Naval Research Laboratory

Washington, DC 20375-5000

NRL Memorandum Report 6479

2

Tests of Gating Algorithms for Tracking of Multiple Objects
I. Theory, Requirements, and Performance Measures

J.M. PICONE, M. ZUNIGA,* J. UHLMANN** AND J.P. BORIS

Center for Computational Physics Developments
Laboratory for Computational Physics and Fluid Dynamics (LCPFD)

**Science Applications International Corporation, McLean, VA 22102*

***Berkeley Research Associates, Springfield, VA 22150*

June 30, 1989

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6479	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b OFFICE SYMBOL *(If applicable)* Code 4440	7a NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* Washington, DC 20375-5000	7b ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SDIO 8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)* Washington, DC 20301-7100	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
63223C			DN155-097

11 TITLE *(Include Security Classification)*
Test of Gating Algorithms for Tracking of Multiple Objects-I. Theory, Requirements, and Performance Measures

12 PERSONAL AUTHOR(S)
Picone, J.M., Zuniga,* M. Uhlmann,** J. and Boris, J.P.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Interim	FROM 2/88 TO 2/89	1989 June 30	25

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Multiple - target tracking Gating Near-neighbor algorithms

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report provides the necessary foundation for tests of selected gating algorithms for multiple-target tracking. We investigate a nonhierarchical cluster algorithm and several near-neighbor algorithms. The discussion covers accuracy and scaling of cost with the number of objects being tracked. We estimate the scaling theoretically and discuss data set requirements and appropriate quantitative measures of performance.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL J.M.Picone	22b TELEPHONE *(Include Area Code)* (202) 767-6326	22c OFFICE SYMBOL Code 4440

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS

Summary

1. Define N to be the number of objects in a scenario. Our interest is in values as high as $10^4 < N \leq 10^5$. We have performed initial theoretical analysis of "gating" (selection of candidate track-report pairs).

   a. A straightforward brute force algorithm should scale as N^2.

   b. A nonhierarchical cluster algorithm used in multihypothesis tracking [1] should scale at best as $N^{3/2}$ and probably closer to N^2. The scaling of this algorithm depends primarily on the track and report data and on the scenario. Obtaining $N^{3/2}$ scaling is fortuitous and is *not* within the control of the programmer.

   c. The worst scaling part of the LCPFD near-neighbor algorithms should vary as $N \log N$.

2. The calculation of scores for pairings of tracks with reports is the most expensive part of gating for the brute force and cluster algorithms. Near-neighbor algorithms should reduce the number of score calculations considerably. This part of gating should increase linearly with N for near-neighbor algorithms, as N^2 for brute force and between $N^{3/2}$ and N^2 for the cluster algorithm.

3. The two aspects which these tests will measure are scaling of the cost with N and accuracy of the result. For accuracy two measures of performance will be used:

   a. $\mathcal{P}_1 \equiv$ the number of highest-valued pairs *actually found* divided by the number of highest-valued pairs which *should have been found.*

   b. $\mathcal{P}_2 \equiv \mathcal{P}_1 \times$ the sum of scores of pairs actually found divided by the sum of scores of pairs which should have been found.

4. To measure scaling with N, the normalized cost of each major step in the gating process for several values of N will be fit to the functions βN^α and $N(\alpha \log N + \beta)$, where α and β are constants to be evaluated, in order to determine the most likely functional dependence. We will express cost in terms of elapsed central processor time normalized by the time required for a given algorithm at the smallest value of N.

5. Data set size requirements differ for the measurement of scaling and accuracy. For *scaling*, N must vary at least from 100 to 10,000 and perhaps up to 60,000, since the systems of interest might contain as many as 10^4 to 10^5 objects. Data sets covering the range of values $100 \leq N \leq 1,000$ appear to be adequate for measuring accuracy.

6. For measuring accuracy of the algorithms and to cover the various situations which affect scaling of the cluster algorithm, the scenarios should cover a range of cluster sizes. Clusters should vary from an average size of 1 up to the maximum cluster size to be expected in a realistic situation. In addition, the data sets must include scenarios in which reports from each sensor scan interval have approximately the same time stamps and scenarios in which the opposite is true.

TESTS OF GATING ALGORITHMS FOR TRACKING OF MULTIPLE OBJECTS I. THEORY, REQUIREMENTS, AND PERFORMANCE MEASURES

I. Introduction

The purpose of this paper is to describe measures of performance and data set requirements for testing algorithms which identify and select candidate associations of tracks with sensor data. The act of finding these candidate associations is commonly called "gating" [2], and we shall use that term. For our purposes, the sensor data come in the form of "reports" on individual objects or on clusters of objects which cannot be resolved separately. Tracking algorithms treat such clusters as single objects until their constituents can be resolved.

Given the set of all possible pairings of an individual track with an individual report, a gating algorithm selects a subset containing the most likely pairings. Each pair selected must therefore have a probability of association that exceeds or equals some minimum value or "threshold." In this way, gating reduces the number of candidate pairs that must be considered in constructing tracks. Ultimately one hopes that this reduction will permit near-real-time data processing and tracking of the objects being observed.

