultaneously track multiple vehicles using measurements from spatially separated sensors and learn the intrinsic parameters of sensors. Each state of the Markov chain in [15] is a possible association but, unlike our model, a uniform prior is used, assuming a fixed number of objects, no missing observations, and no false alarms.

The remainder of this paper is structured as follows. We formally state the (discrete-time) general multiple-target tracking problem in Section II. In Section III, we present a general purpose MCMCDA algorithm for multiple-target tracking. The algorithm is applied in simulation to extreme situations and its performance is compared with the greedy algorithm and MHT in Section IV.

II. General Multiple-Target Tracking

A. 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 $[t_{\mathrm{i}}^k, t_{\mathrm{f}}^k] \subset [1, T]$ 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. At each time, an existing target persists with probability $1 - p_{\mathrm{z}}$ and disppears with probability p_{z}. The number of objects arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_{\mathrm{b}} V$ where λ_{b} is the birth rate of new objects per unit time, per unit volume. 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

$$x_{t+1}^k = F^k(x_t^k) + w_t^k \qquad \text{for } t = t_{\mathrm{i}}^k, \ldots, t_{\mathrm{f}}^k - 1,$$

where $w_t^k \in \mathbb{R}^d$ are white noise processes. The noisy observation of the state of the object is measured with a 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 $(\lambda_{\mathrm{f}} V)$ 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, including 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:

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

where $v_t^j \in \mathbb{R}^m$ are white noise processes and $u_t \sim \mathrm{Unif}(\mathcal{R})$ is a random process for false alarms. Notice that, with probability $1 - p_{\mathrm{d}}$, the object is not detected and we call this a missing observation. We assume that targets are indistinguishable in this paper. But, if observations include target type or attribute information, the state variable can be extended to include target type information.

Under the data-oriented approach, the multiple-target tracking problem is to partition the observations such that the posterior is maximized, i.e., the maximum a posteriori (MAP) estimate. Under the Bayesian approach, if we are given a function defined on Ω, the collection of all partitions of observations (see below for its definition), we seek the expected value of the function given the observations. The MAP estimate found under the data-oriented approach may not be robust but it is sometimes more convenient when representing the estimated parameters of varying dimensions.

B. Probabilistic Model

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 algorithm. If an object is observed k

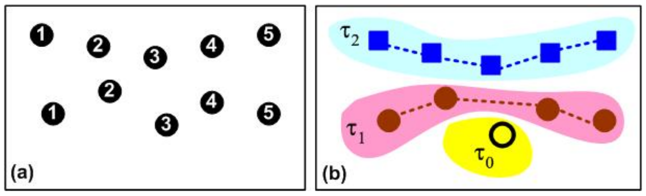

Fig. 1. (a) An example of observations Y (each circle represents an observation and numbers represnt observation times); (b) an example of a partition ω of Y

times at $t_1, t_2, \ldots, t_k$, its dynamic and measurement models can be expressed as:

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

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. Let $\bar{x}_t$ be the expected value of x_t given $y_1, \ldots, y_{t-1}$; $\bar{P}_t$ be the covariance of x_t given $y_1, \ldots, y_{t-1}$; $\hat{x}_t$ be the expected value of x_t given $y_1, \ldots, y_t$; and $\hat{P}_t$ be the covariance of x_t given $y_1, \ldots, y_t$.

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 the observations τ_k. However, we assume there is a deterministic function that returns a set of estimated states given a set of observations, 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. 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. An example of a partition is shown in Fig. 1.

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 states of an object independently since each object moves independently from the other objects. For each track $\tau \in \omega$, we apply the Kalman filter to estimate the states $\bar{x}_t(\tau)$ and covariances $B_t(\tau)$, where $B_t(\tau) = C\bar{P}_t(\tau)C^T + R$ is the conditional observation covariance at time t for the track τ.

Let e_t be the number of targets from time $t-1$ and a_t be the number of new targets at time t. Let z_t be the number of targets terminated at time t and $c_t = e_t - z_t$. Let d_t be the number of detections at time t and $u_t = e_t - z_t + a_t - d_t$ be the number of undetected targets. Finally, let $f_t = n_t - d_t$ be the number of false alarms. It can be shown that the posterior of ω is:

$$\begin{aligned} P(\omega|Y) &= \tfrac{1}{Z} \prod_{t=1}^{T} p_z^{z_t}(1-p_z)^{c_t} p_d^{d_t}(1-p_d)^{u_t} \lambda_b^{a_t} \lambda_f^{f_t} \\ &\times \prod_{\tau \in \omega \setminus \{\tau_0\}} \prod_{i=1}^{|\tau|-1} \mathcal{N}(\tau(t_{i+1}) | C\bar{x}_{t_{i+1}}(\tau), B_{t_{i+1}}(\tau)), \end{aligned} \tag{2}$$

where Z is a normalizing constant and $\mathcal{N}(\cdot|\mu, \Sigma)$ is the Gaussian density function with mean μ and covariance matrix Σ. Now under the data-oriented, combinatorial optimization approach, our goal is to find a partition of observations such that $P(\omega|Y)$ is maximized.

