REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1995	Professional Paper

4. TITLE AND SUBTITLE

A DATA FUSION SIMULATION ALGORITHM BASED ON CONFUSION MATRIX PERFORMANCE METRICS

5. FUNDING NUMBERS

PR: ECB2
PE: 060223YN
WU: DN300086

6. AUTHOR(S)

G. F. Kramer

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152–5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Data fusion is an increasingly important capability in operational command and control systems. A fundamental function of data fusion processing associates unidentified contact reports from multiple sensors to a common target, so that the commander has a consistent tactical picture from which to make decisions. Clearly computer simulations should model the data fusion aspect of the command and control process. However, since data fusion algorithms are typically computationally intensive, they have been neglected in these simulations.

This paper describes a new approach for low resolution simulation of data fusion processing. Operational data fusion systems are evaluated on known data sets using four performance metrics calculated from the confusion matrix—a tabulation of the number of contact reports from each ground truth track assigned to each constructed track. The simulation algorithm described in this paper makes probabilistic data fusion decisions based on values input for these metrics. This paper also describes an object-oriented implementation of the algorithm in C++ and presents numerical results from a simulation study of the algorithm itself, showing its ability to produce data fusion results with the input performance characteristics.

Published in *Proceedings of Military, Government and Aerospace Simulation*, Simulation Series, 82, April 1995.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Data Fusion Probabilistic Algorithm C++	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

A DATA FUSION SIMULATION ALGORITHM BASED ON CONFUSION MATRIX PERFORMANCE METRICS

G. Fred Kramer
Naval Command, Control and Ocean Surveillance Center
RDT&E Division, Code 422
53560 Hull St.
San Diego, CA 92152-5001
gkramer@nosc.mil

KEYWORDS

Data fusion, probabilistic algorithm, C++.

ABSTRACT

Data fusion is an increasingly important capability in operational command and control systems. A fundamental function of data fusion processing associates unidentified contact reports from multiple sensors to a common target, so that the commander has a consistent tactical picture from which to make decisions. Clearly computer simulations should model the data fusion aspect of the command and control process. However, since data fusion algorithms are typically computationally intensive, they have been neglected in these simulations.

This paper describes a new approach for low resolution simulation of data fusion processing. Operational data fusion systems are evaluated on known data sets using four performance metrics calculated from the confusion matrix—a tabulation of the number of contact reports from each ground truth track assigned to each constructed track. The simulation algorithm described in this paper makes probabilistic data fusion decisions based on values input for these metrics. This paper also describes an object-oriented implementation of the algorithm in C++ and presents numerical results from a simulation study of the algorithm itself, showing its ability to produce data fusion results with the input performance characteristics.

INTRODUCTION

In a report to Congress, summarized by Tolchin (1989), the Departments of Defense and Energy identified data fusion as one of the 22 technologies critical to national security and the long-term qualitative superiority of U.S. weapons systems. The directors of the Department of Defense (DOD) command and control (C^2) research and development centers have formed a group to coordinate data fusion issues among the services. This group has defined data fusion as: "A process that deals with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats as well as their significance." (White 1991) Data fusion is an aspect of virtually all C^2 systems in one way or another.

The list also identified computer simulation and modeling as a critical technology. Computers have long been used for a variety of defense-related simulation purposes, including tactics development and assessment, weapons requirements and design, and operational training. However, recent increases in computer processing speed, the development of high-speed, high-capacity networks, and the emergence of object-oriented programming languages, such as C++, offer the opportunity for even more sophisticated simulations. Large-scale simulations will be put together from a menu of object models developed and maintained by geographically separated organizations with expertise in specific areas. These object models undoubtedly will be developed at variable resolutions. For example, high resolution models for sensor design studies may include detailed environmental data and solve complicated propagation equations, while low resolution models appropriate for campaign-level simulations may only use the simple "cookie cutter" detection approach. Davis and Huber (1992) discuss a number of issues related to variable resolution modeling and conclude that hierarchical models provide the best approach.

This paper presents work supporting an Office of Naval Research task to investigate high performance

computing for C^2 modeling. The goal was to develop data fusion object models at several resolutions and use them for analysis and benchmark performance testing on a variety of single processor and multiple processor platforms. This paper proposes a unique, low resolution data fusion simulation algorithm, describes its implementation in a C++ class hierarchy that could support data fusion simulations at other resolutions, and provides numerical results of a simulation study of the new algorithm itself.

