Multitarget Tracking in Clutter: Fast Algorithms for Data Association

B. ZHOU, Member, IEEE

N. K. BOSE, Fellow, IEEE
Pennsylvania State University

Three fast algorithms have been developed to solve the problem of data association in multitarget tracking in clutter. In the first algorithm, the problem of data association is identified as an *exhaustive search problem* in general. Subsequently, a mathematical model is proposed for the problem of data association in the joint probabilistic data association filter (JPDAF). Based on the model, a depth-first search (DFS) approach is developed for the fast generation of data association hypotheses and the computation of the conditional probabilities of the hypotheses in the JPDAF. When the density of targets is moderate, a second algorithm which is more efficient is developed to directly compute *a posteriori* probabilities in the JPDAF without generating the data association hypotheses. In the third algorithm, the effect of interference due to closely spaced targets is simplified. This leads us to develop an approach to approximately compute the *a posteriori* probabilities in the JPDAF. The computational complexity of the above algorithms is analyzed in the worst case as well as in the average case.

Manuscript received November 11, 1990. Released for publication April 15, 1992.

IEEE Log No. T-AES/29/2/04590.

This work was supported by SDIO/IST and managed by the Office of Naval Research under Contract N00014-86-K-0542.

Authors' address: Department of Electrical and Computer Engineering, The Spatial and Temporal Signal Processing Center, Pennsylvania State University, University Park, PA 16802.

I. INTRODUCTION

In multitarget tracking in clutter, often there are more than one measurement available for updating the state of a single target. To solve the problem of data association between targets and measurements, two typical approaches have been reported in the literature in the 1970s. One is called the *target-oriented* approach in which each measurement is assumed to have originated from either a known target or clutter, as in *probabilistic data association filter* (PDAF) [1, 2] and *joint probabilistic data association filter* (JPDAF) [1, 3]. The other is called a *measurement-oriented* approach in which each measurement is hypothesized to have originated from either a known target, a new target, or clutter [4]. Recently, a new approach is proposed in [5, ch. 3] to deal with the data association problem in multitarget tracking. The new approach is called a *track-oriented* approach in which each track is hypothesized to be either undetected, terminated, associated with a measurement, or linked to the start of a maneuver. In these approaches, the number of data association hypotheses could increase rapidly with the increase in the number of targets and the number of measurements. Therefore, in a multitarget tracking algorithm, the computational cost in generating the data association hypotheses would be excessive when the number of targets and the number of measurements are large. In the PDAF, the computational cost for data association is reduced drastically by isolating targets from each other. Only the measurements which lie in the validation gate of a target are considered in data association. Since the PDAF ignores the interference from other targets, it may, sometimes, cause mistracking of closely spaced targets, as discussed in [3]. This difficulty is greatly reduced by using the JPDAF [3], which accounts for the fact that a measurement which falls inside the intersection of the validation gates of several targets could have originated from any one of these targets or from clutter. In the JPDAF, targets are divided into clusters. The targets are in the same cluster if there is at least one measurement inside each of the intersections of their validation gates. The computational cost for data association is reduced by grouping targets into clusters. The number of different data association hypotheses in each cluster is still an exponential function of the number of targets in the cluster. Due to the complexity of the problem of data association, the implementation of a multitarget tracking algorithm has only been carried out in a 2-target case [3, 4], except in [6], where JPDA filtering and smoothing were applied to track 11 nonmaneuvering and maneuvering targets.

To reduce significantly the computational cost for data association, several approximations to the JPDAF and the algorithm in [4] have been reported in literature. A first version of the approximate or

"cheap" JPDAF was published in [7], where the probability of association of target t with measurement j was computed by an *ad hoc* formula. The "cheap JPDAF" in [7] performed fairly well in the case of two targets but failed to track four targets [8]. In [8], the JPDAF was emulated by a neural network which is capable of handling two to six targets and three to twenty measurements. However, a comprehensive analysis [9] reveals several drawbacks of the neural network approach developed in [8]. In [10], a fast JPDA algorithm was proposed for the computation of the β_j^ts in the JPDAF. For $j > 0$, $\beta_j^t(k)$ is the *a posteriori* probability that measurement j originated from target t. For $j = 0$, $\beta_0^t(k)$ is the *a posteriori* probability that no measurement originated from target t. In the fast JPDA algorithm, the computation of the β_j^ts was implemented using enormous number of vector inner product operations. An optical processor was suggested to approximately realize these operations. Three algorithms, which perform much better in the average case than the fast JPDA algorithm, are proposed here. Alternatively, Nagarajan, et al., [11] arranged the hypotheses in the measurement-oriented approach [4] in a special order so that the probabilities of the hypotheses are proportional to the product of certain probability factors already evaluated. The algorithm locates the N globally best hypotheses without evaluating all of them.

A description of the problem of interest is provided in Section II and an alternate formula for the conditional probabilities of the data association hypotheses is derived to simplify the notation in the development of three fast algorithms for computation of the β_j^ts. In Section IIIA a comparison between a general exhaustive search problem and the one of data association in multitarget tracking is discussed. This leads to a mathematical model for the problem of data association. In Section IIIB some unique features of the problem of data association are pointed out as a prelude to a specialized depth-first search (DFS) algorithm for generating all data association hypotheses. In Section IIIC the advantage of the DFS algorithm in the computation of the conditional probabilities of the data association hypotheses is further demonstrated. In Section IIID the computational complexity of the proposed DFS algorithm is analyzed. In Section IV an algorithm is proposed to compute β_j^t directly when the density of targets is not very high. An approximation of the direct computation algorithm is presented in Section V. Finally, in Section VI computer simulation is carried out for tracking multiple targets in different scenario.

II. PROBLEM FORMULATION

Let m and n be the numbers of measurements and targets, respectively, in a particular cluster. In the JPDAF, the computation of the β_j^ts begins with the construction of a so-called validation matrix Ω for the n targets and m measurements. The validation matrix Ω is a $m \times (n+1)$ rectangular matrix defined as [3]

$$\Omega = [\omega_{jt}] = \begin{array}{c} \overbrace{}^{t} \\ \begin{pmatrix} 1 & \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ 1 & \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_{m1} & \omega_{m2} & \cdots & \omega_{mn} \end{pmatrix} \end{array} \begin{matrix} 0\ 1\ 2\ \cdots\ n \\ \\ 1 \\ 2 \\ \vdots \\ m \end{matrix} \} j \tag{1}$$

where $\omega_{j0} = 1$ implies that measurement j could originate from clutter, $\omega_{jt} = 1$ if measurement j is inside the validation gate of target t and $\omega_{jt} = 0$ if measurement j is outside the validation gate of target t for $j = 1,2,\ldots,m$, and $t = 1,2,\ldots,n$. Based on the validation matrix Ω, data association hypotheses (or feasible events [3]) are generated subject to the following two restrictions: (1) each measurement can have only one origin (either a specific target or clutter), and (2) no more than one measurement originates from a target. This leads to a combinatorial problem where the number of data association hypotheses increases exponentially with the number of targets and the number of measurements.