III. MCMC DATA ASSOCIATION ALGORITHM

In this section, we develop an MCMC sampler to solve the multiple-target tracking problem. MCMC-based algorithms play a significant role in many fields such as physics, statistics, economics, and engineering [2]. In some cases, MCMC is the only known general algorithm that finds a good approximate solution to a complex problem in polynomial time [9]. MCMC techniques have been applied to complex probability distribution integration problems, counting problems such as #P-complete problems, and combinatorial optimization problems [9], [2]. The MCMC approach applied to combinatorial optimization problems is generally known as simulated annealing.

MCMC is a general method to generate samples from a distribution π by constructing a Markov chain $\mathcal{M}$ whose states are ω and whose stationary distribution is $\pi(\omega)$. 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

$$A(\omega, \omega') = \min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right), \tag{3}$$

otherwise the sampler stays at ω, so that the detailed balance is satisfied. If we make sure that $\mathcal{M}$ is irreducible and aperiodic, then $\mathcal{M}$ converges to its stationary distribution by the ergodic theorem. Hence, for a given bounded function f, the average value of f over the sampled states converges to $\mathbb{E}_\pi f(\omega)$. Notice that it only requires an ability to compute the ratio $\pi(\omega')/\pi(\omega)$ avoiding the need to normalize π.

The MCMC data association (MCMCDA) algorithm is described in Algorithm 1. MCMCDA is an MCMC algorithm whose state space is Ω described in Section II-B and whose stationary distribution is the posterior (2). The proposal distribution for MCMCDA consists of five types of moves. They are

1) birth/death move pair;
2) split/merge move pair;
3) extension/reduction move pair;
4) track update move; and
5) track switch move.

Algorithm 1 (MCMC Data Association):

```
Input: $Y, n_{\text{mc}}, \omega_{\text{init}}$
Output: $\hat{\omega}$

$\omega \leftarrow \omega_{\text{init}}$; $\hat{\omega} \leftarrow \omega_{\text{init}}$
for $n = 1$ to $n_{\text{mc}}$
   sample $m$ from $\xi_K(\cdot)$
   propose $\omega'$ based on $m$ and $\omega$ (described below)
   sample $U$ from $\text{Unif}[0, 1]$
   $\omega \leftarrow \omega'$ if $U < A(\omega, \omega')$
   $\hat{\omega} \leftarrow \omega$ if $p(\omega|Y)/p(\hat{\omega}|Y) > 1$
end
```

The MCMCDA moves are graphically illustrated in Fig. 2. We index each move by an integer such that $m = 1$ for a birth move, $m = 2$ for a death move and so on. The move m is chosen randomly from the distribution $\xi_K(m)$ where K is the number of tracks of the current partition ω. 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 $\xi_1(m = 4) = \xi_1(m = 8) = 0$. For other values of K and m, we assume $\xi_K(m) > 0$. The inputs for MCMCDA are the set of all observations Y, the number of samples n_{mc}, and the initial state ω_{init}. At each step of the algorithm, ω is the current state of the Markov chain. The acceptance probability $A(\omega, \omega')$ is defined in (3) where $\pi(\omega) = P(\omega|Y)$ from (2).

In Algorithm 1, we use MCMC to find a solution to a combinatorial optimization problem. So it can be considered as simulated annealing at a constant temperature. No burn-in samples are used since we are simply looking for a partition which maximizes the posterior. In addition, the memory requirement of the algorithm is at its bare minimum. Instead of keeping all $\{\omega(n)\}_{n=1}^{n_{\text{mc}}}$, we can simply keep the partition with the maximum posterior, $\hat{\omega}$. If the algorithm is used for an integration problem to estimate $\mathbb{E}_{P(\omega|Y)} f(\omega)$ for some bounded function f, e.g., average link travel times, we will need burn-in samples and need to maintain the sufficient statistics for the desired expectation.

In order to make the algorithm more efficient, we make two additional assumptions: (1) the maximal directional speed 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 its type and terrain conditions. The second assumption is a user-defined parameter. Let $p_{\text{dt}}(s) = 1 - (1 - p_{\text{d}})^s$ be the probability the object is observed at least once out of s sampling times. Then, for given $\bar{p}_{\text{dt}}$, we set $\bar{d} \geq \log(1 - \bar{p}_{\text{dt}})/\log(1 - p_{\text{d}})$ to detect a track with probability larger than $\bar{p}_{\text{dt}}$. For example, given $p_{\text{d}} = .7$ and $\bar{p}_{\text{dt}} = .99$, a track is detected with probability larger than .99 for $\bar{d} \geq 4$. We will now assume that these two new conditions are added to the definition of Ω so each element $\omega \in \Omega$ satisfies these two additional assumptions.

