JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 31, No. 1, January–February 2008

Distributed Multitarget Tracking and Identity Management

Songhwai Oh*
University of California, Merced, California 95344
Inseok Hwang†
Purdue University, West Lafayette, Indiana 47906
and
Shankar Sastry‡
University of California, Berkeley, California 94720

DOI: 10.2514/1.26237

The problem of tracking multiple targets and managing their identities in sensor networks is considered. Each sensor is assumed to have its own surveillance region and an ability to communicate with its neighboring sensors. We propose a scalable, distributed, multitarget-tracking and identity-management algorithm that can track an unknown number of targets and manage their identities efficiently in a distributed sensor network environment. Distributed multitarget tracking and identity management finds a globally consistent solution by maintaining local consistency among neighboring sensors. Distributed multitarget tracking and identity management consists of data association, multitarget tracking, identity management, and identity and track fusion. The data-association and multitarget-tracking problems are efficiently solved by Markov chain Monte Carlo data association, which can track an unknown number of targets. Distributed multitarget tracking and identity management manages identities of targets based on the identity-mass-flow framework. This framework prevents exponential growth in computation and storage of target-to-track association probabilities. Using identity and track fusion, distributed multitarget tracking and identity management maintains consistent identities and tracks among neighboring sensors. The performance and features of distributed multitarget tracking and identity management are extensively evaluated in simulation.

I. Introduction

RECENT advances in sensor technology and wireless communication have led to the concept of a sensor network [1–3]. A sensor network is a network of local sensor nodes that have sensing, processing, and communication capabilities [4,5]. Sensor networks have received growing interest in a variety of applications, including building comfort control [6], habitat and environment monitoring [7], traffic control [8], manufacturing and plant automation [9], service robotics [10], and battlefield surveillance and enemy tracking in military applications [11,12] (also see [2] and references therein). To fully exploit the capability of sensor networks, algorithms for sensor networks need to be *scalable* (i.e., adding/deleting sensors into a sensor network can be handled efficiently) and *distributed* (i.e., the algorithm can be implemented in individual sensors). In this paper, we develop a scalable distributed multitarget-tracking and identity-management (DMTIM) system that can keep track of multiple maneuvering targets and their identities in a sensor network.

Multitarget tracking is an important mathematical framework in many applications, including surveillance [13], computer vision [14,15], and network security [16]. The essence of the multitarget-tracking problem is to find tracks[§] from noisy measurements. Unlike single-target tracking, the multitarget problem is complicated by the uncertainty about the source of each measurement; that is, the associations between measurements and targets are not known. This data-association problem in multitarget tracking is to work out which measurements were generated by which targets; more precisely, we require a partition of measurements such that each element of a partition is a collection of measurements generated by a single target or clutter.

The most frequently used multitarget-tracking algorithms include the nearest-neighbor filter (NNF) [13,14], joint probabilistic data-association (JPDA) filter [13], and multiple hypothesis tracker (MHT) [17–19]. The NNF processes the new measurements in some order and associates each with the target for which the predicted position is closest, thereby selecting a single association after each scan. Although the NNF is easy to implement and fast, as shown in [13], it can break down under nontrivial circumstances and it is not suitable for the problems considered in this paper. JPDA is a suboptimal approach to the Bayesian filter. At each time step, instead of finding a single best association between measurements and tracks, JPDA computes association probabilities for all measurement-target pairs.

An association probability between measurement j and target k is the probability that measurement j is originated from target k. Given each association, the state of a target is estimated by a filtering algorithm, and this conditional state estimate is weighted by its association probability. Then the state of a target is estimated by summing over the weighted conditional estimates. MHT maintains multiple hypotheses associating past measurements with targets. When a new set of measurements arrives, a new set of hypotheses is formed from each previous hypothesis. The algorithm returns a hypothesis with the highest posterior as a solution. However, the track initiation and termination are difficult with JPDA, and both JPDA and MHT require large memory and computation cycles (the time and space complexities of MHT are higher than those of JPDA). Because MHT can initiate and terminate tracks, the tracking task can be easily distributed in a network of sensors, whereas this is difficult with JPDA. Our distributed tracking algorithm is inspired by the distributed tracking algorithm based on MHT by Chong et al. [19], in

Presented as Paper 5838 at the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, 15–18 August 2005; received 29 June 2006; revision received 6 November 2006; accepted for publication 1 December 2006. Copyright © 2007 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/08 $10.00 in correspondence with the CCC.

*School of Engineering; songhwai.oh@ucmerced.edu.
†School of Aeronautics and Astronautics; ihwang@purdue.edu.
‡Department of Electrical Engineering and Computer Science; sastry@eecs.berkeley.edu.

§The sequence of states that a target follows during its lifetime is called a *track*.

which the tracking task is distributed and tracks are hierarchically merged. We follow a similar approach, but our algorithm is based on a more computationally efficient multitarget-tracking algorithm.

The tracks estimated by a multitarget-tracking algorithm are usually used by other applications. These applications make decisions or reallocate resources based on the estimated tracks (e.g., pursuer assignment and path planning in pursuit–evasion games [20]). But a decision based on a single set of tracks may be risky, because tracks do not fully exhibit the uncertainty in the identities of targets accumulated from continuous interactions among crossing or nearby targets. For example, when two targets are moving close to each other, there can be multiple interpretations of the event. Figure 1a shows measurements about positions of two targets over time. Figures 1b and 1c show two possible interpretations made from the measurements shown in Fig. 1a. Clearly, there can be more than two interpretations, due to measurement noise and identity uncertainty, but a multitarget-tracking algorithm can only display a single interpretation (usually the one with the highest likelihood or an interpretation with expected positions). Because the number of possible interpretations grows exponentially as more measurements are collected, we cannot display all possible interpretations. In this paper, we assume that the communication medium is bandwidth-limited; hence, it is necessary to represent the uncertainty about the identities in the most compact representation.

This issue can be addressed by identity management [21,22]. An identity is assigned to a target when it first appears; the *identity belief* associated with a target at any future point in time is represented as a probability distribution of the identity of the target over all existing identities. Thus, when two targets cross each other, the uncertainty in this crossing is represented by changes in the identity beliefs. However, the available identity-management algorithms [21,22] work for the cases in which the number of targets in a sensor network is known and constant and their trajectories are available to local sensors. As a result, the existing algorithms are applicable to limited situations and it is difficult to scale them for a large sensor network. Hence, to handle general situations arising in a large-scale sensor network, a scalable and autonomous approach is required and it demands a new identity-management system that can handle an unknown and time-varying number of targets.

Although target tracking and identity management are closely related, only the recent work by Hwang et al. [23,24] describes an algorithmic framework for systematically tracking multiple maneuvering targets and maintaining their identities from noisy measurements. The algorithm has been shown to perform well for the case in which there is a single sensor (radar) and the number of targets are known and constant. Thus, this algorithm cannot be used for sensor network applications in which there are many sensors and the number of targets is varying over time. In this paper, we extend the results in [24] and propose a multitarget-tracking and identity-management system that is scalable, autonomous, and distributed. The system is based on three new algorithms: multitarget-tracking, identity-management, and identity- and track-fusion algorithms. The multitarget-tracking algorithm is a hierarchical implementation of Markov chain Monte Carlo data association (MCMCDA) [25,26] with a sliding window. For identity-management and fusion algorithms, distributed multitarget identity management (DMIM) is proposed. DMIM is a scalable, event-driven, query-based algorithm for local maintenance of identity beliefs and the incorporation of identity information from nearby sensors. Global identity estimates are generated in DMIM using identity fusion, which is posed as an optimization problem such that the fused identity minimizes a cost function that represents a performance criterion.

MCMCDA can track an unknown number of targets in real time and is an approximation to the optimal Bayesian filter. Monte Carlo methods have been applied to multitarget tracking. In [27,28], the particle filter has been applied to multitarget tracking, but the data-association problem is solved by JPDA. Unlike JPDA or MHT, which enumerates all possible associations to solve data-association problems, MCMCDA randomly samples the region in which the posterior is concentrated using Markov chain Monte Carlo (MCMC) techniques, making it more computationally efficient [25,26]. The simulation study in [25] showed that MCMCDA was computationally efficient compared with MHT with heuristics (i.e., pruning, gating, clustering, N-scan-back logic, and k-best hypotheses). MCMCDA outperformed MHT when the false-alarm rate is high or the number of targets is large [25]. MCMCDA is suitable for distributed sensor networks because it can autonomously initiate and terminate tracks. It has also been shown that MCMCDA is robust against packet losses and communication delays [29]. MCMC was first used to solve data-association problems by Pasula et al. [30,31], who showed it to be effective for multicamera traffic surveillance problems involving hundreds of vehicles. More recently, in [32], MCMC was used to approximate the association probabilities in JPDA and was shown to outperform Fitzgerald's cheap JPDA. MCMC has also been used for problems that are roughly isomorphic to the data-association problem, including state estimation in the switching Kalman filter [33] and stereo correspondence in computer vision [15].

The DMIM algorithm combines local maintenance of identity beliefs with a query-based protocol for the transfer and fusion of identity information between sensors. The algorithm is appropriate for distributed sensor network scenarios because it has the capability to reduce the uncertainty of the global identity estimates by fusing local estimates of the identity of a target collected by each sensor [22]. The identity-mass-flow framework is used to maintain a local estimate of identity for a fixed set of maneuvering targets [21,22]. This framework prevents exponential growth in computation and storage of target-to-track association probabilities. We develop a distributed version of this centralized algorithm (DMIM) that allows an unknown time-varying number of maneuvering targets in a sensor network. The DMIM algorithm is scalable with respect to the number of targets and number of sensors and can handle dynamic scenarios in which targets maneuver into and through the sensor network.

A key component of DMIM is the fusion of tracks and identity beliefs between neighboring sensors. The tracks estimated by neighboring sensors are hierarchically merged using MCMCDA to maintain local consistency. Identity fusion in DMIM is based on principles of information theory. Information-theoretic methods were initially developed to understand data communication and storage limits [34] and have subsequently been applied to problems such as target localization [35] and fault detection [36]. The identity-fusion algorithm incorporates metrics from information theory as performance criteria when determining global belief estimates. Specifically, Shannon information, Chernoff information, and the

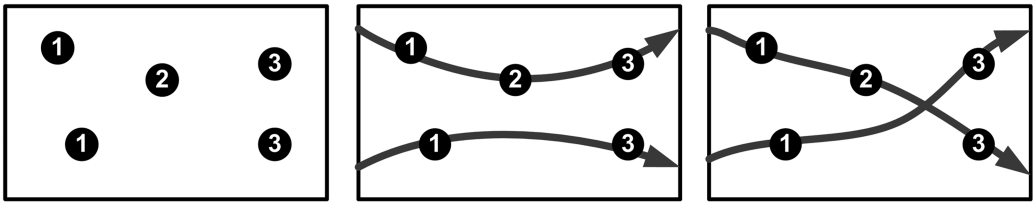


Fig. 1 An example of different interpretation measurements: a) measurements about the positions of two targets, b) one interpretation made from measurements shown in Fig. 1a (a solid line represents a track of a target), and c) another interpretation; each circle represents a measurement and numbers represent measurement times.

sum of Kullback–Leibler distances are used as cost functions. Minimizing these cost functions allows us to find locally consistent identity beliefs. Because local estimates can be combined under a query-based protocol, the identity-fusion algorithm lends itself to a distributed sensor network in which targets maneuver in and out of the sensing range of individual sensors.

In this paper, a distributed air traffic control system is used as an example of sensor networks. Although each air traffic controller is usually more capable than low-power sensor nodes described in [4,5], the efficiency of the system described in this paper makes the system applicable for a wide range of applications, including low-power or heterogeneous sensor networks. In the current air traffic control system, there is no algorithm that can perform simultaneous aircraft tracking and identity management, and most air traffic control systems are based on centralized computation. A controller performs identity management manually through voice communications, and the current techniques to resolve identity uncertainty of aircraft involve considerable communication between pilots and controllers, hence increasing workloads on both parties. The techniques developed in this paper mitigate this problem as much as possible by automating identity management; in this manner, local information is automatically incorporated using aircraft motion models. In situations under which this is not sufficient, the system can be used to alert the controller before a safety-critical situation occurs. In these scenarios, other sources of identity information such as intermittent transponder reports or physical characteristics of the aircraft may also be used in our system.

This paper is organized as follows: The overall architecture of the DMTIM is presented in Sec. II and the components of DMTIM are described in the following sections. In Sec. III, we formally describe the multitarget-tracking problem and describe MCMCDA for data association and multitarget tracking. We also compare MCMCDA against MHT in simulation. The algorithms for the identity management are described in Sec. IV. The tracks and identities estimated by neighboring sensors are combined using the identity- and track-fusion algorithms described in Sec. IV. We demonstrate and evaluate a DMTIM system in simulation in Sec. V.

II. System Architecture of Distributed Multitarget Tracking and Identity Management

The main focus of this paper is the problem of tracking multiple targets and managing their identities in sensor networks. Each sensor is assumed to have its own surveillance region and an ability to communicate with its neighboring sensors. A simple two-sensor example is shown in Fig. 2, in which the circles represent the surveillance regions of the sensors. Each sensor is assumed to have the capability of tracking multiple targets and managing the identities of targets within its surveillance region. The problem gets complicated, because the number of targets within the surveillance region of a sensor changes over time. For example, some targets may come from the surveillance regions of neighboring sensors, some targets may have not yet been registered into the identity-management system, and some targets may leave the surveillance region of the current sensor. For a large network, a centralized approach to multitarget tracking and identity management is not feasible and a scalable distributed approach is required.

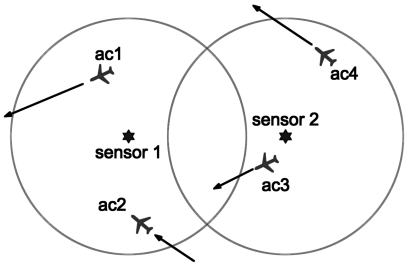


Fig. 2 A distributed multitarget-tracking and identity-management scenario for a two-sensor network.

This paper proposes a scalable DMTIM system that can track an unknown time-varying number of maneuvering targets and manage their identities efficiently in a distributed sensor network. The key highlights of DMTIM include modularity, the compact representation of identity and tracks to reduce communication load, and event-driven mechanisms for identity and track management.

The identity of a target is maintained by a *belief vector*. When there are K known identities, the belief vector of the target at time t is $b(t) \in [0, 1]^K$. The ith element $b_i(t)$ of $b(t)$ represents the probability that the identity of the target is the ith identity and

$$\sum_{i=1}^{K} b(t) = 1$$

For multiple targets, we have a *belief matrix* $B(t)$ for which the columns are belief vectors of the targets. Thus, entry $B_{ij}(t)$ represents the probability that target j can be identified as label (or name) i at time t.

The overall architecture of DMTIM is shown in Fig. 3. DMTIM includes the data-association and multitarget-tracking (DAMTT) module and the DMIM module, which contains the identity-management (IM) and identity- and track-fusion (ITF) submodules. At each sensor, the DAMTT module estimates the number of targets and tracks in its surveillance region, using its local measurements, and computes a mixing matrix and local information that are used by the IM module. A mixing matrix stores information about the interactions among targets, and local information contains identity information about a target. A mixing matrix and local information are described in detail in Sec. III. Upon receiving a mixing matrix and local information, the IM module updates its belief matrix. Then the sensor transmits its updated belief matrix and estimated tracks to its neighboring sensors using the communication unit.