DATA FUSION ALGORITHMS AND THEIR EVALUATION

The DOD data fusion group (White 1991) has defined four levels of data fusion processing. Level 1 processing is characterized by data alignment, data/object correlation, and position and identity estimation. The goal of this level is a complete and accurate representation of the targets in a tactical scene. Level 2 processes use the tactical picture and attempt to explain the situation. Level 3 processes interpret the situation in terms of the threat to friendly forces. Finally, Level 4 processes monitor the previous three levels to evaluate the effectiveness of the surveillance and data fusion processing for the purpose of resource management.

Most data fusion development efforts have focused on the Level 1 problem, which historically has been called report-to-track or track-to-track correlation. Simply stated, the goal is to construct target tracks from incoming contact reports such that each constructed track (CT) contains reports from one and only one ground truth track (GT) and each GT is represented by one and only one CT.

Kendall, Stuart, and Ord (1983) describe two types of statistical classification procedures that have been applied to the Level 1 problem when the contact reports contain numerical data. In the situations with a training data set (previously identified contact reports from each GT), discrimination procedures develop a classification rule that identifies new, unidentified contact reports as one of the known GT's. In other situations, clustering techniques divide contact reports into CT's that are as distinct as possible. In discrimination the number of GT's is known, while in clustering it is to be determined. Batch and sequential procedures have been developed for both discrimination and clustering. Most report-to-track correlation algorithms are sequential clustering procedures.

Performance evaluation of statistical data fusion algorithms is an important aspect of their development. For discrimination methods, well-founded techniques, such as crossvalidation, can be used to assess overall classification performance by estimating error rates. However, evaluating clustering performance is not as straightforward. Typically these algorithms have parameters that can be adjusted to optimize some aspect of performance, for example, number of CT's created. In this case there are several aspects to performance and a variety of measures is necessary. Most are calculated from the "so-called" confusion matrix, which is a common basis for evaluating classification procedures on identified data sets. In the data fusion context, the confusion matrix has one column for each GT and one row for each CT. The (i,j) entries are the number of reports from the j^{th} GT the algorithm assigns to the i^{th} CT.

Fox, Mills, Thode, Crespo, and Mendiguchia (1991) define four performance measures that correspond to the possible decisions a Level 1 data fusion algorithm can make for each contact report: correct assignment, mis-assignment, fragmentation, and ambiguity. An algorithm makes a correct assignment when it puts the contact report in a CT that represents the correct GT. An algorithm makes a mis-assignment when it puts the contact report in a CT that is represents an incorrect GT. For evaluation purposes, the "correct GT" for a CT is assumed to be the one with the most contact reports assigned, even though that might not be the GT that initiated the CT. Note that a single GT may be represented by more than one CT, but that a CT will represent only one GT. (Ties need to be arbitrarily decided.) Fragmentation occurs when the algorithm creates a new CT for a particular GT when there is already an existing CT established for the GT. The contact creating the extra track is counted as a fragmentation error, but subsequent contacts in that CT are considered correct decisions, as long as the largest number of assigned reports come from that GT. An algorithm makes an ambiguous assignment when it assigns the contact report to no CT. For ease of evaluation, the set of ambiguous contacts report is usually identified by a single pseudo-CT, e.g., CT 0. The performance measures in Fox *et al.* (1991) are the ratios of number of contacts assigned in each decision category to total number of contacts processed.

To calculate these performance measures, let N be the total number of contacts processed, $N_a \le N$ be the number that are ambiguous, #GT be the number of ground truth tracks (= number of columns in the confusion matrix), and #CT the number of constructed tracks (= number of rows). Then,

$$car = \frac{\sum(\text{row}_i\_\text{maximum}) - \max(\#CT - \#GT, 0)}{N} \quad (1)$$

$$mar = \frac{(N - N_a) - \sum(\text{row}_i\_\text{maximum})}{N} \quad (2)$$

$$fr = \frac{\max(\#CT - \#GT, 0)}{N} \quad (3)$$

$$ar = \frac{N_a}{N} \quad (4)$$

where *car* is correct assignment ratio, *mar* is mis-assignment ratio, *fr* is fragmentation ratio, and *ar* is ambiguity ratio. The summation in equations (1) and (2) totals the largest entries in each row. The maximum function in equations (1) and (3) handles the case of fewer constructed tracks than ground truth tracks. It is clear that these performance metrics add to 1 and hence consider the range of algorithm decisions. However, their ability to measure the quality of data fusion algorithm decisions is less apparent. In the extreme, perfect data fusion performance corresponds to a square confusion matrix with all the non-zero entries along a diagonal; in this case we can easily see that the correct assignment ratio is 1 and the other metrics are 0. Fox *et al.* (1991) discuss these and other performance metrics in more detail and provide some numerical examples.