Each feasible event $\mathcal{E}$ is represented by a hypothesis matrix $\hat{\Omega}$ in [3]. $\hat{\Omega}$ has the same size as the validation matrix Ω. A typical element in $\hat{\Omega}$ is denoted by $\hat{\omega}_{jt}$ where $\hat{\omega}_{jt} = 1$ only if measurement j is hypothesized to be associated with clutter ($t = 0$) or target t ($t \neq 0$). After each $\hat{\Omega}$ is obtained, the conditional probability of the corresponding data association hypothesis or feasible event is calculated by a formula given in [3]. A simplified version of this formula is given in (2), where time index k is omitted.

$$P(\mathcal{E}(\hat{\Omega}) \mid Z) = \frac{1}{c}(P_0)^{\min(n,m)-m_a} \prod_{j:\hat{\omega}_{jt}=1} P_{jt} \tag{2}$$

$$\text{for} \quad j = 1,2,\ldots,m \quad \text{and} \quad t = 1,2,\ldots,n$$

where Z is the set of all measurements received up to current time index k, c is a normalization constant, m_a is the number of targets detected in this feasible event $\mathcal{E}$, $\hat{\omega}_{jt} = 1$ indicates that measurement j is associated with target t in the event, and

$$P_{jt} = \begin{cases} N(\tilde{z}_j^t; 0, S^t) P_D, & \text{if} \quad \omega_{jt} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$P_{0t} = \lambda(1 - P_D) = P_0 \tag{4}$$

$$\text{for} \quad j = 1,2,\ldots,m, \quad \text{and} \quad t = 1,2,\ldots,n.$$

In (3) and (4), λ is the clutter density, P_D is the probability of detection, and $N(\tilde{z}_j^t; 0, S^t)$ is a normal distribution density function with zero mean and

covariance matrix S^t. It is understood that the normalization constant c in (2) is obtained by the summation, $\sum_{\mathcal{E}(\hat{\Omega})} P(\mathcal{E}(\hat{\Omega}) \mid Z)$. Therefore, c is omitted hereafter. The *a posteriori* probability β_j^t is computed from the conditional probabilities in (2) by

$$\beta_j^t = \sum_{\mathcal{E}(\hat{\Omega})} P(\mathcal{E}(\hat{\Omega}) \mid Z)\hat{\omega}_{jt} \tag{5}$$

$$\beta_0^t = 1 - \sum_{j=1}^{m} \beta_j^t$$

$$\text{for} \quad j = 1,2,\ldots,m, \quad \text{and} \quad t = 1,2,\ldots,n. \tag{6}$$

In order to have a better understanding of how each β_j^t is calculated, consider an example. Suppose, there are two targets. In one radar scan, three measurements are received and suppose one of them falls inside the intersection of the validation gates of the two targets. The mathematical representation of this situation may be described by a validation matrix Ω:

$$\Omega = \overbrace{\begin{matrix} 0 & 1 & 2 \end{matrix}}^{t} \begin{matrix} \\ \\ \\ \end{matrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \Bigg\} j. \tag{7}$$

With the two restrictions given above, 8 feasible events, $\mathcal{E}(\hat{\Omega}_l)$ $(l = 0,1,\ldots,7)$, may be constructed. The hypothesis matrices are

$$\hat{\Omega}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \hat{\Omega}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\hat{\Omega}_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \hat{\Omega}_3 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\hat{\Omega}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \hat{\Omega}_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\hat{\Omega}_6 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \hat{\Omega}_7 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The conditional probability $P(\mathcal{E}(\hat{\Omega}_l) \mid Z)$ can be easily calculated by using (2). Subsequently, all β_j^ts may be computed. For example,

$$\beta_1^1 = P(\mathcal{E}(\hat{\Omega}_1) \mid Z) + P(\mathcal{E}(\hat{\Omega}_2) \mid Z) + P(\mathcal{E}(\hat{\Omega}_3) \mid Z).$$

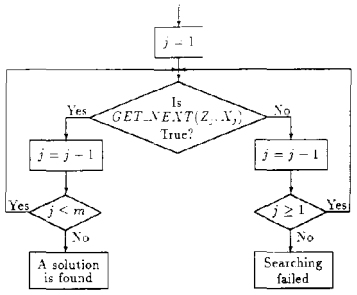

Fig. 1. Flowchart of DFS procedure.

The problems of interest here in the implementation of the JPDAF for any number of targets and measurements are the efficient generation of hypothesis matrices $\hat{\Omega}_l$ and fast computation of the β_j^ts.

III. DFS ALGORITHM

A. Mathematical Model for Data Association

As pointed out earlier, data association in multitarget tracking is a combinatorial problem. The computational cost of data association increases exponentially with n and m. The efficiency of the algorithm used in the generation of the data association hypotheses is especially important when n and m are large. In order to develop an efficient algorithm to generate all data association hypotheses, a mathematical model is developed for data association.

A well-known model for a combinatorial problem is called *exhaustive search with certain constraints* [12]. The general description of a typical exhaustive search problem is the following.

There are m variables X_j $(j = 1,2,\ldots,m)$. The value of each X_j belongs to a set Z_j, where Z_j is finite and linearly ordered. A candidate solution for this problem is an m-tuple

$$(X_1, X_2, \ldots, X_m).$$

The objective here is to find either one solution or all solutions under the imposed constraints.

An efficient algorithm to solve the above problem is a DFS procedure which involves backtracking [12]. In the DFS procedure, a solution is found by checking the m variables in an m-tuple one by one from left to right, as shown in Fig. 1. In Fig. 1, GET_NEXT(Z_j, X_j) is a logical function. It is true only if $X_j \in Z_j$, X_j has not been used in the DFS procedure yet, and $X_1, X_2, \ldots$, and X_j satisfy the given constraints. Otherwise, it is false.

In the context of tracking multitarget in clutter, however, data association can be modeled as an

exhaustive search problem with a set of proper notations. Let X_j ($j = 1,2,\ldots,m$) denote measurement j. The value of X_j identifies the target or clutter which is hypothesized to be associated with measurement j. For example, $X_j = 3$ implies that measurement j is hypothesized to be associated with target 3. However, if $X_j = 0$, measurement j is hypothesized to be associated with clutter. Therefore, the set Z_j may be defined by

$$Z_j = \{t \mid \omega_{jt} = 1\}, \qquad j = 1,2,\ldots,m, \quad \text{and} \quad t = 0,1,2,\ldots,n. \tag{8}$$

Note that since ω_{j0} is equal to 1, therefore, $0 \in Z_j$ for $j = 1,2,\ldots,m$. If measurement j falls inside the validation gate of target t, then, $t \in Z_j$. For the example discussed in the previous section, the Z_js ($j = 1,2,3$) are

$$Z_1 = \{0,1\}, \qquad Z_2 = \{0,1,2\}, \qquad Z_3 = \{0,2\}.$$

In the JPDAF scenario, the two constraints which have to be satisfied for a feasible event can be easily translated into the language of exhaustive search problem for data association. Hence, an m-tuple,

$$(X_1, X_2, \ldots, X_p, \ldots, X_q, \ldots, X_m),$$

is a solution if the following two constraints are satisfied.