In DMTIM, instead of sharing raw measurement data among neighboring sensors, the neighboring sensors share identity information in the form of a belief matrix and estimated tracks. As a result, we can reduce the overall communication load of the network. When the updated belief matrices and estimated tracks from neighboring sensors are available, the ITF module combines identity information and tracks and maintains local consistency among sensors. For example, if the same target is seen by sensor 1 and sensor 2, then the ITF module makes sure that sensor 1 and sensor 2 share the same information about the target. The ITF module also combines multiple tracks of the same target into a single track and adds an entry for the identity of a new target into the belief matrix. In following sections, the components of DMTIM are described in detail.

III. Data Association and Multitarget Tracking

The DAMTT module of DMTIM takes in sensor measurements and outputs a mixing matrix, state estimates, and local information. The DAMTT module needs to be able to track an unknown number of targets to distribute tracking tasks, because the number of targets in each sensor's surveillance region changes over time. In this section, we first describe a general formulation that allows the uncertainties in the number of targets and appearance and disappearance times of targets. Then we describe an algorithm for solving the general multitarget-tracking problem. Finally, the computations of mixing matrices, state estimates, and local information are described.

A. Multitarget-Tracking Problem

Let $T \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the number of targets that appear in the surveillance region $\mathcal{R}$ during the surveillance period. Each target $k \in \{1, \ldots, K\}$ moves in $\mathcal{R}$ for some duration $[t_i^k, t_f^k] \subset [1, T]$. Notice that the exact values of K and $\{t_i^k, t_f^k\}$ are unknown. Each target arises at a random position in $\mathcal{R}$ at t_i^k, moves independently around $\mathcal{R}$ until t_f^k, and then disappears. At each time, an existing target persists with probability $1 - p_z$ and disappears with probability p_z. The number of targets arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_b V_{\mathcal{R}}$, where

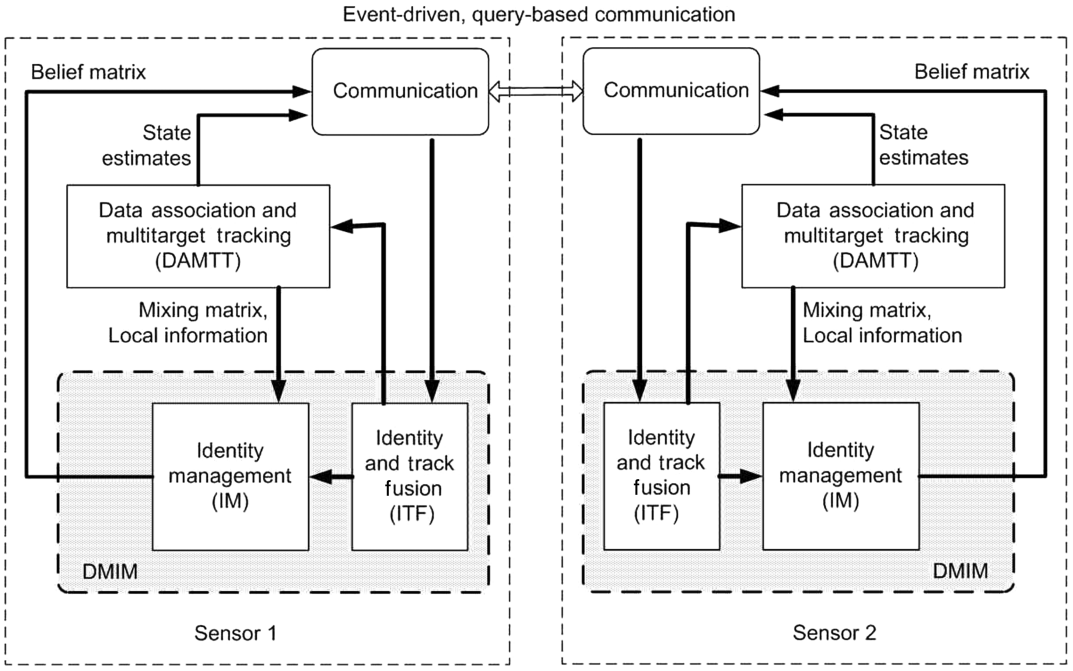


Fig. 3 Architecture of a DMTIM system for a two-sensor example.

λ_b is the birth rate of new targets per unit time, per unit volume, and $V_{\mathcal{R}}$ is the volume of $\mathcal{R}$. The initial position of a new target is uniformly distributed over $\mathcal{R}$.

Let $F^k: \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be the discrete-time dynamics of the target k, where n_x is the dimension of the state variable, and let $x^k(t) \in \mathbb{R}^{n_x}$ be the state of the target k at time t. The target k moves according to

$$x^k(t+1) = F^k[x^k(t)] + w^k(t) \quad \text{for } t = t_i^k, t_i^k + 1, \ldots, t_f^k - 1 \quad (1)$$

where $w^k(t) \in \mathbb{R}^{n_x}$ are white-noise processes. The white-noise process is included to model nonrectilinear motions of targets. The noisy measurement (or observation) of the state of the target is made with a detection probability p_d. Notice that with probability $1 - p_d$, the target is not detected and we call this a *missing observation*. There are also false alarms, and the number of false alarms has a Poisson distribution with a parameter $\lambda_f V_{\mathcal{R}}$, where λ_f is the false-alarm rate per unit time, per unit volume. Let $n(t)$ be the number of measurements at time t, including both noisy measurements and false alarms. Let $y^j(t) \in \mathbb{R}^{n_y}$ be the jth measurement at time t for $j = 1, \ldots, n(t)$, where n_y is the dimension of each measurement vector. Each target generates a unique measurement at each sampling time if it is detected. Let $H^j: \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ be the measurement model. Then the measurements are generated as follows:

$$y^j(t) = \begin{cases} H^j[x^k(t)] + v^j(t) & \text{if } j\text{th measurement is from } x^k(t) \\ u(t) & \text{otherwise} \end{cases} \quad (2)$$

where $v^j(t) \in \mathbb{R}^{n_y}$ are white-noise processes and $u(t) \sim \text{unif}(\mathcal{R})$ is a random process for false alarms. We assume that the targets are indistinguishable in this paper, but if measurements include target-type or attribute information, then the state variable can be extended to include target-type information as done in [30].

The objective of the multitarget-tracking problem is to estimate K, $\{t_i^k, t_f^k\}$ and $\{x^k(t): t_i^k \le t \le t_f^k\}$ for $k = 1, \ldots, K$ from noisy measurements.

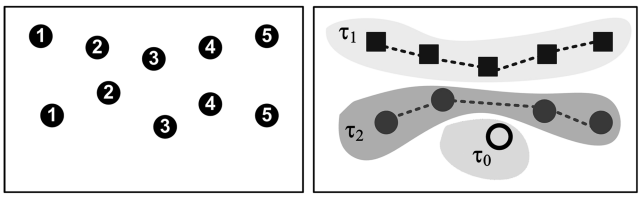
Let $y(t) = \{y^j(t): j = 1, \ldots, n(t)\}$ be all measurements at time t and $Y = \{y(t): 1 \le t \le T\}$ be all measurements from $t = 1$ to $t = T$. Let Ω be a collection of partitions of Y such that for $\omega \in \Omega$, $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$, where τ_0 is a set of false alarms and τ_k is a set of measurements from target k for $k = 1, \ldots, K$. Note that ω is also known as a *joint-association event* in literature. More formally, ω is defined as the following: 1) $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$, 2) $\cup_{k=0}^{K} \tau_k = Y$ and $\tau_i \cap \tau_j = \emptyset$ for $i \ne j$, 3) τ_0 is a set of false alarms, 4) $|\tau_k \cap y(t)| \le 1$ for $k = 1, \ldots, K$ and $t = 1, \ldots, T$, and 5) $|\tau_k| \ge 2$ for $k = 1, \ldots, K$.

An example of a partition is shown in Fig. 4, in which K is the number of tracks for the given partition $\omega \in \Omega$, and $|\tau_k|$ denotes the cardinality of the set τ_k. We call τ_k a track when there is no confusion, although the actual track is the set of estimated states from the measurements τ_k. However, we assume there is a deterministic function that returns a set of estimated states, given a set of measurements, and so no distinction is required. A track is assumed to contain at least two measurements, because we cannot distinguish a track with a single measurement from a false alarm, assuming $\lambda_f > 0$. For special cases in which $p_d = 1$ or $\lambda_f = 0$, the definition of Ω can be adjusted accordingly.

Let $n_e(t-1)$ be the number of targets at time $t-1$, $n_z(t)$ be the number of targets terminated at time t, and $n_c(t) = n_e(t-1) - n_z(t)$ be the number of targets from time $t-1$ that have not terminated at time t. Let $n_b(t)$ be the number of new targets at time t, $n_d(t)$ be the number of actual target detections at time t, and $n_u(t) = n_c(t) + n_b(t) - n_d(t)$ be the number of undetected targets. Finally, let $n_f(t) = n(t) - n_d(t)$ be the number of false alarms. Notice that the numbers $n_z(t)$, $n_c(t)$, $n_d(t)$, $n_u(t)$, $n_b(t)$, and $n_f(t)$ can be computed for given ω.

Using the Bayes rule, it can be shown that the posterior of ω is

$$P(\omega|Y) \propto P(\omega) \cdot P(Y|\omega) \propto \prod_{t=1}^{T} p_z^{n_z(t)} (1-p_z)^{n_c(t)} p_d^{n_d(t)} \times (1-p_d)^{n_u(t)} (\lambda_b V_{\mathcal{R}})^{n_b(t)} (\lambda_f V_{\mathcal{R}})^{n_f(t)} \cdot P(Y|\omega) \quad (3)$$



a) b)

Fig. 4 An example of partition measurements: a) measurements Y (each circle represents a measurement and numbers represent measurement times) and b) partition ω of Y (associations are indicated by dotted lines and hollow circles are false alarms).

where $P(Y|\omega)$ is the likelihood of measurements Y given ω, which can be computed based on the chosen dynamic and measurement models. Our formulation of Eq. (3) is similar to MHT [18], and the derivation of Eq. (3) can be found in [37]. The parameters p_z, p_d, λ_b, and λ_f have been widely used in many multitarget-tracking applications [13,18]. Our experimental and simulation experiences show that our tracking algorithm is not sensitive to changes in these parameters in most cases. For example, the computation of $P(Y|\omega)$ for the linear dynamic and measurement models can be found in [25].

In this paper, we take the maximum a posteriori (MAP) approach to multitarget tracking, similar to MHT. The MAP approach finds a partition of measurements that maximizes $P(\omega|Y)$ and estimates the states of targets based on this partition.

B. State Estimation

Based on our general framework for multitarget tracking described in the previous section, we describe an algorithm for estimating the number of targets and states of targets. The DMTIM system implements the *online* MCMCDA algorithm, which is an approximate implementation of MCMCDA. First, the MCMCDA algorithm is described.

1. MCMCDA

MCMC plays a significant role in many fields such as physics, statistics, economics, and engineering [38–40]. The MCMC method includes algorithms such as Gibbs sampling [41] and the Metropolis–Hastings algorithm [42,43]. MCMC techniques have been applied to complex probability distribution integration problems, counting problems, and combinatorial optimization problems [39]. In some cases, MCMC is the only known general algorithm that finds a good approximate solution to a complex problem in polynomial time [44].

MCMC is a general method to generate samples from a distribution π on a space Ω by constructing a Markov chain $\mathcal{M}$ with states $\omega \in \Omega$ and stationary distribution $\pi(\omega)$. We now describe an MCMC algorithm known as the Metropolis–Hastings algorithm. 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

Algorithm 1 MCMCDA

Input: Y, n_{mc}, an ω_{init}
Output: $\hat{\omega}$
  $\omega = \omega_{\text{init}}$; $\hat{\omega} = \omega_{\text{init}}$
  for $n = 1$ to n_{mc} do
    propose ω' based on ω
    sample U from Unif[0, 1]
    $\omega = \omega'$ if $U < A(\omega, \omega')$
    $\hat{\omega} = \omega$ if $p(\omega|Y)/p(\hat{\omega}|Y) > 1$
  end for

$$A(\omega, \omega') = \min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right) \tag{4}$$

otherwise, the sampler stays at ω, and so 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 [45].

The MCMC data-association (MCMCDA) algorithm [25] is described in Algorithm 1. MCMCDA is an MCMC algorithm for which the state space Ω is described in Sec. III.A and for which the stationary distribution is the posterior (3). The proposal distribution for MCMCDA consists of five types of moves (a total of eight moves). They are the 1) birth/death move pair, 2) split/merge move pair, 3) extension/reduction move pair, 4) track-update move, and 5) track-switch move.

The MCMCDA moves are graphically illustrated in Fig. 5. 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, and 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, and 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 measurements 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 Eq. (4), in which $\pi(\omega) = P(\omega|Y)$ from Eq. (3). The output $\hat{\omega}$ approximates the MAP estimate $\arg\max P(\omega|Y)$. Given $\hat{\omega}$, the states

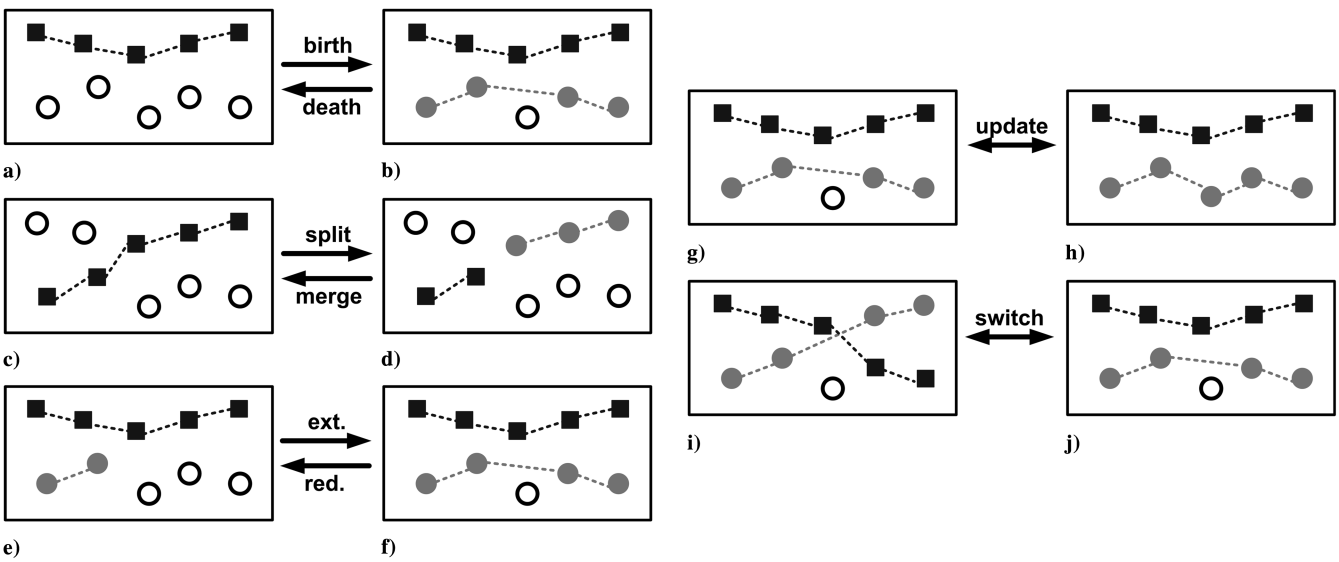


Fig. 5 Graphical illustration of MCMCDA moves (associations are indicated by dotted lines and hollow circles are false alarms). Each move proposes a new joint-association event ω' that is a modification of the current joint-association event ω. The birth move proposes ω' by forming a new track from the set of false alarms (Fig. 5a to Fig. 5b). The death move proposes ω' by combining one of the existing tracks into the set of false alarms (Fig. 5b to Fig. 5a). The split move splits a track from ω into two tracks (Fig. 5c to Fig. 5d); whereas the merge move combines two tracks in ω into a single track (Fig. 5d to Fig. 5c). The extension move extends an existing track in ω (Fig. 5e to Fig. 5f) and the reduction move reduces an existing track in ω (Fig. 5f to Fig. 5e). The track-update move chooses a track in ω and assigns different measurements from the set of false alarms (Fig. 5g to Fig. 5h and the reverse). The track-switch move chooses two track from ω and switches some measurement-to-track associations (Fig. 5i to Fig. 5j and the reverse).

of the targets can be easily computed by any filtering algorithm because the associations between the targets and the measurements are known. Algorithm 1 can be also used to find the Bayesian estimates of the target states (see [26] for more detail).