A DATA FUSION SIMULATION ALGORITHM USING CONFUSION MATRIX METRICS

Data fusion might be an aspect of computer simulation studies for a variety of reasons. First, in a campaign simulation, tactical decisions depend on the state of the scenario, which is known and input by the model user. The simulation will have added realism if the decisions are based on a perceived tactical situation, not the actual ground truth situation. The perceived situation will, of course, be different from the actual one. Data fusion algorithms (as a key part of C^2 systems) are the mechanism by which tactical commanders get their view of the operational situation. Second, the effect of data fusion performance on the tactical outcome might be of interest in C^2 studies. Finally, the simulation might be focused on the data fusion process itself, and different algorithmic approaches evaluated in a number of scenarios.

Since actual data fusion algorithms are not required in the first two modeling situations, a low level resolution approach can be used. That is, in these cases the data fusion processing can be simulated probabilisitcally since ground truth is known. The algorithm described here uses the performance metrics *car*, *mar*, *fr*, and *ar* as simulation parameters representing the average data fusion performance desired. So, the first step is to assign values between 0 and 1 to each of these four parameters, such that their sum is 1. Simulated data fusion processing of a contact report begins by generating a random number u uniformly on [0,1]. If $0 \le u < car$, the algorithm makes a correct assignment decision; if $car \le u < car + mar$, a mis-assignment decision; if $car + mar \le u < car + mar + fr$, a fragmentation decision; and an ambiguity decision otherwise.

While the general implementation of the four decisions is straightforward, a number of special cases need to be recognized and handled by the data fusion simulation algorithm. Table 1 defines the five possible cases. The GT column indicates whether the GT in the current contact report has already been processed; the CT column denotes whether there already is a constructed track associated with this GT (if an old GT) or whether there are any CT's at all (if new a GT); and the MGT column whether there is more than one GT with associated CT's. The reasons for these particular columns will become apparent as we discuss the cases. The next set of columns in Table 1 provide the data fusion action taken in each of the five cases for possible values of u.

Case 1 is the general case in which all four decisions can be implemented. In case 2 only one GT has associated CT(s), although more than one GT may have been processed.

Table 1. Simulated Data Fusion Algorithm Processing Cases

Cases				Data Fusion Decision			
Num	GT	CT	MGT[1]	$0 \le u < car$	$car \le u < car + mar$	$car + mar \le u < car + mar + fr$	$car + mar + fr \le u \le 1$
1	yes	yes[2]	yes	correct assign	mis-assign	fragment	ambiguous
2	yes	yes[2]	no	correct assign	fragment		ambiguous
3	yes	no[2]	—	create CT	mis-assign		ambiguous
4	no	yes[3]	—	create CT	mis-assign		ambiguous
5	no	no[3]	—	create first CT			

[1]Multiple GT's with associated CT's [2]Associated to GT [3]Any CT, associated or not

In case 2, mis-assignments are not possible and are combined with fragmentation actions. Case 3 represents the situation in which the GT has been seen before, but no CT has been created for it, because the previous contacts reports were erroneously processed. In this case fragmentation actions are not possible and are grouped with mis-assignments. Case 4 represents the situation in which the GT has not been seen before, but there are existing CT's. This case, as with case 3, does not allow fragmentation actions, which are again combined with mis-assignments. In cases 3 and 4, the correct assignment action is to create a new track. Finally, case 5 is the special case of the first contact report processed. No matter what values of the confusion matrix metrics are specified, the decision is always to create the first track with the first contact report.