1) If $p \neq q$, $X_p \neq 0$, and $X_q \neq 0$, then $X_p \neq X_q$.
2) If $X_p = X_q$ and $p \neq q$, then $X_p = X_q = 0$.

All data association hypotheses can be generated by solving the exhaustive search problem described above.

B. Generation of Data Association Hypotheses

A specialized DFS algorithm for the generation of data association hypotheses is proposed here. Before the DFS algorithm is given, it would be worthwhile to observe some differences between a general exhaustive search problem and the problem of data association.

Usually, there are no solutions that are known in advance in a general exhaustive search problem. However, in the problem of data association, a solution, which is always known in advance, is $(0,0,\ldots,0)$. The remaining solutions can be generated systematically from various valid combinations of non-zero values of the elements. In order to illustrate how this idea works, consider again the example in the previous sections. Given $Z_1 = \{0,1\}$, $Z_2 = \{0,1,2\}$, and $Z_3 = \{0,2\}$, the starting solution is $(0,0,0)$. The next solution we look at is $(1,0,0)$, obtained by setting $X_1 = 1$. Since 1 is already in the previous solution, the only non-zero value X_2 may have is 2. Therefore, after $(1,0,0)$, the solution $(1,2,0)$ is obtained. Now, all non-zero values in Z_2 which can been used to generate new solutions have been used up. Reset X_2 to 0 and go on to check if there is any non-zero value in Z_3 which will lead us to a new solution. In this case, by setting X_3 to 2, another solution, $(1,0,2)$, is found. After all non-zero values in both Z_2 and Z_3 are used up, X_1 is reset to 0 and is kept as 0 for the remainder of the search process. At this stage, $X_2 = 1$ will not cause any conflict. Hence, a new solution, $(0,1,0)$ is generated. This process can go on until all solutions are found. In this particular example, the complete set of solutions is

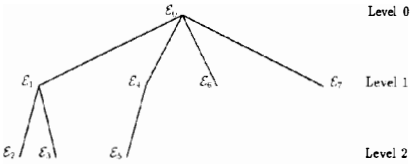

Fig. 2. Graph arrangement of $\mathcal{E}_l$.

$$\mathcal{E}_0 = (0,0,0), \quad \mathcal{E}_1 = (1,0,0), \quad \mathcal{E}_2 = (1,2,0),$$
$$\mathcal{E}_3 = (1,0,2), \quad \mathcal{E}_4 = (0,1,0), \quad \mathcal{E}_5 = (0,1,2),$$
$$\mathcal{E}_6 = (0,2,0), \quad \mathcal{E}_7 = (0,0,2).$$

For a better view of the solution generation process, all solutions can be arranged into a tree, which is called *hypotheses tree*, as shown in Fig. 2. Each node in the tree stands for a solution or a data association hypothesis. The root of the tree is $\mathcal{E}_0$, which implies that all 3 measurements are hypothesized to be associated with clutter. Each node at level i is associated with i non-zero variables in the corresponding 3-tuple. This also implies that i out of 3 measurements are hypothesized to be associated with i out of 2 targets. In Fig. 2, the height of the tree is 2. In general, the height of the tree is $\min(n,m)$ since i must be less than or equal to both n and m.

Only non-zero variables in a hypothesis need be stored in a stack of integer pairs when implementing of the DFS algorithm. For example, $\mathcal{E}_4$ $(= (0,1,0))$ is represented by $((2,1))$. Both convey the information that, in the current context, measurement 2 is associated with target 1 and the remaining two measurements are associated with clutter. The first integer in the pair indicates which variable is non-zero in $\mathcal{E}_4$ and the second integer is the value of that variable. In general, any m-tuple with L non-zero variables is represented by

$$\bar{X} = \begin{pmatrix} (j_1, X_{j_1}) \\ (j_2, X_{j_2}) \\ \vdots \\ (j_L, X_{j_L}) \end{pmatrix} \tag{9}$$

where $X_{j_1}, X_{j_2}, \ldots$, and X_{j_L} are non-zero and L is the pointer of the stack $\bar{X}$ which is pointing at the last pair of integers in the stack. The flowchart

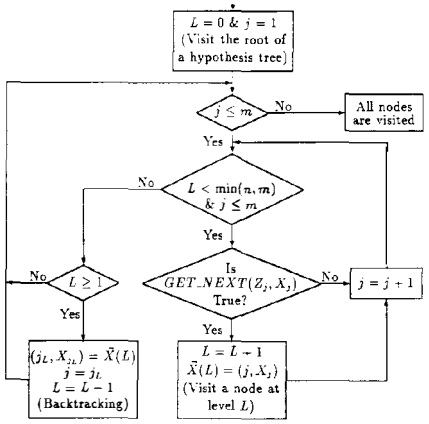

Fig. 3. Flowchart of DFS procedure for data association.

of the DFS algorithm for the generation of the data association hypotheses is shown in Fig. 3. The proposed DFS algorithm is designated to visit every node in a hypothesis tree. Similar as in Fig. 1, GET_NEXT(Z_j, X_j) in Fig. 3 is a logical function. It is true only if $X_j \in Z_j$ and X_j is not equal to $X_1, X_2, \ldots,$ and X_{j_L}. Otherwise, it is false.

C. Computation of *a posteriori* Probabilities

After each data association hypothesis is generated, the conditional probability of the hypothesis is computed by using the formula in (2). However, if the data association hypotheses are arranged in such an order that a factor of a conditional probability is included in another conditional probability, a resulting savings in the overall computation is feasible by the sharing the common factor. Suppose that there are 3 targets and 3 measurements and that the following sequence of solutions or data association hypotheses are generated.

$$\mathcal{E}_0 = (0,0,0), \qquad \mathcal{E}_1 = (1,0,0),$$
$$\mathcal{E}_2 = (1,2,0), \qquad \mathcal{E}_3 = (1,2,3).$$

Recalling that Z denotes the set of all measurements received up to the current time index, the conditional probabilities of the data association hypotheses $\mathcal{E}_l$ ($l = 0,1,2,3$) can be computed using (2).

$$P(\mathcal{E}_0 \mid Z) = (P_0)^3, \qquad P(\mathcal{E}_1 \mid Z) = (P_0)^2 P_{11},$$
$$P(\mathcal{E}_2 \mid Z) = P_0 P_{11} P_{22}, \qquad P(\mathcal{E}_3 \mid Z) = P_{11} P_{22} P_{33}.$$

