Shared-Memory Parallelization of the Data Association Problem in Multitarget Tracking

Robert L. Popp, Krishna R. Pattipati, *Fellow*, *IEEE*, Yaakov Bar-Shalom, *Fellow*, *IEEE*, and Reda A. Ammar, *Senior Member*, *IEEE*

Abstract—The focus of this paper is to present the results of our investigation and evaluation of various shared-memory parallelizations of the data association problem in multitarget tracking. The multitarget tracking algorithm developed was for a sparse air traffic surveillance problem, and is based on an Interacting Multiple Model (IMM) state estimator embedded into the (2D) assignment framework. The IMM estimator imposes a computational burden in terms of both space and time complexity, since more than one filter model is used to calculate state estimates, covariances, and likelihood functions. In fact, contrary to conventional wisdom, for sparse multitarget tracking problems, we show that the assignment (or data association) problem is *not* the major computational bottleneck. Instead, the *interface* to the assignment problem, namely, computing the rather numerous gating tests and IMM state estimates, covariance calculations, and likelihood function evaluations (used as cost coefficients in the assignment problem), is the major source of the workload. Using a measurement database based on two FAA air traffic control radars, we show that a "coarse-grained" (dynamic) parallelization *across* the numerous tracks found in a multitarget tracking problem is robust, scalable, and demonstrates superior computational performance to previously proposed "fine-grained" (static) parallelizations *within* the IMM.

Index Terms—Air traffic surveillance, multitarget tracking, Interacting Multiple Model (IMM) estimator, shared-memory MIMD multiprocessor, data association, assignment problem.

1 Introduction

1.1 Motivation

With the availability and widespread usage of general-purpose shared-memory MIMD multiprocessor systems, there is increased interest in the parallelization of new types of problems that were not addressable in the past. In particular, surface-based, airborne, and space-based surveillance are several such examples. In general, the objective of a surveillance and tracking algorithm is to detect an unknown number of targets in the presence of spurious observations and occasional missed detections, and to estimate their states (e.g., position, velocity, acceleration) using sensor measurements of (possibly) unknown origin, and contaminated by noise and clutter. To be computationally feasible, multitarget tracking algorithms often require large computational resources because of the limited performance of such algorithms on conventional uniprocessor systems [18], [19]. Motivated by the numerous advantages offered by shared-memory multiprocessing systems, e.g., cost, availability, and widespread usage, the focus of the present work was to adapt a serial multitarget tracking algorithm developed for a sparse air traffic surveillance problem [26] to such systems.

Typical of many realistic air traffic surveillance problems are:

1) the existence of multiple targets in the surveillance region, and
2) uncertain target dynamics, namely, target motion having constant course and speed (nonmaneuvering) segments interspersed with arbitrary maneuvers (often characterized as piecewise constant).

Consequently, in developing a tracking solution to such problems, one must resolve two inherently related subproblems, namely, filtering, i.e., *state estimation*, of sensor data over time, and fusion, i.e., *data association*, of sensor data across sensor scans, wherein the main goals are overcoming the inherent limitations of real-world sensors (accuracy and reliability) due to noise and clutter, while not sacrificing computational tractability.

In a typical air traffic surveillance problem, multiple targets with dynamic target motions must be tracked. Consequently, a state estimator that is *flexible* and *adaptable* is essential. The classical Kalman filter (KF)—the workhorse of state estimation—has been found to provide only marginal tracking performance for such problems because of the single motion model assumption [6], [15], i.e., target motion cannot be modeled well via a single set of state equations given the uncertain target dynamics inherent to such problems. However, unlike the KF, the Interacting Multiple Model (IMM) estimator [6], [7] has proven to be very effective for such problems. To achieve superior tracking performance, the IMM estimator, consisting of a finite number

• *R.L. Popp is with BBN Systems and Technologies, 70 Fawcett St., Cambridge, MA 02138. E-mail: rpopp@bbn.com*
• *K.R. Pattipati and Y. Bar-Shalom are with the Department of Electrical and Systems Engineering, University of Connecticut, 260 Glenbrook Rd., Storrs, CT 06269-3157. E-mail: krishna@sol.uconn.edu.; ybs@ee.uconn.edu*
• *R.A. Ammar is with the Department of Computer Science and Engineering, University of Connecticut, 191 Auditorium Rd., Storrs, CT 06269-3155. E-mail: reda@cse.uconn.edu.*

Manuscript received 18 Aug. 1995.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 104155.

of filters, each modeling different target motions, contains an explicit provision for the target's filter to probabilistically "switch" from one motion model to another [10]. However, the IMM estimator also imposes an increased computational burden in terms of additional state estimates, covariance calculations, and likelihood function evaluations.

Rarely does a realistic air traffic surveillance problem involve tracking the state of a single known target. Consequently, inherent to such problems is the data association problem. Data association is the decision process of linking measurements (from successive scans) of a common origin (i.e., a target or false alarm) such that each measurement is associated with at most one origin. The nature of the multitarget data in the surveillance region often motivates the choice of algorithm to use in solving the data association problem. For example, in this work, the FAA multitarget data can be characterized as sparse and nonstressful, i.e., uncluttered, not very noisy, absent of many crossing, splitting, merging tracks. For these types of multitarget tracking problems, a single scan processing approach for data association has been found to work rather well [9]. Consequently, in this work, we solve the data association problem by embedding a state estimator, such as a Kalman filter or the IMM estimator, into the assignment framework, i.e., two-dimensional (2D) assignment problem. In such a formulation, the state estimator provides a score (i.e., *likelihood*) of how likely a particular measurement from a given scan originated from a particular target track. The problem is then one of finding an assignment, in a (global) *maximum likelihood* sense, of measurements from the latest scan to the most likely tracks from the previous scans. Any well-known 2D assignment algorithm can be used in finding the optimal solution, e.g., auction, shortest augmenting path, interior point methods.

Historically, the most computationally intensive aspect of multitarget tracking has been the task of solving the data association problem; that is, partitioning the scan measurements into tracks and false alarms so that accurate estimates of true tracks can be recovered. This is especially true for dense, highly cluttered, and/or highly contentious multitarget scenarios. However, as we will show in this work, contrary to conventional wisdom, for sparse multitarget tracking problems, the assignment (or data association) problem is *not* the major computational bottleneck. Instead, the *interface* to the assignment problem, namely, computing the rather numerous gating tests and IMM state estimates, covariance calculations, and likelihood function evaluations (used as cost coefficients in the assignment problem), is the major source of the workload.

1.2 Related Research

A literature survey of parallel state estimators developed for air traffic surveillance problems reveals a plethora of "fine-grained" parallelizations proposed over the years [3], [4], [12], [25]. With respect to the KF, typically, many numerically intensive computations inherent to state estimation were parallelized, such as coordinate transformations, state and covariance estimates, linear algebraic operations, such as matrix multiplication, Cholesky (square root) factorization, and inner products. With respect to the IMM state estimator, Atherton and Lin [3] and Averbuch et al. [4] developed fine-grained parallelizations on a (four-processor) distributed-memory transputer and shared-memory MIMD multiprocessor architecture, respectively. Essentially, the multiple filter models of the IMM estimator are run in parallel, while the other components of the estimator (i.e., interaction, update, and combination steps) are run sequentially. Using an IMM estimator configured with nine and 13 filter models, Atherton and Lin obtained speedups (efficiencies)[1] of 2.34 (59 percent) and 2.73 (68 percent), respectively. Averbuch et al. were able to obtain a speedup (efficiency) of three (75 percent) for an IMM estimator configured with 12 filter models. However, as we will show in this paper, a fine-grained parallelization *within* the IMM estimator proves to be computationally inadequate in the context of a multitarget tracking problem. In particular, the performance of a fine-grained parallelization is *dependent* on the number of models used in the IMM, with marginal speedups obtained only when the number of filter models used is unrealistically high. In fact, when the IMM estimator is configured with a number of models considered suitable[2] for an air traffic surveillance problem (e.g., two-three), the execution time of the fine-grained parallelization is greater than sequential time.

Current work documented in the literature [1], [2], [9], [19], [24] tends to view the data association problem in multitarget tracking as :

1) a nearest-neighbors problem, wherein an overall distance function between measurement and track pairings is minimized, or,
2) a statistical estimation problem, wherein a set of measurements is associated with a set of tracks and false alarms by an unknown random permutation.

Nearest neighbor data association is problematic because it causes the state estimator to become overconfident. Consequently, with some probability, incorrect measurements will be fed into the filter, thereby losing tracks [7]. Data association based on statistical estimation techniques, as exemplified by Reid's classical Multiple Hypotheses Tracking (MHT) method [24] and MHT variants, such as the track branching/splitting algorithm of [2], and the model-based algorithms of [19], use a deferred decision approach to data association. Final decisions pertaining to difficult data association situations can be postponed until more information, such as the next scan(s) of data, are received. For stressful multitarget scenarios, postponing these decisions works rather well, whereas the lack of information in the 2D assignment approach (single scan processing) may lead to improperly partitioned

1. The standard definitions of speedup and efficiency are used, i.e., $speedup \overset{\Delta}{=} \frac{\tau_1}{\tau_p}$, where τ_1 (τ_p) denotes the sequential (parallel) execution time utilizing one (p) processor(s), and $efficiency \overset{\Delta}{=} \frac{speedup}{p}$, respectively.