Our purpose is to test various gating algorithms intended for tracking large numbers (10^4 to 10^5) of moving objects. The algorithms fall under three categories, discussed in the subsections below.

I.1. Brute Force

The "brute force" approach computes a likelihood (probability) or "score" for each possible pair and selects those pairs with scores that equal or exceed the threshold. This algorithm should find all such pairs and thus provides a baseline for determining accuracy of the other algorithms. The cost equation is

$$C_{bf} = N_T N_R (C_e + C_s + C_d) \quad . \tag{1.1}$$

In Eq.(1.1), N_T is the number of tracks, N_R is the number of reports received during a given time interval (sensor scan interval), C_e is the cost (in central processor unit seconds (CPU s)) of extrapolating a track state vector and covariance matrix to the time of a report, C_s is the cost of computing a likelihood or score for one track-report pair, and C_d is the cost of comparing two data, in this case the score for a track-report pair and the threshold score, in order to select the candidate pairs. Notice that Eq.(1.1) and similar cost relationships in Eq.(1.3) and Table 1 are schematic rather than exhaustive. Our intent is to reveal the underlying scaling of the various algorithms and to provide a basis for comparing the overall costs of corresponding steps in the algorithms.

Manuscript approved March 13, 1989.

In Eq.(1.1), assuming that N_T and N_R are both proportional to the actual number of objects N, C_{bf} scales as N^2. Given present computer hardware, the brute force approach is unacceptably expensive when $N \approx 10^4$ or 10^5. We note that the various per item costs might be far from equal and expect that

$$C_s > C_e > C_d \text{ (and possibly } C_s \gg C_e > C_d) \ , \tag{1.2}$$

since C_s requires many more operations than track extrapolation or simple data comparison. In some formulations, the score calculation could implicitly include the cost of track extrapolation.

I.2. Nonhierarchical Cluster Algorithm [1]

This algorithm uses spatial clusters that have been identified primarily for the purpose of reducing the number of different hypotheses which must be followed in a multiple hypothesis tracking (MHT) algorithm. The approximate cost may be computed from the steps shown in Table 1. To reduce the scaling from brute force, one first averages the tracks within a cluster, so that the cluster itself may be treated as a "supertrack." The algorithm calculates a score for each cluster-report pairing. The extrapolation of the cluster (supertrack) to the time of the report is implicit in the algorithm. The scores are then compared to the minimum acceptable value or threshold. For each cluster with an acceptable score, the algorithm then computes a score for each constituent track and again compares the score to the threshold to find the candidate track-report pairs. In Table 1, C_a is the average cost of computing a cluster averaged "supertrack", N_C is the number of clusters, N_C' is the number of selected clusters (usually 1 or so), and N_{TC} is the average number of tracks per cluster. Notice that if each cluster contains one track, $N_C = N_T$ and the scaling of the cost is $N_T N_R$ as in the brute force approach. However, if some clusters contain more than one track, then $N_C < N_T$ and the scaling might be better than brute force. In fact the Appendix shows that the optimum case occurs when there are approximately $N_C = \sqrt{N_T}$ clusters, each containing on average $\sqrt{N_T}$ tracks, giving a cost of

$$\mathcal{C}_c(\text{min}) \approx 2(C_e + C_s + C_d) N_R \sqrt{N_T} \ . \tag{1.3}$$

Thus, this algorithm scales, at best, as approximately $N^{3/2}$ and at worst, as N^2. The scaling depends primarily on the track and report data and on the scenario. Obtaining $N^{3/2}$ scaling is fortuitous and not within the control of the programmer. As indicated at the beginning of this subsection, clusters are important in the generation and maintenance of hypotheses. The case of $N_C \approx \sqrt{N_T}$ and $N_{TC} \approx \sqrt{N_T}$, while optimum for gating, is not optimum for hypothesis generation and maintenance, since the number of potential hypotheses (or scenes) in a cluster varies as $(N_{TC})!$ (factorial). In addition, if $\sqrt{N}$ exceeds the maximum cluster size in a realistic situation, the optimum case for gating with this algorithm will not occur.

Finally, this "cluster" algorithm should not be confused with other cluster algorithms that one finds in the computational literature [3, 4]. The latter are hierarchical and scale as $N \log N$. For this reason, we view the above nonhierarchical cluster algorithm as a sophisticated brute force algorithm worthy of consideration because of its simplicity and potential improvement over conventional brute force techniques.

I.3. Near-Neighbor Algorithms [5].

Table 1 shows the costs of steps used in a typical near-neighbor algorithm for gating when the reports correspond to different observation times within some finite time interval (scan). The quantities A_n and A_s are constant coefficients of order 1. The example here is that of projecting the tracks forward to M_n different times t_n within the time interval, where M_n is a small number (≈ 5) [6]. For each report, the algorithm identifies the closest data structure in time and then searches it for all tracks falling within some radius of the position of the report. The search radius depends on the time t_R of the report, the time t_n corresponding to the data structure selected, the track velocities, and the threshold score. Upon finding a small number (denoted N_n) of candidate tracks, the algorithm computes scores and makes the final gating decisions. Notice that in theory this algorithm computes far fewer scores than the cluster algorithm, and thus minimizes the most costly calculations. Setting up the data structures scales as $N_T \log N_T$, but the costs per operation are very small relative to C_s.