An MCMC algorithm can be specialized and made more efficient by incorporating the domain-specific knowledge. In multitarget tracking, we can make two assumptions:

1) The maximum directional speed of any target in $\mathcal{R}$ is less than $\bar{v}$.

2) The number of consecutive missing measurements of any track is less than $\bar{d}$.

The first assumption is reasonable in a surveillance scenario because, in many cases, the maximum speed of a vehicle is generally known based on the vehicle type and terrain conditions. The second assumption is a user-defined parameter. Let $\bar{p}_{dt}(s) = 1 - (1 - p_d)^s$ be the probability that a target is observed at least once out of s measurement times. Then for a given $\bar{p}_{dt}$, we set $\bar{d} = \lceil \log(1 - \bar{p}_{dt}) / \log(1 - p_d) \rceil$ to detect a track with probability of at least $\bar{p}_{dt}$. For example, given $p_d = 0.7$ and $\bar{p}_{dt} = 0.99$, a track is detected with a probability larger than 0.99 for $\bar{d} \geq 4$. We will now assume that these two new conditions are added to the definition of Ω so that each element $\omega \in \Omega$ satisfies these two additional assumptions.

We use a data structure, called a neighborhood tree of measurements, which groups temporally separated measurements based on distances, to propose a new partition ω' in Algorithm 1. A neighborhood tree of measurements is defined as

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

for $d = 1, \ldots, \bar{d}$, $j = 1, \ldots, n_t$, and $t = 1, \ldots, T-1$, where $\|\cdot\|$ is the Euclidean distance, and the parameter d allows missing measurements. The use of this neighborhood tree makes the algorithm more scalable, because distant measurements 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 measurements.

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 with indexing of time; that is, 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. For a birth move (Fig. 5a to Fig. 5b), 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 target. Then we choose d_1 from the distribution ζ. Let

$$L_{d_1}^1 = \left\{y_{t_1}^j \colon L_{d_1}\left(y_{t_1}^j\right) \neq \emptyset, y_{t_1}^j \notin \tau_k(t_1), j = 1, \ldots, n_{t_1}, k = 1, \ldots, K\right\}$$

$L_{d_1}^1$ is a set of measurements 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 $L_{d_1}^1$. If $L_{d_1}^1$ is empty, the move is rejected, because the move is not reversible. Once the initial measurement is chosen, we then choose the subsequent measurements 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

$$L_{d_i}[\tau_{K'}(t_{i-1})] \setminus \{\tau_k(t_{i-1} + d_i) \colon k = 1, \ldots, K\}$$

unless this set is empty. But, for $i = 3, 4, \ldots$, the process of adding measurements to $\tau_{K'}$ terminates with probability p_z. If $|\tau_{K'}| \leq 1$, then the move is rejected. We then propose this modified partition in which $\omega'^1 = \omega^1 \cup \{\tau_{K'}\}$ and $\omega'^0 = \{\tau_0 \setminus \tau_{K'}\}$. For a death move (Fig. 5b to Fig. 5a), we simply choose k u.a.r. from $\{1, \ldots, K\}$, delete the kth track, and propose a new partition in which $\omega'^1 = \omega^1 \setminus \{\tau_k\}$ and $\omega'^0 = \{\tau_0 \cup \tau_k\}$.

b. Split and Merge Moves. For a split move (Fig. 5c to Fig. 5d), we select $\tau_s(t_r)$ u.a.r. from

$$\{\tau_k(t_i) \colon |\tau_k| \geq 4, i = 2, \ldots, |\tau_k| - 2, k = 1, \ldots, K\}$$

Then we split the track τ_s into τ_{s_1} and τ_{s_2} such that $\tau_{s_1} = \{\tau_s(t_i) \colon i = 1, \ldots, r\}$ and $\tau_{s_2} = \{\tau_s(t_i) \colon i = r+1, \ldots, |\tau_s|\}$. The modified track partition becomes $\omega'^1 = (\omega^1 \setminus \{\tau_s\}) \cup \{\tau_{s_1}\} \cup \{\tau_{s_2}\}$ and $\omega'^0 = \omega^0$. For a merge move (Fig. 5d to Fig. 5c), we consider the following set of possible merge move pairs:

$$M_{\text{sp}} = \{[\tau_{k_1}(t_f), \tau_{k_2}(t_1)] \colon \tau_{k_2}(t_1) \in L_{t_1 - t_f}[\tau_{k_1}(t_f)], f = |\tau_{k_1}| \text{ for } k_1 \neq k_2, 1 \leq k_1, k_2 \leq K\}$$

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 in which $\omega'^1 = [\omega^1 \setminus (\{\tau_{s_1}\} \cup \{\tau_{s_2}\})] \cup \{\tau_s\}$ and $\omega'^0 = \omega^0$.

c. Extension and Reduction Moves. In a track-extension move (Fig. 5e to Fig. 5f), we select a track τ u.a.r. from K available tracks in ω. We reassign measurements for τ after the disappearance time $t_{|\tau|}$, as done in the track-birth move. For a track-reduction move (Fig. 5f to Fig. 5e), 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 measurements assigned to τ after the time t_{r+1}.

d. Track–Update Move. In a track-update move (Fig. 5g to Fig. 5h), 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 measurements for τ after the time t_r as done in the track-birth move.

e. Track–Switch Move. For a track-switch move (Fig. 5i to Fig. 5j), we select a pair of measurements $[\tau_{k_1}(t_p), \tau_{k_2}(t_q)]$ 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 $d = t_{p+1} - t_q$, $d' = t_{q+1} - t_p$, $0 < d$, and $d' \leq \bar{d}$. Then we let

$$\tau_{k_1} = \{\tau_{k_1}(t_1), \ldots, \tau_{k_1}(t_p), \tau_{k_2}(t_{q+1}), \ldots, \tau_{k_2}(t_{|\tau_{k_2}|})\}$$

$$\tau_{k_2} = \{\tau_{k_2}(t_1), \ldots, \tau_{k_2}(t_q), \tau_{k_1}(t_{p+1}), \ldots, \tau_{k_1}(t_{|\tau_{k_1}|})\}$$

2. *Online MCMCDA*

Although the computational complexity of the MCMCDA algorithm described in Sec. III.B.1 is lighter than MHT [25], it grows as more measurements are collected. Because recent measurements are more relevant to the current states, good estimates of the current states can still be found from recent measurements. Based on this idea, we propose an online MCMCDA algorithm for which the estimates are based on measurements from a window of time $[t_{\text{curr}} - t_{\text{win}} + 1, \ldots, t_{\text{curr}}]$, where t_{curr} is the current time and t_{win} is the size of a window. Hence, at all times, only a finite number of measurements are kept by the algorithm. This online implementation of MCMCDA is suboptimal, because it considers only a subset of past measurements. At each time step, we use the previous estimate to initialize MCMCDA and run MCMCDA on the measurements belonging to the current window. The measurements belonging to the current window are

$$Y_w = \{y^j(t) \colon 1 \leq j \leq n(t), t_{\text{curr}} - t_{\text{win}} + 1 \leq t \leq t_{\text{curr}}\}$$

At time $t_{\text{curr}} + 1$, the measurements at time $t_{\text{curr}} - t_{\text{win}} + 1$ are removed from Y_w and a new set of measurements is appended to Y_w. Any delayed measurements are inserted into the appropriate slots. Then we initialized the Markov chain with the previously estimated tracks and executes Algorithm 1 on Y_w.

Each time that online MCMCDA is executed, it finds the partition $\hat{\omega}$ that approximates the MAP estimate of the multitarget-tracking problem and state estimates for all tracks in $\hat{\omega}$. For each track $\tau \in \hat{\omega}$, we compare it with the tracks of previously identified targets. If τ does not share any measurements with the tracks of previously identified targets, we declare τ as a new target. Then the current sensor makes a query about the identity of τ to its neighboring

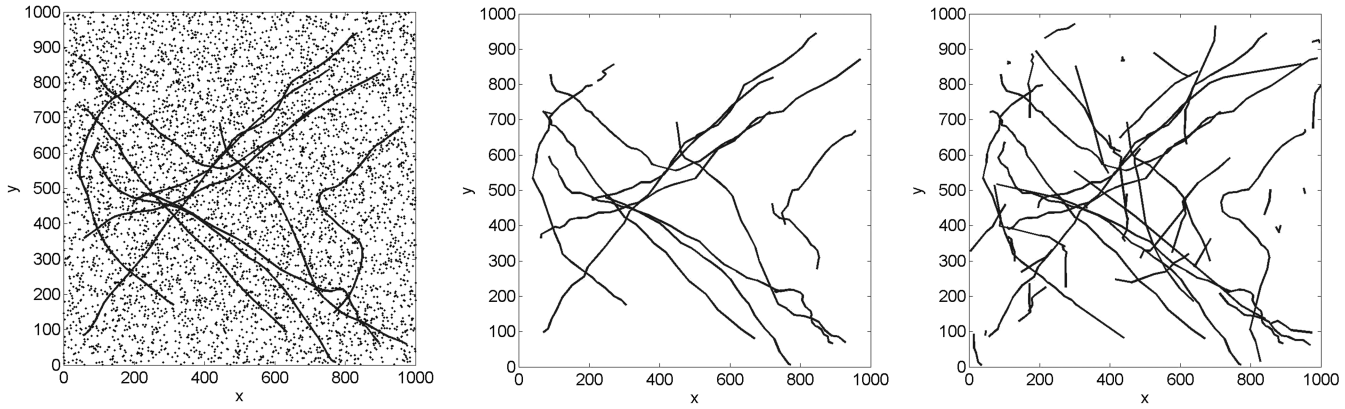


Fig. 6 Online MCMCDA vs MHT: a) an example used in Sec. III.B.3 [trajectories of targets are shown in solid lines and measurements are shown in dots (from $t = 1$ to 100)]; b) tracks estimated by online MCMCDA with 10,000 samples at each simulation time (total execution time is 33.5 s); and c) tracks estimated by MHT (total execution time is 130 s). Although MHT reports many spurious tracks and requires a longer running time, online MCMCDA provides excellent track estimates with less running time.

sensors. If the identity of τ is known to the neighboring sensors, its identity is copied to the current sensor. Otherwise, a new identity is created for τ. The identity of a target is deleted when the track of the target is terminated. In Sec. IV.A, we describe how the belief matrix is updated upon changes in the number of identities.

3. Comparison Between MCMCDA and MHT

The performance of MCMCDA is extensively compared against MHT in [25]. The simulation study in [25] compared MCMCDA against MHT with heuristics (i.e., pruning, gating, clustering, N-scan-back logic, and k-best hypotheses) by varying the false-alarm rate, the number of targets (or target density), and the detection rate. MCMCDA outperformed MHT when the false-alarm rate was high or the number of targets was large, and it required significantly less computation time [25]. At varying detection rate, MCMCDA achieved performance similar to MHT. For more information about the comparison between MCMCDA and MHT, see [25]. In this section, an illustrative example is described to demonstrate the robustness of MCMCDA.

An example of tracking multiple targets in a dense clutter environment is used to compare online MCMCDA against MHT (see Fig. 6a). For this example, the surveillance duration is $T = 100$, and there are 10 targets appearing and disappearing at random times over the surveillance region $\mathcal{R} = [0, 1000] \times [0, 1000]$. The other parameters are $p_d = .9$, $\lambda_f = 5.0 \times 10^{-5}$, $\bar{d} = 10$, and $\bar{v} = 100$ unit lengths per unit time. A uniform mass function is used for each $\xi_K(\cdot)$, and $\zeta(d)$ is computed based on p_d. The state vector of a target is $x = [x, \dot{x}, y, \dot{y}]^T$, where (x, y) is a position along the usual x and y axes and $(\dot{x}, \dot{y})$ is a velocity vector. Simple linear dynamics and measurement models are used in simulation:

$$x^k(t+1) = Ax^k(t) + Gw^k(t), \qquad y^j(t) = Cx^k(t) + v^j(t) \quad (5)$$

where

$$A = \begin{bmatrix} 1 & \delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} \delta^2/2 & 0 \\ \delta & 0 \\ 0 & \delta^2/2 \\ 0 & \delta \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}^T$$

δ is the sampling period, $w^k(t)$ is a zero-mean Gaussian process with covariance $Q = \text{diag}(25, 25)$, and $v^j(t)$ is a zero-mean Gaussian process with covariance $R = \text{diag}(25, 25)$.

The online MCMCDA algorithm is written in C^{++} with MATLAB interfaces. The size of the sliding window is $t_{\text{win}} = 10$ and we used 10,000 samples at each simulation time. We used the C^{++} implementation of MHT[¶] [46], which implements pruning, gating, clustering, N-scan-back logic, and k-best hypotheses. The following parameters are used for MHT: the maximum number of hypotheses in a group is 1000, the maximum track tree depth is 5, and the maximum Mahalanobis distance is 11.8. All simulations are run on a PC with a 2.6-GHz Intel processor.

Figure 6b shows the tracks estimated by online MCMCDA, and its overall execution time was 33.5 s. The tracks estimated by MHT are shown in Fig. 6c, and its overall execution time was 130.0 s. Even with a longer execution time, MHT reports many spurious tracks. On the other hand, online MCMCDA provides excellent track estimates with less running time.

C. Mixing Matrix

Suppose there are K targets and K identities in the surveillance region of the current sensor. Then the problem of managing identities of multiple targets is to match each target to its identity over time. For this, we use the idea of the identity-mass flow [21]. The idea of the identity-mass flow is that an identity is treated as a unit mass assigned to a target. These masses cannot be destroyed or created and they flow from one target into another through a *mixing matrix* $M(t)$ at time t. A mixing matrix is a $K \times K$ matrix for which the element $M_{ij}(t)$ represents the probability that target i at time $t-1$ has become target j at time t. Thus, a mixing matrix stores information about the interactions among targets for a single sampling period. It is a doubly stochastic matrix; that is, its column sums and row sums are equal to 1.

Let $K'(t)$ be the number of targets estimated by the online MCMCDA at time t. We are interested in computing a mixing matrix for the targets that are present both at time $t-1$ and t. Let K be the number of targets present at time $t-1$ and t (i.e., $K = \min[K'(t-1), K'(t)]$), and this excludes any disappeared targets or new targets. Without loss of generality, assume that the first K targets are present at time $t-1$ and t. Let

$$\hat{x}(t-1) = \{\hat{x}^k(t-1)\colon 1 \le k \le K\}$$

be the state estimates of targets at time $t-1$ and

$$\hat{x}(t) = \{\hat{x}^k(t)\colon 1 \le k \le K\}$$

be the current state estimates computed by online MCMCDA. The mixing matrix entry $M_{ij}(t)$ represents the probability that the target with state $\hat{x}^i(t-1)$ at time $t-1$ has become the target with state $\hat{x}^j(t)$ at time t. In theory, we can use Algorithm 1 to compute this probability. Instead, we use a single-scan version of MCMCDA to reduce the computation time. Single-scan MCMCDA uses concepts from JPDA [26]. We first encode this target-to-target association event in a bipartite graph. Let $G = (U, V, E)$ be a bipartite graph, where $U = \{1, \ldots, K\}$ is a set of target indices at time $t-1$, $V =$

[¶]Data available online at http://www.adastral.ucl.ac.uk/~icox/.