Observe that P_{11} is the common factor used in the computation of $P(\mathcal{E}_l \mid Z)$ ($l = 1,2,3$) and $P_{11}P_{22}$ is used in the computation of $P(\mathcal{E}_l \mid Z)$ ($l = 2,3$). If P_0, $(P_0)^2$, $(P_0)^3$, and the P_{jt}s are computed in advance (preprocessing) for this example and the two common factors mentioned above are used in the computation of $P(\mathcal{E}_l \mid Z)$ ($l = 0,1,2,3$), then the number of multiplications required in the computation of the $P(\mathcal{E}_l \mid Z)$s is 4. Let $ND_i(n,m)$ and $M(n,m)$ denote, respectively, the number of nodes at level i and the total number of multiplications required in the computation of the $P(\mathcal{E}_l) \mid \mathcal{Z})$s. It can be shown [13] that

$$M(n,m) = ND_1(n,m) + 2 \sum_{i=2}^{\min(n,m)-1} ND_i(n,m) + ND_{\min(n,m)}(n,m). \tag{10}$$

After each $P(\mathcal{E}_l) \mid \mathcal{Z})$ is computed, the β_j^is and the normalization constant c can be updated. In the above example, these updates are indicated below.

$$\beta_1^1 + P(\mathcal{E}_2) \mid \mathcal{Z}) \rightarrow \beta_1^1 \tag{11}$$
$$\beta_2^2 + P(\mathcal{E}_2) \mid \mathcal{Z}) \rightarrow \beta_2^2 \tag{12}$$
$$c + P(\mathcal{E}_2) \mid \mathcal{Z}) \rightarrow c. \tag{13}$$

In general, $i + 1$ additions are required for each hypothesis at level i in a hypothesis tree. Let $A(n,m)$ denote the total number of additions. Then,

$$A(n,m) = \sum_{i=0}^{\min(n,m)} (i+1) ND_i(n,m). \tag{14}$$

The computational complexity of the DFS algorithm depends on the numbcr of multiplications, $M(n,m)$, and the number of additions, $A(n,m)$.

D. Computational Complexity

Since both $M(n,m)$ and $A(n,m)$ are functions of $ND_i(n,m)$, the analysis of the computational complexity in the following is focused on the calculation of $ND_i(n,m)$ in the worst case as well as in the average casc. According to the mathematical model for data association, the m variables, X_j ($j = 1,2,\ldots,m$), are used to denote the m measurements. Each variable X_j is defined on a set Z_j. In the worst case, Z_j assumes the following form:

$$Z_j = \{0,1,2,\ldots,n\} \qquad \text{for} \quad j = 1,2,\ldots,m. \tag{15}$$

In (15), each Z_j contains n non-zero values. In the average case, however, the number of non-zero values in each Z_j is much less than n. Conventionally, it is assumed that there are two to three measurements inside the validation gate of a target in the average case. In the following, the worst case analysis is considered first. Then, the average case analysis is discussed.

1) *Worst Case Analysis*: In the worst case, each Z_j contains n non-zero values, as given in (15). A direct interpretation of this situation is that the m measurements received fall inside the intersection of the validation gates of the n targets. Therefore, each measurement may be associated with either any one of

TABLE I
Sample Values of $M(n,m)/A(n,m)$ in DFS Algorithm

n	m	$M(n,m)/A(n,m)$	n	m	$M(n,m)/A(n,m)$
3	6	318/787	5	10	96890/339041
4	8	5072/14849	6	12	2218992/9081085

the n targets or clutter. The number of nodes at level i in a hypothesis tree in the worst case is then obtained as

$$ND_i(n,m) = C_m^i A_n^i, \qquad \text{for} \quad 0 \le i \le \min(n,m) \tag{16}$$

where

$$C_m^i = \frac{m!}{i!(m-i)!}, \qquad \text{for} \quad 0 \le i \le m \tag{17}$$

and

$$A_n^i = \frac{n!}{(n-i)!}, \qquad \text{for} \quad 0 \le i \le n. \tag{18}$$

Substitute $ND_i(n,m)$ into (10) and (14), to get

$$M(n,m) = C_m^1 A_n^1 + 2 \sum_{i=2}^{\min(n,m)-1} C_m^i A_n^i + C_m^{\min(n,m)} A_n^{\min(n,m)} \tag{19}$$

$$A(n,m) = \sum_{i=0}^{\min(n,m)} (i+1) C_m^i A_n^i. \tag{20}$$

With a few manipulations, the above representations may be simplified to

$$M(n,m) = 2S(n,m) - 2 - nm - C_m^{\min(n,m)} A_n^{\min(n,m)} \tag{21}$$

$$A(n,m) = S(n,m) + nmS(n-1,m-1) \tag{22}$$

where

$$S(n,m) = \sum_{i=0}^{\min(n,m)} C_m^i A_n^i. \tag{23}$$

The variables $S(n,m)$ denotes the number of data association hypotheses or the number of nodes in a hypothesis tree in the worst case. It can be shown that $S(n,m)$ satisfies the following difference equation [13],

$$S(n,m) = nS(n-1,m-1) + S(n,m-1) \tag{24}$$

with boundary conditions,

$$S(1,m) = m+1, \tag{25}$$

$$S(n,1) = n+1. \tag{26}$$

In Table I, some $M(n,m)$s and $A(n,m)$s are listed with different numbers of targets and measurements. From the listed values, we can see that the computational cost is increasing with the increase of n and m. In the average case, however, $M(n,m)$ and $A(n,m)$ are expected to be much less than the ones given in (21) and (22), respectively. In the next section, upper bounds for $M(n,m)$ and $A(n,m)$ in the average case are proposed.

TABLE II
Upper Bounds of $M(n,m)$ and $A(n,m)$ in the DFS Algorithm

n	$M(n,m)$	$A(n,m)$
3	90	208
4	417	1024
5	1788	4864
6	7443	22528
7	30558	102400

2) *Average Case Analysis*: Commonly, it is assumed that there are on the average two to three measurements inside the validation gate of each target and that the density of targets is moderate. Under these assumptions, the total number of measurements m is most likely to fall in the following closed interval,

$$n \le m \le 2n+1. \tag{27}$$

Then, the upper bounds for the number of nodes at each level in a hypothesis tree can be obtained as below [13].

$$ND_i(n,m) \le C_n^i 3^i, \qquad \text{for} \quad 0 \le i \le n. \tag{28}$$

Accordingly, $M(n,m)$ and $A(n,m)$ are bounded from above as

$$M(n,m) < 2*4^n - 2 - 3n - 3^n \tag{29}$$

$$A(n,m) < 4^n + 3n*4^{n-1}. \tag{30}$$

As shown in (29) and (30), the upper bounds of $M(n,m)$ and $A(n,m)$ are independent of m under the given assumptions.