2. Unsatisfactory tracking performance may result in an IMM estimator with a large number of filter models, since many of the models will differ significantly from the system mode in effect at a particular time, yielding excessive "competition" from the "unnecessary" filter models.

measurements into tracks and false alarms. However, for nonstressful multitarget scenarios, as is the case in the present work, single scan processing suffices. In theory, MHT can provide the optimal tracking solution; however, the computational feasibility of MHT is questionable since an enumerative search of numerous hypothesized permutations is required to determine their likelihoods, with the complexity increasing exponentially with the number of feasible tracks. As examples of MHT in parallel computing environments, in [1], Allen describes efficient mappings of an MHT algorithm onto massively parallel computers, while, in [19], Pattipati et al. describe efficient mappings of a model-based MHT algorithm onto general-purpose shared-memory MIMD multiprocessors. In [2], Atherton et al. describe an efficient parallel track branching/splitting algorithm on a distributed-memory message-passing transputer architecture.

1.3 Scope and Organization of Paper

To date, there has been a lack of efficient and practical parallelizations of the data association problem in multitarget tracking problems on general-purpose shared-memory MIMD multiprocessor systems. Filling this gap is the primary focus of the present work.

We begin this paper by describing, in Section 2,

1) our multitarget tracking problem based on two FAA air traffic control radars located in upstate New York, and
2) the multitarget tracking algorithm developed in a previous research effort [26] to solve this problem.

In Section 3, we provide a workload analysis of the sequential implementation of our multitarget tracking algorithm, while in Section 4, we describe “fine-“ and “coarse-grained” parallelizations developed for a general-purpose shared-memory MIMD multiprocessor. The fine-grained parallelization, similar to the ones described in [3], [4], is static and *within* the state estimator. The coarse-grained parallelization, as we propose in this research, is dynamic and *across* the numerous tracks found in our multitarget tracking problem.

As we will show in this paper, the coarse-grained parallelization is a practical choice for multitarget tracking problems because it is robust, scalable, and has excellent computational performance. In Sections 5 and 6, we provide an analytical proof and various performance results, respectively, that show the superior performance of the coarse-grained parallelization over the fine-grained parallelization. In Section 7, we provide some concluding remarks concerning this research.

2 Multitarget Tracking Algorithm

In this section, we provide a brief exposition of our multitarget tracking algorithm, termed IMM-2D, illustrated in Fig. 1. The algorithm consists of four primary components:

1) *Obtain Scan Measurements*,
2) The *Data Association Interface Problem*, which, in our tracking approach, is primarily concerned with coarse and fine gating (defined shortly), and IMM state estimation,

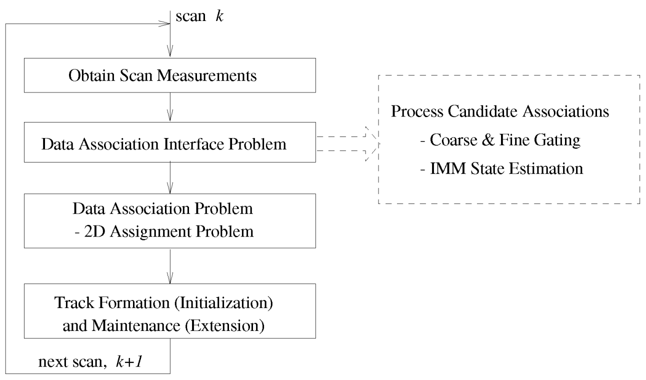


Fig. 1. Block diagram of our multitarget tracking algorithm.

3) The *Data Association Problem* (via a 2D assignment problem), and
4) Track Formation and Maintenance. Interested readers can find a more thorough presentation of IMM-2D in [20], [21].

2.1 Obtain Scan Measurements

Since it is not the main focus of this paper, we only present a simplified description of the air traffic surveillance problem here and refer interested readers to [20], [21], [26] for more details. The surveillance system is at a location referred to as the fusion center. The sensors in this surveillance system, two L-band FAA air traffic control radars located at Remsen and Dansville, New York, respectively, asynchronously transmit scans of data, approximately every 10 seconds, to the fusion center. The full database consists of 210 scans (98 from Remsen and 112 from Dansville), while a modified version of the database (for which we report in this paper) consists of 55 scans (26 from Remsen and 29 from Dansville). A particular scan, say scan k, received at the fusion center consists of $M(k)$ detection reports containing a number of primary radar (skin) returns (i.e., slant range, azimuth angle) and secondary (beacon) returns (i.e., slant range, azimuth angle, altitude). Because state estimation is done in a three-dimensional (3D) Cartesian coordinate system, conversion from a radar’s polar frame of reference is necessary [7], [26].

2.2 Data Association Interface Problem

In this section, we describe the two tasks that must be performed when interfacing with the data association (2D assignment) problem, specifically, *state estimation*, via an IMM estimator, and various coarse and fine *gating* tests. Of these two tasks, the former is the *single most costly* in terms of *processing cost*, and comprises a significant fraction of the workload in the IMM-2D tracking algorithm.

2.2.1 IMM State Estimation

For the air traffic surveillance problem used in this work, the IMM state estimator was employed for state estimation. Since it is not the main focus of the present work, we omit a detailed exposition of the underlying theory defining the IMM and simply provide a brief discussion of the state estimation problem in general. Interested readers can refer to

Appendix A and [6], [7] for a more detailed presentation of the material.

State estimation provides :

1) a score of how likely a particular measurement from a given scan originated from a particular target track, i.e., provides a *measurement-to-track* association likelihood that serves as the basis for an assignment cost coefficient in the 2D assignment (data association) problem, and
2) an estimate of the target state.

In the state estimation problem, we assume that the target state evolves according to a known linear dynamic model, corrupted by process noise, and driven by a known input, i.e.,

$$x\left(t_{m_k}\right) = F(\delta)\; x\left(t_{m_{k-1}}\right) + G(\delta)\, v\left(t_{m_{k-1}}\right), \tag{1}$$

where, in the present work, the target (or track) state $x(\cdot) = \left[x\, y\, z\, \dot{x}\, \dot{y}\, \dot{z}\right]'$ is a second order kinematic model with 3D coordinates, $\delta = t_{m_k} - t_{m_{k-1}}$ is the time interval, $F(\cdot)$ is the state transition matrix, $G(\cdot)$ is the disturbance matrix, and $v(\cdot)$ is zero-mean, white Gaussian *process* noise with (known) covariance matrix $Q(\cdot)$. Furthermore, the measurements are linear functions of the target state corrupted by measurement noise, i.e.,

$$z\left(t_{m_k}, m_k\right) = Hx\left(t_{m_k}\right) + w\left(m_k\right), \tag{2}$$

where $m_k = 1, \ldots, M(k)$ denotes the m_kth measurement from the kth scan, $H = [I\ 0\ 0]$ is the measurement matrix, and $w(\cdot)$ is zero-mean, white Gaussian *measurement* noise with (known) covariance matrix $R(\cdot)$.

2.2.2 Coarse and Fine Gating

To reduce the number of likelihood functions to evaluate, which, in terms of processing cost, is the single most costly operation to perform, *gating*—a pruning technique to filter out highly unlikely candidate associations—is used. A track *gate* is the region in measurement space in which the true measurement of interest will lie, in view of all uncertainties, with some (high) probability [7]. A measurement within the gate is a candidate for association to the corresponding track.

Define the set of *candidate associations* at the kth scan by

$$C(k) \triangleq \left\{\left(n, m_k\right) : \left(n, m_k\right) \in N\left(k-1\right) \times M\left(k\right)\right\}, \tag{3}$$

where $N(k-1) \times M(k)$ denotes the cross product of the track and measurement sets. In IMM-2D, for each $(n, m_k) \in C(k)$, a *coarse* gating test is first invoked, denoted by

$$\mathcal{G}_c : C(k) \to \{0, 1\}, \tag{4}$$

which consists of a *maximum velocity* gate test (i.e., measurement falls within the track's maximum velocity gate), and, if applicable, a high process noise Kalman filter *elliptical* gate test. $\mathcal{G}_c(n, m_k) = 1$ denotes that measurement m_k fell within both of track n's maximum velocity and elliptical gates, while $\mathcal{G}_c(n, m_k) = 0$ denotes that measurement m_k fell outside either of track n's gates. For maximum velocity gating, we use the following test,

$$\frac{\left\| z\left(t_{m_k}, m_k\right) - H\hat{x}\left(t_{m_{k-1}}\right) \right\|}{\left(t_{m_k} - t_{m_{k-1}}\right)} \overset{?}{>} MAX\_SPEED, \tag{5}$$

where the relation "$\overset{?}{>}$" compares the l.h.s. term with the r.h.s. term, and returns true if the former is greater than the latter, and false, otherwise. The residual terms in the numerator above are defined as follows: $z\left(t_{m_k}, m_k\right)$ is the m_kth measurement in scan k having time stamp t_{m_k}, H is the measurement matrix, and $\hat{x}\left(t_{m_{k-1}}\right)$ is the track state estimate at time $t_{m_{k-1}}$. The MAX_SPEED parameter is an assumed upper bound on the target's maximum velocity. For elliptical gating, we use a log-likelihood ratio test, i.e.,

$$-\log\left(\frac{P_D \Lambda_{kf}\left(n, m_k\right)}{\Lambda_{kf}\left(0, m_k\right)}\right) \overset{?}{>} 0, \tag{6}$$

where $\Lambda_{kf}(\cdot)$ denotes the likelihood function provided by a Kalman filter, the detection probability $P_D = 0.95$ is based on FAA standards [26], and the false alarm pdf $\Lambda_{kf}(0, m_k)$ is the spatial density of the false measurements (assumed uniformly probable over the sensor's field of view) [7].