$\{1, \ldots, K\}$ is a set of target indices at time t, and

$$E = \{(i, j): i \in U, j \in V, P[\hat{x}^j(t)|\hat{x}^i(t-1)] > \epsilon_0\}$$

for some small $\epsilon_0 > 0$. A feasible target-to-target association event ϕ is a *matching* in G (i.e., a subset $E' \subset E$ such that no two edges in E' share a vertex). Let $\Phi = \{\phi\}$ be the set of all feasible target-to-target association events. The posterior of ϕ given $\hat{x}(t)$ and $\hat{x}(t-1)$ can be computed using the Bayes rule:

$$\begin{aligned} P[\phi|\hat{x}(t), \hat{x}(t-1)] &= \frac{1}{Z} P[\hat{x}(t)|\hat{x}(t-1), \phi] P[\hat{x}(t-1)|\phi] P(\phi) \\ &= \frac{1}{Z} \left\{ \prod_{(i,j)\in\phi} P[\hat{x}^j(t)|\hat{x}^i(t-1)] \right\} P[\hat{x}(t-1)|\phi] P(\phi) \end{aligned} \quad (6)$$

where Z is a normalizing constant. We assume that matchings of the same size are equally likely; hence,

$$P(\phi) \propto \binom{|E|}{|\phi|}^{-1} \propto |\phi|!(|E| - |\phi|)!$$

where $|E|$ is the number of edges in E, and $|\phi|$ is the number of matches in ϕ. When computing a mixing matrix, we assume that $\{\hat{x}^i(t-1)\}$ are fixed for the targets with known identities. So we simply set $P(\hat{x}(t-1)|\phi) = 1$.

Under these assumptions, the posterior (6) reduces to

$$P[\phi|\hat{x}(t), \hat{x}(t-1)] = \frac{1}{Z'} \left\{ \prod_{(i,j)\in\phi} P[\hat{x}^j(t)|\hat{x}^i(t-1)] \right\} |\phi|!(|E| - |\phi|)! \quad (7)$$

where Z' is another normalizing constant. Each $P[\hat{x}^j(t)|\hat{x}^i(t-1)]$ can be computed using the target's dynamic model; hence, the posterior (7) can be computed up to a normalizing constant. Notice that $P[\hat{x}^j(t)|\hat{x}^i(t-1)]$ can be computed exactly for linear-Gaussian dynamic models and approximately for the cases when the dynamic model is nonlinear and/or the noise processes are non-Gaussian, using methods such as linearization, unscented filtering [47], interacting multiple models [48], and particle filters [49]. Hence, our objective is to compute the mixing matrix based on our estimates $\{P[\hat{x}^j(t)|\hat{x}^i(t-1)]\}$.

Let ϕ_{ij} be the event such that the target with state $\hat{x}^i(t-1)$ becomes the target with state $\hat{x}^j(t)$. Now the mixing probability $M_{ij}(t)$ can be computed as

$$M_{ij}(t) = P[\phi_{ij}|\hat{x}(t), \hat{x}(t-1)] = \sum_{\phi\in\Phi:\ (i,j)\in\phi} P[\phi|\hat{x}(t), \hat{x}(t-1)] \quad (8)$$

The computation of $M_{ij}(t)$ requires a summation over the posteriors, hence the enumeration of all joint-association events. The exact computation of a mixing matrix is NP-hard. More generally, the exact computation of association probabilities in JPDA is NP-hard [50] because the related problem of finding the permanent of a 0-1 matrix is ♯P-complete [51].** Hence, for a large problem (i.e., when the number of targets is large), we need to seek an efficient approximation algorithm. In the remainder of this section, we describe a polynomial-time approximation algorithm based on MCMC: namely, the Metropolis–Hastings algorithm. The algorithm for computing a mixing matrix is shown in Algorithm 2. The inputs to Algorithm 2 are the graph G, state estimates $\hat{x}(t)$ and $\hat{x}(t-1)$, an initial Markov chain state ϕ_{init} that is chosen randomly from Φ, the number of MCMC samples n_{mc}, and the number of burn-in MCMC samples n_{bi}. The mixing matrix is estimated by the Monte Carlo integration of samples after the burn-in period. For more information about burn-in samples and general MCMC techniques, we refer readers to [38]. Notice that we only need to compute the ratio

Algorithm 2 MCMC for mixing matrix computation

Input: n_{bi}, n_{mc}, $G = (U, V, E)$, $\hat{x}(t)$, $\hat{x}(t-1)$, and ϕ_{init}
Output: $\hat{M}(t)$
  $\phi = \phi_{\text{init}}$; $\hat{M}(t) = \mathbf{0}^{K\times K}$
  for $n = 1$ to n_{mc} do
    choose $e = (u, e) \in E$ uniformly at random
    if $e \in \phi$, then
      $\phi' = \phi - e$
    else if both u and v are unmatched in ϕ, then
      $\phi' = \phi + e$
    else if exactly one of u and v is matched in ϕ and e' is the matching edge, then
      $\phi' = \phi + e - e'$
    else
      $\phi' = \phi$
    end if
    $\phi = \phi'$ with probability

$$\min\left(1, \frac{P[\phi'|\hat{x}(t), \hat{x}(t-1)]}{P[\phi|\hat{x}(t), \hat{x}(t-1)]}\right)$$

    if $n > n_{\text{bi}}$ then
      for each $(i, j) \in \phi$ do
        $\hat{M}_{ij}(t) = \hat{M}_{ij}(t) + 1$;
      end for
    end if
  end for
  for each $(i, j) \in \{(i', j'): 1 \le i', j' \le K\}$ do
    $\hat{M}_{ij}(t) = \hat{M}_{ij}(t)/(n_{\text{mc}} - n_{\text{bi}})$
  end for

$$P[\phi'|\hat{x}(t), \hat{x}(t-1)]/P[\phi|\hat{x}(t), \hat{x}(t-1)]$$

avoiding the need to normalize $P[\phi|\hat{x}(t), \hat{x}(t-1)]$.

Although heuristic approaches do not guarantee asymptotic optimality and may fail in some situations, Algorithm 2 can approximate the mixing matrix $M = [M_{ij}]$ (the time index is suppressed) in polynomial time with guaranteed error bounds.

Theorem 1: For any $0 < \epsilon_1, \epsilon_2 < 1$ and $0 < \eta < 0.5$, with time complexity

$$\mathcal{O}\left(\epsilon_0^{-4}\epsilon_1^{-2}\epsilon_2^{-1} \log \eta^{-1} K^8 \left[K \log \epsilon_0^{-1} + \log\left(\epsilon_1^{-1}\epsilon_2^{-1}\right)\right]\right) \quad (9)$$

Algorithm 2 finds estimates $\hat{M}_{ij}$ for M_{ij} with probability at least $1 - \eta$, such that for $M_{ij} \ge \epsilon_2$, $\hat{M}_{ij}$ estimates M_{ij} within ratio $1 + \epsilon_1$ [i.e., $(1 - \epsilon_1)M_{ij} \le \hat{M}_{ij} \le (1 + \epsilon_1)M_{ij}$] and for $M_{ij} < \epsilon_2$, $|\hat{M}_{ij} - M_{ij}| \le (1 + \epsilon_1)\epsilon_2$.

Theorem 1 is a corollary of Theorem 2 and Theorem 3 of [52] after replacing the number of measurements N in [52] with the number of targets K and letting $R = |E|/\epsilon_0$ and $m_5(K, N) = \mathcal{O}(K \log \epsilon_0^{-1})$. The idea of the proof is as follows: It is well known that the *mixing time* of a Markov chain determines the rate of convergence of the Markov chain to its stationarity [44]. We first show that the mixing time of MCMCDA is polynomial in the size of the problem and $\log \epsilon^{-1}$, where ϵ is a precision parameter [52]. Then it is easy to show that the estimates computed by MCMCDA are very close to their true values.

D. Local Information

In some applications, identity information about a target (local information) could be obtained from sensors that can measure its physical attributes or from the target's dynamic characteristics. When local information is available, we use local information to decrease the uncertainty of the belief matrix measured by entropy. MCMCDA, described in Sec. III.B.1, allows an efficient way to compute local information from both the latest and past measurements. Another benefit of MCMCDA is that local information can be computed simultaneously while the number of targets and tracks of all targets are estimated. For identity k, let N_{jk} be the number of times the jth latest measurement is associated with the initial measurement identified by k after the initial n_{bi} samples while

**A ♯P-complete problem is computationally equivalent to computing the number of accepting computations of a polynomial-time nondeterministic Turing machine, and ♯P contains NP [44].

running Algorithm 1, where n_{bi} is the number of initial burn-in samples and $n_{\text{bi}} < n_{\text{mc}}$. When Algorithm 1 terminates, we compute $\gamma^k = (\gamma_1^k, \ldots, \gamma_{n(t)}^k)^T$ for identity k, where $\gamma_j^k = N_{jk}/(n_{\text{mc}} - n_{\text{bi}})$. Then we form local information from γ^k by resizing the vector according to the latest measurements assigned in state estimates and normalizing the resized vector.

IV. Distributed Multitarget Identity Management

A. Identity Management

The IM module consists of belief-matrix update and local-information incorporation. In the IM module, a mixing matrix and local information from the DAMTT module are used to update the belief matrix.

1. *Belief-Matrix Update*

The belief-matrix-update block maintains identity information stored in a belief matrix $B(t)$ for which the columns are belief vectors of the targets over time. The evolution of this belief matrix is governed by a mixing matrix $M(t)$, which stores interaction information for a single time step. Then the belief matrix is updated by [21]

$$B(t) = B(t-1)M(t) \tag{10}$$

We can show that Eq. (10) keeps row and column sums of the belief-matrix constant when the numbers of targets and identities are the same. However, this is not the case for distributed identity management, because the number of the targets within the surveillance region of individual sensors may change over time. There are two possible cases: a target leaves or enters the surveillance region of a sensor. When a target leaves, we delete the corresponding column in the belief matrix managed by the sensor. When a target enters the surveillance region of a sensor, there are two possible cases:

1) The target comes from the surveillance region of another sensor, which may be queried.

2) The target comes from the outside of the surveillance region of a sensor network.

For these cases, we propose Algorithm 3, a scalable, event-driven, query-based belief-matrix-update algorithm.

For distributed identity management, a belief matrix managed by each sensor may not be a square matrix, but might more likely be a skinny matrix, which has more rows than columns. The belief matrix may not be a doubly stochastic matrix for which the row and column sums are equal to one, but it should be a stochastic matrix with column sums equal to one. Its row sums remain constant, because an identity mass cannot be destroyed or created. Because the evolution of the belief matrix is governed by Eq. (10), these characteristics of the belief matrix are preserved over time.

2. *Local-Information Incorporation*

When local information is available, we use local information to decrease the uncertainty of the belief matrix measured by entropy. The entropy (Shannon information) of an $L \times K$ belief matrix is defined as

$$H[B(t)] \triangleq -\sum_{i=1}^{L}\sum_{j=1}^{K} B_{ij}(t)\log B_{ij}(t) \tag{11}$$

Then the problem is how to incorporate this information to the belief matrix. From the idea of the identity-mass flow and the characteristics of Eq. (10), we know that the belief matrix should have the following properties: its column sums are equal to one and its row sums remain constant. However, if we replace a column in the belief matrix with local information, it is not guaranteed that the new belief matrix has the preceding properties. For a nonnegative square matrix, the Sinkhorn algorithm [53] can be used to scale a matrix to achieve specified row and column sums [54,55]; that is, we scale a new belief matrix so that its row and column sums remain the same as those of the belief matrix before local-information incorporation. However, because the belief matrix is, in general, a nonnegative rectangular matrix, such iteration may not converge [54–56]. But the question whether a given matrix is *almost* scalable (i.e., the matrix can be scaled within $\epsilon > 0$) can be decided in polynomial time [56]. Thus, we can efficiently check whether the available local information can be incorporated. Thus, local information can be incorporated when it makes the new belief matrix almost scalable. Even though the new belief matrix is almost scalable, local information incorporated may not necessarily decrease the uncertainty (entropy) of the belief matrix. Therefore, local information is incorporated only when it reduces the uncertainty of the belief matrix. The local-information-incorporation algorithm is described in Algorithm 4.

B. Identity and Track Fusion

For DMTIM, the identity and track fusion is crucial to compute the global information of the system from information provided by local sensors. In this section, we explain how the ITF modules combines state estimates and belief vectors of the same target from neighboring sensors.

1. *Identity Fusion*

We now consider the problem of combining two belief vectors of the same target from two different sensors. Identity fusion can be formulated as an optimization problem such that the fused identity is the one that minimizes a cost function, which represents a performance criterion. For optimization, we propose three different cost functions: Shannon information, Chernoff information, and the sum of Kullback–Leibler distances. We then derive a Bayesian identity-fusion method and discuss the relationship between the Bayesian method and the optimization algorithms.

a. Shannon Information. The Shannon information is defined as

$$H(b') = \sum_{i=1}^{n} -b'(i)\log b'(i) \tag{12}$$

Algorithm 3 Event-driven, query-based belief-matrix update

For sensor i and target k,
if target k leaves the surveillance region of sensor i, then
  delete the corresponding column in the belief matrix.
end if
if a target enters the surveillance region of sensor i, then
  send a query about the identity of target k.
  if there is an answer "yes" and the belief vector of target k is received, then
    augment the belief matrix with the belief vector received.
  else
    augment the belief matrix with a belief vector with a new identity assigned to the target.
  end if
end if

Algorithm 4 Local-information incorporation

Given local information (belief vector) of a target and a belief matrix $B(t)$.
Make a matrix $B'(t)$ by replacing the column corresponding to the target in $B(t)$ with the local information.
Operator S represents the matrix scaling process in [56].
if $B'(t)$ is almost scalable, then
  $B_{\text{new}}(t) := \mathsf{S}[B'(t)]$
  if $H[N_{\text{new}}(t)] \leq H[B(t)]$ then
    $B(t) := B_{\text{new}}(t)$
  else
    $B(t) := B(t)$
  end if
else
  $B(t) := B(t)$
end if

where $b' \in [0,1]^n$ with

$$\sum_i b'(i) = 1$$

The Shannon information (also known as *entropy*) is a measure of the uncertainty of a system. Thus, the minimization of the Shannon information selects a belief vector that is most informative in the sense of minimum entropy. Suppose b_1 and b_2 are belief vectors of target t computed by sensor 1 and sensor 2, respectively. Because the most common data-fusion algorithms compute a *linear combination* of two data, we propose the following fusion strategy:

$$b' = \lambda b_1 + (1-\lambda) b_2 \tag{13}$$

where $\lambda \in [0,1]$ and $b_i \in [0,1]^n$ with

$$\sum_{j=1}^{n} b_i(j) = 1$$

for $i \in \{1,2\}$ and

$$\sum_{j=1}^{n} b'(j) = 1$$

Then the problem of computing the fused belief vector becomes a problem of finding a weight λ that minimizes the cost function in Eq. (12). If we use the fusion strategy in Eq. (13), then the Shannon information of the new fused belief vector is

$$H(b') = H[\lambda b_1 + (1-\lambda) b_2] \geq \lambda H(b_1) + (1-\lambda) H(b_2) \tag{14}$$