In summary, the worst scaling part should be superior in scaling to the brute force and cluster approaches. The near-neighbor approach offers the additional advantage that the poorest scaling step does not involve a score calculation. These are the features that have led to massive savings in molecular dynamics calculations and have been the main selling points for the present application.

I.4. Further Comments

The above discussion indicates the following potential advantages for near-neighbor algorithms:

(1) The inherent scaling of the near-neighbor algorithms should be superior by a factor of $N/\log N$ relative to brute force and by a factor of at least $\sqrt{N}/\log N$ over the cluster algorithm and perhaps closer to $N/\log N$.

TABLE I
Cost of Steps in Association

	Cluster	Cost	Near-Neighbor	Cost
1.	Compute cluster averages ($t = t_{s0}$)	$N_T C_a$	Construct data structure ($t = t_{s0}$)	$A_n C_d\ N_T \ln N_T$
2.	(Extrapolate clusters)*	$N_C C_e N_R$	e.g., Extrapolate data structure M_n times †	$M_n N_T \times$ $(C'_e + A_n C_d \ln N_T)$
3.a.	Compute scores	$N_C C_s N_R$	Find candidate tracks	$A_s (\ln N_T) N_R$
b.	Select clusters	$N_C C_d N_R$		
4.a.	Compute track scores	$N'_C N_{TC} \times$ $(C_s + C_e) N_R$	Compute track scores	$N_n C_s N_R$
b.	Select tracks	$N'_C N_{TC} C_d N_R$	Select tracks	$N_n C_d N_R$

* Implicit in algorithm
† $C'_e < C_e$

(2) The most expensive calculation, that of gating scores, scales as N^2 for brute force and between $N^{3/2}$ and N^2 for the cluster algorithm. The same step scales only linearly with N in the current implementation of the near-neighbor algorithms.

(3) In fact, the number of score calculations *per report* remains relatively *constant* with N in the case of near-neighbor algorithms but *increases* as N and at least $\sqrt{N}$ *in the other cases.*

The purpose of the present tests are twofold:

(1) To determine the expense of the above algorithms in performing gating. This addresses directly the validity of the above theoretical analysis and the accompanying conclusions for systems containing large numbers of objects. In Section II, we consider scaling and actual cost separately.

(2) To determine the accuracy of the above algorithms in performing gating (see Section II.2). In other words, what percentage of the candidate pairs with scores above threshold do the algorithms find? An alternative form of this question is, "For each report, what percentage of the N_g or fewer highest-valued tracks with scores exceeding the threshold does the algorithm find?" Here N_g is the limit usually set by a gating algorithm on the number of tracks which may be associated with each report and depends on some constraint, such as the required speed or available computer memory. Another approach is to weight this percentage by the scores of the pairs found, so that missing lower-valued pairs is less important [7]. As indicated by the form of the second question, the possibility exists that, for a given report, fewer than N_g tracks have scores exceeding the threshold (Section II.2).

Two sources of data and algorithms are available for the tests. First, a realistic Tracker-Correlator (TRC) Code [1] using the above cluster algorithm can provide data on tracks and reports, and the portion of actual code dealing with the cluster algorithm could be part of the overall test package. The available TRC can comfortably handle sets of order 10 to 100 objects and not many more than 1000 objects. For that reason, the authors plan to conduct tests on the three algorithms using simpler, "simulated" tracks and reports numbering up to 6.4×10^4, which is more representative of possible scenarios. The authors plan to write the computer code for all three algorithms The tests will take place on a SUN workstation.

The next two sections of this paper deal with the data set requirements and the appropriate measures of performance for the tests. The fourth section identifies other important parameters and factors in the definition of data sets.

II. Data Requirements

As indicated above, the tests will address two criteria for selection of a gating algorithm: expense and accuracy. The data set requirements for determining these two features might be different. We will consider them in turn.

II.1. Cost Measurement

Cost or expense involves (1) the various costs per operation, denoted C_s, C_e, etc. in the previous section, and (2) the scaling of the algorithms with the number of objects N. Consider scaling first. Given reasonable computer programming techniques and within a computer hardware class, the scaling of the algorithms (or their major steps) with N should be reasonably independent of the particular machine used and of the quality of the actual programming. Thus, scaling with N is one aspect which these tests should determine with reasonable generality.

The actual cost in terms of CPU seconds, however, will also depend on the costs per operation of the particular computer code on the particular machine and will depend on the quality of programming. The measurement of CPU time is important when one is seeking the maximum speed of a given algorithm on a specific machine.

For the present tests, since absolute speed is not the goal of the overall project, scaling with N should be the most important aspect of cost to measure. To accomplish this requires performance data for different scenarios, individually containing different numbers of objects. To account for situations which are more favorable or less favorable to the cluster algorithm, the scenarios should also cover the range of clustering from an average size of 1 to $N_{TC}^{\max}$, the maximum number of objects in a cluster for a realistic situation.