Define the set of candidate associations passing the coarse gating test $\mathcal{G}_c(\cdot)$ by

$$\mathcal{L}(k) \triangleq \left\{\left(n, m_k\right) : \left(n, m_k\right) \in C(k), \mathcal{G}_c\left(n, m_k\right) = 1\right\}, \tag{7}$$

where each $(n, m_k) \in \mathcal{L}(k)$ requires the application of a *fine* gating test (based on the IMM estimator). Denote the fine gating test by

$$\mathcal{G}_f : \mathcal{L}(k) \to \{0, 1\}, \tag{8}$$

where $\mathcal{G}_f(n, m_k) = 0$ denotes that candidate association (n, m_k) is not to participate in the data association (2D assignment) problem because it is more likely that measurement m_k corresponds to a false alarm than to track n. Conversely, $\mathcal{G}_f(n, m_k) = 1$ denotes that candidate association (n, m_k) is to participate in the data association problem, where the cost of assigning measurement m_k to track n is defined next.

2.3 Data Association Problem

Data association is the decision process of *linking* measurements (from successive scans) of a common origin (i.e., a target or false alarm) such that each measurement is associated with at most one origin. In IMM-2D, we formulate the data association problem as a 2D assignment problem. Specifically, $M(k)$ measurements from the latest scan k are to be assigned to the $N(k-1)$ most likely existing tracks from the previous scans using a global cost minimization function [7] (based on a maximum likelihood function). Specifically, let $n = 0, \ldots, N(k-1)$ denote a particular track from the set of existing tracks (including a *dummy* track $n = 0$), and $m_k = 0, \ldots, M(k)$ denote a particular measurement from the latest set (scan) of measurements (including a *dummy* measurement $m_k = 0$). Define the binary *assignment* variable

$$d_{nm_k} = \begin{cases} 1 & \text{if measurement } m_k \text{ is assigned to track } n \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Note that $d_{n0} = 1$ implies that track n is *unassociated* and has missed a detection at scan k. Furthermore, $d_{0m_k} = 1$ implies that measurement m_k is unassociated, that is, not assigned to any of the $N(k - 1)$ previously established tracks, but, instead, assigned to the dummy track (false alarm). Since measurement errors within a scan are independent of each other, maximizing the likelihood function, consisting of the joint pdf-probability [7] of measurements, given their origins and the corresponding detection events, over the set of feasible assignments, can be cast into the following 2D assignment problem

$$\begin{aligned} &\min \sum_{n=0}^{N(k-1)} \sum_{m_k=0}^{M(k)} c_{nm_k} d_{nm_k} \\ &\text{s.t. } \sum_{m_k=0}^{M(k)} d_{nm_k} = 1 \quad n = 1, \ldots, N(k-1) \\ &\qquad \sum_{n=0}^{N(k-1)} d_{nm_k} = 1 \quad m_k = 1, \ldots, M(k), \end{aligned} \quad (10)$$

where the cost of assigning measurement m_k to track n is

$$c_{nm_k} = \begin{cases} 0 & \text{if } n = 0 \text{ or } m_k = 0 \\ -\log\left(\frac{\Lambda(n, m_k)}{\Lambda(0, m_k)}\right) & \text{if } -\log(\cdot) < 0 \\ \infty & \text{otherwise} \end{cases} \quad (11)$$

The numerator in the above $-\log(\cdot)$ expression, based on a likelihood function calculation $\Lambda(\cdot)$ from the IMM estimator (via (28) in Appendix A), is the likelihood that the m_kth measurement at scan k originated from the nth track, and the denominator is the likelihood that the m_kth measurement corresponds to none of the existing tracks (i.e., a false alarm). The likelihood of false alarms, i.e., $\Lambda(0, m_k)$, is assumed uniformly probable over the sensor's field of view [7].

2.4 Track Formation and Maintenance

Unassociated measurements in the 2D assignment problem are used to initialize new tracks. Initializing (forming) tracks based on a single measurement is somewhat different from the common two point differencing technique [6]. However, in air traffic surveillance, it is beneficial, since the difference in times between scans is often relatively short. Associated measurements in the 2D assignment problem are used to extend existing tracks, i.e., the corresponding track state vector is updated via (33) in Appendix A with the new measurement at scan k based on the solution to the 2D assignment problem. Tracks that go unassociated (i.e., not assigned a measurement) in the 2D assignment problem for a specified threshold of consecutive scans are terminated (dropped). In this work, the drop track threshold spans approximately 100 seconds, which roughly corresponds to 20 consecutive scans.

3 Workload Analysis

Historically, the most computationally-intensive aspect of multitarget tracking has been the task of solving the data association problem, that is, partitioning the scan measurements into tracks and false alarms so that accurate estimates of true tracks can be recovered. However, as shown in Table 1, the vast majority of the workload in IMM-2D when run on the multiprocessor architectures used in this research (described in the next section) involves processing the set of candidate associations in *interfacing* with the 2D assignment (data association) problem. In fact, this constitutes 94.3 percent, 94.7 percent, 95.2 percent, and 96 percent of the workload when IMM-2D is configured with one, two, three, and five filter models in the IMM, respectively. Recall, as described in Section 2.2, interfacing with the 2D assignment problem consists of performing numerous gating tests and assignment cost coefficient computations (i.e., the log-likelihood function (11) based on the IMM state estimator). If one were to follow conventional wisdom, namely, parallelize the data association (assignment) problem, 1.1 percent, 0.8 percent, 0.6 percent, and 0.2 percent of the workload would have been targeted for parallelization, respectively. Clearly, for a sparse air traffic surveillance problem such as the one we have in the present work, a parallel assignment algorithm would yield, at best, minimal benefits. However, although not negating the significance of the interface problem, for dense, highly cluttered, and/or highly contentious multitarget scenarios, where data association may subsume a significant fraction of the total computation, a parallel assignment algorithm, such as those described in [5], [8], [13], would provide increased benefit. In this work, since it is the significant computational bottleneck, we developed parallelizations of the *interface* to the 2D assignment (data association) problem in IMM-2D, of which we now describe.

4 IMM-2D Shared-Memory Parallelization

The shared-memory computing environment used for this work consisted of several general-purpose MIMD multiprocessors, namely, a two- and four-processor SPARCstation 20, and a 12-processor SPARCcenter 2000. A simple model of the four-processor SPARCstation 20 architecture is illustrated in Fig. 2 with various hardware specifications provided. The software utilized consisted of the Solaris 2.4.2 development environment, which included the SunOS 5.4.2 UNIX operating system (OS) and the multithreaded system architecture,

TABLE 1
Workload Distribution within IMM-2D

IMM-2D COMPONENT	# OF FILTER MODELS IN IMM			
	1 (KF)	2	3	5
Obtain Scan Measurements	0.8%	0.6%	0.5%	0.3%
Data Association Interface (Gating, IMM)	94.3%	94.7%	95.2%	96%
Data Association Problem (2D Assignment)	1.1%	0.8%	0.6%	0.2%
Track Maintenance and Formation	3.8%	3.9%	3.7%	3.5%

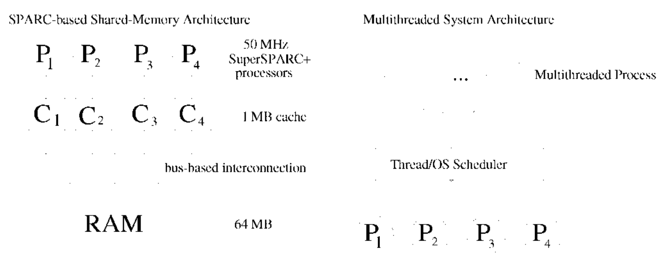


Fig. 2. Model of shared-memory architecture.

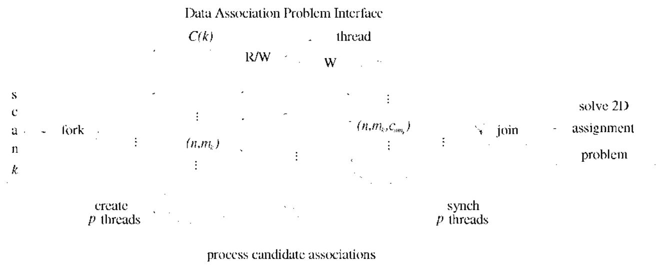

Data Association Problem Interface
C(k)
thread
R/W
W
scan k
fork
(n,m_i)
(n,m_i,c_{nm_i})
join
solve 2D assignment problem
create p threads
synch p threads
process candidate associations

Fig. 3. Task graph of IMM-2D shared-memory parallelization.

which we used as our parallel processing interface.

Multithreading separates a UNIX process into lightweight independent *threads*, each of which (concurrently) executes a sequence of the process's instructions. As depicted in Fig. 2, threads are dispatched across the processor set indirectly via a two-level scheduling hierarchy. Threads implicitly communicate via shared memory; consequently, synchronization mechanisms (e.g., mutual exclusion) must be supported to allow threads to cooperate in accessing shared data. Because of the numerous performance advantages offered by the threads model, we chose multithreading, as opposed to multiprocessing, as our avenue to exploit the multiple processor resources available in our multiprocessor system. Unlike a process, a thread requires very little interaction with the OS. Consequently, threads can be processed quickly (i.e., created, maintained, destroyed, blocked, activated, etc.), thousands can be present at one time, and synchronization and context switching between them can be accomplished rapidly. More details concerning multithreading can be found in [16], [22].