From Eq. (14), we can see that the minimum is always achieved at either $\lambda = 0$ or $\lambda = 1$. This means that a fused belief vector that has the minimum Shannon information is either of the two given belief vectors, which is a *hard* choice. For some applications, such as identity management in this paper, the hard choice may not be desirable because it ignores one possibility completely and thus might quickly lead to a wrong answer over time, if not immediately. Thus, we propose a *soft* decision method that has $\lambda \in (0,1)$ for almost all cases. Motivated by the fact that Shannon-information minimization chooses a belief vector that has the minimum entropy, we propose to use the inverse of the Shannon information of a belief vector as a weight. Thus, we put large confidence on a belief vector that has small Shannon information. Then a new belief vector $b' = [b'(i)]$ is

$$b'(i) = \frac{H(b_1)^{-1} b_1(i)}{H(b_1)^{-1} + H(b_2)^{-1}} + \frac{H(b_2)^{-1} b_2(i)}{H(b_1)^{-1} + H(b_2)^{-1}} \tag{15}$$

From Eqs. (13) and (15), we get

$$\lambda = \frac{H(b_1)^{-1}}{H(b_1)^{-1} + H(b_2)^{-1}} = \frac{H(b_2)}{H(b_1) + H(b_2)} \tag{16}$$

When $H(b_1) = H(b_2) = 0$, we set $\lambda = \frac{1}{2}$; $\lambda = 0$ if $H(b_2) = 0$ (no uncertainty in b_2); and $\lambda = 1$ when $H(b_1) = 0$ (no uncertainty in b_1). In these cases, the fused belief vector computed by the proposed fusion algorithm is a belief vector that has no uncertainty. This fusion algorithm is a soft decision method because the fused data are a convex combination of the two given data with a larger weight on the data that have smaller entropy. From Eqs. (14) and (16), the Shannon information of the new belief $H(b')$ has the property that

$$H(b') \geq \frac{2H(b_1)H(b_1)}{H(b_1) + H(b_2)} \quad \text{or} \quad 2H(b')^{-1} \leq H(b_1)^{-1} + H(b_2)^{-1} \tag{17}$$

Inequality (17) tells us that the achievable minimum uncertainty of the fused belief vector with the fusion strategy in Eq. (13) with Eq. (16) as a weight is lower-bounded by uncertainties of the given information. In other words, the maximum achievable certainty (inverse of the Shannon information) is upper-bounded by the arithmetic mean of the inverse of the Shannon information of the given belief vectors. If we use the fusion strategy in Eq. (13), we can also derive the upper bound of the Shannon information of the new belief vector:

$$H(b') \leq \lambda^2 H(b_1) + (1-\lambda)^2 H(b_2) + \lambda(1-\lambda)[H(b_1) + H(b_2) + D(b_1 \parallel b_2) + D(b_2 \parallel b_1)] \tag{18}$$

where

$$D(p \parallel q) \triangleq \sum_i p(i) \log\left(\frac{p(i)}{q(i)}\right)$$

is the Kullback–Leibler distance [57]. If we use λ in Eq. (16), then

$$H(b') \leq \frac{2H(b_1)H(b_1)}{H(b_1) + H(b_2)} + \frac{H(b_1)H(b_2)[D(b_1 \parallel b_2) + D(b_2 \parallel b_1)]}{[H(b_1) + H(b_2)]^2} \tag{19}$$

Thus, we can analytically compute the upper and lower bounds of the Shannon information of the new belief vector using the fusion strategy in Eq. (13) with Eq. (16). If we interpret the error covariance matrix as an uncertainty measure of an estimate of a continuous random variable, we find that Eq. (17) could be interpreted as an analogy of the *Cramér–Rao lower bound* of the mean-squared error (error covariance) of any unbiased estimator, in the sense that the achievable minimum uncertainty has a lower bound. Thus, the Shannon-information cost function would be useful when we have good knowledge about the performance and/or fidelity of each sensor, because we can get a solution that has lower entropy by weighing information that has lower entropy than the other. However, if we do not have such knowledge, we may get a biased solution by consistently putting more confidence on one piece of information (possibly the wrong one) than the other.

b. Chernoff Information. The Chernoff information is defined as

$$C(b_1, b_2) = -\min_{0 \leq \lambda \leq 1} \log\left(\sum_{i=1}^{n} b_1(i)^{\lambda} b_2(i)^{1-\lambda}\right) \tag{20}$$

If λ^* minimizes Eq. (20), the new belief vector b' ($[b'(i)]$ for $i = \{1, 2, \cdots, n\}$) is

$$b'(i) = \frac{b_1(i)^{\lambda^*} b_2(i)^{1-\lambda^*}}{\sum_{j=1}^{n} b_1(j)^{\lambda^*} b_2(j)^{1-\lambda^*}} \tag{21}$$

The new belief vector in Eq. (21) satisfies [57,58]

$$D(b' \parallel b_1) = D(b' \parallel b_2) \tag{22}$$

This fusion strategy is different from that in Eq. (13), which is a convex combination of the two data. From Eq. (22), the minimization of the Chernoff information is equivalent to finding a function that is in the *middle* of the two original functions, in which the middle is defined in terms of the Kullback–Leibler distance. In other words, Chernoff-information minimization could be interpreted as selecting a probability vector that is "equally close" in terms of the Kullback–Leibler distance to the original probability vectors. This fusion algorithm does not put more confidence on one than the other. Thus, this cost function could be useful when we do not know the quality of information obtained from individual sensors; by choosing the middle point of the two pieces of information, we could minimize the bias over time. However, the fused belief vector computed by the Chernoff-information-minimization algorithm may have larger entropy than that computed by the algorithm in Eq. (13) with Eq. (16).

c. Sum of the Kullback–Leibler Distances. Because the Kullback–Leibler distance is not symmetric, we consider two possible optimization problems:

Minimize

$$D(b' \parallel b_1) + D(b' \parallel b_2) \tag{23}$$

subject to

$$\sum_{j=1}^{n} b'(j) = 1 \qquad b'(j) \geq 0$$

Minimize

$$D(b_1 \parallel b') + D(b_2 \parallel b') \tag{24}$$

subject to

$$\sum_{j=1}^{n} b'(j) = 1 \qquad b'(j) \geq 0$$

where $b'(j)$ is the jth element of a vector b'. Let us first consider the optimization problem in Eq. (23). The Lagrangian is given by

$$L(b', \lambda) = D(b' \parallel b_1) + D(b' \parallel b_2) + \lambda\left(\sum_{j=1}^{n} b'(j) - 1\right) \tag{25}$$

To get an optimal solution, we set the derivatives of L with respect to $b'(i)$ and λ to be equal to zero. Then we get a new belief vector:

$$b'(i) = \frac{\sqrt{b_1(i)b_2(i)}}{\sum_{j=1}^{n}\sqrt{b_1(j)b_2(j)}} \tag{26}$$

From Eq. (26), we see that the fused data are a geometric mean of the given data. The fused data are the same as those in Eq. (21) for Chernoff-information minimization when $\lambda^* = \frac{1}{2}$. Thus, this data-fusion strategy can be interpreted as a special case of the Chernoff-information-minimization method.

Now let us consider the optimization problem in Eq. (24). The Lagrangian is given by

$$L(b', \lambda) = D(b_1 \parallel b') + D(b_2 \parallel b') + \lambda\left(\sum_{j=1}^{n} b'(j) - 1\right) \tag{27}$$

Similarly, we get an optimal solution

$$b'(i) = \frac{b_1(i) + b_2(i)}{\sum_{j=1}^{n}[b_1(j) + b_2(j)]} = \frac{b_1(i) + b_2(i)}{2} \tag{28}$$

In this case, the fused data are the arithmetic mean of the given data. This fusion strategy is the same as that in Eq. (13) when $\lambda = \frac{1}{2}$. Thus, from Eqs. (14) and (18), we get the lower and upper bounds of Shannon information of the new belief vector:

$$\begin{aligned} H(b') &\geq \frac{H(b_1) + H(b_2)}{2} \\ H(b') &\leq \frac{H(b_1) + H(b_2)}{2} + \frac{D(b_1 \parallel b_2) + D(b_2 \parallel b_1)}{4} \end{aligned} \tag{29}$$

Therefore, the fusion algorithms obtained by solving the optimization problems in Eq. (23) or Eq. (24) are to average the given data either geometrically or arithmetically. This is similar to Chernoff information minimization and thus these fusion strategies would be useful when we want to get unbiased fused data in situations in which we do not have good a prior information about a system. An example would be a case in which information from one sensor is wrong due to failure of the sensor or the malicious intent of the sensor that is unknown a priori. These information-fusion strategies would be robust to this wrong information, because they do not put more confidence on one (possibly incorrect information) than the other, but average them to compute a fused belief vector.

d. Bayesian Approach. In this section, we derive a fused belief vector using a Bayesian approach. Suppose the target's identity $\theta \in \{1, 2, \cdots, N\}$ and, without loss of generality, suppose there are two sensors. Denote events X_1 and X_2 to be measurements at sensor 1 and sensor 2, respectively. We are assumed to be given information $b_1(\theta) \triangleq P(\theta|X_1)$ from sensor 1 and $b_2(\theta) \triangleq P(\theta|X_2)$ from sensor 2, in which $P(\cdot|\cdot)$ is a conditional probability. Then the problem of identity fusion is to find the a posteriori probability $P(\theta|X_1, X_2)$. We assume $P(X_1, X_2|\theta) = P(X_1|\theta)P(X_2|\theta)$ because given the identity of a target, the events that are observed by sensor 1 or sensor 2 are independent in distributed identity management. Using the Bayes rule, we get

$$P(\theta|X_1, X_2) = \frac{P(X_1|\theta)P(X_2|\theta)P(\theta)}{P(X_1, X_2)} \tag{30}$$

Because

$$P(\theta|X_i) = \frac{P(X_i|\theta)P(\theta)}{P(X_i)}$$

for $i \in \{1, 2\}$, we obtain

$$P(\theta|X_1, X_2) = \frac{b_1(\theta)b_2(\theta)}{P(\theta)}\frac{P(X_1)P(X_2)}{P(X_1, X_2)} \tag{31}$$

Therefore, a fused belief vector is

$$b'(\hat{\theta}) = \arg\max_{\theta} P(\theta|X_1, X_2) = \frac{b_1(\theta)b_2(\theta)}{P(\theta)} \cdot \frac{1}{c} \tag{32}$$

where c is a normalization constant. This is an interesting result because the fused data do depend only on the given data $[b_1(\theta), b_2(\theta)]$ and the a priori probability $P(\theta)$. Thus, if we knew a priori information, we could compute the a posteriori probability (i.e., the fused data). However, because we may not know the a priori probability for some applications such as distributed identity management in this paper, we cannot compute the fused data from Eq. (32). To compute the fused data for this case, we have to assume $P(\theta)$ either from the characteristics of the systems or from that of applications. For example, if we assume the a priori probability as a geometric mean of the given data due to lack of information about a system $[P(\theta) = \sqrt{b_1(\theta)b_2(\theta)}]$, then we can get exactly the same result as that in Eq. (26), which minimizes the sum of Kullback–Leibler distances to the original data in Eq. (23). Thus, the a posteriori probability is the same as the a priori probability; that is, we cannot extract any information from the given data. From Bayesian analysis, we can see that the data-fusion strategies such as Chernoff-information minimization and the minimization of the sum of Kullback–Leibler distances in Eq. (23) compute the solution in a similar form to the solution produced by the Bayesian approach.

In this paper, we are considering the case in which good knowledge about the performance/fidelity of all sensors is available; hence, the Shannon-information method is used. For a comprehensive comparison of identity-fusion methods, see [22].

2. Track Fusion

Because each sensor maintains its own set of tracks, there can be multiple tracks from the same target maintained by different sensors. To resolve this inconsistency, we do track-level data association to combine tracks from different sensors, as described in [29]. Let ω_i be the set of tracks maintained by sensor i and NB_i be a set of neighboring sensors around i, including i itself. Let

$$Y'_{\mathrm{w}} = \{\tau_k(t)\colon \tau_k \in \omega_j, 1 \leq t \leq T, 1 \leq k \leq |\omega_j|, j \in \mathrm{NB}_i\}$$

be a set of measurements of all identified targets. We form a set of combined measurements Y_w from Y'_w by combining measurements made from overlapping surveillance regions and keeping the remaining measurements.[††] We then form a new set of tracks ω_{init} from $\{\tau \in \omega_j\colon j \in \mathrm{NB}_i\}$ while making sure that constraints defined in Sec. III.A are satisfied. Then we run Algorithm 1 on the set of

[††]In our current implementation, when multiple measurements from different sensors are in close proximity (measured by the Mahalanobis distance using the measurement covariance matrix), their mean value is used in Y_w instead. In our future implementation, we plan to allow multiple measurements over the overlapping regions by relaxing constraints in Sec. III.A.

combined measurements Y_w with the initial state ω_{init} to find locally consistent tracks.

V. Simulation Results

In this section, we present two sets of simulations to illustrate the features of the DMTIM algorithm. There are stationary sensors (e.g., air traffic control radars) tracking multiple aircraft through two-dimensional space. The sensing range of each sensor is assumed to be circular with a radius of 10 km, and a pair of sensors can communicate if they are within the communication radius of 20 km. We first present a two-sensor scenario with three aircraft and two sensors and describe the behavior of DMTIM in detail. Then we describe an example with five aircraft and seven sensors to demonstrate a complete DMTIM system. Moderate-sized examples are chosen for better illustration.

The nonlinear dynamics of maneuvering aircraft is modeled using interacting multiple models [48] with two linear kinematic models based on the discrete-time linear dynamics (5):

Model 1 second-order kinematic model:

$$A(v^k(t)=1)=\begin{bmatrix}1&\delta&0&0\\0&1&0&0\\0&0&1&\delta\\0&0&0&1\end{bmatrix}$$

$$G(v^k(t)=1)=\begin{bmatrix}\delta^2/2&0\\\delta&0\\0&\delta^2/2\\0&\delta\end{bmatrix}$$

and $Q(v_t^k=1)=\text{diag}(0.1,0.1)$, where $v^k(t)$ indexes the kinematic model of target k operating at time t and δ is the sampling period. The state vector is $x=(x_1,\dot{x}_1,x_2,\dot{x}_2)^T$. This model assumes that the variation in a velocity component is a discrete-time white-noise acceleration [59].

Model 2 third-order kinematic model:

$$A(v^k(t)=2)=\begin{bmatrix}1&\delta&\delta^2/2&0&0&0\\0&1&\delta&0&0&0\\0&0&1&0&0&0\\0&0&0&1&\delta&\delta^2/2\\0&0&0&0&1&\delta\\0&0&0&0&0&1\end{bmatrix}$$

$$G(v^k(t)=2)=\begin{bmatrix}\delta^2/2&0\\\delta&0\\1&0\\0&\delta^2/2\\0&\delta\\0&1\end{bmatrix}$$

and $Q(v_t^k=2)=\text{diag}(20,20)$. The state vector is $x=(x_1,\dot{x}_1,\ddot{x}_1,x_2,\dot{x}_2,\ddot{x}_2)^T$. This is a third-order kinematic model with accelerations modeled as a discrete-time Wiener process [59].

The measurement model given in Eq. (5) is used for both models, in which $R=\text{diag}(200,200)$. The other parameters are $p_d=0.98$, $\lambda_f=1.0\times10^{-7}$, and $\delta=5$. The online MCMCDA algorithm described in Sec. III.B.3 is used in DMTIM.