Let N^- and N^+ denote the minimum and maximum values of N covered by the set of scenarios. What are the smallest acceptable values of these numbers? Given scenarios with similar distributions of clusters and cluster sizes, the minimum value N^-should be several times the maximum number of objects per cluster. Alternatively, N^-should be several times N_g, the maximum allowed number of tracks selected by the gating algorithm for each report. In either case, the gating algorithm should then eliminate a significant number tracks from consideration for pairing with any report and a trivial result will be avoided. In the scenarios presently envisioned, the maximum number of objects in a cluster will be of order 10, as will N_g. In that case, we expect that

$$N^- \geq 100 \quad . \qquad (2.1)$$

Now consider N^+. To determine scaling for applications to systems with large numbers of objects, say $N \geq 10^4$, N^+should at least be 10^4, so that extrapolation to the next power of 10 or so will be as believable as possible. In addition, one would like to use $N^+ \geq 100N^-$ to make sure that the scaling functionality is stable over an appreciable range of values for N. Equation (2.1) then gives us $N^+ \geq 10^4$.

We can also ask a related question about N^+: "For what value will the range (N^-, N^+) be large enough to discriminate the best possible scaling of the nonhierarchical cluster algorithm from the worst scaling claimed for the near-neighbor algorithms?" Thus we must distinguish $N^{3/2}$ from $N \log N$. At this point we cannot estimate the scatter in the data over the range of scenarios, but we can compute the differences which will arise for various values of $N^-$and N^+. As a measure, we compute the ratio of maximum to minimum costs for the two scaling functions and divide one by the other:

$$\mathcal{R}(N^-, N^+) = \frac{\sqrt{N^+}}{\sqrt{N^-}} \times \frac{\log N^-}{\log N^+} \quad . \tag{2.2}$$

Figure 1 depicts the meaning of $\mathcal{R}$ in terms of a graph of cost $\mathcal{C}$ versus N. To compare the two different algorithms, we normalize the respective cost values within the range (N^-, N^+) to the values at $N = N^-$ (see Eq.(3.9)). Then $\mathcal{C}$ starts out at a value of 1 in both cases. At $N = N^+$, the ratio of normalized costs for the two functions will be $\mathcal{R}(N^-, N^+)$. Thus $\mathcal{R}(N^-, N^+)$ gives the maximum deviation of the two functions. Now define $\mathcal{R}(N^-, N)$ to be the ratio of the normalized values of the two functions for any N. This we obtain by replacing $N^+$by N in Eq.(2.2). As N decreases in value from N^+, $\mathcal{R}(N^-, N)$ approaches 1, meaning that the differences between the functions become less easy to discriminate.

Now choose N^-= 100, the lowest value indicated in Eq.(2.1). For $N = N^+ = 1000$, we have $\mathcal{R} \approx 2$. Unless the scatter in the data are too large, this should be enough to distinguish ***that a scaling difference exists*** between the cluster and near-neighbor approaches. Since the differences decrease with N, the range of N from 100 to 1000 might be marginal for determining the ***actual scaling functionality*** of the two approaches. At $N = N^+ \equiv 10^4$, we obtain $\mathcal{R} = 5$, and for $N = N^+ \equiv 6 \times 10^4$, we obtain $\mathcal{R} = 10$. Thus to determine the actual scaling functionality, scenarios ranging from $N = 100$ to $N = 10^4$ or 6×10^4 seem to be much better.

II.2 Accuracy Determination

Now consider the question of accuracy, and define the ideal result as follows. For a given report r, Let N_{T_r} be the number of tracks with scores equal to or exceeding the

