Technical Report 1648
June 1994

Measures of Effectiveness for the Distributed Data Fusion Problem

V. Broman
J. Pack

Approved for Public Release; Distribution is Unlimited

Executive Summary

Objective

The objective of this work was to identify useful approaches for measuring the effectiveness of data fusion systems applicable to the case of a distributed processing architecture.

Approach

A literature search for related work dealing with measures of effectiveness (MOEs) was performed, followed by an analytical evaluation of their generalizability to the multi-target, multi-hypothesis, distributed architecture case.

Results

Most of the MOEs seen in previous work can be applied only with great caution here. A localization MOE based on target density functions with distinguishable areas of interest for each processing node can be applied directly. Certain entropy methods might be applicable if a coordinate system appropriate to the problem could be defined.

Contents

1 Introduction **1**

2 What part of the system is being evaluated? **1**

3 Single-target MOEs **2**

4 Multi-target MOEs **4**

4.1 Statistics based on the confusion matrix 5
4.2 Distinct tracks are not independent subproblems 6
4.3 Target density functions 8
4.4 Search area target value relation 9
4.4.1 System concept 9
4.4.2 Notations and definition 10
4.4.3 Lake bottom analogy 11
4.4.4 Graphic example 12
4.4.5 Computation of R and $L(A)$ 12
4.4.6 Recursive triangularization approach 13
4.4.7 Approach of tracing enclosing contours 14
4.5 Entropy measures 15

5 Distributed Data Fusion MOEs **16**

6 EVAL Program **19**

6.1 Computational strategy 19
6.2 Instructions for using EVAL 19
6.2.1 Input files 20
6.2.2 Output files 20
6.2.3 Input parameters 21

7 References **22**

1 Introduction

Data fusion software systems have been evaluated with a variety of Measures of Effectiveness (MOEs) in the past, many of them being *ad hoc* measures supported mainly by heuristic reasoning, applied to a sparse sample of the kinds of input data expected to be encountered in operational use. Often, the confidence that users have in a system is due not so much to the fact that it achieved passing scores from a battery of MOEs as the fact that, in running the system, the users usually obtained credible results, and the system's failures did not seem unreasonable. Part of the difficulty in formulating formal data fusion MOEs is due to "data fusion" not being one problem but a whole family of superficially similar problems. Each Data Fusion System (DFS) has its own unique user requirements and intended environmental conditions, all of which strongly affect the design process and the definition of success. As a consequence, the field of data fusion does not have a small toolbox of generally accepted and understood measures like "array gain," "maximum relative error," or "SPECmarks." Another part of the difficulty lies in the fast increasing complexity and generality of the fusion algorithms being introduced nowadays, as well as the increasing comprehensiveness of the fused world view to be displayed for the user. The increasing capacity of these systems must be tested by means of MOEs of greater generality and power than those used in the past.

This report surveys MOEs applicable to data fusion systems addressing Level 1 of the taxonomy of data fusion processes identified by the Joint Directors of Labs Data Fusion Subpanel, (reference 1), particularly as they apply to the detection, association, and localization subproblems, with less attention paid to the classification subproblem. These are presented starting with those applicable to the simplest fusion systems and ending with the most complex type, i.e., the multiple-target, multiple-sensor, multiple-hypothesis, distributed data fusion system. The applicability of each and its shortcomings are discussed, with the perspective oriented toward applications involving Navy ASW.

2 What part of the system is being evaluated?

Essentially all of the MOEs applied to sensor data fusion systems actually measure the effectiveness of the DFS software and the sensor hardware and software in combination, as they operate in a certain environment and target scenario. In order to evaluate the sensors and the DFS by themselves, the environment and target scenario are controlled, sampling them over an operationally significant range of values. It can be difficult to evaluate just the DFS *by itself* because the sample space of potential scenarios, environments, and sensor suites is so huge and complex.

Directly comparing two candidate DFSs can be hindered by the fact that different DFSs tend to make different assumptions about how much side information

will be included in the input streams and which preprocessing functions may or may not be required upstream of the DFS, i.e., by the sensors. This can make a competition between two DFSs difficult to design fairly, since reasonable default actions and default values have to be assigned for missing functions and parameters.

Isolating the performance of the DFS itself and answering questions such as: "How much benefit can we get by using multiple hypotheses?" or "How much more performance might be achieved by an optimal system?" can be approached by comparing the performance of the system under consideration against a related, but idealized, system. There does not seem to be any known performance bounds for tracker/correlators analogous to the Cramer-Rao bound, but an upper bound can be obtained by allowing the algorithm to cheat, e.g., by using access to information on the true data associations to avoid association errors, and by performing Target Motion Analysis with an asymptotically optimal estimation algorithm, and initializing any iterative estimation processes with the true value of the parameter. Lower bounds on performance might be obtained from runs of any simple, least effective (but still rational) algorithm, e.g., a nearest-neighbor tracker. Sometimes, this least effective algorithm can be implemented by simply configuring the DFS being evaluated to use minimal memory, prune maximally, etc.