A. Two-Sensor Scenario

A simple, yet illustrative, scenario with three aircraft is shown in Fig. 7a. The aircraft labeled *A* and *B* are previously registered and aircraft labeled *X* is unknown to the identity-management system. The sensor on the left is denoted by sensor 1 and the sensor on the right is denoted by sensor 2.

The DAMTT module of each sensor estimates the number of targets (Fig. 7b) and estimates tracks of targets, as shown in Figs. 7c and 7d. In Fig. 7b, the events in which the number of targets changes are indicated by dotted vertical lines. The belief vector for each target (i.e., a column of the belief matrix) computed by the IM module is shown in Figs. 8a and 8b. At time 1, sensor 1 knows about target 1 and its belief vector is $(b_{A,1}^1, b_{B,1}^1)^T=(0.8,0.2)^T$, where $b_{j,k}^i$ is the probability that target k of sensor i can be identified as label j, and

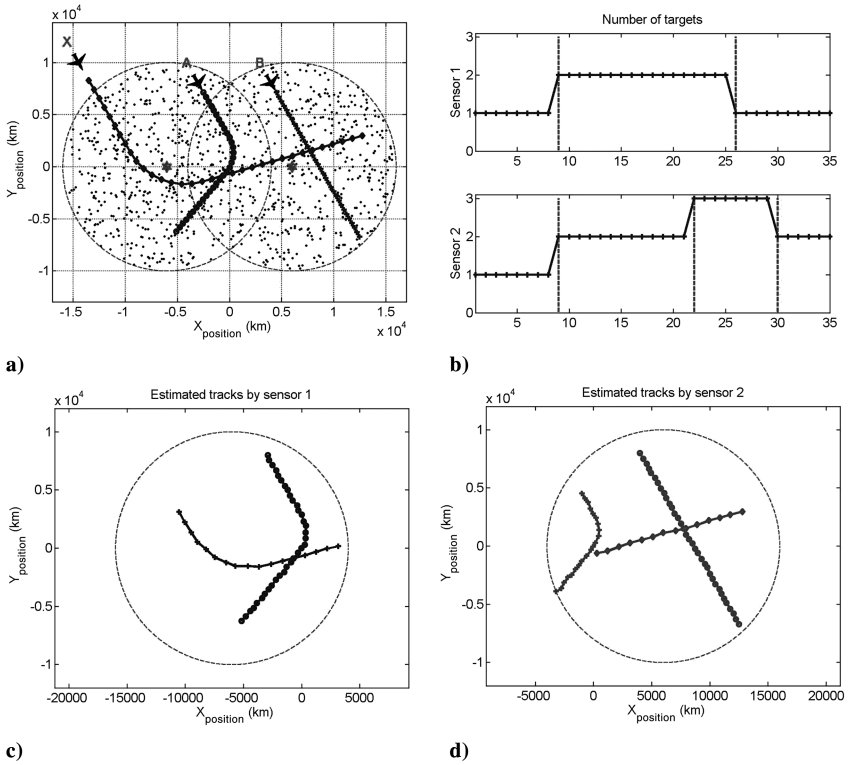


Fig. 7 Two-sensor scenario: a) trajectories for three aircraft, superimposed with accumulated measurements marked as dots and sensor positions marked by ⋆, b) estimated number of targets by each sensor, c) tracks estimated by sensor 1, and d) tracks estimated by sensor 2. (A track of a target is shown after its first detection.)

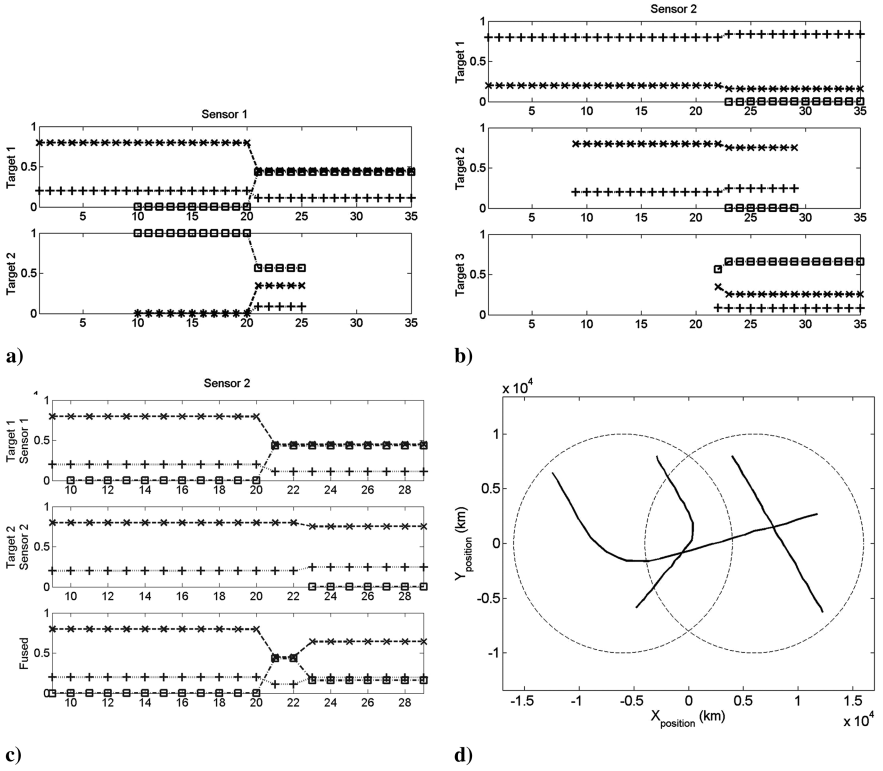


Fig. 8 Fusion of belief vectors and tracks: a) local belief vectors computed by sensor 1, b) local belief vectors computed by sensor 2, c) fused belief vectors between target 1 of sensor 1 and target 2 of sensor 2, and d) fused tracks. The symbols ×, +, and □ denote aircraft *A*, aircraft *B*, and aircraft *X*, respectively.

sensor 2 knows about its target 1 and its belief vector is $(b^2_{A,1}, b^2_{B,1})^T = (0.2, 0.8)^T$. At time 9, sensor 1 detects a new target (target 2 of sensor 1) and assigns a new identity X because the target is unknown to its neighboring sensors. The updated belief vectors are shown in Fig. 8a. At the same time, sensor 2 detects a new target (target 2 of sensor 2) and its identity and state-estimate information is transferred from sensor 1, because its track is recognized as target 1 of sensor 1. The updated belief vectors are shown in Fig. 8b. At time 22, sensor 2 detects a new target (target 3 of sensor 2) and its identity and state-estimate information is transferred from sensor 1, because its track is recognized as target 2 of sensor 1. At time 26, target 2 of sensor 1 leaves the surveillance region of sensor 1 and information about target 2 is removed from sensor 1. At time 30, target 2 of sensor 2 leaves the surveillance region of sensor 2 and information about target 2 is removed from sensor 2.

For illustration purpose, Figs. 8a and 8b are showing the local belief vectors at each sensor before ITF. At time 21, aircraft A and aircraft X cross one another and the uncertainty about identity is increased, as shown in Fig. 8a. For example, the belief that target 1 of sensor 1 can be identified with aircraft A is reduced from 0.8 to 0.45. However, ITF can reduce this uncertainty by fusing the belief vector of target 1 of sensor 1 and the belief vector of target 2 of sensor 2. We use Shannon information for ITF because we are considering a cooperative situation and it has been shown that Shannon information is superior against the other criteria in terms of cooperative efficiency [22]. The fused belief vector is shown in Fig. 8c, in which the belief vectors from times 9 to 29 are shown. When target 3 of sensor 2 appears at time 23 (i.e., one time step after its detection), the identity uncertainty is reduced by ITF. For example, the (fused) belief that target 1 of sensor 1 can be identified with aircraft A is increased from 0.45 to 0.64, as shown in the bottom plot of Fig. 8c. Finally, the tracks estimated by each sensor in a distributed manner are fused by ITF and the fused tracks are shown in Fig. 8d.

B. Seven-Sensor Scenario

There are seven air traffic control radars (ATC1 through ATC7) and five aircraft (T1 through T5), as shown in Fig. 9, which represents the state of the system at simulation time $t = 17$. The left frame of Fig. 9 shows the position and heading of each aircraft, along with the sensing range (circle) of each sensor and measurements (dots). As the simulation time progresses, the left frame of Fig. 9 will show the estimated tracks and event logs. The right frame of Fig. 9 shows belief vectors of aircraft at each sensor. The order of the belief vector is dictated by the order in which the target is registered to the corresponding sensor. Out of five targets, only three targets (T1, T2, and T3) are known to the system initially, and the targets T4 and T5 are unknown to the system. At time $t = 1$, targets T1, T2, and T3 are registered to ATC1 and target T3 is registered to ATC2. The identities ID1, ID2, and ID3 are associated to targets T1, T2, and T3, respectively. At $t = 8$, ATC1 terminates T3 because the target moves away from the sensing region of ATC1. At $t = 15$, ATC3 detects a new target and the target is labeled as A3-T1 and identity ID4 is assigned to this target. At $t = 16$, ATC6 detects a new target and the target is labeled as A6-T1 and identity ID5 is assigned to this target. At the same time, ATC1 transfers T1 to ATC3 (see Fig. 9). The snapshots at $t = 51$ and 52 are shown in Figs. 10 and 11, respectively. The uncertainty about the identities at $t = 52$ can be observed from the mixing of belief vectors.

At $t = 80$, the belief matrix of ATC7 is

$$B_{\mathrm{ATC7}}(80) = \begin{bmatrix} 0.2572 & 0.0514 & 0.0343 & 0.6571 & 0 \\ 0.4928 & 0.0986 & 0.0657 & 0.3429 & 0 \end{bmatrix}^T \quad (33)$$

where the rows 1 through 5 of the belief matrix $B_{\mathrm{ATC7}}(80)$ correspond to identities ID1 through ID5, and the first and second columns of the belief matrix correspond to targets A3-T1 and T1, respectively.

At the same time, local information about target A3-T1 is obtained by ATC7, and target A3-T1 is now thought to have belief $[0.1\ \ 0\ \ 0\ \ 0.9\ \ 0]^T$. This local information is incorporated according to Algorithm 4. First, a new matrix $B'_{\mathrm{ATC7}}(80)$ is formed by replacing the column for target A3-T1 with local information, where

$$B'_{\mathrm{ATC7}}(80) = \begin{bmatrix} 0.1 & 0 & 0 & 0.9 & 0 \\ 0.4928 & 0.0986 & 0.0657 & 0.3429 & 0 \end{bmatrix}^T \quad (34)$$

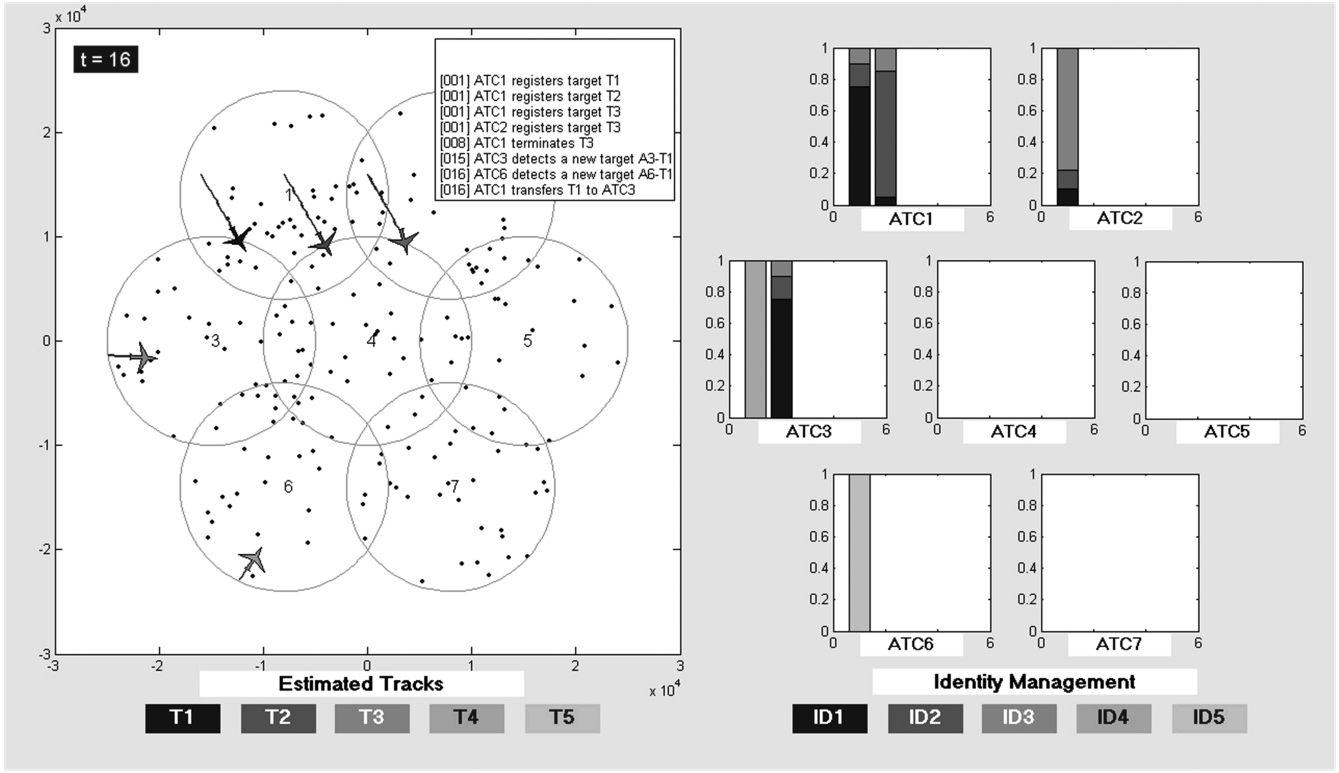


Fig. 9 Simulation time $t = 16$; the left frame shows the estimated tracks and event logs. Circles and dots represent sensing ranges and measurements, respectively. The grayscale of aircraft corresponds the true identity based on the grayscale shown on the left. The right frame shows the belief vectors maintained by each sensor. The grayscale of each segment of the bar graph represents an identity based on the grayscale shown on the right. At $t = 15$, ATC3 also detects a new target and the target is labeled as A3-T1 and identity ID4 is assigned to this target. At $t = 16$, ATC6 detects a new target and the target is labeled as A6-T1 and identity ID5 is assigned to this target. At the same time, ATC1 transfers T1 to ATC3.