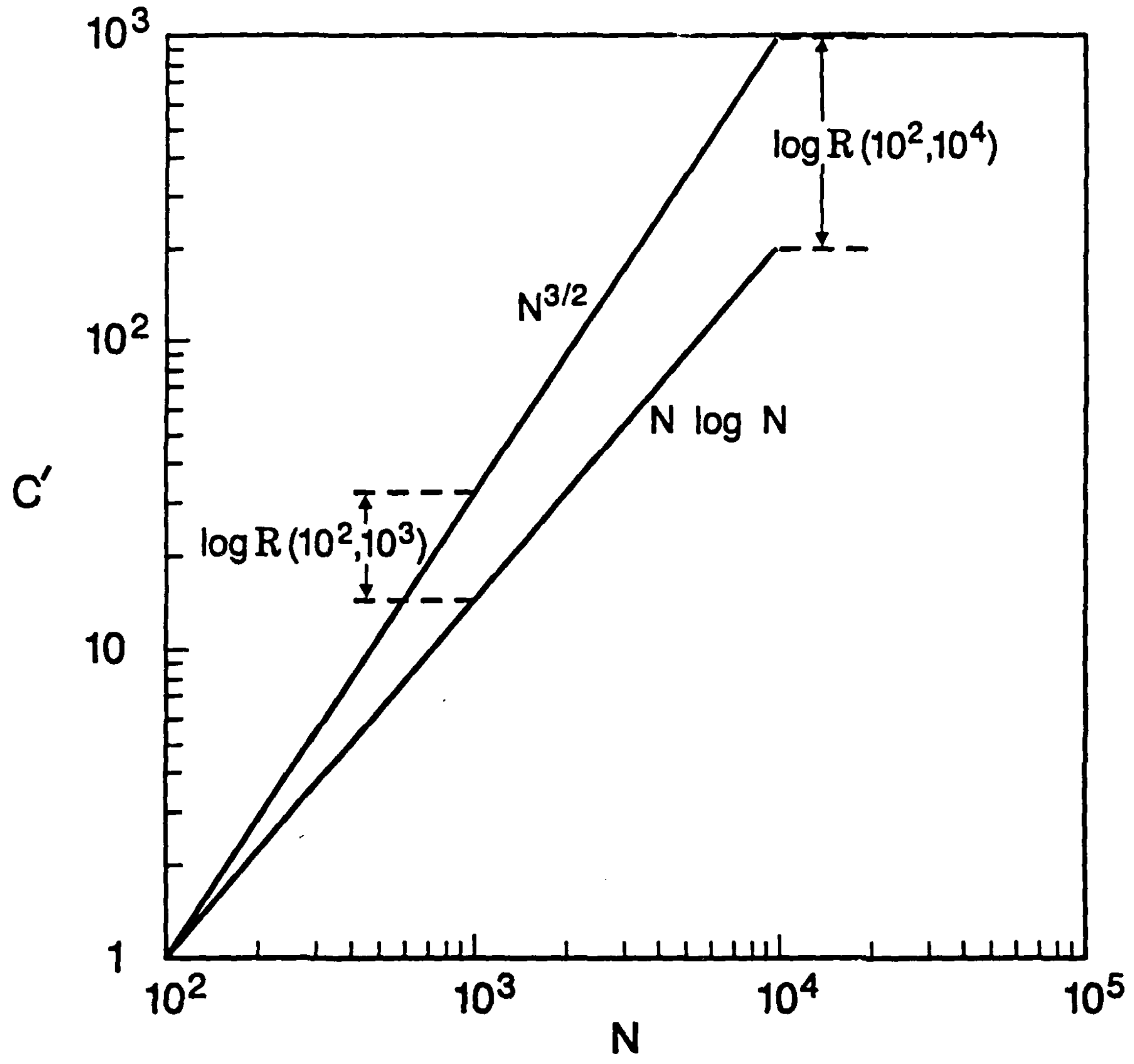

Figure 1. Log-Log plot of normalized cost Eq.(3.9) versus number of objects N for costs varying as $N^{3/2}$ and $N \log N$. Since the scales are logarithmic, the distances marked with arrows are proportional to the logarithms of the ratio $\mathcal{R}(N^-, N)$ of the two normalized cost functions for $N = 10^3$ and 10^4, respectively.

threshold. Remember also that the gating algorithm usually selects no more than N_g tracks per report. Then if $N_{T_r} < N_g$, the ideal result is to identify all N_{T_r} pairings of tracks with report r. If $N_{T_r} \geq N_g$, the ideal result is to identify the N_g highest-valued pairs. The performance of the various algorithms should be measured against the ideal result for each scenario considered. One might wish to vary the threshold to determine its effects on the process.

Once again, we desire a set of scenarios covering a range (N^-, N^+) of values for N, the number of objects. Equation (2.1) limits N^- to a value of 100 or greater to ensure a nontrivial result. However, for the determination of accuracy, a value of $N^+ = 6 \times 10^4$ is potentially too high. The two most important factors in accuracy appear to be the degree of clustering inherent in each scenario and the variation of observation times (time "stamps") of the reports gathered during each sensor scan interval. A scenario with little spatial clustering of the moving objects (separations exceeding sensor resolution) and in which new batches of reports have approximately the same time stamps might be better for near-neighbor algorithms than a scenario having the opposite properties (see Section IV.2). We expect these factors to have less of an influence on the accuracy of the cluster algorithm.

Of the two factors, only clustering should affect the value of N^+. To cover the range of possibilities, each scenario should contain clusters with an average size somewhere in the range from 1 to $N_{TC}^{\max}$, the maximum cluster size to be expected in any scenario. To insure a nontrivial result, many more than one cluster should be present. This leads us to propose that N^+ be greater than $N_{TC}^{\max}$ by a factor of at least 10. In addition, to cover a reasonable range of values for N, N^+should be greater than N^- by a factor a least 10. Since $N^- \geq 100$ and $N_{TC}^{\max}$ is of order 10, these considerations indicate that $N^+ \geq 1000$. Thus scenarios covering the range from 100 to 1000 objects should be useful for determining accuracy of the algorithms.

II.3. Summary of Data Requirements

For measuring expense of the algorithms, the tests should emphasize scaling with the number of objects N. The data sets should consist of various scenarios covering a range $100 \leq N \leq 6 \times 10^4$. The data sets currently within the capability of the realistic correlator-tracker code will not satisfy these requirements at the upper end of the range and could be marginal for determining scaling.

For measuring accuracy of the algorithms and to cover the situations which affect the scaling of the cluster algorithm, the scenarios should cover a range in average cluster size from a value of 1 up to the maximum size expected in realistic situations. Data sets covering $100 \leq N \leq 1000$ should be useful and perhaps adequate for measuring accuracy. In addition, the data sets must include scenarios in which reports from each sensor scan interval have approximately the same time stamps and scenarios in which the opposite is true.