Pseudo-code for the simulated data fusion algorithm is provided in Figure 1. It is a set of nested if-then-else constructions to identify and process the five cases described in Table 1. Required processing actions are indicated by boldface type. The code in Figure 1 reflects two arbitrary decisions. In the case of a correct assignment decision, there may be more than one CT to which the contact report could be assigned. The convention applied is to assign it to the CT created most recently, represented by the highest CT number. This tends to somewhat reflect how actual data fusion algorithms behave. The convention could just as easily been to assign it to the first CT created or randomly to one of the CT's associated with the GT. For mis-assignment decisions, there is usually more than one CT to which the contact report can be mis-assigned. The algorithm randomly selects one of these CT's. Both of these design decisions do not affect the values of confusion matrix performance metrics calculated from the results of the simulated data fusion algorithm.

```
Assign performance values: car, mar, fr, ar
Compute random number u from [0,1]
if (existing ground truth (GT)) {
 if (existing constructed track (CT) for this GT) {
  if (more than one GT for set of CT's) {
   if (u < car) Correctly assign to latest existing CT
   else if (u < car+mar) Incorrectly assign to random
CT
   else if (u < car+mar+fr) Incorrectly create new CT
   else Report is ambiguous
  }
  else {
   // Only one GT for set of CT's
   if (u < car) Correctly assign to latest existing CT
   else if (u < car+mar+fr) Incorrectly create new CT
   else Report is ambiguous
  }
 }
 else {
  // No existing CT for this GT
  if (u < car) Correctly create new CT
  else if (u < car+mar+fr) Incorrectly assign to random
CT
  else Report is ambiguous
 }
else {
 // New GT
 Add to GT list
 if (At least one CT) {
  if (u < car) Correctly create new CT
  else if (u < car+mar+fr) Incorrectly assign to random
CT
  else Report is ambiguous
 }
 else
  // No CT's
  Correctly create first track
}
```

Figure 1. Pseudo-Code for Simulated Data Fusion Algorithm

OBJECT-ORIENTED MODELING FOR DATA FUSION

Even though the data fusion simulation algorithm described in the previous section is quite simple, as data fusion algorithms go, it can nonetheless be implemented using the object-oriented paradigm. This structure can be used for more complicated algorithms as well. This section is not meant to present a rigorous object-oriented design methodology, but rather discuss some of the issues involved.

The first step is a description of the problem. In this case, we need a (simulated) data fusion algorithm that assigns contact reports to tracks. So, our first breakdown of required objects would be: (1) data fusion algorithm; (2) report; and (3) track. A track object should contain the reports assigned to it. Hence, we need a fourth object, a list of reports. Finally, since we expect to deal with more than one track, we need a list of tracks as the fifth object.

A rough view of how these objects might interact is shown in Figure 2. A Report object is read by a Data_Fusion object which processes the Report against the Track_List (i.e., data base) and assigns the Report to a specific Track object in the Track_List or creates a new Track in the Track_List. The Track object maintains a Report_List object containing the Reports assigned to that Track.

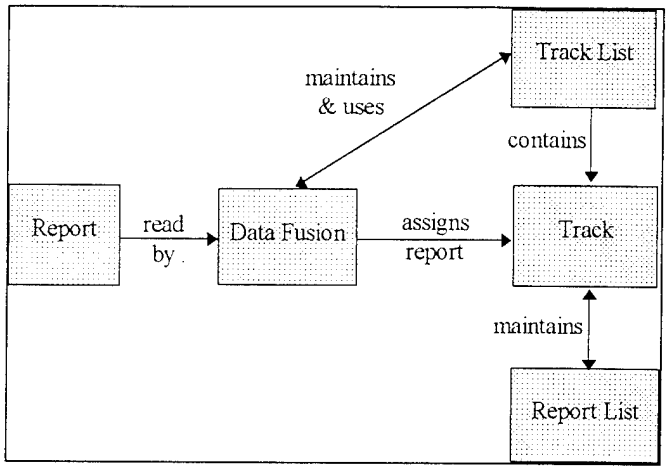

Figure 2. Data Fusion Object Interactions

Figure 3 shows C++ class declarations for these objects. Note that these represent just a bare minimum of data members and member functions to illustrate the idea. In addition, for clarity all declarations are public. Data_Fusion is a base class for the DF_Simulation class. Other derived classes in the hierarchy could represent different algorithmic approaches to data fusion. The Data_Fusion base class essentially only manages the track data base for any derived class through the Track_List data member trklst. This base class also defines a virtual function named process, which each derived class will re-define to implement specific data fusion algorithms. Hence, process takes a Report reference as an argument and returns the track number index to which the algorithm assigns the report. The DF_Simulation class implements the data fusion simulation algorithm discussed in the previous section. Hence, this class has four data members for the confusion matrix performance metrics: correct assignment ratio, etc. Its member function redefines the virtual function process.