3 Single-target MOEs

The well-known Probability of Detection, P_D, and the Probability of False Alarm, P_{FA}, measures seen in signal-processing textbooks assess the capability of the DFS to perform detection, i.e., to distinguish objects from noise, under various circumstances. The usual analysis tests a binary hypothesis: noise-only versus noise plus signal from a single target. Attempts to apply this analysis to multiple-target detectors have several problems. Aliasing effects like side-lobes, multi-path propagation, shadowing, and convergence zone propagation make it questionable whether an energy peak measured in a certain beam/bin is due to one target or several. This invalidates likelihood calculations based on hypothesizing a single target versus no targets. In practical operational systems, the detection decisions made are generally soft decisions, announced in several stages, e.g., data are labeled as threshold crossings, potential tracks, confirmed tracks, system tracks, and lost tracks.

An additional factor that complicates the analysis of detection in DFSs is that most detectors are expected to not only filter out noise peaks but also to eliminate clutter, where the definition of "clutter" can be vague but varies according to the application. A detector may or may not be expected to "detect" sea mounts, shrimp beds, ice features, shipping lanes, or consort vessels, for example. This creates a large gray area where detection is not clearly distinct from classification.

Even where the classification task being performed is clearly separated from detection, its analysis can often resemble the analysis of detection, simply because the critical decision being made is threat vs. non-threat, given that the target is already correctly detected. This was the type of classification requirement named, for example, in the SQQ-89 improvement program request for proposals, where errors were measured in terms of probability (or frequency) of false alert and the probability (or frequency) of missed threat. A similar type of binary analysis can be applied to the case where exact classification into one of many categories is required by measuring the probability of correct or incorrect classification, assuming that the question of which incorrect classification was assigned is not of primary importance.

Work at Paramax, (reference 2), has applied the concept of a Bayesian Percent Attribute Miss distance (BPAM) to classification, in the case where there is a Bayesian prior distribution available in the form of a complete Order of Battle, and where the fusion system describes the track attributes in the form of propositional bodies of evidence, e.g., Dempster/Shafer measures of belief, as in reference 3. In this case, they define a metric on the propositional bodies of evidence, and then define the BPAM MOE to be the distance between the actual output and a classification output that was both 100% correct and 100% certain. While the strong requirement of knowing the Order of Battle in advance can be worked around, the applicability of this approach is still limited by the assumption that for each track there *is* a true target classification. If we have to deal with multiple targets, and there is a possibility of tracks being impure, i.e., composed of data originating from more than one target, then defining a "true classification" for each track becomes problematic.

For the problem of localizing single targets, the direct analog of the BPAM MOE is that of "radial miss distance" and its one dimensional analog often used with passive sensors, "range error." A radial miss distance is simply the magnitude of the vector difference between the best estimate of target position reported by the DFS and the true target position. This MOE takes no account of the uncertainty attached to the localization estimate and is motivated by the idea that if a weapon were fired on that track it would have to be aimed at the best position estimate, regardless of its uncertainty, and therefore the probability of kill would directly depend on the distance between the aim point and the true target position. The accuracy of the velocity estimate can generally be assessed by supposing that the present track state is dead-reckoned ahead to some future time at which the radial miss distance is measured again. Then the increase in position error over time is attributable to the error in the velocity components of the track state.

Instead of Euclidean distances, some other distance metric can be used to measure miss distances. A range error MOE evaluates the magnitude of the down-range component of the vector difference between the aim-point and the true target position. This is appropriate for passive sensor systems where the down-range component generally dominates the error. A Mahalanobis Miss

Distance MOE simply weights the position error by the inverse covariance of the estimate provided by the tracker, i.e., if the estimated track state mean and covariance are $\hat{\mu}$ and $\hat{\Sigma}$ and the true state is μ, then the Mahalanobis Miss Distance has this value.

$$(\hat{\mu} - \mu)^T \Sigma^{-1} (\hat{\mu} - \mu)$$

If the track position is modeled as a Gaussian random variable, then the likelihood attributed by the tracker to the true position is a monotonic function (an exponential) of the Mahalanobis Miss Distance. Averaged over many repetitions, this MOE does not so much measure the strength of the tracking algorithm as the proper scaling of the error sensitivity analysis, since a conservative algorithm could simply play it safe by inflating all its covariance estimates so as to achieve small Mahalanobis Miss Distances. Hall and Linn, (reference 4), have shown the susceptibility of the Mahalanobis distance measure to degradation by noise.