III. Measures of Performance

III.1 Accuracy

We now identify measures of performance which seem to be appropriate for these tests. For accuracy the baseline will be the "ideal result" defined in the previous section. Given that the classic brute force approach computes a score for each possible pair and compares the scores to the threshold value, the result of the brute force algorithm for each report will be identical with the ideal result. Denote by $\mathcal{I}$ the set of all track-report pairs constituting the ideal result for the entire set of reports. Then if $\mathcal{A}_b$ is the set of pairs identified by the brute force algorithm, we have $\mathcal{I} = \mathcal{A}_b$. We must measure the intersection of $\mathcal{I}$ with the sets $\mathcal{A}_c$ and $\mathcal{A}_n$ of pairs actually found by the cluster and near-neighbor algorithms, respectively. Thus the performance $\mathcal{P}$ of the near-neighbor algorithm relative to accuracy is

$$\mathcal{P} \equiv \frac{\mathcal{M}(\mathcal{A}_n \cap \mathcal{I})}{\mathcal{M}(\mathcal{I})} \quad , \tag{3.1}$$

where $\mathcal{M}$ denotes a measure function. A similar equation holds for the cluster algorithm. Two performance measures have been mentioned previously [7] but not specified in an equation:

(1) Define $\mathcal{M}_1(S)$ to be the number of elements in a set S. Compute the first performance measure $\mathcal{P}_1$ as the percentage of pairs in $\mathcal{I}$ which are also contained in the set $\mathcal{A}$ of pairs found by a near-neighbor or cluster algorithm,

$$\mathcal{P}_1 \equiv \frac{\mathcal{M}_1(\mathcal{I} \cap \mathcal{A})}{\mathcal{M}_1(\mathcal{I})} \quad . \tag{3.2}$$

(2) For a set S, define $\mathcal{M}_2(S)$ as the product of $\mathcal{M}_1(S)$ and the sum of scores in S. This gives us a second performance measure, equivalent to multiplying $\mathcal{P}_1$ by the ratio of the sums of scores for the elements in the sets $\mathcal{A} \cap \mathcal{I}$ and the ideal set $\mathcal{I}$,

$$\mathcal{P}_2 \equiv \mathcal{P}_1 \times \frac{\sum_{p \in \mathcal{A} \cap \mathcal{I}} S_p}{\sum_{p \in \mathcal{I}} S_p} \quad . \tag{3.3}$$

In Eq.(3.3), the score or likelihood for the pth pair is S_p. The second measure $\mathcal{P}_2$ has the advantage that the penalty for missing a lower-valued pair in $\mathcal{I}$ is less than for missing a higher-valued pair. Thus the relative importance of the various pairs comes into the measure of performance.

A variation on the above approach is possible. Denote by $\mathcal{I}_r$ the set of track-report pairs in the ideal result *for a given report r*:

$$\mathcal{I}_r = \left\{ (t,r) \mid t \in \mathcal{T} \text{ and } (t,r) \in \mathcal{I} \right\} \quad , \tag{3.4}$$

where $\mathcal{T}$ is the set of all tracks. Thus $\mathcal{I}_r$ is the ideal result for a specific report r, and we have

$$\mathcal{I} = \bigcup_r \mathcal{I}_r \quad . \tag{3.5a}$$

Similar equations hold for the sets $\mathcal{A}_{cr}$ and $\mathcal{A}_{nr}$ of tracks identified by the cluster and near-neighbor gating algorithms for a report r:

$$\mathcal{A}_c = \bigcup_r \mathcal{A}_{cr} \tag{3.5b}$$

$$\mathcal{A}_n = \bigcup_r \mathcal{A}_{nr} \tag{3.5c}$$

We can now compute the above performance measures separately for each report r and average them over all reports. For report r and the set $\mathcal{A}_r$ of tracks identified by the cluster or near-neighbor gating algorithm, define the performance measures to be

$$\mathcal{P}_{r1} \equiv \frac{\mathcal{M}_1(\mathcal{I}_r \cap \mathcal{A}_r)}{\mathcal{M}_1(\mathcal{I}_r)} \tag{3.6}$$

and

$$\mathcal{P}_{r2} \equiv \mathcal{P}_{r1} \times \frac{\sum_{p \in \mathcal{A}_r \cap \mathcal{I}_r} S_p}{\sum_{p \in \mathcal{I}_r} S_p} \quad . \tag{3.7}$$

Then the overall performance measures corresponding to $\mathcal{P}_1$ and $\mathcal{P}_2$ are the averages of $\mathcal{P}_{r1}$ and $\mathcal{P}_{r2}$ over all reports:

$$\mathcal{P}_i' = \frac{1}{N_R}\sum_{r=1}^{N_R}\mathcal{P}_{ri}, \quad i = 1,\ 2. \tag{3.8}$$

Experience indicates that the measures averaged over reports $\mathcal{P}_1'$ and $\mathcal{P}_2'$ might deviate from the value 1.0 more than the measures for the total set of reports $\mathcal{P}_1$ and $\mathcal{P}_2$. However, the corresponding measures should be quite similar. Note also that Eqs.(3.6) and (3.7) assume that $\mathcal{I}_r \neq \emptyset$, the null set.