We now introduce a data structure which is used to

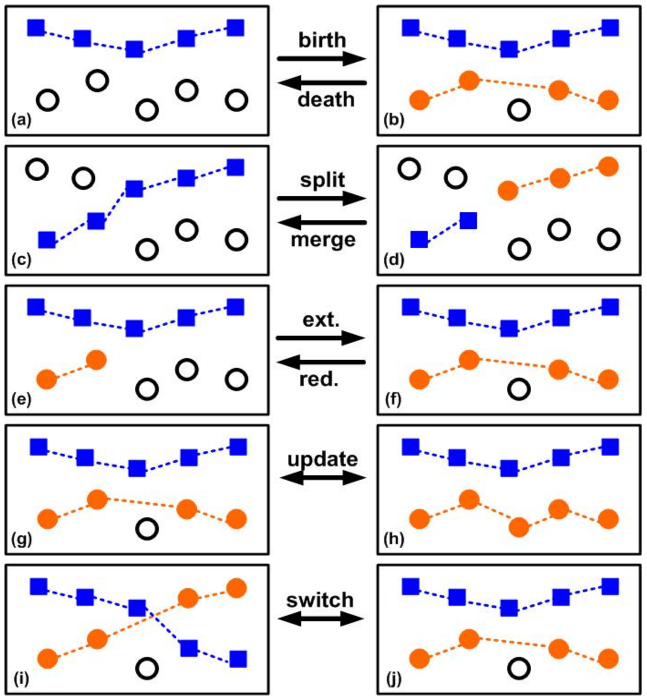

Fig. 2. Graphical illustration of MCMCDA moves (associations are indicated by dotted lines and hollow circles are false alarms)

propose a new partition ω' in Algorithm 1. We define a neighborhood tree of observations as

$$L_d(y_t^j) = \{y_{t+d}^k \in y_{t+d} : \|y_t^j - y_{t+d}^k\| \leq d \cdot \bar{v}\}$$

for $d = 1, \ldots, \bar{d}$, $j = 1, \ldots, n_t$ and $t = 1, \ldots, T - 1$. Here $\|\cdot\|$ is the usual Euclidean distance. This neighborhood tree groups temporally separated observations based on their distances. The parameter d allows missing observations. The use of this neighborhood tree makes the algorithm more scalable since distant observations will be considered separately and makes the computations of the proposal distribution easier. It is similar to the clustering technique used in MHT but L_d is fixed for a given set of observations.

We now describe each move of the sampler in detail. First, let $\zeta(d)$ be a distribution of a random variable d taking values from $\{1, 2, \ldots, \bar{d}\}$. We assume the current state of the chain is $\omega = \omega^0 \cup \omega^1 \in \Omega$, where $\omega^0 = \{\tau_0\}$ and $\omega^1 = \{\tau_1, \ldots, \tau_K\}$. The proposed partition is denoted by $\omega' = \omega'^0 \cup \omega'^1 \in \Omega$. Note the abuse of notation below with indexing of time, i.e., when we say $\tau(t_i)$, t_i means the time at which a target corresponding to the track τ is observed i times.

A. Birth and Death Moves (Fig. 2, $a \leftrightarrow b$)

For a birth move, we increase the number of tracks from K to $K' = K + 1$ and select t_1 uniformly at random (u.a.r.) from $\{1, \ldots, T - 1\}$ as an appearance time of a new track. Let $\tau_{K'}$ be the track of this new object. Then we choose d_1 from the distribution ζ. Let $L_{d_1}^1 = \{y_{t_1}^j : L_{d_1}(y_{t_1}^j) \neq \emptyset, y_{t_1}^j \notin \tau_k(t_1), j = 1, \ldots, n_{t_1}, k = 1, \ldots, K\}$. $L_{d_1}^1$ is a set of observations at t_1 such that, for any $y \in L_{d_1}^1$, y does not belong to other tracks and y has at least one descendant in $L_{d_1}(y)$. We choose $\tau_{K'}(t_1)$ u.a.r. from $L^1_{d_1}$. If $L^1_{d_1}$ is empty, the move is rejected since the move is not reversible. Once the initial observation is chosen, we then choose the subsequent observations for the track $\tau_{K'}$. For $i = 2, 3, \ldots$, we choose d_i from ζ and choose $\tau_{K'}(t_i)$ u.a.r. from $L_{d_i}(\tau_{K'}(t_{i-1})) \setminus \{\tau_k(t_{i-1}+d_i) : k = 1, \ldots, K\}$ unless this set is empty. But, for $i = 3, 4, \ldots$, the process of adding observations to $\tau_{K'}$ terminates with probability γ, where $0 < \gamma < 1$. If $|\tau_{K'}| \leq 1$, the move is rejected. We then propose this modified partition where $\omega'^1 = \omega^1 \cup \{\tau_{K'}\}$ and $\omega'^0 = \{\tau_0 \setminus \tau_{K'}\}$. For a death move, we simply choose k u.a.r. from $\{1, \ldots, K\}$ and delete the k-th track and propose a new partition where $\omega'^1 = \omega^1 \setminus \{\tau_k\}$ and $\omega'^0 = \{\tau_0 \cup \tau_k\}$.