4.1 Coarse-Grained Parallelization

The coarse-grained shared-memory parallelization of the interface to the 2D assignment problem in IMM-2D is based on the supervisor/worker model (see Fig. 3). The supervisor thread initially *forks* a specified number of worker threads, say p, to process the set of candidate associations, $C(k)$, defined by (3) in Section 2.2.2. Once forked, the supervisor awaits processing of $C(k)$ to be completed by the p worker threads via a *join* operation. Worker threads, asynchronously and in parallel, process a specified number of candidate associations per serialized critical section access across mutually exclusive track and measurement data. Recall, as described in Section 3, the processing of a candidate measurement-to-track association consists of performing two coarse gating tests (the second test is performed only if the first succeeds), and, if the measurement falls within both of the track's gates, a fine gating test is applied, i.e., the log-likelihood function (11), based on the IMM state estimator. Since the processing cost corresponding to each candidate association is not uniform (depends on the results of gating), dynamic scheduling of candidate associations across threads is employed. In this way, because candidate associations are dynamically scheduled, maximum load balancing is achieved [11]. Upon processing of $C(k)$ by the worker threads, the supervisor solves the global 2D assignment problem.

4.2 Fine-Grained Parallelization

As pointed out in [3], [4], the IMM approach to target tracking is highly suited to parallel processing, since the different filter models may be assigned to different processors. This obvious fine-grained parallelization is apparent from the general structure of the IMM (see Fig. 10 in Appendix A). In the fine-grained parallelization, similar to those described in [3], [4], each candidate association in $C(k)$ passing the two coarse gating tests will have an assignment cost coefficient (i.e., log-likelihood function (11)) computed based on a parallelized IMM state estimator. However, the set $C(k)$ is sequentially iterated over and the two coarse gating tests are sequentially processed. Note that, in this approach, the parallelization is within the IMM for each track processed, i.e., the IMM estimator itself is parallelized, as opposed to across the multiple tracks found in a multitarget tracking problem, as is the case for the coarse-grained parallelization, i.e., the processing of candidate associations is parallelized.

5 Analytical Analysis

In this section, we present analytical models for the coarse- and fine-grained parallelizations using the micro time cost analysis technique as described in [23]. We then compare the derived time cost expressions corresponding to the parallelizations, and show that the former is superior, in terms of computational performance, to the latter.

Given a particular scan k, assume that we have $N = N(k-1)$ tracks, $M = M(k)$ measurements, and $NM = |C(k)| = N(k-1)M(k)$ candidate associations to process at scan k. Also, assume that we have p threads (or processors), an IMM estimator with r models, and, without loss of generality, assume that $r = zp$, $z \in \mathcal{N}^+$ (z a positive integer). In the fine-grained parallelization, p threads in parallel process r filter models of the IMM estimator, each thread processing over z models. In the coarse-grained parallelization, p threads in parallel process the set of $C(k)$ candidate associations, sequentially iterating over the r filter models when computing assignment cost coefficients. To simplify the analysis, assume all similar computations take an equivalent amount of time.[3]

The general computational structure model (CSM) [23]

3. We make this assumption primarily to simplify the analysis by avoiding minimization and/or maximization functions. Moreover, because we employ a homogeneous data partitioning strategy where each thread executes exactly the same operations, but on different data segments, and the operations executed are not data dependent, such an assumption is not unreasonable.

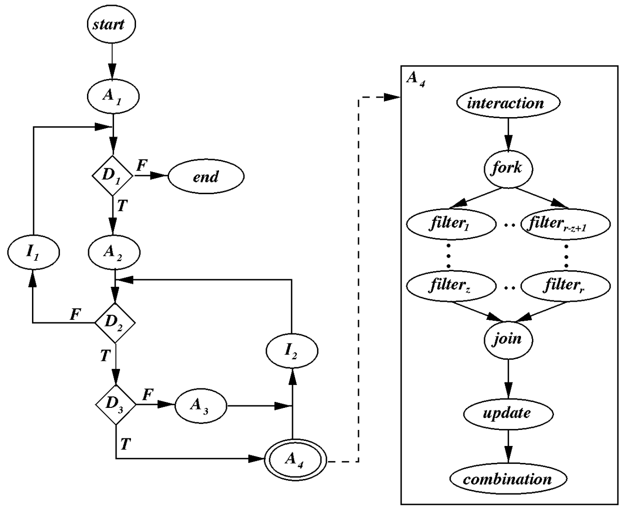


Fig. 4. CSM for the fine-grained parallelization.

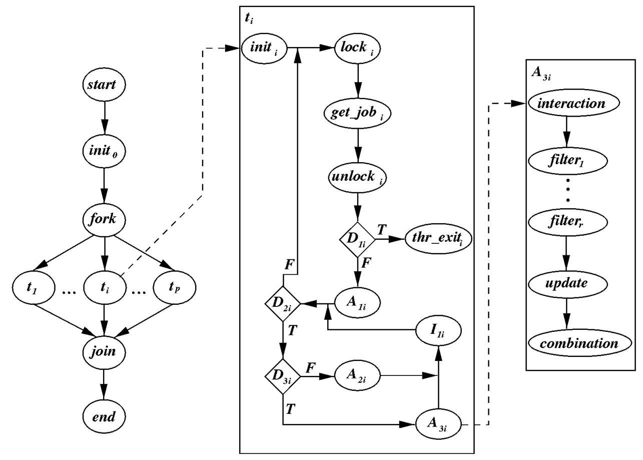


Fig. 5. CSM for the coarse-grained parallelization.

for the fine-grained parallelization is illustrated in Fig. 4. All the primitive operations and/or basic utility routines in this computation are specified as follows:

$fork : thr\_create(\;)\, p$ times
$join : thr\_join(\;)\, p$ times
$filter_m$: for $i = 1, \ldots, p$, thread i computes z filters $filter_m$, $m = (i-1)z + 1, \ldots, iz$
$A_1 : n \leftarrow 1$
$A_2 : m_k \leftarrow 1$
$A_3 : C[n, m_k] \leftarrow 0$
$A_4 : C[n, m_k] \leftarrow c_{nm_k}$ (11) in parallel
$I_1 : n \leftarrow n + 1$
$I_2 : m_k \leftarrow m_k + 1$
$D_1 : n \le N$
$D_2 : m_k \le M$
$D_3 : \mathcal{G}_c(n, m_k) = 1 \;\&\&\; \mathcal{G}_f(n, m_k) = 1$

where A_i, $i = 1, \ldots, 4$ denotes an assignment node, D_j, $j = 1, \ldots, 3$ a decision node, and I_k, $k = 1, 2$ an increment node. To derive the number of times each node is executed, we utilize the flow analysis techniques as described in [23]. The time cost for the fine-grained parallel computation is

$$\begin{aligned} T_{fine} &= T_{A_1} + T_{D_1=F} + N\left(T_{D_1=T} + T_{A_2} + T_{D_2=F} + T_{I_1}\right) + \\ &\quad NM\left[\left(T_{D_2=T} + T_{I_2}\right) + (1-\alpha)\left(T_{D_3=F} + T_{A_3}\right) + \alpha\left(T_{D_3=T} + T_{A_4}\right)\right] \\ &= T_{A_1} + T_{D_1=F} + N\left(T_{D_1=T} + T_{A_2} + T_{D_2=F} + T_{I_1}\right) + \\ &\quad NM\left[\left(T_{D_2=T} + T_{I_2}\right) + (1-\alpha)\left(T_{D_3=F} + T_{A_3}\right) + \right. \\ &\quad \left. \alpha\left(T_{D_3=T} + T_I + T_{fork} + zT_{filter_m} + T_{join} + T_U + T_c\right)\right] \end{aligned} \tag{12}$$

where T_I, T_U, and T_C denote the interaction, update, and combination time costs, respectively, within the IMM estimator (see Appendix A for details), $T_{D_l} = T$ $\left(T_{D_l} = F\right)$, for $l = 1, 2, 3$, denotes the time cost for the *true* (*false*) branch to execute, α denotes the number of times a log-likelihood function (11) is evaluated, and $1 - \alpha$, the number of times a log-likelihood function is not evaluated due to the gating tests.

The general CSM for the coarse-grained parallelization is illustrated in Fig. 5. All the primitive operations and/or basic utility routines in this computation are specified as follows:

$init_0$: $track\_id \leftarrow 1; measurement\_id \leftarrow 1$
$fork$: $thr\_create()\, p$ times
$join$: $thr\_join()\, p$ times
t_i: ith thread calls procedure process_ assoc()
$init_i$: $init\_TSD()$– get thread specific data segment
$(un)lock_i$: read/write (un)lock on *mutex*
get_job_i: $n \leftarrow track\_id, m_k \leftarrow measurement\_id$,
then update indices $track\_id$ & $measurement\_id$
$D_{1_i} : n > N$
$D_{2_i} : j < \left((M = 1) \,\&\&\, \left(m_k + tasksize\right)\right)$
$D_{3_i} : \mathcal{G}_c(n, m_k) = 1 \;\&\&\; \mathcal{G}_f(n, m_k) = 1$
$A_{1_i} : j \leftarrow m_k$
$A_{2_i} : C[n, j] \leftarrow 0$
$\mathrm{A}_{3_i} : C[n, j] \leftarrow c_{nj}$ (11) in serial
$\mathrm{I}_{1_i} : j \leftarrow j + 1$