The Track_List class manages a list of Tracks. Its data members reflect the number of tracks in the list and the array of Track pointers. Member functions allow reading the Track_List object from and writing it to a file. The overloaded array reference operator ([ ]) allows access to a specific Track in the list. The overloaded additive assignment operator (+=) allows adding a new Track to the list, and the remove function allows deleting a Track from the list by specifying its index number. The Track class has a data member track_number for identification purposes and a Report_List data member for managing the Reports assigned to the Track by the Data_Fusion object. The Report_List class is analogous to the Track_List class. Finally, the Report class has two basic data members, the ground truth track number and the constructed track number. Certainly in actual applications, additional data members and member functions would be added to these

```
class Data_Fusion {
 public:
   Track_List trklst;
   virtual int process (const Report&) {return 0;}
};
class DF_Simulation : public Data_Fusion {
 public:
   double correct_assignment_ratio, mis_assignment_ratio,
          fragmentation_ratio, ambiguity_ratio;
   setratios (const double&, const double&, const double&,
    const double&);
   int process (const Report&);
};
class Track_List {
 public:
   int number_tracks;
   Track **trk_ptr;
   int Read (const char *filename);
   Track& operator[] (const int&);
   Track_List& operator += (const Track&);
   void Remove (const int&);
   int Write (const char *filename);
};
class Track {
 public:
   int track_number;
   Report_List rptlst;
};
class Report_List {
 public:
   int number_reports;
   Report **rpt_ptr;
   int Read (const char *filename);
   Report& operator[] (const int&);
   Report_List& operator+= (const Report&);
    void Remove (const int&);
   int Write (const char *filename);
};
class Report {
 public:
   int GT_Number; // for evaluation purposes
   int CT_Number; // assigned by data fusion algorithm
};
```

Figure 3. Data Fusion Related C++ Class Declarations

classes either directly or through inheritance.

Figure 4 shows a C++ code fragment that implements the object interactions in Figure 2 using object instances from the classes in Figure 3. The code illustrates the basics of object-oriented programming to instantiate a data fusion object and use it for processing Reports. This example certainly does

```
Report rpt;
DF_Simulation good_df_sim;
int trkndx;
good_df_sim.setratios(0.85, 0.075, 0.025, 0.05);
        .   .   .
// loop on reports
{
        .   .   .
  trkndx = good_df_sim.process(rpt);
  if (trkndx <= good_df_sim.trklst.number_tracks)
    good_df_sim.trklst[trkndx].rptlst += rpt;
  else {
    Track temp_trk;
    temp_trk.track_number = trkndx;
    good_df_sim.trklst += temp_trk;
    good_df_sim.trklst[trkndx].rptlst += rpt;
  }
  rpt.CT_Number = trkndx;
}
```

Figure 4. C++ Code Fragment for Data Fusion

not represent robust code, particularly since the function definitions have not been specified. The first line in Figure 4 creates a Report object named rpt that is used to hold Report values for processing. The second line instantiates a DF_Simulation object named good_df_sim and line four assigns appropriate values to its data members. The main context of the code fragment is a loop on reports such that the variable rpt contains current Report values during each iteration. The good_df_sim process function returns the Track number index to which the data fusion simulation algorithm assigns rpt. If this is less than or equal to the number of Tracks in the Track_List for good_df_sim, then the Report is appended to the Report_List for that Track. If not, a temporary Track is created with the new track number, added to good_df_sim's Track_List, and the Report appended to this new Track's Report_List. Finally, the track index number is assigned to the current Report. The important thing to note about this example is the simplicity of the source code once the objects have been defined.

NUMERICAL RESULTS

An object-oriented version of the data fusion simulation algorithm was implemented in Borland C++ on a DOS PC and numerical results were obtained for four scenarios and six different sets of performance parameters. The scenarios were: 1 track with 100 reports; 10 tracks with 25 reports each; 25 tracks with 10 reports each; and 100 tracks with 1 report each. The parameter settings were chosen to represent a data fusion algorithm with good performance, as reflected in Fox *et al.* (1991), an algorithm with mediocre performance, and four limiting cases. The algorithm was run 100 times on each of the 24 scenario-parameter combinations and a confusion matrix and performance metrics calculated for each run. The performance metrics were then averaged over the 100 runs, with the results expected to match the input performance parameters. In the scenarios with multiple tracks and multiple reports per track, the inner loop was on tracks.