It is worth noting that $C_n^i 3^i$ is a loose upper bound of $ND_i(n,m)$ for $i \ge 2$. As a result, the upper bounds of $M(n,m)$ and $A(n,m)$ in (29) and (30) may be much larger than the actual values of $M(n,m)$ and $A(n,m)$. In Table II, some sample values of the upper bounds of $M(n,m)$ and $A(n,m)$ are given in order to compare the performance of the DFS algorithm developed here with that of the fast JPDA algorithm in [10]. In Table III, the numbers of multiplications and additions required in the fast JPDA algorithm are given in the case when m is in the range given in (27). As shown in Table III, the computational cost in the fast JPDA algorithm is higher than that of the DFS algorithm when m is in the upper part of the range given in (27). Furthermore, the actual computational cost of the DFS algorithm could be much lower than that shown in Table II since the numbers shown in Table II are upper bounds of $M(n,m)$ and $A(n,m)$ for the DFS algorithm, Therefore, the DFS algorithm is more efficient than the fast JPDA algorithm [10] in the average case.

TABLE III
Number of Multiplications/Additions in the Fast JPDA Algorithm

	n				
m	3	4	5	6	7
n	51/63	260/304	1045/1175	3654/3996	11655/12495
$n-1$	78/132	432/620	1830/2370	6624/8022	21630/25032
$n-2$	111/240	672/1152	3030/4480	11466/15360	38661/48384
$n-3$	150/396	992/1988	4780/8000	19020/28134	66626/90300
$n-4$	195/609	1404/3232	7235/13590	30360/49440	110922/162855
$n-5$	-	1920/5004	10570/22100	46824/83622	178836/284172
$n-6$	-	-	14980/34595	70044/136584	279958/480571
$n-7$	-	-	-	101976/216138	426636/789194
$n-8$	-	-	-	-	634473/1261155

IV. DIRECT COMPUTATION OF β_j^t

The computational cost for the generation of the data association hypotheses is very high when n and m are large. To speed up the computation of the β_j^ts, an algorithm which applies when the density of targets is moderate is proposed in this section. This procedure does not require the generation of the data association hypotheses. The formulae for computing β_j^t directly are given below for the cases when n equals 1, 2, 3, or 4. The proofs are available in [13].

1) For $n = 1$:

$$\beta_j^t = P_{jt} \qquad \text{for} \quad j = 1,2,\ldots,m \tag{31}$$

$$\beta_0^t = P_0. \tag{32}$$

2) For $n = 2$:

$$\beta_j^{t_1} = P_{jt_1}(P_{t_2} - P_{jt_2}) \tag{33}$$

$$\beta_0^{t_1} = P_0 P_{t_2} \tag{34}$$

where $t_1 \neq t_2$ and $P_t = P_0 + \sum_{j=1}^m P_{jt}$, for $t = 1,2$.

3) For $n = 3$:

$$\beta_j^{t_1} = P_{jt_1}[(P_{t_2} - P_{jt_2})(P_{t_3} - P_{t_3}) - (G(t_2,t_3) - G_j(t_2,t_3))] \tag{35}$$

$$\beta_0^{t_1} = P_0(P_{t_2}P_{t_3} - G(t_2,t_3)) \tag{36}$$

where $t_p \neq t_q$ if $p \neq q$, $G_j(t_2,t_3) = P_{jt_2}P_{jt_2}$, and $G(t_2,t_3) = \sum_{j=1}^m G_j(t_2,t_3)$.

4) For $n = 4$:

$$\begin{aligned}\beta_j^{t_1} = P_{jt_1}\Bigg[&\prod_{i=2}^{4}(P_{t_i} - P_{jt_i}) - (P_{t_2} - P_{jt_2}) \\ &\times (G(t_3,t_4) - G_j(t_3,t_4)) \\ &- (P_{t_3} - P_{jt_3})(G(t_2,t_4) - G_j(t_2,t_4)) \\ &- (P_{t_4} - P_{jt_4})(G(t_2,t_3) - G_j(t_2,t_3)) \\ &+ 2(G(t_2,t_3,t_4) - G_j(t_2,t_3,t_4))\Bigg]\end{aligned} \tag{37}$$

TABLE IV
Operation Counts for DC Algorithm

n	Multiplications	Additions
2	$2m + 1$	$4m$
3	$9m - 2$	$18m - 2$
4	$38m - 7$	$70m - 6$

$$\begin{aligned}\beta_0^{t_1} = P_0\Bigg(&\prod_{i=2}^{4} P_{t_i} - P_{t_2}G(t_3,t_4) - P_{t_3}G(t_2,t_4) \\ &- P_{t_4}G(t_2,t_3) + 2G(t_2,t_3,t_4)\Bigg)\end{aligned} \tag{38}$$

where $t_p \neq t_q$ if $p \neq q$, $G_j(t_2,t_3,t_4) = P_{jt_2}P_{jt_2}P_{jt_4}$, and $G(t_2,t_3,t_4) = \sum_{j=1}^m G_j(t_2,t_3,t_4)$.

When the number of targets is larger than 4, the equations for computing β_j^t can be derived. However, the complicated equations themselves may require a lot of memory. In a small system, excessive requirement for memory may not be suitable. Therefore, in this section, the equations for direct computation of β_j^t are given for the cases in which the number of targets in a cluster is less than or equal to 4. The normalization constant c in all cases is obtained from

$$c = \sum_{l=0}^{m} \beta_l^t. \tag{39}$$

In Table IV, the numbers of multiplications and additions required in the direct computation (DC) of the β_j^ts are given for $n = 2,3,4$ in the worst case. It is not difficult to verify that the DC algorithm requires less multiplications and additions than the DFS algorithm for $n = 2,3,4$, and any m. An upper bound on the number of multiplications and additions for the DC algorithm in the average case can be obtained by substituting $m = 3$ in Table IV under the assumption that the number of measurements inside the validation gate of each target is 3 or less.

V. APPROXIMATION OF β_j^t

In this section, an algorithm is proposed to compute β_j^t approximately without computing the conditional probabilities of the data association hypotheses in the JPDAF. In the original JPDAF, the interference from all the targets in the same cluster is considered in the computation of β_j^t. This results in high computational cost for the data association. To reduce the computational cost for the data association, only the interference from the neighbors of target t is considered in the approximation of β_j^t. A target is a neighbor of target t if there is at least one measurement inside the intersection of the validation gates of the two targets. In the DC algorithm, each β_j^t is computed as follows.

$$\beta_j^t = P_{jt} F_{jt} \quad \text{for} \quad t = 1,2,\ldots,n, \quad \text{and} \quad j = 0,1,2,\ldots,m. \tag{40}$$

In (40), F_{jt} is the interference from the other targets in the same cluster. Comparing (40) with (31), the interference F_{jt} may be considered as a weight on P_{jt}.