The time cost for the coarse-grained parallel computation is

$$\begin{aligned}
T_{coarse} &= T_{init_0} + T_{fork} + T_{join} + T_{t_i} \\
&= T_{init_0} + T_{fork} + T_{join} + T_{init_i} + \\
&\quad \left(1 + \tfrac{NM}{P}\right)\left(T_{lock_i} + T_{get\_job_i} + T_{unlock_i}\right) + T_{D_{1_i}=T} + \\
&\quad \tfrac{NM}{P}\Big[T_{I_{1_i}} + T_{D_{1_i}=F} + T_{A_{1_i}} + T_{D_{2_i}=F} + T_{D_{2_i}=T} + \\
&\quad (1-\alpha)\left(T_{D_{3_i}=F} + T_{A_{2_i}}\right) + \alpha\left(T_{D_{3_i}=T} + T_{A_{3_i}}\right)\Big] \\
&= T_{init_0} + T_{fork} + T_{join} + T_{init_i} + \\
&\quad \left(1 + \tfrac{NM}{P}\right)\left(T_{lock_i} + T_{get\_job_i} + T_{unlock_i}\right) + \\
&\quad T_{D_{1_i}=T} + \tfrac{NM}{P}\Big[T_{I_{1_i}} + T_{D_{1_i}=F} + T_{A_{1_i}} + T_{D_{2_i}=F} + \\
&\quad T_{D_{2_i}=T} + (1-\alpha)\left(T_{D_{3_i}=F} + T_{A_{2_i}}\right) + \\
&\quad \alpha\left(T_{D_{3_i}=T} + T_{I_i} + rT_{filter_i} + T_{U_i} + T_{C_i}\right)\Big]. \qquad (13)
\end{aligned}$$

Note that, for each of the $i = 1, \ldots, p$ threads in the coarse-grained computation, the time cost to access the critical section $T_{get\_job_i}$ is nondeterministic because of a *wait* time cost in trying to acquire the mutex lock *mutex*. Therefore, define

$$T_{get\_job_i} \triangleq T_{W_i} + T_{A_i}, \qquad (14)$$

where T_{W_i} denotes the wait time cost for the ith thread, and T_{A_i} denotes the time cost for the ith thread to execute the critical section (i.e., obtain a specified number of candidate associations and update the track and measurement indices *track_id* & *measurement_id*, respectively). Thus, T_{W_i} is a random variable for which we need to determine an expected time cost. Given p threads trying to acquire the mutex lock *mutex*, and, considering the worst case mutex lock access policy (i.e., assume a round-robin policy), since the thread may not necessarily acquire the mutex lock within p tries, we use a geometric distribution to model its probability distribution. Letting A denote the event "a thread acquires *mutex* on a given try," then

$$P\{A\} = \frac{1}{p}, \qquad (15)$$

and the expected time cost for T_{W_i} (in the worst case) is

$$\begin{aligned}
\mathbf{E}\left\{T_{W_i}\right\} &= \sum_{j=0}^{\infty} T_{A_i} j\left(1 - \frac{1}{p}\right)^j \frac{1}{p} = \frac{T_{A_i}}{p}\sum_{j=0}^{\infty} j\left(1 - \frac{1}{p}\right)^j \\
&= \frac{T_{A_i}}{p}\frac{\left(1 - \frac{1}{p}\right)}{\left(1 - \left(1 - \frac{1}{p}\right)\right)^2} = T_{A_i}(p-1) \ . \qquad (16)
\end{aligned}$$

Substituting (16) into (14), we obtain

$$T_{get\_job_i} = pT_{A_i}, \qquad (17)$$

Assume that the *base time costs* for all assignment, decision, increment, lock, and unlock nodes have unit cost. Further, assume that the worst case time cost in executing the critical section is $T_{A_i} = 5$ (i.e, two assignments, two increments, and one decision node). Hence, after some algebraic simplifications, we get

$$\begin{aligned}
T_{fine} &= 2 + 4N + \frac{4-\alpha}{NM} + NM\alpha \\
&\quad \left(T_I + T_{fork} + zT_{filter_m} + T_{join} + T_U + T_C\right) \qquad (18)
\end{aligned}$$

$$\begin{aligned}
T_{coarse} &= 6 + 5p + NM\left(\frac{14-\alpha}{NM}\right) + T_{fork_i} + T_{join_i} + \\
&\quad \frac{NM\alpha}{p}\left(T_{I_i} + rT_{filter_i} + T_{U_i} + T_{C_i}\right) \qquad (19)
\end{aligned}$$

From (18) and (19), define constants

$$R = 2 + 4N + \frac{4-\alpha}{NM}$$

and

$$S = 6 + 5p + NM\left(\frac{14-\alpha}{p}\right)$$

where R and S are insignificant constants that are dominated by the remaining terms in their respective equations. To determine how they (relatively) compare, we form the following relationship between the two time cost equations:

$$T_{fine} - T_{coarse} \overset{?}{>} 0 \qquad (20)$$

$$\begin{aligned}
&\left[R + NM\alpha\left(T_I + T_{fork} + zT_{filter_m} + T_{join} + T_U + T_C\right)\right] - \\
&\left[S + T_{fork_i} + T_{join_i} + \tfrac{NM\alpha}{p}\left(T_{I_i} + rT_{filter_i} + T_{U_i} + T_{C_i}\right)\right] \overset{?}{>} 0 \\
&\qquad (21)
\end{aligned}$$

Again, after some algebraic simplifications, we get

$$\begin{aligned}
&\frac{p(R-S)}{NM\alpha} + p\left(T_{fork} + T_{join}\right)\left(1 - \frac{1}{NM\alpha}\right) + \\
&\left(T_I + T_U + T_C\right)(p-1) + T_{filter}(pz - r) \overset{?}{>} 0 \qquad (22)
\end{aligned}$$

$$\begin{aligned}
&\frac{p(R-S)}{NM\alpha} + p\left(T_{fork} + T_{join}\right)\left(1 - \frac{1}{NM\alpha}\right) + \\
&\left(T_I + T_U + T_C\right)(p-1) > 0 \qquad (23)
\end{aligned}$$

Clearly, irrespective of the number of threads (processors) p, filter models r, and the number of candidate associations NM for a given scan, the coarse-grained parallel computation has a smaller time cost than the fine-grained parallel computation. In the next section, we provide empirical evidence that further supports this assertion.

6 Results

In this section, based on the two- and four-processor SPARCstation 20, and the 12-processor SPARCcenter 2000, we demonstrate the computational superiority of the coarse-grained parallelization of IMM-2D, making comparisons with both the fine-grained parallelization and a serial implementation of IMM-2D. In the fine-grained parallelization, the multiple filter models of the IMM estimator are evenly distributed across the processor set in a manner similar to the approach described in [3], [4]. Since the multiprocessors used in this work are time-shared systems, the execution times of IMM-

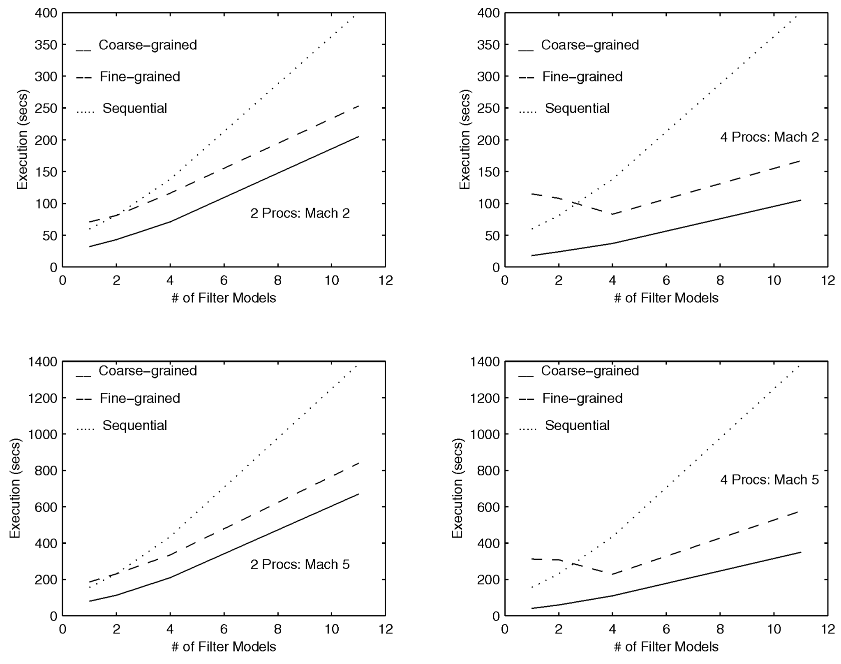


Fig. 6. Execution time of IMM-2D on a two- and four-processor SPARCstation 20 using various coarse gating policies. Horizontal axis denotes # of filter models in IMM configuration.

2D depended, in part, on random system events such as the system load and thread schedule order. Consequently, Monte Carlo simulations were performed, and all results presented represent the means of those simulations with standard errors less than three percent.

6.1 Execution Time

In Fig. 6, we plot the execution times of the sequential, the coarse- and fine-grained parallelizations of IMM-2D on the two- and four-processor SPARCstation 20. Note that *Mach 2* and *Mach 5* simply denote the speed of an aircraft to be two and five times the speed of sound, respectively. The significance of these two parameters is as follows. As described in Section 2.2.2, one of the coarse gating policies employed in IMM-2D is based on an aircraft's maximum velocity. Increasing the magnitude of the *Mach* parameter will increase the spatial area of the track's maximum velocity gate, thus increasing the processing cost associated with the track since a greater number of measurements will fall within the gate.

Clearly, the coarse-grained parallelization demonstrates superior execution time performance over the fine-grained parallelization for any number of models used in the state estimator. Moreover, the performance of the fine-grained parallelization is dependent on the number of models used, yielding marginal performance improvements only when using an unrealistically large number of filter models. Furthermore, when configured with a small to moderate number of models (three or less) that would be considered practical for an air traffic surveillance problem, the fine-grained parallelization yields *execution time greater than sequential time*.