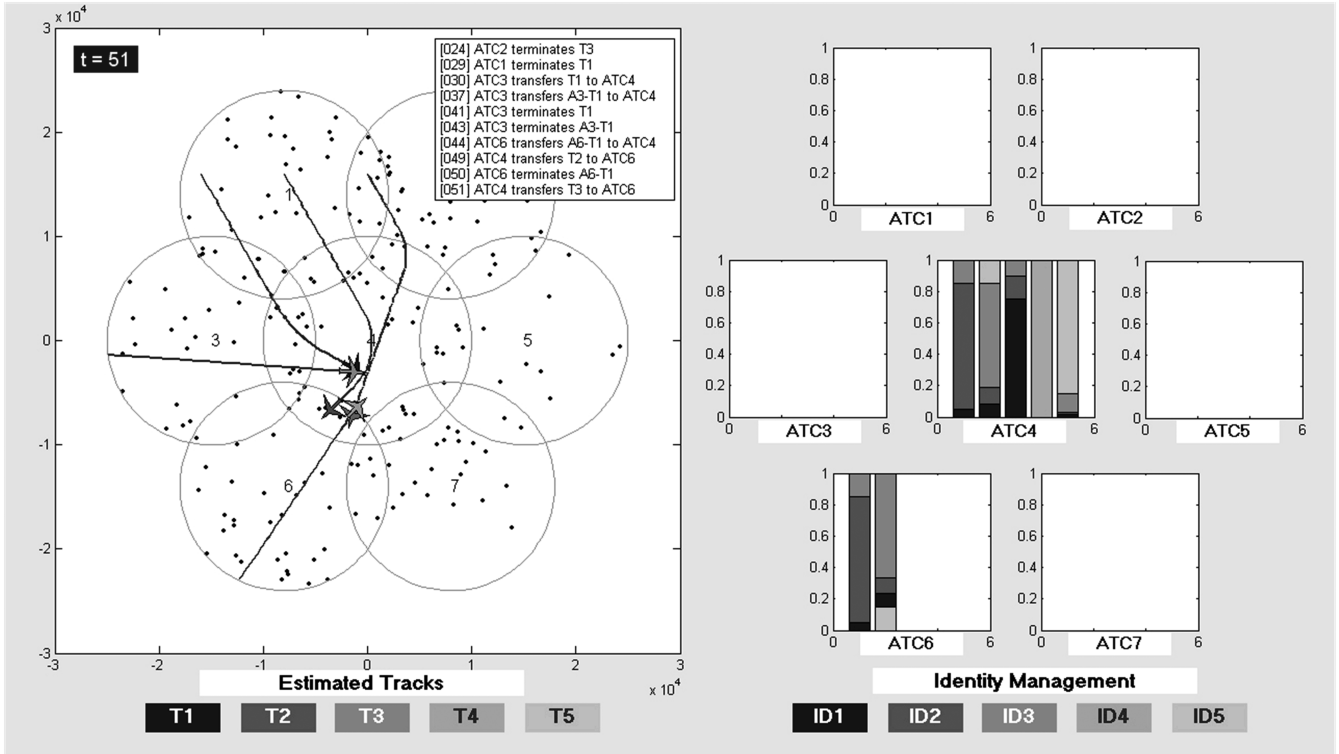


Fig. 10 Simulation time $t = 51$; at $t = 23$, ATC1 terminates T2; at $t = 24$, ATC2 terminates T3; at $t = 29$, ATC1 terminates T1; at $t = 30$, ATC3 transfers T1 to ATC4; at $t = 37$, ATC3 transfers A3-T1 to ATC4; at $t = 41$, ATC3 terminates T1; at $t = 43$, ATC3 terminates A3-T1; at $t = 44$, ATC6 transfers A6-T1 to ATC4; at $t = 49$, ATC4 transfers T2 to ATC6; at $t = 50$, ATC6 terminates A6-T1, targets T3 and A6-T1 move close to each other, and the belief vectors of targets T3 and A6-T1 are mixed in ATC4; at $t = 51$, ATC4 transfers T3 to ATC6.

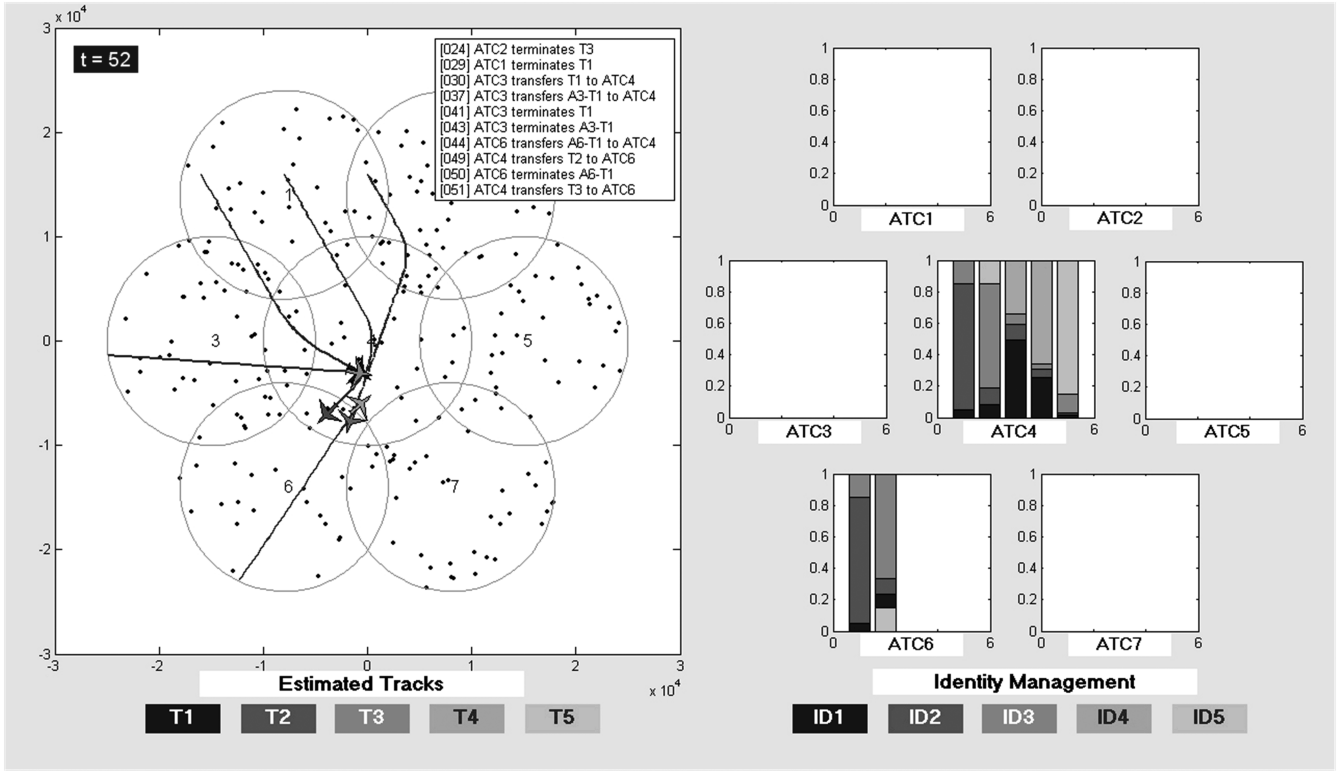


Fig. 11 Simulation time $t = 52$; at $t = 52$, targets T1 and A3-T1 move close to each other and the belief vectors of targets T1 and A3-T1 are mixed.

Because $B'_{\mathrm{ATC7}}(80)$ is almost scalable, Algorithm 4 computes the scaled belief matrix:

$$B^{\mathrm{new}}_{\mathrm{ATC7}}(80) = \begin{bmatrix} 0.1879 & 0 & 0 & 0.8121 & 0 \\ 0.5621 & 0.15 & 0.1 & 0.1879 & 0 \end{bmatrix}^T \tag{35}$$

The scaled belief matrix $B^{\mathrm{new}}_{\mathrm{ATC7}}(80)$ decreases entropy from 2.9091 to 2.3602. Hence, we assign $B_{\mathrm{ATC7}}(80) = B^{\mathrm{new}}_{\mathrm{ATC7}}(80)$. See Fig. 12 and compare it against the updated belief matrix in Fig. 13.

To evaluate the performance of DMTIM, we compared DMTIM against the centralized version of the algorithm. In the centralized version, there is a single ATC at [0,0], with a larger sensing radius of 26 km. The same set of measurements are used, except those on

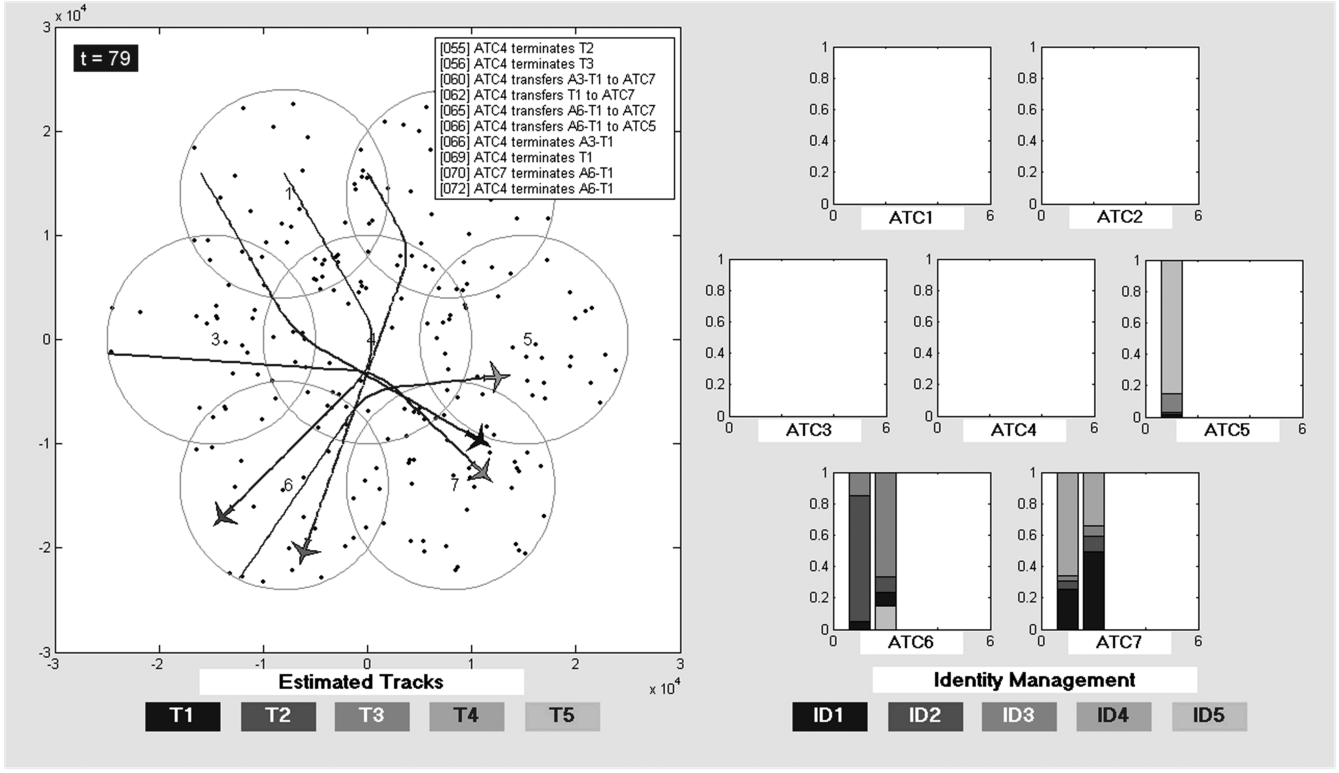


Fig. 12 Simulation time $t = 79$; before the belief-matrix update by ATC7.

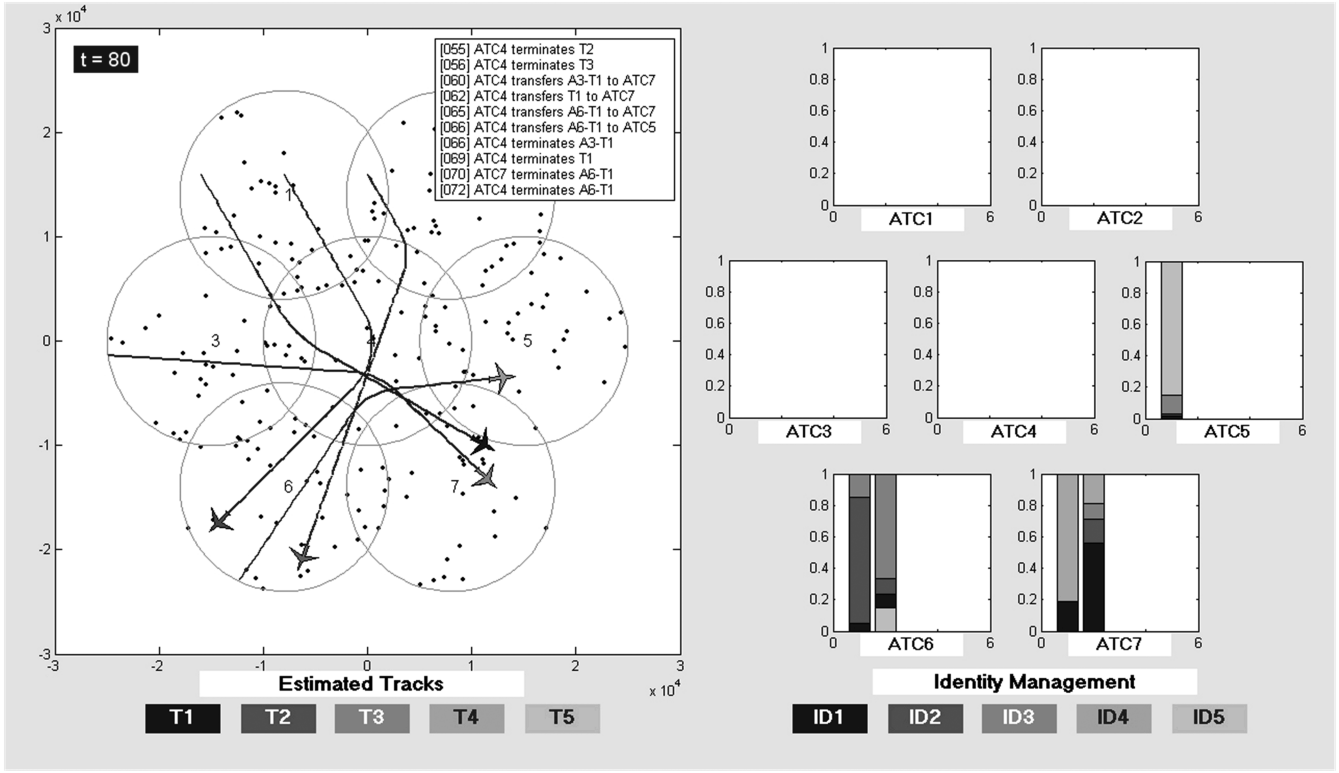


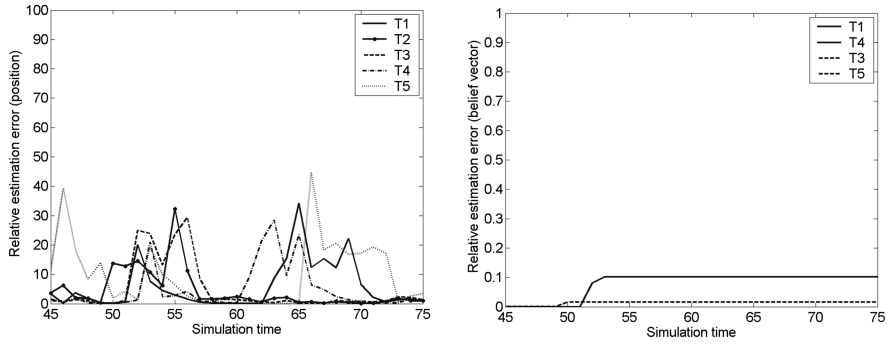
Fig. 13 Simulation time $t = 80$; the belief matrix is updated due to the local information about target A3-T1 obtained by ATC7; see the text for detail.

the overlapping sensing regions. When there are multiple measurements made by overlapping ATCs, a randomly chosen measurement is used by the centralized algorithm. The performance comparison between DMTIM and the centralized algorithm is made by comparing belief vectors and state estimates from both algorithms. The relative estimation errors in the position and belief-vector estimations are shown in Fig. 14. The relative position estimation error is high when there is confusion or when the same targets are detected by multiple sensors. The latter is due to the fact that a measurement from a single target can be different for a different ATC. Because the norm of standard deviations of position can be 20.77 when $\nu_t^k = 1$ and 81.54 when $\nu_t^k = 2$ (including both process and measurement noises), the position estimates of DMTIM are close to the estimates of the centralized algorithm. At $t = 52$ and 53, the relative position estimates are high for all targets, and this is when the relative belief-vector-estimation error increases. As shown in Fig. 14b, the belief-vector-estimation error may increase over time, hence, it is necessary to perform the local-information incorporation to reduce the error, as described earlier. The centralized algorithm required more computation time than DMTIM because it processes more measurements each time. Hence, the centralized algorithm is not a practical solution for a large-scale sensor network.