To reduce the computational cost, in the approximation of β_j^t, only the interferences from the neighbors of target t are considered in the computation of F_{jt}. Let B^t denote the set of neighbors of target t. That is, a target belongs to B^t if there is at least one measurement inside the intersection of the validation gates of this target and target t. Then, the computation of F_{jt} is simplified as shown below.

$$F_{jt} = P_0 + \sum_{i \in B^t} \sum_{\substack{l=1 \\ l \neq j}}^{m} P_{li} \quad \text{for} \quad j = 1,2,\ldots,m \tag{41}$$

$$F_{0t} = P_0 + \sum_{i \in B^t} \sum_{l=1}^{m} P_{li}. \tag{42}$$

Substitute F_{jt} into (40), to obtain

$$\beta_j^t = P_{jt} F_{jt} \quad \text{for} \quad t = 1,2,\ldots,n, \quad \text{and} \quad j = 0,1,2,\ldots,m. \tag{43}$$

Finally, β_j^t is normalized by

$$c^t = \sum_{j=0}^{m} \beta_j^t. \tag{44}$$

In general, $c^t \neq c^\tau$ if $t \neq \tau$. The approximate computation (AC) algorithm is equivalent to the DC algorithm if $n = 2$. Comparing with PDAF [2], the computational cost of the AC algorithm is increased slightly because of the need to compute F_{jt}. Furthermore, it is expected that the performance of the AC algorithm could be close to that of the JPDAF because the interference from neighboring targets of target t is considered in the computation of β_j^t. The computer simulation of the AC algorithm is presented in Section VI.

TABLE V
Initial Positions and Velocities of 16 Targets

Target	Position (km)		Velocity (km/s)	
i	x	y	$\dot{x}$	$\dot{y}$
1	3.0	3.0	0.72	0.12
2	3.0	7.8	0.72	-0.12
3	10.0	2.0	0.45	0.60
4	6.0	10.0	0.62	0.35
5	6.0	12.0	0.62	0.35
6	5.0	24.0	0.50	0.22
7	5.0	30.0	0.50	-0.50
8	4.0	34.0	0.70	-0.31
9	5.0	20.5	0.62	0.00
10	4.0	18.0	0.70	0.00
11	11.0	7.0	0.45	0.55
12	18.0	34.2	0.10	-0.68
13	3.0	29.0	0.71	0.00
14	11.0	33.0	0.60	-0.60
15	12.0	7.0	0.35	0.65
16	4.0	26.0	0.60	0.25

TABLE VI
Maneuver Parameters of Target 10 and Target 11

Trajectory	Acceleration	Turn Started	Turn Finished
i	a_c (m/s^2)	at (s)	at (s)
10	-0.05	10	25
11	0.08	15	30

VI. COMPUTER SIMULATION

In the simulation, the dynamic models for the targets have been digitized using the sampling period T normalized to 1 s and the state vectors have been represented in 2-dimensional Cartesian coordinates (or in (X,Y) plane). Furthermore, only position measurements have been assumed to be available. The surveillance region used in the simulation is a 35 km by 35 km square and the initial positions and velocities of 16 targets in 2-dimensional Cartesian coordinates are given in Table V. The performance of the three algorithms proposed are tested in two separate cases in the simulation. In the first case, suppose that the dynamic characteristics of all targets are known, i.e., every target moves at a constant velocity. In the second case, however, some targets could make a maneuver at any time. In the simulation, target 10 and 11 are capable of making a turn with a centripetal acceleration. The maneuver parameters of the two targets are given in Table VI.

Since every target is nonmaneuvering in the first case, the generic target dynamic model has the

following form:

$$\mathbf{x}(k+1) = F\mathbf{x}(k) + G\mathbf{w}(k) \tag{45}$$

$$\mathbf{z}(k) = H\mathbf{x}(k) + \mathbf{v}(k) \tag{46}$$

where

$$\mathbf{x}(k) = (x(k) \quad \dot{x}(k) \quad y(k) \quad \dot{y}(k))'$$

$$\mathbf{w}(k) = (w_1(k) \quad w_2(k))'$$

$$E\{\mathbf{w}(k)\} = 0$$

$$E\{\mathbf{w}(k)\mathbf{w}'(j)\} = Q\delta(k-j)$$

$$F = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$G = \begin{pmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

and

$$E\{\mathbf{v}(k)\} = 0; \qquad E\{\mathbf{v}(k)\mathbf{v}'(j)\} = R\delta(k-j).$$

For maneuvering targets, a higher order of dynamic model is required to describe the characteristics of these targets. In the simulation, the Singer's model developed in [14] is used for all targets in the second case. The state and observation equations in the Singer's model are described by

$$\mathbf{x}_m(k+1) = F_m\mathbf{x}_m(k) + \mathbf{u}(k) \tag{47}$$

$$\mathbf{z}(k) = H_m\mathbf{x}_m(k) + \mathbf{v}(k). \tag{48}$$

In (47) and (48), $\mathbf{x}_m(k)$, F_m, and H_m are defined as

$$\mathbf{x}_m(k) = (x(k) \quad \dot{x}(k) \quad \ddot{x}(k) \quad y(k) \quad \dot{y}(k) \quad \ddot{y}(k))'$$

$$F_m = \begin{pmatrix} f(T) & 0 \\ 0 & f(T) \end{pmatrix};$$

$$f(T) = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix}$$

$$H_m = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The statistical properties of $\mathbf{v}(k)$ in (48) are identical to those of $\mathbf{v}(k)$ in (46). In (47) and (48), the two noise terms, $\mathbf{u}(k)$ and $\mathbf{v}(k)$ are still assumed to be mutually independent. The state noise $\mathbf{u}(k)$ is a zero-mean Gaussian white random process with covariance matrix $Q_m(k)$ of the following form:

$$Q_m(k) = 2\alpha \begin{pmatrix} \sigma_x(k)q(T) & 0 \\ 0 & \sigma_y(k)q(T) \end{pmatrix};$$

$$q(T) = \begin{pmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{pmatrix}$$

where α is the maneuver correlation coefficient, and $\sigma_x(k)$ and $\sigma_y(k)$ are, respectively, the X and Y components of the maneuver magnitude variance. Both $\sigma_x(k)$ and $\sigma_y(k)$ are adaptively estimated in the simulation.

The correct returns from a target are generated by adding noise to the computed true position of the target. The standard deviation of the measurement noise has been selected as $\sqrt{R_{ii}} = 0.15$ km for both the X and Y components. The correct return would pass a detector with probability of detection $P_D = 0.99$. The clutter is generated uniformly over the whole surveillance region. The total number of clutter returns observed in the region is a Poisson random number. The density of the clutter λ is selected to be $0.05/\text{km}^2$. This has given an average of 0.5–2 clutter per gate in the examples given below. The threshold g^2 for the validation gate is set to 17.0.