The computational bottleneck in the fine-grained parallelization is primarily twofold:

1) Since the computations associated with the gating tests in processing candidate associations occur prior to invoking the IMM to obtain the assignment cost coefficients, the fine-grained parallelization (within the state estimator) is unable to process these independent computations in parallel, and
2) There is a large overhead cost corresponding to a substantial number of fork-join operations in computing the assignment cost coefficients. Specifically, the fine-grained parallelization suffers from a poor *computation-to-communication* ratio.

In the coarse-grained parallelization, the bottleneck is the numerous serialized accesses to the shared task queue when aquiring candidate associations to process.

6.2 Efficiency

In Fig. 7, we plot the parallel efficiency for both the coarse- and fine-grained parallelizations of IMM-2D. Clearly, as Fig. 7 illustrates, the computational performance of the coarse-grained parallelization is *independent* of the number of filter models used in the IMM estimator, whereas the fine-grained parallelization performs rather inefficiently unless the number of filter models used is unrealistically high. Furthermore, as shown in Fig. 7, near-unity efficiency, and, given a large enough problem (e.g., two processors using Mach 5 coarse gating policy), even greater than unity efficiency (i.e., *superlinear speedup*) is possible. When nonalgorithmic issues, such as context switches, effective memory size and access costs, and scheduling order, are considered, superlinear speedups in practice may indeed occur [14], [17].

In Fig. 7, note the efficiency improvement for the fine-grained parallelization as the number of filter models in the IMM estimator increases (these results are consistent with those in [3], [4]). However, in a real-time sense, these so-called efficiency improvements are actually misleading. In

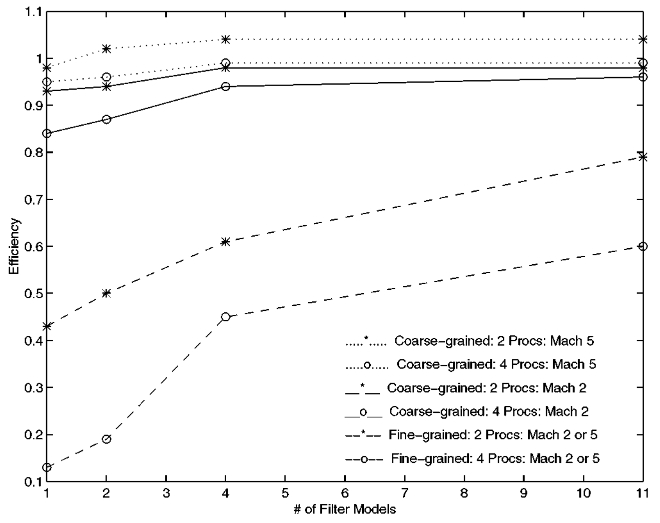


Fig. 7. Parallel efficiency of IMM-2D on a two- and four-processor SPARCstation 20 using various coarse gating policies. Horizontal axis denotes number of filter models in IMM configuration.

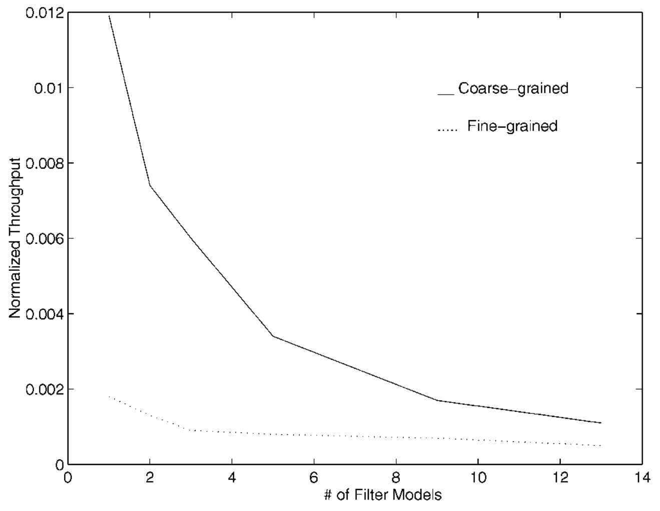


Fig. 8. Normalized throughput $\left(\tau_p^{-1}\right)$ of IMM-2D on a four-processor SPARCstation 20. Horizontal axis denotes number of filter models in IMM configuration.

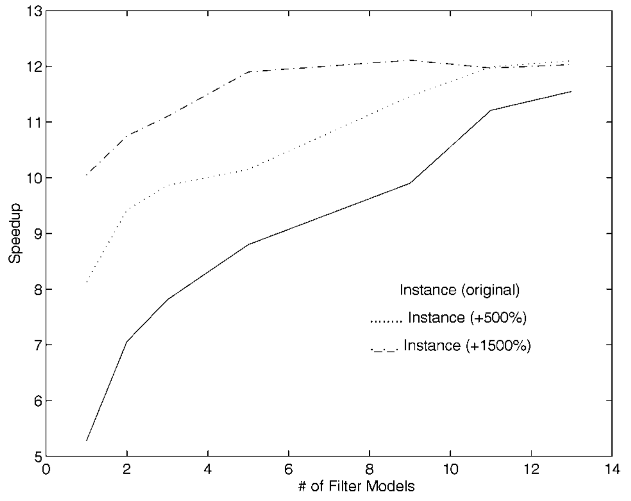


Fig. 9. Scale-up of IMM-2D on a 12-processor SPARCcenter 2000 for several problem instances. Horizontal axis denotes number of filter models in IMM configuration.

Fig. 8, we plot the throughput $\left(\tau_p^{-1}\right)$ for the coarse-grained and fine-grained parallelizations of IMM-2D. This plot may be viewed as a measure of the real-time effectiveness (*speed*) of IMM-2D in terms of the number of candidate associations processed per unit time. Clearly, as this plot demonstrates, increasing the number of filter models in the IMM estimator, and, hence, increasing the processing workload, decreases the throughput of IMM-2D. However, what it also shows is that, based on the *flatness* of its curve, a fine-grained parallelization is no better in terms of throughput when the IMM estimator is configured with many filter models, than when configured with fewer models (i.e., it does not get faster, even when it has less processing to do). The coarse-grained parallelization, on the other hand, shows significant improvement, in an absolute and relative sense, as the number of filter models in the IMM estimator decreases. This is a rather important result given that in practical air traffic surveillance problems, typically a small number of filter models are necessary in the IMM estimator [15].

6.3 Scale-Up

Many factors determine the scale-up performance of a parallel algorithm, in particular, the multiprocessor architecture and the problem size are directly related. In the context of multitarget tracking, the problem size is a function of numerous factors, including the number of actual and false alarm tracks, noise/clutter intensity, target density, contentiousness of the assignment problem, and the number of filter models used in the IMM. When considering *scalability*, a highly desirable goal would be for IMM-2D to have the ability to maintain the same *high* level of performance (e.g., efficiency) on larger multiprocessor systems as it did on smaller ones, and do so in terms of *any* of the factors that can influence the problem size. Certainly, IMM-2D having this property would make it *robust*, and it could easily adapt, without modification, to other diverse multitarget tracking problems.

Unlike the fine-grained parallelization, the coarse-grained parallelization of IMM-2D scales when any of the factors influencing problem size increase. In particular, in Fig. 9, we plot the speedup for the coarse-grained parallelization on a 12-processor SPARCcenter 2000 for several problem instances. In *(Instance (original))*, all factors influencing the problem size are unchanged, whereas, in *(Instance (+500 percent))* and *(Instance (+1,500 percent))*, a more dense multitarget scenario was simulated by increasing the problem size (relative to *(Instance (original))*) in terms of track set size by 500 percent and 1,500 percent, respectively. From the plot, we can see marginal speedup results when the problem size is small (i.e., *(Instance (original))*) relative to the particular multiprocessor architecture. However, as the problem size increases, the scalability of the coarse-grained parallelization is evident, approaching near linear speedup with fewer filter models, and linear with many.