III.2. Scaling

For scaling we must compute the cost of each step in the gating process, as identified in Table 1, for each scenario. The set of scenarios will cover a range of values $(N^-,\, N^+)$ of the number of objects N. The most convenient unit of cost to measure is central processor unit (CPU) seconds. By treating each step of gating separately, we will be able to isolate the portions of the gating process with different scaling with N, as indicated by Table 1. To eliminate programming differences between the various candidate algorithms, we will normalize the cost to the value at $N = N^-$:

$$\mathcal{C}'(N) \equiv \frac{\mathcal{C}(N)}{\mathcal{C}(N^-)} \ . \tag{3.9}$$

Thus the lowest value of $\mathcal{C}'$ for each of the algorithms will be 1.

Once sufficient cases have been run to obtain costs at five or more values of N in the range $(N^-,\, N^+)$, we will fit the functions $\log(\mathcal{C}_c')$ (cluster algorithm) and $\log(\mathcal{C}_n')$ (near-neighbor algorithm) to the function $f(N) = \alpha\log(N) + \beta$. The quantity α is the slope of the curve and measures the power of N with which a step varies. Thus, α should be 2 for the brute force algorithm and fall between 2 and 3/2 for the cluster algorithm. For those steps which might scale as $N\log N$, we will also fit the function $\mathcal{C}'N^-/N$ to $f(N)$ to determine whether the slope is constant and equal to $\frac{1}{\log(N^-)}$ over the range of values for N. The accuracy of the results will depend critically on the scatter in the data, which we cannot estimate at present.

IV. Further Considerations

Several additional factors could have measurable effects on the results, and taking them into account could increase the number of different scenarios necessary for the tests. Ancillary data from the tests might also be useful to designers of operational systems in the future. We will briefly discuss these ideas below.

IV.1 Types of Sensors

The two main types of sensors to be considered are active (radar) and passive (e.g., infrared (IR) or electro-optical (EO)). The radar data are three-dimensional while the passive data come in the form of bearings (two angles) and are effectively two-dimensional. The two types of data represent different programming problems for near-neighbor algorithms. We plan to concentrate on radar data at the outset and treat the IR problem in the second stage of the tests.

IV.2. Temporal Distribution of Data

Two possibilities for scenarios exist [5]. Either the observations by a sensor during a scan all correspond to approximately the same time (i.e., the scan period is short relative to the time scale on which the objects change position significantly), or the observations are distributed over a nonnegligible scan time. The former case is advantageous for near-neighbor algorithms, since the track file data structure must be set up only once per scan and the reports all have roughly the same time "stamp" as the tracks, making calculation of scores easier. The latter case is more difficult because the observation times are different and candidate tracks must be projected to each of those times to compute the respective scores. Table 1 presents a near-neighbor method under the assumption of the latter case. Presumably both situations should be addressed by the tests.

IV.3. Choice of Threshold Score

The threshold score for determining the acceptable track-report pairings will determine the average number of tracks selected by the gating algorithm per report. This will also determine the maximum number of hypotheses which could be created. As the threshold score increases, the accuracy of the near-neighbor algorithms should also increase, since the acceptable tracks must be closer to each report. In this case, we should reach the situation of having to investigate only the nearest one or two tracks for each report. The situation in which reports have different observation times should also be easier when the threshold score is higher.

IV.4. Number of Different Scenarios

The data set for each N should include a number of different scenarios for each value of the average cluster size and each representative sensor scan interval. In addition the data sets must cover several different values of N. We see in this the possibility of another "combinatorial explosion" and will have to determine the minimal number of data sets necessary during the course of the tests. The basis for this decision will likely be the scatter observed in the results during the initial testing phase.

IV.5. Coefficients of Scaling Factors

As indicated previously, the tests should emphasize scaling and not absolute speed. However, from the type of hardware used for the tests and the actual CPU time expenditures for the gating tests, one can often estimate speed on other systems. Thus, even if the software is not optimized and even though speed is hardware dependent, elapsed CPU time can be useful to system designers who wish to employ the algorithms that we have developed. While we will not present such data as part of the test results, we can use the data to estimate speed of near-neighbor algorithms on other machines, if so requested.

V. Acknowledgements

The a· .hors gratefully acknowledge funding support from the Strategic Defense Initiative Organization through the NRL Information Technology Division Battle Management Program, which is guided by Mr. M. S. McBurnett. We also greatly appreciate helpful discussions with Mr. J. H. Gardner and Mr. M. S. McBurnett of NRL, Dr. D. Kierstead of Daniel H. Wagner, Associates, and Dr. L. Filippelli of Ball Systems Engineering Division.

References

1. J. A. Erbacher and B. Belkin, *Tracker/Correlator Algorithm Design Document* (Ball Systems Engineering Division, Arlington, Virginia, 1988).

2. S. S. Blackman, *Multiple-Target Tracking with Radar Applications* (Artech House, Dedham, Massachusetts, 1986).

3. A. W. Appel, *SIAM J. Sci. Stat. Comp.* **6**(1), 85 (1985).

4. D. H. Porter and J. G. Jernigan, U. Minn. Supercomp. Inst. Report UMSI 86/59 (1986).

5. J. M. Picone, J. P. Boris, M. Zuniga, J. Uhlmann, and S. G. Lambrakos, "Near-Neighbor Algorithms for Processing Bearing Data," *NRL Memo. Rpt.*, in preparation (1989).