The efficiency of the three algorithms is estimated in terms of central processing unit (CPU) time used in the data association in a single sample run. The simulation is performed on a VAX 8550. The exact CPU time for each example could vary from time to time, depending upon the load of the computer. The robustness of the three algorithm is evaluated using the success rate in tracking both maneuvering and nonmaneuvering targets in clutter. The success rate means the number of sample runs finished without lost tracking of a single target in 100 different sample runs.

In the simulation, two examples are used to track 5 and 16 targets with different dynamic models given in (45) and (47). The initial parameters of the 5 targets in the first example are taken from targets 4, 6, 7, 11, and 13 in Table V. In the second example, all targets listed in Table V are used. When all targets are supposed to be nonmaneuvering, the two examples are shown in Figs. 4 and 6. If the Singer's model in (47) is used for all targets, the trajectories of all targets in the two examples are shown in Figs. 5 and 7.

The CPU times used in the data association in the three algorithms in one sample run are listed in Table VII. In the first example, as shown in Figs. 4 and 5, the largest size of cluster of targets is 3. But most of the time during simulation, the sizes of clusters are 1 and, occasionally, 2. When there is one target in a cluster, a PDAF is applied for tracking this target. In the three algorithms, the equations used to compute β_j^t are the same when the size of a cluster is

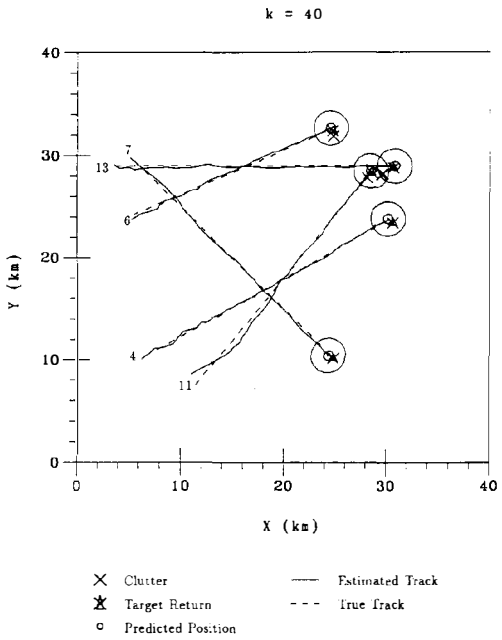

Fig. 4. Tracking 5 nonmaneuvering targets.

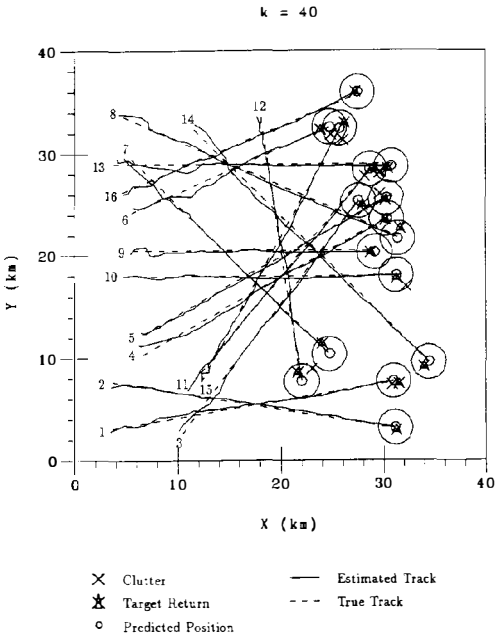

Fig. 6. Tracking 16 nonmaneuvering targets.

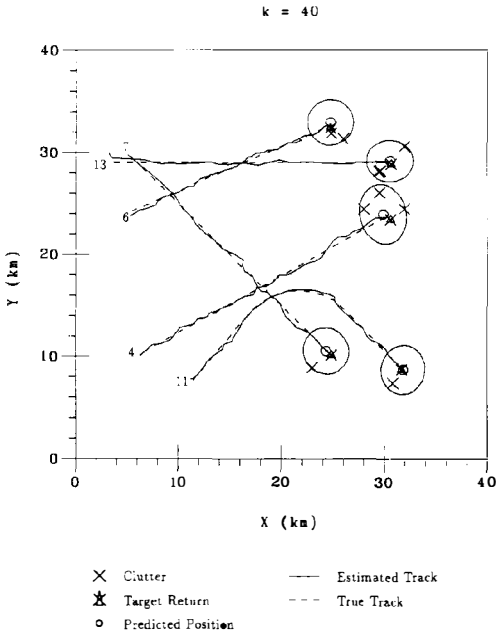

Fig. 5. Tracking 5 maneuvering targets.

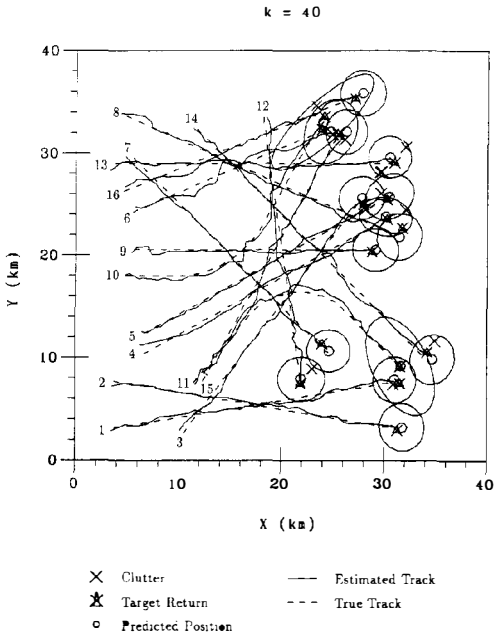

Fig. 7. Tracking 16 maneuvering targets.

1. As a result, in this example, the CPU times used in the data association are almost the same in the three algorithms for tracking nonmaneuvering and maneuvering targets. However, in the second example as shown in Figs. 6 and 7, the size of the largest cluster is frequently ranging from 6 to 8, and it is sometimes as large as 9. This leads to a significant increase of the CPU time used for data association in the DFS algorithm. For the AC algorithm, the increase of the CPU time is proportional to the increase of the number of targets because of the way in which the data association is considered in this algorithm. Although the AC algorithm is much more efficient than the DFS algorithm when the targets are nonmaneuvering, the AC algorithm is not suitable for use to track 16 targets when there are possible maneuvers. Since the size of

TABLE VII
CPU Time Used in Data Association in Seconds (One Sample RUN)

# of targets	Nonmaneuvering (45)			Maneuvering (47)		
n	DFS	AC	DC	DFS	AC	DC
5	0.14	0.14	0.09	0.22	0.11	0.12
16	35.98	0.39	N/A	78.80	N/A	N/A

TABLE VIII
CPU Time Used in Data Association in Seconds (One Sample Run)

# of targets	Nonmaneuvering (45)			Maneuvering (47)		
n	DFS	AC	DC	DFS	AC	DC
5	100	100	100	94	94	94
16	85	80	N/A	32	N/A	N/A

the cluster of targets is frequently larger than 4 in this example, the DC algorithm is not applicable.