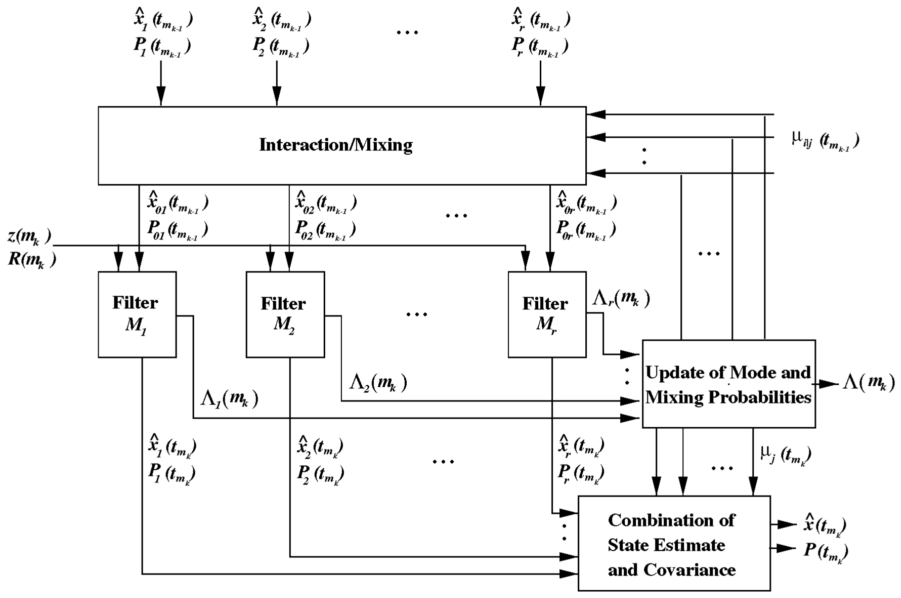

$\hat{x}_1(t_{m_{k-1}})$ $P_1(t_{m_{k-1}})$
$\hat{x}_2(t_{m_{k-1}})$ $P_2(t_{m_{k-1}})$
$\hat{x}_r(t_{m_{k-1}})$ $P_r(t_{m_{k-1}})$
Interaction/Mixing
$\mu_{i|j}(t_{m_{k-1}})$
$\hat{x}_{01}(t_{m_{k-1}})$ $P_{01}(t_{m_{k-1}})$
$\hat{x}_{02}(t_{m_{k-1}})$ $P_{02}(t_{m_{k-1}})$
$\hat{x}_{0r}(t_{m_{k-1}})$ $P_{0r}(t_{m_{k-1}})$
$z(m_k)$
$R(m_k)$
Filter M_1
Filter M_2
Filter M_r
$\Lambda_r(m_k)$
$\Lambda_2(m_k)$
$\Lambda_1(m_k)$
Update of Mode and Mixing Probabilities
$\Lambda(m_k)$
$\hat{x}_1(t_{m_k})$ $P_1(t_{m_k})$
$\hat{x}_2(t_{m_k})$ $P_2(t_{m_k})$
$\hat{x}_r(t_{m_k})$ $P_r(t_{m_k})$
$\mu_j(t_{m_k})$
Combination of State Estimate and Covariance
$\hat{x}(t_{m_k})$
$P(t_{m_k})$

Fig. 10. Augmented IMM estimator.

7 Conclusions

We have presented our investigation and evaluation of various shared-memory parallelizations of the data association problem in multitarget tracking for general-purpose shared-memory MIMD multiprocessor systems. A sparse measurement database, based on two FAA air traffic control radars, served as the data for our multitarget tracking problem. We showed that, for sparse multitarget tracking problems, the assignment (or data association) problem is not the major computational bottleneck. Instead, the interface to the assignment problem, namely, computing the rather numerous gating tests and IMM state estimates, covariance calculations, and likelihood function evaluations (used as cost coefficients in the assignment problem), is the major source of the workload. Because of the shortcomings associated with previously proposed fine-grained parallelizations of the IMM estimator in a multitarget tracking problem, we proposed a scalable, robust coarse-grained parallelization across tracks that has excellent performance. Analytically and empirically, we showed that the coarse-grained parallelization has superior execution time performance over the fine-grained parallelization for any number of filter models used in the IMM estimator. Furthermore, we showed that the coarse-grained parallelization can realize superlinear speedups independent of the number of filter models used in the IMM estimator. On the other hand, the performance of the fine-grained parallelization, being dependent on the number of filter models used, yielded negligible throughput for any number of filter models, marginal speedups when many models were used, and worse execution time than sequential time when three or less filter models were used.

Appendix A
IMM State Estimation

The IMM estimator used in IMM-2D [26], illustrated in Fig. 10, is an augmented version of the standard IMM estimator [6]. As Fig. 10 depicts, the various computations involved in one cycle of the IMM estimator can be divided into:

1) interaction/mixing,
2) filtering,
3) update of mode and mixing probabilities, and
4) combination of mode-matched state estimates and covariances.

The IMM assumes the motion of a target for which a corresponding track will develop follows one of r possible filter models (or modes) between two successive detections. The augmented IMM estimator combines the likelihoods $\Lambda_j(\cdot)$, $j = 1, \ldots, r$ from the individual mode-matched filters to yield an overall likelihood $\Lambda(\cdot)$. We omit detailed exposition of the IMM algorithm here, since it is not the main focus of this paper, and simply present a brief summary of the relevant equations. We refer interested readers to [26] for a more descriptive presentation of the material.

The **mixed state estimate** and **covariance** are given by

$$\hat{x}_{0j}\left(t_{m_{k-1}}\right) = \sum_{i=1}^{r} \hat{x}_i\left(t_{m_{k-1}}\right)\mu_{i|j}\left(t_{m_{k-1}}\right) \tag{24}$$

$$P_{0j}\left(t_{m_{k-1}}\right) = \sum\nolimits_{i=1}^{r}\left[\hat{x}_i\left(t_{m_{k-1}}\right) - \hat{x}_{0j}\left(t_{m_{k-1}}\right)\right]\left[\hat{x}_i\left(t_{m_{k-1}}\right) - \hat{x}_{0j}\left(t_{m_{k-1}}\right)\right]' \mu_{i|j}\left(t_{m_{k-1}}\right) + \sum\nolimits_{i=1}^{r} P_i\left(t_{m_{k-1}}\right)\mu_{i|j}\left(t_{m_{k-1}}\right) \tag{25}$$

The **mixing probability** is given by

$$\mu_{i|j}\left(t_{m_{k-1}}\right) = \frac{\mu_i\left(t_{m_{k-1}}\right)}{\sum_{i=1}^{r} \mu_i\left(t_{m_{k-1}}\right)} \tag{26}$$

where $\pi_{ij} \triangleq \mathrm{P}\left\{u\left(t_{m_k}\right) = j \middle| u\left(t_{m_{k-1}}\right) = i\right\}$ is the **model transition probability**.

The **mode probability** is given by

$$\mu_j\left(t_{m_k}\right) = \frac{\Lambda_j(m_k)\sum_{i=1}^{r} \mu_i\left(t_{m_{k-1}}\right)\pi_{ij}}{\Lambda(m_k)} \tag{27}$$

The **combined likelihood of the IMM estimator** is given by

$$\Lambda(m_k) = \sum_{j=1}^{r}\sum_{i=1}^{r} \Lambda_j(m_k)\mu_i\left(t_{m_{k-1}}\right)\pi_{ij} \tag{28}$$

and each filter model (mode) **likelihood function** is given by

$$\Lambda_j(m_k) = \left|2\pi S_j(m_k)\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\alpha_j(m_k)\right\} \tag{29}$$

where the **normalized innovation squared** is

$$\alpha_j(m_k) = \nu_j(m_k)'\left[S_j(m_k)\right]^{-1}\nu_j(m_k) \tag{30}$$

and the **measurement residual** and **residual covariance** are

$$\nu_j(m_k) = z(m_k) - H\,F_j(\delta_k)\hat{x}_{0j}\left(t_{m_{k-1}}\right) \tag{31}$$

$$S_j(m_k) = R(m_k) + H\left[F_j(\delta_k)P_{0j}\left(t_{m_{k-1}}\right)F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)'\right]H' \tag{32}$$

where $\delta_k = t_{m_k} - t_{m_{k-1}}$.

The **updated state estimate** and **covariance** are given by

$$\hat{x}_j\left(t_{m_k}\right) = F_j(\delta_k)\hat{x}_{0j}\left(t_{m_{k-1}}\right) + W_j(m_k)\nu_j(m_k) \tag{33}$$

$$P_j\left(t_{m_k}\right) = \left[F_j(\delta_k)P_{0j}\left(t_{m_{k-1}}\right)F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)'\right] - W_j(m_k)S_j(m_k)W_j(m_k)' \tag{34}$$

where the **filter gain** is

$$W_j(m_k) = \left[F_j(\delta_k)P_{0j}\left(t_{m_{k-1}}\right)F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)'\right]H'\left[S_j(m_k)\right]^{-1} \tag{35}$$

And, last, the **combined state estimate** and **covariance** are given by

$$\hat{x}\left(t_{m_k}\right) = \sum_{j=1}^{r} \hat{x}_j\left(t_{m_k}\right)\mu_j\left(t_{m_k}\right) \tag{36}$$

$$P\left(t_{m_k}\right) = \sum_{j=1}^{r}\left[\hat{x}_j\left(t_{m_k}\right) - \hat{x}\left(t_{m_k}\right)\right]\left[\hat{x}_j\left(t_{m_k}\right) - \hat{x}\left(t_{m_k}\right)\right]'\mu_j\left(t_{m_k}\right) \sum_{j=1}^{r} P_j\left(t_{m_k}\right)\,\mu_j\left(t_{m_k}\right). \tag{37}$$

Acknowledgments

We would like to thank the anonymous referees for their technical and editorial advice, which substantially improved the quality of the paper. We would also like to thank the Booth Research Center at the University of Connecticut, and Rome Laboratory, in particular, Mr. Rick Gassner, for providing the data sets that enabled much of this work to take place. Research was conducted at the University of Connecticut and sponsored by grants AFOSR F409620-93-1-0399, ONR/BMDO N00014-91-J-1950, and NASA NGT-303104.

References

[1] T. Allen, "Multiple Hypothesis Tracking Algorithms for Massively Parallel Computers," *Proc. SPIE Conf. Signal and Data Processing of Small Targets*, Orlando, Fla., 1992.

[2] D. Atherton, E. Gul, A. Kountzeris, and M. Kharbouch, "Tracking Multiple Targets Using Parallel Processing," *IEE Proc. Control, Theory, and Applications*, vol. 137, no. 6, pp. 225-234, July 1990.

[3] D. Atherton and H. Lin, "Parallel Implementation of IMM Tracking Algorithm Using Transputers," *IEE Proc. Radar, Sonar Navigation*, vol. 141, no. 6, pp. 325-332, Dec. 1994.

[4] A. Averbuch, S. Itzikowitz, and T. Kapon, "Parallel Implementation of Multiple Model Tracking Algorithms," *IEEE Trans. Parallel and Distributed Systems*, vol. 2, no. 2, pp. 242-252, Apr. 1991.

[5] E. Balas, D. Miller, J. Pekny, and P. Toth, "A Parallel Shortest Augmenting Path Algorithm for the Assignment Problem," *J. ACM*, vol. 38, no. 4, pp. 985-1,004, Oct. 1991.