4 Multi-target MOEs

A multitarget tracker outputs one or more tracks, each of which represents the data associated with a certain hypothesized target and contains a state probability density conditioned on that data. If the tracker does a good job of estimating target positions and motions, then it may be that the number of tracks produced by the tracker equals the number of targets present in the region of interest. If this is the case, then it may be possible to define a one-to-one correspondence between the tracks and the actual targets, based on nearest-neighbor matching. It often happens that the tracker produces more tracks than there are targets. This can occur if one set of measurements fails to be associated with other measurements originating from the same target. Another possibility is that the tracker produces fewer tracks than there are actual targets. This case might result from measurement sets emanating from different targets being erroneously combined to produce one track. In general, the situation can become very complex with several false associations and missed associations occurring in the same scenario.

The single-target MOEs sketched in the last section fail to generalize easily to the multiple-target case because of the difficulty of assigning a single target to correspond to a track that is impure, or of assigning a single track to correspond to a target from which the data set has been fragmented instead of fully associated. However, if the data association problem is unambiguous and easy, then the MOEs obtained from each track/target pair can be averaged together almost as if they were just independent repetitions of a single-target experiment. Obviously, if such averaging is contemplated, an MOE must be chosen for which averages have a useful meaning. For example, average Radial Miss Distance is unlikely to have any more than heuristic significance, because the position error distribution is likely to be different for each track. However,

if the relationship between Radial Miss Distance and "Probability of Kill" for a particular missile-delivered weapon is known, then averaging the Probability of Kill over the set of all tracks produces a meaningful overall Probability of Kill for the particular weapon, DFS, and scenario being considered. This could be a sensible application of a single-target MOE to a multi-target tracking problem, assuming no difficulty in the data association and little interaction between the tracks.

4.1 Statistics based on the confusion matrix

Several of the MOEs commonly used to assess performance of the data association function of a DFS are based on the "confusion matrix." The confusion matrix is constructed by grouping all the measurements processed by the DFS according to which track they were placed in and which target they truly originated from. The element in the i'th row and j'th column of the matrix, C_{ij}, is the count of measurements originating from target j and placed in track i. In order to obtain these counts, both the DFS output and a knowledge of the true associations for all the measurements must be available. If the data association process is perfectly successful, then the confusion matrix will be square and contain exactly one nonzero element in each row and column, and by permuting rows and columns one could obtain a diagonal confusion matrix.

The Track Purity (TP) MOE for track i is computed from the confusion matrix according to the following definition.

$$\mathrm{TP}_i = \frac{\max_j C_{ij}}{\sum_j C_{ij}}$$

The Correct Assignment Ratio (CAR) MOE for target j is analogously defined by this equation.

$$\mathrm{CAR}_j = \frac{\max_i C_{ij}}{\sum_i C_{ij}}$$

Weighted averages of Track Purity taken over all tracks (TPWA), or of Correct Assignment Ratio taken over all targets (CARWA), have a particularly convenient form if the weight given each track is the number of measurements put in that track, and if the weight given each target is the number of measurements originating from that target. Here are the resulting definitions of the MOE weighted averages.

$$\mathrm{TPWA} = \frac{\sum_i \max_j C_{ij}}{\sum_i \sum_j C_{ij}}$$

$$\mathrm{CARWA} = \frac{\sum_j \max_i C_{ij}}{\sum_j \sum_i C_{ij}}$$

Figure 1 shows an example confusion matrix generated at scan 90 by the NCCOSC RDT&E Division ADM algorithm working on the scenario designated

by S9. At this scan, nine tracks are generated in two clusters, labeled 2 and 4. There are eight targets in the scenario, but only seven have been detected up to this scan. Target 47 has not yet been detected.

From figure 1 we can see that all 22 detections in track 23 originated from the same target (target 41). The TP for this track is therefore 1.00 or 100%. For track 65, 19 of the 20 detections originated from the same target (target 2); therefore the track purity is 0.95 or 95%. Tracks 11, 77, 26, 13, 54, and 2 also have a track purity score of 1.00.

Target 32 has a CAR of 0.52, which is equal to the fraction 11/21. A total of 21 detections emanated from target 32, and eleven came from the same track. The correct assignment ratio for targets 1, 2, 3, 31, and 41 is 1.00 since all detections from any of these targets are found in the same track.