B. Split and Merge Moves (Fig. 2, $c \leftrightarrow d$)

For a split move, we select $\tau_s(t_r)$ u.a.r. from $\{\tau_k(t_i) : |\tau_k| \geq 4, i = 2, \ldots, |\tau_k| - 2, k = 1, \ldots, K\}$. Then we split the track τ_s into τ_{s_1} and τ_{s_2} such that $\tau_{s_1} = \{\tau_s(t_i) : i = 1, \ldots, r\}$ and $\tau_{s_2} = \{\tau_s(t_i) : i = r+1, \ldots, |\tau_s|\}$. The modified track partition becomes $\omega'^1 = (\omega^1 \setminus \{\tau_s\}) \cup \{\tau_{s_1}\} \cup \{\tau_{s_2}\}$ and the false alarm partition ω'^0 is updated accordingly. For a merge move, we consider the set

$$\begin{aligned} M \quad = \quad & \{(\tau_{k_1}(t_f), \tau_{k_2}(t_1)) : \tau_{k_2}(t_1) \in L_{t_1 - t_f}(\tau_{k_1}(t_f)), \\ & f = |\tau_{k_1}| \text{ for } k_1 \neq k_2, 1 \leq k_1, k_2 \leq K\}. \end{aligned}$$

We select a pair $(\tau_{s_1}(t_f), \tau_{s_2}(t_1))$ u.a.r. from M. The tracks are combined into a single track $\tau_s = \tau_{s_1} \cup \tau_{s_2}$. Then we propose a new partition where $\omega'^1 = (\omega^1 \setminus (\{\tau_{s_1}\} \cup \{\tau_{s_2}\})) \cup \{\tau_s\}$ and ω'^0 with appropriate rearrangements.

C. Extension and Reduction Moves (Fig. 2, $e \leftrightarrow f$)

In a track extension move, we select a track τ u.a.r. from K available tracks in ω. We reassign observations for τ after the disappearance time $t_{|\tau|}$ as done in the track birth move. For a track reduction move, we select a track τ u.a.r. from K available tracks in ω and r u.a.r. from $\{2, \ldots, |\tau| - 1\}$. We shorten the track τ to $\{\tau(t_1), \ldots, \tau(t_r)\}$ by removing the observations assigned to τ after the time t_{r+1}.

D. Track Update Move (Fig. 2, $g \leftrightarrow h$)

In a track update move, we select a track τ u.a.r. from K available tracks in ω. Then we pick r u.a.r. from $\{1, 2, \ldots, |\tau|\}$ and reassign observations for τ after the time t_r as done in the track birth move.

E. Track Switch Move (Fig. 2, $i \leftrightarrow j$)

For a track switch move, we select a pair of observations $(\tau_{k_1}(t_p), \tau_{k_2}(t_q))$ from two different tracks such that, $\tau_{k_1}(t_{p+1}) \in L_d(\tau_{k_2}(t_q))$ and $\tau_{k_2}(t_{q+1}) \in L_{d'}(\tau_{k_1}(t_p))$, where $d = t_{p+1} - t_q$, $d' = t_{q+1} - t_p$ and $0 < d, d' \leq \bar{d}$. Then we let

$$\begin{aligned} \tau_{k_1} &= \{\tau_{k_1}(t_1), \ldots, \tau_{k_1}(t_p), \tau_{k_2}(t_{q+1}), \ldots, \tau_{k_2}(t_{|\tau_{k_2}|})\} \\ \tau_{k_2} &= \{\tau_{k_2}(t_1), \ldots, \tau_{k_2}(t_q), \tau_{k_1}(t_{p+1}), \ldots, \tau_{k_1}(t_{|\tau_{k_1}|})\}. \end{aligned}$$

Theorem 1: Suppose that $0 < p_z, p_d < 1$ and $\lambda_b, \lambda_f > 0$. If $\zeta(d) > 0$, for all $d \in \{1, \ldots, \bar{d}\}$, then the Markov chain designed by Algorithm 1 is irreducible.

Proof: The birth and death moves are sufficient to illustrate the irreducibility of the chain. Since $0 < p_z, p_d < 1$ and $\lambda_b, \lambda_f > 0$, $P(\omega|Y) > 0$ for all $\omega \in \Omega$. Take an arbitrary partition $\omega \in \Omega$, say $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$. Now consider the partition $\omega' \in \Omega$, such that $\omega' = \{\tau'_0\}$, i.e., ω' assigns all observations as false alarms. Since ω is arbitrary, the chain is irreducible if the chain can move from ω' to ω and from ω to ω'. For the move from ω' to ω, consider K consecutive birth moves: $\omega_0 = \omega', \omega_1 = \{\{\tau'_0 \setminus \tau_1\}, \tau_1\}, \ldots, \omega_K = \{\{\tau'_0 \setminus \{\cup_{i=1}^K \tau_i\}\}, \tau_1, \ldots, \tau_K\} = \omega$. Since $\omega \in \Omega$, all tracks τ_k are legal, i.e., τ_k satisfies the constraints described in Section II-B and, for $i = 1, \ldots, |\tau_k| - 1$, $\tau_k(t_{i+1}) \in L_d(\tau_k(t_i))$ for $1 \leq d = t_{i+1} - t_i \leq \bar{d}$. Thus, $\omega_k \in \Omega$ for all k. Because $\zeta(d) > 0$ and all tracks τ_k are legal, the probability of proposing τ_k at ω_{k-1} by the birth move is positive and $q(\omega_k, \omega_{k+1}) > 0$. For the move from ω to ω', consider K consecutive death moves: $\omega_K = \omega, \omega_{K-1}, \ldots, \omega_0 = \omega'$. The probability of removing the track τ_k at ω_k by the death move is positive and $q(\omega_{k+1}, \omega_k) > 0$. Since $P(\omega_k|Y) > 0$ for all k, the chain can move from ω' to ω and from ω to ω'. Hence, the chain is irreducible. ■