[6] Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques and Software*. Boston: Artech House, 1993.

[7] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques.* Storrs, Conn.: YBS Publishing, 1995.

[8] D. Bertsekas and D. Castanon, "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," *Parallel Computing*, vol. 17, pp. 707-732, 1991.

[9] S. Blackman, "Association and Fusion of Multiple Sensor Data," *Multitarget-Multisensor Tracking: Advanced Applications*, vol. I, Y. Bar-Shalom, ed., Artech House, 1990.

[10] H. Blom and Y. Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Trans. Automatic Control*, vol. 33, no. 8, pp. 780-783, Aug. 1988.

[11] S. Dandamudi and P. Cheng, "A Hierarchical Task Queue Organization for Shared-Memory Multiprocessor Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 1, pp. 1-16, Jan. 1995.

[12] U. Desai and B. Das, "Parallel Algorithms for Kalman Filtering," *Proc. American Control Conf.*, Boston, June 1985.

[13] A. Goldberg, S. Plotkin, D. Shmoys, and E. Tardos, "Using Interior-Point Methods for Fast Parallel Algorithms for Bipartite Matching and Related Problems," *SIAM J. Computing*, vol. 21, no. 1, pp. 140-150, Feb. 1992.

[14] R. Janben, "A Note on Superlinear Speedup," *Parallel Computing*, vol. 4, pp. 211-213, 1987.

[15] X. Li and Y. Bar-Shalom, "Design of an Interacting Multiple Model Algorithm for Air Traffic Control Tracking," *IEEE Trans. Control Systems Technology: Special Issue on Air Traffic Control*, vol. 1, no. 3, pp. 186-194, Sept. 1993.

[16] Sun Microsystems, Inc., "Multithreaded Programming Guide,". Mountain View, Calif.: Sun Microsystems, Inc., 1994.

[17] D. Parkinson, "Parallel Efficiency can be Greater than Unity," *Parallel Computing*, vol. 3, pp. 261-262, 1986.

[18] K. Pattipati, S. Deb, Y. Bar-Shalom, and R. Washburn, "A New Relaxation Algorithm and Passive Sensor Data Association," *IEEE Trans. Automatic Control*, vol. 37, no. 2, pp. 197-213, Feb. 1992.

[19] K. Pattipati, T. Kurien, R. Lee, and P. Luh, "On Mapping a Tracking Algorithm onto Parallel Processors," *IEEE Trans. Aerospace and Electronic Systems*, vol. 26, no. 5, pp. 774-791, Sept. 1990.

[20] R. Popp, K. Pattipati, Y. Bar-Shalom, and M. Yeddanapudi, "Parallelization of a Multiple Model Multitarget Tracking Algorithm with Superlinear Speedups," *IEEE Trans. Aerospace and Electronic Systems*, vol. 33, no. 1, pp. 281-289, Jan. 1997.

[21] R. Popp, K. Pattipati, and Y. Bar-Shalom, "Multitarget Tracking Algorithm Parallelizations for Air Traffic Surveillance," to appear, *Annals of Operations Research: Special Issue on Parallel Optimization*, 1997.

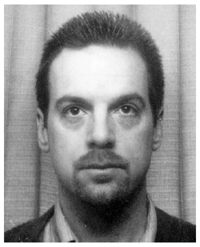
[22] M. Powell, S. Kleiman, S. Barton, D. Shah, D. Stein, and M. Weeks, "SunOS Multithread Architecture," *Proc. USENIX '91 Conf.*, Dallas, Tex., pp. 1-14, 1991.

[23] B. Qin, H. Sholl, and R. Ammar, "Micro Time Cost Analysis of Parallel Computations," *IEEE Trans. Computers*, vol. 40, no. 5, pp. 613-628, May 1991.

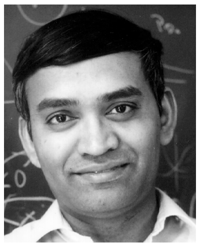
[24] D. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. Automatic Control*, vol. 24, no. 6, pp. 423-432, Dec. 1979.

[25] I.B. Rhodes, "A Parallel Decomposition for Kalman Filters," *IEEE Trans. Automatic Control*, vol. 35, no. 3, pp. 322-324, Mar. 1990.

[26] M. Yeddanapudi, Y. Bar-Shalom, and K. R. Pattipati, "IMM Estimation for Multitarget-Multisensor Air Traffic Surveillance," *Proc. 34th IEEE CDC*, New Orleans, Dec. 1995; also appears *IEEE Proc.*, Jan. 1997.//**name of proceeding?**//

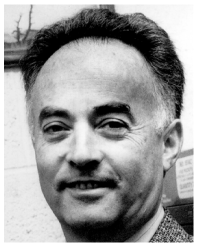


Robert L. Popp received the BA and MA degrees in computer science from Boston University, and the PhD degree in electrical and systems engineering from the University of Connecticut. In 1994 and 1995, he was a visiting research scientist at Rome Laboratory, an Air Force R&D laboratory. Since 1996, he has been with BBN Systems and Technologies in Cambridge, Massachusetts, where he is a scientist with the Advanced Systems Development Group. His primary research interests are in the area of surveillance information processing, parallel and distributed computing, and optimization theory. He is a member of Phi Beta Kappa, Upsilon Pi Epsilon, and he is listed in *Who's Who in the East*.



Krishna R. Pattipati received the BTech degree in electrical engineering with highest honors from the Indian Institute of Technology, Kharagpur, and the MS and PhD degrees in systems engineering from the University of Connecticut. From 1980 to 1986, Dr. Pattipati was employed with Alphatech, Inc., Burlington, Massachusetts, where he supervised and performed research on artificial intelligence and systems theory approaches to human decision modeling, multitarget tracking, and automated testing. Since 1986, he has been with the University of Connecticut, where he is a professor of electrical and systems engineering and director of the SOL (Systems Optimization Lab). He is also President of QUALTECH Systems, Inc., Storrs, Connecticut, a small business specializing in software tools and solutions for testability, maintainability, and quality control. He has served as a consultant to Alphatech, Inc., and the IBM T. J. Watson Research Center.

Dr. Pattipati was selected by the IEEE Systems, Man, and Cybernetics (SMC) Society as the Outstanding Young Engineer of 1984, and received the Centennial Key to the Future award. He won the best technical paper awards at the 1985, 1990, and 1994 IEEE Autotest Conferences. He was elected a fellow of the IEEE for his contributions to "Discrete-Optimization Algorithms for Large-Scale Systems Applications and Team Decision Making." He served as the vice chairman for invited sessions of the IEEE International SMC Conference, Boston, Massachusetts, 1989. He is currently serving as an associate editor of the *IEEE Transactions on Systems, Man, Cybernetics*, and is a member of the administrative committee of the IEEE SMC Society.



Yaakov Bar-Shalom received the BS and MS degrees from the Technion, Israel Institute of Technology, and the PhD degree from Princeton University, all in electrical engineering. From 1970 to 1976, Dr. Bar-Shalom was employed with Systems Control, Inc., Palo Alto, California. Currently, he is a professor of electrical and systems engineering and director of the ESP Lab (Estimation and Signal Processing) at the University of Connecticut. His research interests are in estimation theory and stochastic adaptive control, and he has published more than 200 papers in these areas. He coauthored the monograph *Tracking and Data Association* (Academic Press, 1988), the graduate text *Estimation and Tracking: Principles, Techniques and Software* (Artech House, 1993), the text *Multitarget-Multisensor Tracking: Principles and Techniques* (YBS Publishing, 1995), and edited the books *Multitarget-Multisensor Tracking: Applications and Advances* (Artech House, vol. I, 1990; vol. II, 1992). He has been elected a fellow of the IEEE for "contributions to the theory of stochastic systems and of multitarget tracking." He has been a consultant to numerous companies, and originated the series of *Multitarget-Multisensor Tracking* short courses offered via UCLA Extension, at government laboratories, private companies, and overseas.

During 1976 and 1977, Dr. Bar-Shalom served as an associate editor of the *IEEE Transactions on Automatic Control*, and, from 1978 to 1981, as an associate editor of *Automatica*. He was program chairman of the 1982 American Control Conference, general chairman of the 1985 ACC, and cochairman of the 1989 IEEE International Conference on Control and Applications. During 1983-87, he served as chairman of the Conference Activities Board of the IEEE Control Systems Society, and, during 1987-89, was a member of the Board of Governors of the IEEE Control Systems Society. In 1987, he received the IEEE CSS Distinguished Member Award. Since 1995, he has been a distinguished lecturer of the IEEE AESS. He is corecipient of the M. Barry Carlton Award for the best paper in the *IEEE Transactions on Aerospace and Electronic Systems* in 1995.

Reda A. Ammar received the BS degree in electrical engineering from Cairo University, the BS degree in mathematics from Ein Shames University, Egypt, and the MS and PhD degrees in computer science and engineering from the University of Connecticut. Currently, he is an associate professor with the Department of Computer Science and Engineering at the University of Connecticut. His primary research interests are in the area of software performance engineering, parallel processing, computer-aided performance engineering, modeling, and optimization of sequential and parallel computations. Dr. Ammar has been chairman of three International Conferences on Parallel and Distributed Computing, and is presently an associate editor of two professional journals. His publications record exceeds 115 papers in professional journals, conferences, and workshops. He is a senior member of the IEEE, a member of the IEEE Computer Society, chairman of the IEEE technical committee on Simulation, a member of ACM and ISCA, and a member of the SIMULATION society.