VI. Conclusions

We proposed a scalable distributed multitarget-tracking and identity-management (DMTIM) algorithm that can track an unknown number of targets and manage their identities efficiently in a distributed sensor network environment. A decision based on a single set of tracks can be risky, due to the uncertainty in the identities of targets accumulated from continuous interactions among crossing or nearby targets. DMTIM provides a scalable and distributed solution to the multitarget-tracking and identity-management problem by combining an efficient data-association algorithm and identity-management methods. The novelty of the system architecture of DMTIM includes modularity, the compact representation of identity and tracks to reduce communication load, and event-driven mechanisms for identity and track management.

DMTIM consists of data association, multitarget tracking, identity management, and identity and track fusion. The data-association and multitarget-tracking problems are efficiently solved by MCMCDA.



a) b)

Fig. 14 Relative estimation error in a) the position of each target and b) belief vectors of each target from simulation time $t = 45$ to $t = 75$. For each target, the figures plot the maximum estimation error among all ATCs at each simulation time. The estimation error at each ATC is computed by calculating the vector norm of the difference between estimates made by the ATC and the centralized algorithm.

DMTIM efficiently incorporates local information about the identity of a target to reduce the uncertainty in the system and maintains local consistency among neighboring sensors via identity and track fusion.

In this paper, the system is evaluated using simple measurement models. We are currently applying more sophisticated measurement models to evaluate the performance of the system. In particular, we are applying our method to the distributed camera network, in which the measurements can be, but are not limited to, position, color, texture, and shape. Currently, the DAMTT module produces the mixing matrix for the IM module. But updated identity information from the IM and ITF modules are not used in the DAMTT module. We are currently developing methods to improve the performance of DAMTT by incorporating updated identity information in DAMTT.

Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) under grant EIA-0122599 and the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the NSF, award number CCF-0424422, and the following organizations: Cisco, Embedded Systems Consortium for Hybrid and Embedded Research (ESCHER) Institute, Hewlett-Packard (HP), IBM, Intel, Microsoft, Oak Ridge National Laboratory (ORNL), Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies. The authors wish to thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Estrin, D., Culler, D., Pister, K., and Sukhatme, G., "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, Vol. 1, No. 1, Jan. 2002, pp. 59–69. doi:10.1109/MPRV.2002.993145

[2] Akyidliz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E., "A Survey on Sensor Networks," *IEEE Communications Magazine*, Vol. 40, No. 8, Aug. 2002, pp. 102–116. doi:10.1109/MCOM.2002.1024422

[3] Gharavi, H., and Kumar, S. P., "Special Issue on Sensor Networks and Applications," *Proceedings of the IEEE*, Vol. 91, No. 8, Aug. 2003, pp. 1151–1153.10.1109/JPROC.2003.814925

[4] Estrin, D., Girod, L., Pottie, G., and Srivastava, M., "Instrumenting the World with Wireless Sensor Networks," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Vol. 4, May 2001, pp. 2033–2036.

[5] Culler, D., Estrin, D., and Srivastava, M., "Overview of Sensor Networks," *Computer*, Vol. 37, No. 8, Aug. 2004, pp. 41–49. doi:10.1109/MC.2004.93

[6] Kintner-Meyer, M., and Conant, R., "Opportunities of Wireless Sensors and Controls for Building Operation," *Energy Engineering : Journal of the Association of Energy Engineers*, Vol. 102, No. 5, 2005, pp. 27–48; also *Proceedings of the 2004 ACEEE Summer Study on Energy Efficiency in Buildings*, American Council for an Energy-Efficient Economy, Washington, DC, 2004, pp. 3-139–3-152; also available online at http://www.buildingsystemsprogram.pnl.gov/wireless/publications/PNNL-SA-41972.pdf.

[7] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A. M., and Estrin, D., "Habitat Monitoring with Sensor Networks," *Communications of the ACM*, Vol. 47, No. 6, 2004, pp. 34–40. doi:10.1145/990680.990704

[8] Nekovee, M., "Ad Hoc Sensor Networks on the Road: The Promises and Challenges of Vehicular Ad Hoc Networks," *Workshop on Ubiquitous Computing and e-Research*, Edinburgh, Scotland, U.K., May 2005.

[9] Willig, A., Matheus, K., and Wolisz, A., "Wireless Technology in Industrial Networks," *Proceedings of the IEEE*, Vol. 93, No. 6, June 2005, pp. 1130–1151. doi:10.1109/JPROC.2005.849717

[10] LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S. B., Sikorski, K., Fox, D., and Borriello, G., "Making Sensor Networks Practical with Robots," *Proceedings of the First International Conference on Pervasive Computing*, Lecture Notes in Computer Science, Vol. 2414, Springer–Verlag, London, 2002, pp. 152–166.

[11] Brooks, R. R., Friedlander, D., Koch, J., and Phoha, S., "Tracking Multiple Targets with Self-Organizing Distributed Ground Sensors," *Journal of Parallel and Distributed Computing*, Vol. 64, No. 7, 2004, pp. 874–884. doi:10.1016/j.jpdc.2003.12.005

[12] Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M., "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer networks*, Vol. 46, No. 5, Dec. 2004, pp. 605–634. doi:10.1016/j.comnet.2004.06.007

[13] Bar-Shalom, Y., and Fortmann, T., *Tracking and Data Association*, Academic Press, New York, 1988.

[14] Cox, I., "A Review of Statistical Data Association Techniques for Motion Correspondence," *International Journal of Computer Vision*, Vol. 10, No. 1, 1993, pp. 53–66. doi:10.1007/BF01440847

[15] Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S., "EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence," *Machine Learning*, Vol. 50, Nos. 1–2, 2003, pp. 45–71. doi:10.1023/A:1020245811187

[16] Cybenko, G., Berk, V., Crespi, V., Gray, R., and Jiang, G., "An Overview of Process Query Systems," *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, Proceedings of SPIE: The International Society for Optical Engineering, Vol. 5403, SPIE—The International Society for Optical Engineering, Bellingham, WA, Apr. 2004, pp. 183–197.

[17] Reid, D., "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control*, Vol. 24, No. 6, Dec. 1979, pp. 843–854. doi:10.1109/TAC.1979.1102177

[18] Kurien, T., "Issues in the Design of Practical Multitarget Tracking Algorithms," *Multitarget-Multisensor Tracking: Advanced Applications*, edited by Y. Bar-Shalom, Artech House, Norwood, MA, 1990.

[19] Chong, C., Mori, S., and Chang, K., "Distributed Multitarget Multisensor Tracking," *Multitarget-Multisensor Tracking: Advanced Applications*, edited by Y. Bar-Shalom, Artech House, Norwood, MA, 1990, pp. 247–295.

[20] Schenato, L., Oh, S., Sastry, S., and Bose, P., "Swarm Coordination for Pursuit Evasion Games Using Sensor Networks," *Proceedings of the 2005 International Conference on Robotics and Automation*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 18–22 Apr. 2005, pp. 2493–2498.

[21] Shin, J., Guibas, L., and Zhao, F., "A Distributed Algorithm for Managing Multi-Target Identities in Wireless Ad-Hoc Sensor Networks," *Information Processing in Sensor Networks (IPSN'03)*, Lecture Notes in Computer Science, Vol. 2634, Springer, New York, Apr. 2003, pp. 223–238.

[22] Hwang, I., Roy, K., Balakrishnan, H., and Tomlin, C., "A Distributed Multiple-Target Identity Management Algorithm in Sensor Networks," *Proceedings of the 43rd IEEE Conference on Decision and Control*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Dec. 2004, pp. 728–734.

[23] Hwang, I., Balakrishnan, H., Roy, K., and Tomlin, C., "Multiple-Target Tracking and Identity Management in Clutter, with Application to Aircraft Tracking," *Proceedings of the 2004 American Control Conference*, Vol. 4, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, June 2004, pp. 3422–3428.

[24] Hwang, I., Balakrishnan, H., Roy, K., and Tomlin, C., "Multiple-Target Tracking and Identity Management with Application to Aircraft Tracking," *Journal of Guidance, Control, and Dynamics* (submitted for publication).

[25] Oh, S., Russell, S., and Sastry, S., "Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems," *Proceedings of the 43rd IEEE Conference on Decision and Control*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Dec. 2004, pp. 735–742.

[26] Oh, S., Russell, S., and Sastry, S., "Markov Chain Monte Carlo Data Association for Multiple-Target Tracking," Univ. of California, TR UCB//ERL M05/19, Berkeley, CA, 2005.

[27] Schulz, D., Burgard, W., Fox, D., and Cremers, A., "Tracking Multiple Moving Targets with a Mobile Robot Using Particle Filters and Statistical Data Association," *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Vol. 2, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2001, pp. 1665–1670.

[28] Karlsson, R., and Gustafsson, F., "Monte Carlo Data Association for Multiple Target Tracking," *Target Tracking: Algorithms &*

Applications, Pt. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2001, pp. 13/1–13/5.

[29] Oh, S., Schenato, L., and Sastry, S., "A Hierarchical Multiple-Target Tracking Algorithm for Sensor Networks," *Proceedings of the 2005 International Conference on Robotics and Automation (ICRA 2005)*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Apr. 2005, pp. 2197–2202.

[30] Pasula, H., Russell, S. J., Ostland, M., and Ritov, Y., "Tracking Many Objects with Many Sensors," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1999, pp. 1160–1171.

[31] Pasula, H., "Identity Uncertainty," Ph.D. Thesis, Computer Science Div., Univ. of California, Berkeley, CA, 2003.

[32] Cong, S., Hong, L., and Wicker, D., "Markov-Chain Monte-Carlo Approach for Association Probability Evaluation," *Control Theory and Applications*, Vol. 151, No. 2, Mar. 2004, pp. 185–193.

[33] Bergman, N., and Doucet, A., "Markov Chain Monte Carlo Data Association for Target Tracking," *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, June 2000, pp. 705–708.10.1049/ip-cta:20040037

[34] Shannon, C. E., "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, July and Oct. 1948, pp. 379–423, 623–656.

[35] Wang, H., Yao, K., Pottie, G., and Estrin, D., "Entropy-Based Sensor Selection Heuristic for Target Localization," *Third International Symposium on Information Processing in Sensor Networks (IPSH 2004)*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Apr. 2004, pp. 36–45.

[36] Joshi, A., Deignan, P., Meckl, P., King, G., and Jennings, K., "Information Theoretic Fault Detection," *Proceedings of the 2005 American Control Conference*, Vol. 3, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, June 2005, pp. 1642–1647.

[37] Oh, S., "Multiple Target Tracking for Surveillance," Univ. of California, TR UCB/ERL MO3/54, Berkeley, CA, 2003.

[38] Gilks, W., Richardson, S., and Spiegelhalter, D. (ed.), *Markov Chain Monte Carlo in Practice*, Interdisciplinary Statistics Series, Chapman and Hall, Norwell, MA, 1996.

[39] Beichl, I., and Sullivan, F., "The Metropolis Algorithm," *Computing in Science and Engineering (1999-) / Computing in Science & Engineering*, Vol. 2, No. 1, 2000, pp. 65–69. doi:10.1109/5992.814660

[40] Eraker, B., "MCMC Analysis of Diffusion Models with Application to Finance," *Journal of Business and Economic Statistics*, Vol. 19, No. 2, 2001, pp. 177–191. doi:10.1198/073500101316970403

[41] Geman, S., and Geman, D., "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, Nov. 1984, pp. 721–741.

[42] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., "Equation of State Calculation by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, No. 6, 1953, pp. 1087–1092. doi:10.1063/1.1699114

[43] Hastings, W., "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, Vol. 57, No. 1, 1970, pp. 97–109. doi:10.1093/biomet/57.1.97

[44] Jerrum, M., and Sinclair, A., "The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration," *Approximations for NP-Hard Problems*, edited by D. Hochbaum, PWS Publishing, Boston, MA, 1996.

[45] Roberts, G., "Markov Chain Concepts Related to Sampling Algorithms," *Markov Chain Monte Carlo in Practice*, edited by W. Gilks, S. Richardson, and D. Spiegelhalter, Interdisciplinary Statistics Series, Chapman and Hall, Norwell, MA, 1996.

[46] Cox, I., "Multiple Hypothesis Tracking Code," http://www.adastral.ucl.ac.uk/~icox/

[47] Julier, S., and Uhlmann, J., "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, Vol. 92, No. 3, 2004, pp. 401–422. doi:10.1109/JPROC.2003.823141

[48] Blom, H. A. P., and Bar-Shalom, Y., "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Transactions on Automatic Control*, Vol. 33, No. 8, Aug. 1988, pp. 780–783. doi:10.1109/9.1299

[49] De Freitas, A. D. J., and Gordon, N. (eds.), *Sequential Monte Carlo Methods In Practice*, Springer–Verlag, New York, 2001.

[50] Collins, J., and Uhlmann, J., "Efficient Gating in Data Association with Multivariate Distributed States," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 28, No. 3, July 1992, pp. 909–916. doi:10.1109/7.256316

[51] Valiant, L., "The Complexity of Computing the Permanent," *Theoretical Computer Science*, Vol. 8, No. 2, 1979, pp. 189–201. doi:10.1016/0304-3975(79)90044-6

[52] Oh, S., and Sastry, S., "A Polynomial-Time Approximation Algorithm for Joint Probabilistic Data Association," *Proceedings of the 2005 American Control Conference*, Vol. 2, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, June 2005, pp. 1283–1288.

[53] Sinkhorn, R., "Diagonal Equivalence to Matrices with Prescribed Row and Column Sums," *American Mathematical Monthly*, Vol. 74, No. 4, 1967, pp. 402–405. doi:10.2307/2314570

[54] Rothblum, U., and Schneider, H., "Scaling of Matrices Which Have Prescribed Row Sums and Column Sums via Optimization," *Linear Algebra and Its Applications*, Vol. 114–115, Mar.–Apr. 1989, pp. 737–764. doi:10.1016/0024-3795(89)90491-6

[55] Linial, N., Samorodnitsky, A., and Wigderson, A., "A Deterministic Strongly Polynomial Algorithm for Matrix Scaling and Approximate Permanents," *Combinatorica : an international Journal of the Janos Bolyai Mathematical Society*, Vol. 20, No. 4, 2000, pp. 545–568. doi:10.1007/s004930070007

[56] Balakrishnan, H., Hwang, I., and Tomlin, C., "Polynomial Approximation Algorithms for Belief Matrix Maintenance in Identity Management," *Proceedings of the 43rd IEEE Conference on Decision and Control*, Vol. 5, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, Dec. 2004, pp. 4874–4879.

[57] Cover, T., and Thomas, J., *Elements of Information Theory*, Wiley-Interscience, New York, 1991.

[58] Kapur, J., and Kesavan, H., *Entropy Optimization Principles with Application*, Academic Press, Inc., New York, 1992.

[59] Lerro, D., and Bar-Shalom, Y., "Interacting Multiple Model Tracking with Target Amplitude Feature," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No. 2, 1993, pp. 494–509. doi:10.1109/7.210086