The Markov chain designed by Algorithm 1 is irreducible (Theorem 1) and aperiodic since there is always a positive probability of staying at the current state in the track update move. In addition, the transitions described in Algorithm 1 satisfy the detailed balance condition since it uses the Metropolis-Hastings kernel (3). Hence, by the ergodic theorem, the chain converges to its stationary distribution. Notice that the other moves are designed to improve the performance of the algorithm.

IV. SIMULATION RESULTS

For the simulations we consider surveillance over a rectangular region on a plane, $\mathcal{R} = [0, L] \times [0, L] \subset \mathbb{R}^2$. The state vector is $x = [x, y, \dot{x}, \dot{y}]^T$ where (x, y) is a position on $\mathcal{R}$ along the usual x and y axes and $(\dot{x}, \dot{y})$ is a velocity vector. The linear system model (1) is used where δ is an interval between observations and

$$A(\delta) = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} G(\delta) = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix} C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T$$

The covariance matrices are $Q = \text{diag}(100, 100)$ and $R = \text{diag}(25, 25)$.

The complexity of multiple-target tracking problems can be measured by several metrics: (1) the intensity of the false alarm rate λ_f; (2) the detection probability p_d; and (3) the density of tracks. The problem gets more challenging with increasing λ_f, decreasing p_d, increasing K, and increasing density of tracks. The number of tracks itself may not make the problem more difficult if they are scattered apart. The difficulty arises when there are many tracks that are moving closely and crossing each other; this is when the ambiguity of data association is greater. Hence, we only consider situations in which tracks move very closely so we can control the density of tracks by the number of tracks.

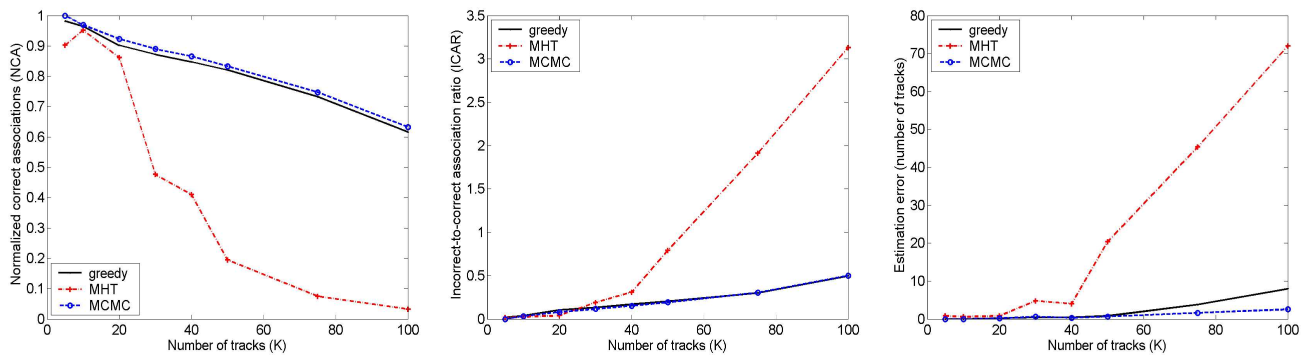

Fig. 3. NCA (left), ICAR (middle), and the estimation error in the number of tracks (right) as functions of a number of tracks

We study the performance of the MCMCDA algorithm against the greedy algorithm and MHT by varying the parameters listed above. The greedy algorithm is a batch-mode nearest neighbor multiple-target tracking algorithm. The algorithm first marks all observations as false alarms, and then picks two unmarked observations at different times to estimate an initial state. Then it forms a canditate track by picking unmarked observations which are the nearest to the predicted states for subsequent time steps. The candidate track is validated as a track and observations associated to the candidate track are marked if the marginal of the candidate track exceeds a threshold. The process is repeated until no more track can be found.

Based on our model described above, we have generated different scenarios. In particular, in all cases, except for the online tracking, half of the new objects appear from the left bottom quadrant of $\mathcal{R}$ and the other half appear from the right bottom quadrant. The actual initial positions are chosen randomly from each quadrant. They all move diagonally so each group of tracks crosses the other group in the middle of $\mathcal{R}$. Also targets move very close to each other and there are also crossovers within each group. The situations we have used for simulations below include very extreme cases and, in our opinion, such complex situations have not appeared in the multiple-target tracking literatures.