The success rates of the three algorithms are listed in Table VIII. When targets are nonmaneuvering, target 11, as shown in Fig. 6, causes most of the mistracking. However, when the dynamic model given in (47) is used, the two maneuvering targets are the major cause of the mistracking. Of course, the different layout of the targets could lead to different success rates. Especially, the success rates might be improved if either target 11 is removed or it is moved to a different position. However, the current layout of the targets could help us understand the limitation of the joint data probabilistic association technique. In all examples, the available success rates of the DFS and the DC algorithms are the same, as expected, when the density of targets is not high. But the DC algorithm is computationally more efficient than the DFS algorithm. The high success rate of the DFS algorithm is accompanied by high computational cost in comparison with the AC algorithm. Although the AC algorithm is much more efficient than the DFS algorithm when the targets are nonmaneuvering, the AC algorithm is not suitable for use to track 16 targets when there are possible maneuvers. The performances of the DFS and the AC algorithms deteriorate with the increase in the number of targets. This difficulty is caused by the increase of the number of measurements inside each gate. Therefore, additional information is needed to reduce the uncertainty in the data association. Practically, the choice of the three algorithm in the design of a tracking system is dependent upon the computational capability and the requirement of tracking accuracy.

VII. CONCLUSION

In this paper, three algorithms have been proposed for fast computation of the *a posteriori* probability β_j^t in the JPDAF. As shown in the simulation, the CPU time is increased drastically when the sizes of clusters of targets are large. Therefore, the DFS algorithm is suitable for implementation in a tracking system and in a ground-based surveillance system with a large centralized computational capability. Furthermore, the JPDAF implemented with the DFS algorithm could serve as a standard for any future attempt to approximate the JPDAF. The advantages of the AC algorithm proposed here are that the computation of the *a posteriori* probability β_j^t is very efficient and that the success rate is relatively high in comparison with the JPDAF without any approximation. Especially, the AC algorithm is suitable for implementation in a multiprocessor system. The DC algorithm is more efficient than the DFS algorithm and can be implemented in a multiprocessor system.

Although the discussion here is focused on the target-oriented approach, i.e., JPDAF [3], it is not difficult to extend the DFS algorithm to the measurement-oriented approach [4] and to the track-oriented approach [10]. Furthermore, the DFS-based technique is applicable to three dimensional validation matrices which occur in Markov models of system parameters for maneuvering targets in clutter [15] and in multiscan correlation [4].

REFERENCES

[1] Bar-Shalom, Y., and Fortmann, T. E. (1988)
Tracking and Data Association.
New York: Academic Press, 1988.

[2] Bar-Shalom, Y., and Tse, E. (1975)
Tracking in a cluttered environment with probabilistic data association.
Automatica, **11** (Sept. 1975), 451–460.

[3] Fortmann, T. E., Bar-Shalom, Y., and Scheffe, M. (1983)
Sonar tracking of multiple targets using joint probabilistic data association.
IEEE Journal of Oceanic Engineering, **OE-8** (July 1983), 173–184.

[4] Reid, D. B. (1979)
An algorithm for tracking multiple targets.
IEEE Transactions on Automatic Control, **AC-24** (Dec. 1979), 843–854.

[5] Bar-Shalom, Y. (1990)
Multitarget-Multisensors Tracking: Advanced Applications.
Norwood, MA: Artech House, 1990.

[6] Mahalanabis, A. K., Zhou, B., and Bose, N. K. (1990)
Improved multi-target tracking in clutter by PDA smoothing.
IEEE Transactions on Aerospace and Electronic Systems, **26** (Jan. 1990), 113–121.

[7] Fitzgerald, R. J. (1986)
Development of practical PDA logic for multitarget tracking by microprocessor.
In *Proceedings of the American Controls Conference*, Seattle, WA, June 1986, 889–898.

[8] Sengupta, D., and Iltis, R. A. (1989)
Neural solution to the multiple target tracking data association problem.
IEEE Transactions on Aerospace and Electronic Systems, **25** (Jan. 1989), 96–108.

[9] Zhou, B., and Bose, N. K. (1993)
A comprehensive analysis of "neural solution to the multitarget tracking data association problem."
IEEE Transactions on Aerospace and Electronic Systems, **29** (Jan. 1993), 260–263.

[10] Fisher, J. L., and Casasent, D. P. (1989)
Fast JPDA multitarget tracking algorithm.
Applied Optics, **28** (Jan. 1989), 371–376.

[11] Nagarajan, V., Chidambara, M. R., and Sharma, R. N. (1987)
Combinatorial problems in multitarget tracking—A comprehensive solution.
IEE Proceedings Part F, Communications, Radar and Signal Processing, **134** (Feb. 1987), 113–118.

[12] Reingold, E. M., Nievergelt, J., and Deo, N. (1977)
Combinatorial Algorithms, Theory and Practice.
Englewood Cliffs, NJ: Prentice-Hall, 1977.

[13] Zhou, B. (1992)
Multitarget tracking in clutter: Algorithms for data association and state estimation.
Ph.D. dissertation, Pennsylvania State University, Dept. of Electrical and Computer Engineering, University Park, PA, May 1992.

[14] Singer, R. A. (1970)
Estimating optimal tracking filter performance for manned maneuvering targets.
IEEE Transactions on Aerospace and Electronic Systems, **AES-6** (July 1970), 473–483.

[15] Sengupta, D., and Iltis, R. A. (1989)
Tracking of multiple maneuvering targets in clutter by joint probabilistic data and maneuver association.
In *Proceedings of the American Controls Conference*, Pittsburg, PA, June 1989, 2696–2701.

N. K. Bose (S'66—M'67—SM'74—F'81) received his B.Tech (with honors), M.S., and Ph.D. degrees, all in electrical engineering, at Indian Institute of Technology, Kharagpur, India, Cornell University, Ithaca, NY and Syracuse University, Syracuse, NY.

He is currently the HRB-Systems Professor and Director of the Spatial and Temporal Signal Processing Center at Pennsylvania State University, University Park, PA.

Dr. Bose is the author or editor of several books and numerous papers. He is currently Editor-in-Chief of *Multidimensional Systems and Signal Processing*.

Bin Zhou (S'86—M'92) received the B.S. degree in electrical engineering from Fudan University, Shanghai, China, in 1982, the M.S. degree in computer engineering from Shanghai Institute of Technical Physics, China, in 1985, and the Ph.D. degree in electrical engineering from Pennsylvania State University, University Park, in 1992.

Dr. Zhou is currently a postdoctoral fellow at the Center for Spatial and Temporal Signal Processing and the Center for Multivariate Analysis, Pennsylvania State University. His research interests are in the areas of multitarget tracking, array signal processing, image processing, and pattern recognition.