6. E. Hyman, S. Eidelman, M. Zuniga, and M. Redmon, "Numerical Modeling of Airblast, Second Year Annual Report," *SAIC Report No. 88/1779*, 3 (1988).

7. D. P. Kierstead, *Memorandum*, D. H. Wagner, Associates, October 14, 1988.

Appendix
Optimal Scaling of the Nonhierarchical Cluster Algorithm

This appendix deals in more detail with the costs of the nonhierarchical cluster algorithm and presents an approximate equation for the optimal scaling of cost with N, the number of objects in the scenario. First, the equations given in Step 4 of Table 1 require further explanation. The computation of scores for all tracks in the clusters that have been selected for association with report r incurs the following expense:

$$C_{4r} = (C_s + C_e) \sum_{\{c\}_r} N_{T_c} \quad . \tag{A.1}$$

Here C_{4r} is the cost of step 4 (Table 1) for report r; $\{c\}_r$ denotes the set of clusters which have scores that exceed the threshold for association with report r; and N_{T_c} is the number of tracks in cluster c. Section I defines the remaining quantities. We now define the bracket notation

$$\langle f_a \rangle \equiv \frac{1}{N_a} \sum_{\{a\}} f_a$$

for an average over a number N_a of quantities which are labelled by the set of indices $\{a\}$, *not* necessarily equal to $\{1, 2, \ldots, N_a\}$. Averages of different sets will use the same bracket notation, even though the sets may have different numbers of elements. Further, if the index set $\{a\}$ depends on another index i, i.e., $\{a\} = \{a\}_i$ then the right bracket will have a subscript i, in other words, $\langle\ \rangle_i$. If the set $\{a\} = \{1, 2, \ldots, N_a\}$, then we will use the usual notation for the sum and need not carry the subscript $_a$ within the brackets:

$$\langle f_a \rangle \equiv \langle f \rangle = \frac{1}{N_a} \sum_{a=1}^{N_a} f_a \quad .$$

The total cost of computing scores and determining which tracks should be associated with each report in the set of data is then

$$\begin{aligned} C_4 &= (C_s + C_e + C_d) \sum_{r=1}^{N_R} \sum_{\{c\}_r} N_{T_c} \\ &= (C_s + C_e + C_d) \sum_{r=1}^{N_R} N'_{c(r)} \left(\frac{1}{N'_{c(r)}} \right) \sum_{\{c\}_r} N_{T_c} \\ &\equiv (C_s + C_e + C_d) \sum_{r=1}^{N_R} N'_{c(r)} \langle N_{T_c} \rangle_r \\ &\equiv (C_s + C_e + C_d)\, N_R \langle N'_{c(r)} \langle N_{T_c} \rangle_r \rangle \end{aligned} \quad . \qquad (A.2)$$

The last line of Eq.(A.2) shows the average over reports of the product of two quantities: (a) the number of clusters selected for each report r and (b) the average over those particular clusters of the number of tracks in a cluster. The superscript $'$ differentiates the number of clusters selected for a report from the total number of clusters N_C. To obtain the relationship shown in Table 1, Step 4, one can assume either that the number of clusters exceeding the threshold score is approximately the same for each report or the number of tracks per cluster is approximately the same. Equation (A.2) then becomes

$$C_4 = N_R \langle N'_c \rangle \langle N_{Tc} \rangle \quad , \qquad (A.3)$$

in which we have left the subscript "r" off of the average quantities for convenience. For presentation of this equation in Table 1, we define $\langle N_{T_c} \rangle \equiv N_{TC}$ and $\langle N'_c \rangle \equiv N'_C$. From Table 1, we see that the total cost for the cluster algorithm is then

$$C_c = N_T C_a + N_C (C_e + C_s + C_d) N_R + N'_C N_{TC} (C_e + C_s + C_d) N_R \quad . \qquad (A.4)$$

Notice that N_{TC}, N_T, and N_C are related through the equation

$$N_{TC} = \frac{N_T}{N_C} \quad . \qquad (A.5)$$

To find an extremum in the cost relative to the number of clusters, one substitutes Eq.(A.5) into Eq.(A.4), computes the partial derivative of the result with respect to N_C and sets the result to zero to obtain

$$N'_C N_T = N_C^2 \quad . \qquad (A.6)$$

An additional partial derivative with respect to N_C is greater than zero when Eq.(A.6) holds, showing that the cost is minimized by Eq.(A.6). The resulting cost equation is

$$\begin{aligned} C_{c\,\min} &= N_T C_a + 2N_R\sqrt{N_T N'_C}(C_e + C_s + C_d) \\ &\approx 2N_R\sqrt{N_T}(C_e + C_s + C_d) \end{aligned} \qquad (A.7)$$

To obtain the second line of Eq.(A.7), we have assumed that the cluster averaging (Step 1, Table 1) costs far less than the score calculation and that $N'_C \approx 1$. Also, for large numbers of objects the term with the worst scaling should dominate.