Since the number of targets is not fixed, it is difficult to compare algorithms using a standard criterion such as the residual mean square error. Hence, we introduce two new metrics to measure the effectiveness of each data association algorithm. Let ω^* be the true partition with which the test case was generated. For $\omega \in \Omega$, we represent the set of all associations in ω as $\mathrm{SA}(\omega) = \{(\tau, t_i^\tau, t_{i+1}^\tau) : i = 1, \ldots, |\tau| - 1, \tau \in \omega\}$, where t_i^τ is the time at which the track τ is observed i times. Let $\mathrm{CA}(\omega) = \{(\tau, t, s) \in \mathrm{SA}(\omega) : \tau(t) = \tau^*(t), \tau(s) = \tau^*(s), \tau^* \in \omega^*\}$ be the set of correct associations in ω relative to ω^*. The two new metrics we will be using are the normalized correct associations (NCA) and incorrect-to-correct association ratio (ICAR):

$$\mathrm{NCA}(\omega) = \frac{|\mathrm{CA}(\omega)|}{|\mathrm{SA}(\omega^*)|} \quad (4)$$

$$\mathrm{ICAR}(\omega) = \frac{|\mathrm{SA}(\omega)| - |\mathrm{CA}(\omega)|}{|\mathrm{CA}(\omega)|}. \quad (5)$$

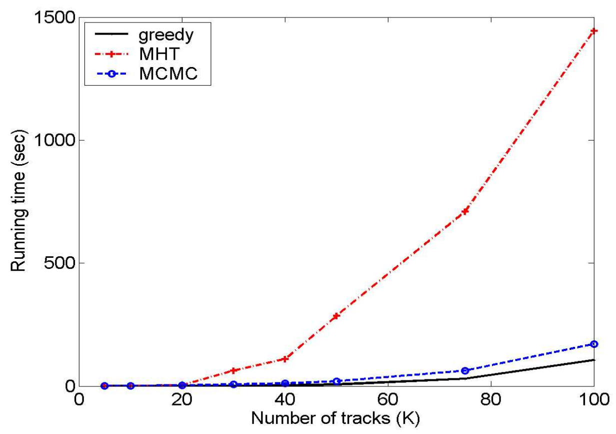

Fig. 4. Average running time vs. number of tracks

NCA measures the ratio between the number of correct associations and the number of associations in the true partition while ICAR measures the number of incorrect associations per correct association. We measure the performance of each algorithm by NCA, ICAR, the estimation error in the number of tracks, $||\omega^*| - |\omega||$, and the running time of the algorithm.

Both MCMCDA and greedy algorithms are written in C++ with Matlab interfaces. We have used the C++ implementation of MHT from [6] [1], which implements pruning, gating, clustering, N-scan-back logic and k-best hypotheses. The parameters for MHT are fine-tuned so that it gives similar performance when there are 10 targets: the maximum number of hypotheses in a group is 1000, the maximum track tree depth is 5, and the maximum Mahalanobis distance is 5.9. All simulations are run on a PC with a 2.6-GHz Intel processor.

A. Experiment I (Number of Tracks)

In this experiment, we vary K from 5 to 100 (the actual values of K are 5, 10, 20, 30, 40, 50, 75 and 100). The other parameters are held fixed: $\mathcal{R} = [0, 1000] \times [0, 1000]$, $T = 10$, $\lambda_f V = 1$, $\bar{v} = 130$ unit lengths per unit time. The main focus of this experiment is to test the accuracy of MCMCDA against other algorithms so the tracks are

[1]http://www.ee.ucl.ac.uk/ ˜icox/

detected at all times, however, we have set $p_{\mathrm{d}} = .9$ for the prior calculation. We have also set $\bar{d} = 1$. Since all tracks are observed, the number of observations increases as the number of tracks increases. For each value of K, we randomly generated five tests. The results for MCMCDA are the average values over 10 repeated runs and the initial state is initialized with the greedy algorithm and 10,000 samples are used. The average NCAs, ICARs and the estimation error in the numbers of tracks for three different algorithms are shown in Fig. 3. The running times of three algorithms are shown in Fig. 4 (the running time of MCMCDA includes the initialization step). Although the maximum number of hypotheses of 1000 per group is a large number, with increasing numbers of tracks, the performance of MHT deteriorates due to pruning. But both greedy and MCMCDA keep good performance, although the greedy algorithm detects a less number of tracks for large K. In addition, the running times of both greedy and MCMCDA are significantly less than that of MHT.