The numerical results are summarized in Table 2. We see that the simulation algorithm reproduces the input parameters values reasonably well. In the cases of good and mediocre performance (the first two sections in Table 2), the results are within expected statistical variation. For the scenarios with one ground truth track, there is no possibility of mis-assignment. For the scenarios with one report per track, there is no possibility of track fragmentation. In-between these

Table 2. Numerical Results From Simulated Data Fusion Algorithm

Scenario		Performance Values			
Trks	Rpts	*car*	*mar*	*fr*	*ar*
1	100	.859	0	.095	.047
10	25	.849	.078	.024	.048
25	10	.852	.077	.022	.048
100	1	.849	.100	0	.051
Actual		.850	.075	.025	.050
1	100	.504	0	.393	.103
10	25	.501	.253	.144	.102
25	10	.503	.280	.119	.099
100	1	.507	.400	0	.095
Actual		.500	.250	.150	.100
1	100	1	0	0	0
10	25	1	0	0	0
25	10	1	0	0	0
100	1	1	0	0	0
Actual		1	0	0	0
1	100	.010	0	.990	0
10	25	.184	.756	.060	0
25	10	.110	.890	0	0
100	1	.010	.990	0	0
Actual		0	1	0	0
1	100	.010	0	.990	0
10	25	.184	.756	.060	0
25	10	.110	.890	0	0
100	1	.010	.990	0	0
Actual		0	0	1	0
1	100	.010	0	0	.990
10	25	.004	0	0	.996
25	10	.004	0	0	.996
100	1	.010	0	0	.990
Actual		0	0	0	1

extremes, the confusion matrix metrics calculated from the results agree very well with input values. For the case of perfect correlation performance (actual *car*=1), the simulation algorithm performs flawlessly. However, for the other limiting cases of input parameter values (zeros and ones), the computed metrics do not match the inputs as closely, but the differences are expected. No data fusion algorithm, real or simulated, can perform totally erroneously—it should always get the first contact report correct. This does not explain all the discrepancies, though. In the scenarios of 10 ground truth tracks with 25 contact reports each and 25 tracks with 10 reports each, the computed correct assignment percentages are 0.184 and 0.110, respectively, even though perfect mis-assignment or total fragmentation is input. This is the aspect where the simulation algorithm and the confusion matrix metrics diverge slightly. The algorithm, in randomly mis-assigning reports to constructed tracks, mis-assigns more incorrect reports to the track than correct ones. The algorithm keeps track of which ground truth track is represented by each constructed track. The computation of performance metrics from the confusion matrix assume the association is based on the most reports assigned. This is not a serious discrepancy, and only affects results when input performance parameters are set at or near the extremes.

CONCLUSIONS

Data fusion is an important aspect of virtually all C^2 systems and should be incorporated into computer simulations for added realism. The probabilistic algorithm for simulating data fusion proposed in this paper can be used as a low resolution model in these applications. The relatively simple algorithm is based on four confusion matrix performance metrics which represent the range of possible decisions a report-to-track correlation algorithm can make: correct assignment, mis-assignment, fragmentation, and ambiguity. The numerical results from the simulation study of the algorithm itself show that it satisfactorily reproduces the input performance metrics. Finally, the object-oriented implementation outlined in this paper can also serve as the framework for additional data fusion algorithm models at other resolution levels.

REFERENCES

Davis, P. K., and R. K. Huber. 1992. "Variable-Resolution Combat Modeling: Motivation, Issues, and Principles." Rand Note N-3400-DARPA, Rand, Santa Monica, CA.

Fox, M. L., G. S. Mills, S. Thode, A. R. Crespo., and D. Mendiguchia. 1991. "Navy Tactical Command System-Afloat (NTCS-A) 1990-91 Correlator/ Tracker Evaluation." NRaD Technical Report 1475. Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA. (Nov.).

Kendall, Sir Maurice, Alan Stuart, and J. Keith Ord. 1983. The Advanced Theory of Statistics Volume 3: Design and Analysis, and Time Series, 4th ed. Macmillan Publishing Co. Inc., New York.

Tolchin, Martin. 1989. "Crucial Technologies: 22 Make the US List." New York Times, March 17, p. C1,3.

White, Franklin E. 1991. "Data Fusion Subpanel Report." In Technical Proceedings of the 1991 Joint Service Data Fusion Symposium, Vol. 1. (Applied Physics Laboratory, John Hopkins University, Laurel, MD, Oct. 7-11), Naval Air Development Center, Warminster, PA. 335-361.