B. Experiment II (False Alarms)

Now the settings are the same as Experiment I but we vary the false alarm rates while the number of tracks is fixed at $K = 10$. The test cases for this experiment are prepared as follows. We first generated five different random scenarios each with 10 tracks. Then, we applied different false alarm rates to generate test cases. The false alarm rates are varied from $\lambda_{\mathrm{f}} V = 1$ to $\lambda_{\mathrm{f}} V = 100$ with an increment of 10. 10,000 samples are used for MCMCDA and the results for MCMCDA are the average values over 10 repeated runs. The average NCAs, ICARs and the estimation error in the numbers of tracks for three different algorithms at different false alarm rates are shown in Fig. 5. It shows the remarkable performance of MCMCDA at high false alarm rates while the other two algorithms perform poorly. The greedy algorithm scores higher in NCA than MCMCDA but poorly in ICAR. In addition, it reports spurious tracks at high false alarm rates. Notice that MHT does not make any correct associations at high false alarm rates, $\lambda_{\mathrm{f}} V \geq 80$, so ICARs for MHT at $\lambda_{\mathrm{f}} V \geq 80$ are not reported.

C. Experiment III (Detection Probability)

In this experiment we vary the detection probability p_{d} from .3 to .9 with an increment of .1 while keeping the other parameters as the previous experiments except $K = 10$, $\lambda_{\mathrm{f}} V = 1$, $T = 15$ and $\bar{d} = 5$. Now the tracks are not observed all the time. For each value of p_{d}, five test cases are randomly generated and the average NCAs, ICARs and the estimation error in the numbers of tracks are shown in Fig. 6. For MCMCDA, we present two cases: MCMC(15K) with 15,000 samples and MCMC(150K) with 150,000 samples. It shows that MCMCDA outperforms the other algorithms at low detection probabilities. At high detection probabilities, MHT scores higher than MCMCDA but it reports a higher number of tracks, meaning that it fragments tracks.

TABLE I
PERFORMANCE OF ONLINE MCMCDA TRACKER
RUNNING TIMES (RT) IN SECONDS

	Number of samples					
	1,000			5,000		
K	NCA	ICAR	RT	NCA	ICAR	RT
100	.95	.19	.06	.98	.13	.28
200	.94	.06	.09	.97	.05	.41
300	.92	.07	.11	.97	.05	.55

Although, in theory, MHT gives an optimal solution in the sense of MAP, it performs poorly when the detection probability is low or the false alarm rate is high due to the heuristics such as pruning and N-scan-back techniques used to reduce the complexity. The heuristics are required parts of MHT in practice. Without the pruning and N-scan-back logic, the problem complexity grows exponentially fast even for a small problem. In practice, MHT with heuristics works well when a few number hypotheses carry most of the weight. When the detection probability is low or the false alarm rate is high, there are many hypotheses with low weights and there is no small set of dominating hypotheses, so MHT cannot perform well. In addition, when the detection probability is high, MHT again suffers from a large number of observations. Another noticeable benefit of the MCMCDA algorithm is that its running time can be regulated by the number of samples and the number of observations but the running time of MHT depends on the complexity of the problem instance and is not predictable in advance.

D. Online MCMCDA Multiple-Target Tracker

The extension of MCMCDA to an online, real-time tracking is a trivial task. We implement a sliding window of size w_{s} using Algorithm 1. At each time step, we use the previous estimate to initialize MCMCDA and run MCMCDA on the observations belonging to the current window. A total of three test cases are generated: (case 1) 100 tracks, (case 2) 200 tracks and (case 3) 300 tracks. The surveillance duration is increased to $T = 1000$ and the surveillance region is now $\mathcal{R} = [0, 10000] \times [0, 10000]$. The other parameters are: $\lambda_{\mathrm{f}} V = 10$, $p_{\mathrm{d}} = .9$, $\bar{d} = 3$, $\bar{v} = 230$ and $w_{\mathrm{s}} = 10$. The objects appear and disappear at random in time and space so the number of tracks changes in time. These test cases represent instances of the general (discrete-time) multiple-target tracking problem. The average NCAs and ICARs over the sliding window and the average execution time per simulation time are shown in Table I. Notice that MCMCDA achieves excellent performance in all cases with less than one second of execution time.

V. CONCLUSIONS

The general (discrete-time) multiple-target tracking problem is described and an MCMCDA algorithm is proposed. Our MCMCDA tracker, a data association algorithm capable of initiating and terminating a varying number of

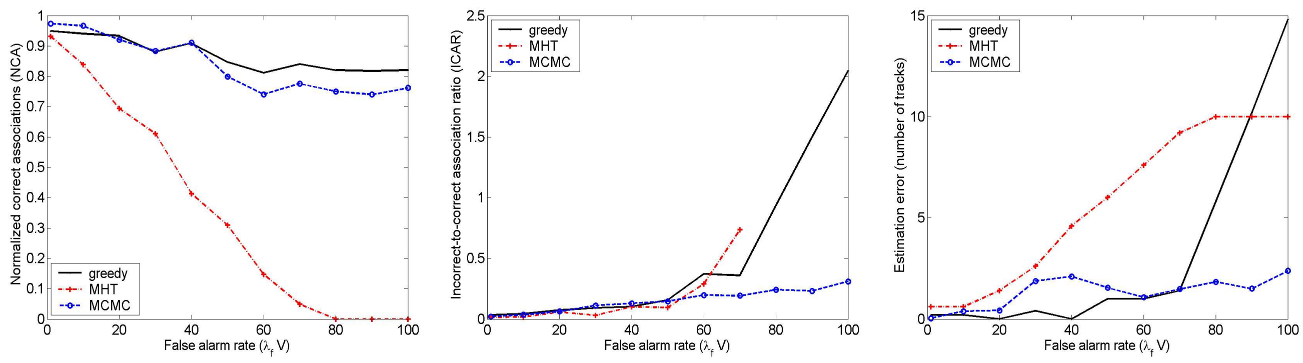

Fig. 5. NCA (left), ICAR (middle), and the estimation error in the number of tracks (right) as functions of false alarm rate

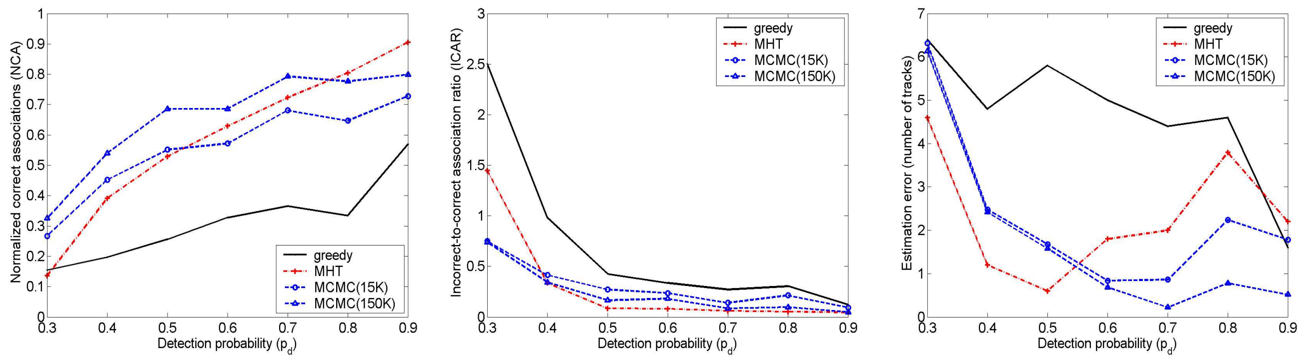

Fig. 6. NCA (left), ICAR (middle), and the estimation error in the number of tracks (right) as functions of detection probability

tracks, is flexible and can easily incorporate any domain specific knowledge to make it more efficient. Instead of searching over the whole solution space, the MCMC algorithm randomly searches over the space where the posterior is concentrated. Our simulation results show remarkable performance of the MCMCDA algorithm under extreme conditions such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates. We have shown that the algorithm can be extended as an online, real-time algorithm with excellent performance.

REFERENCES

[1] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Mathematics in Science and Engineering Series 179 Academic Press, San Diego, CA, 1988.

[2] I. Beichl and F. Sullivan. The Metropolis algorithm. *Computing in Science and Engineering*, 2(1):65–69, 2000.

[3] N. Bergman and A. Doucet. Markov chain Monte Carlo data association for target tracking. In *Proc. of IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

[4] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate distributed states. *IEEE Trans. Aerospace and Electronic Systems*, 28(3), 1992.

[5] I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[6] I.J. Cox and S.L. Hingorani. 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*, pages 437–443, 1994.

[7] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics Series. Chapman and Hall, 1996.

[8] Timothy Huang and Stuart J. Russell. Object identification in a Bayesian context. In *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1276–1283, 1997.

[9] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In Dorit Hochbaum, editor, *Approximations for NP-hard Problems*. PWS Publishing, Boston, MA, 1996.

[10] Thomas Kurien. Issues in the design of practical multitarget tracking algorithms. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*. Artech House: Norwood, MA, 1990.

[11] C. L. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *IEEE Trans. on Automatic Control*, 22:3:302–312, June 1971.

[12] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[13] Songhwai Oh, Jin Kim, and Shankar Sastry. A sampling-based approach to nonparametric dynamic system identification and estimation. In *Proc. of the American Control Conference*, Boston, MA, June 2004.

[14] L.Y. Pao. Multisensor multitarget mixture reduction algorithms for tracking. In *Proc. AIAA Guidance, Navigation, and Control Conf.*, pages 28–37, Monterey, CA, Aug. 1993.

[15] Hanna Pasula, Stuart J. Russell, Michael Ostland, and Yaacov Ritov. Tracking many objects with many sensors. In *Proc. IJCAI-99*, Stockholm, 1999.

[16] A.B. Poore. Multidimensional assignment and multitarget tracking. *Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 19:169–196, 1995.

[17] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transaction on Automatic Control*, 24(6):843–854, December 1979.

[18] R.W. Sittler. An optimal data association problem on surveillance theory. *IEEE Trans. on Military Electronics*, MIL-8:125–139, April 1964.

[19] R. Streit and T. Luginbuhl. Maximum likelihood method for probabilistic multi-hypothesis tracking. In *Proc. SPIE*, volume 2235, pages 394–405, April 1994.

[20] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.