The higher the values of TP and CAR, the better the association performance is, in general. It should be pointed out, however, that both measures are important; one should not monitor only one measure. If the DFS were to place each detection in a different track, then TP would be 1.00 for each track. The CAR values, however, would be quite low. Conversely, if the DFS were to assign all detections to the same track, then the CAR values would be 1.00, but the track purity would be low.

The confusion matrix and the TP and CAR MOEs seem to be very specific to assessing the success of the data association process in a DFS, since only the association results are examined, but this apparent narrowness of focus is illusory. The information used by the DFS to make association decisions comes from the localization and classification processes, and vice versa, and each depends heavily on the others being successful. Since association decisions are so tightly coupled with localization and classification, they cannot really be evaluated separately. However, the confusion matrix does have separate value in post-mortem analysis of a DFS run, e.g., in blaming the bad localization in a particular track on its being impure, or in identifying targets from which the data set was badly fragmented.

4.2 Distinct tracks are not independent subproblems

Single-target localization MOEs do not measure the resolution of a multi-target tracker, that is to say, the ability to distinguish and separately track targets that are close together and might be confounded. This is practically important, for example, because of the need to be certain that nonhostile vessels near targets not be attacked by accident. Close-together tracks tend to interact because of the association ambiguities between them. On the other hand, a fused scene that identified the presence of threats clumped together in a certain region could well be a useful output, even if the exact number of threats or their precise relative positions might be uncertain. A perfect solution of the data association problem may or may not be critical to success in supporting the human decision-maker.

The capability of identifying cross-fixes by passive sensors, and of rejecting

```
          SCENARIO: S9                ALGORITHM:  ADM6

SCAN:  90     TIME (min): 32.000     Last Detection Itemid:  336

    Valid Track   Track
       Index      Number     Cluster
          1         23          2
          2         65          4
          3         11          4
          4         77          4
          5         26          4
          6         13          4
          7         44          4
          8         54          4
         11          2          4

 Search areas (km2):
    4.770    2.744   13.656   36.987   30.736 6189.205 3252.394

    CONFUSION MATRIX
   1   2   3  31  32  41  44  47 << targets
                                    TP
   0   0   0   0   0  22   0   0  1.000
   0  19   0   0   1   0   0   0  0.950
  33   0   0   0   0   0   0   0  1.000
   0   0  22   0   0   0   0   0  1.000
   0   0   0  22   0   0   0   0  1.000
   0   0   0   0   6   0   0   0  1.000
   0   0   0   0   3   0   7   0  0.700
   0   0   0   0  11   0   0   0  1.000
   0   0   0   0   0   0   1   0  1.000
Correct Assignment Ratios:
    1.00    1.00    1.00    0.00
1.00    1.00    0.52    0.88

Track Purity Weighted Average                0.972789
Correct Assignment Ratio Weighted Average    0.925170
```

Figure 1: Sample excerpt from cfm.dat.

ghost cross-fixes, is another example of a localization effectiveness issue that is not easily related to a single-target MOE like radial miss distance.

4.3 Target density functions

These shortcomings of MOEs designed for single-target trackers highlight the need for localization MOEs that deal with the fused track picture as a composite whole, not just as a list of independent tracks. The central concept applied to address this concern is the target density function, which measures the expected number of targets per unit area in each infinitesimal box in the surveillance region. Let the track-state probability density function for track i be written $p_i(x)$. Each $p_i(x)$ might be represented, say, as a convex combination of one or more Gaussian densities, or perhaps of one or more densities uniform over polygonal regions. (The usefulness of employing convex combinations of elementary density functions lies in the flexibility of being able to represent state densities that are multimodal or nearly singular.) Whatever the form in which $p_i(x)$ appears, the target density function, V, describing the whole scene, is the following sum over all i.

$$V(x) = \sum_i p_i(x) \cdot 1$$

This formula applies to nearest-neighbor trackers that maintain only one association hypothesis at a time. To generalize to the multi-hypothesis case, we need only sum over the entire list of all tracks in all association hypotheses, multiplying the state densities by the probability q_i that the track belongs to the true association hypothesis.

$$V(x) = \sum_i p_i(x) q_i$$

This track probability q_i is computable simply as the sum of the probabilities of all the hypotheses that contain this i'th track. As demonstrated in section 4.4, this target density function can be further generalized by multiplying each of the terms by some kind of cost or utility associated with each track, e.g., by the expected payoff of attacking whatever target underlies the track. The result would be a kind of expected payoff distribution.

Several statistics can be derived from a track density function that provide a composite view of localization effectiveness and that are appropriate for multi-target, multi-hypothesis DFSs. The simplest is just the integral of V^2, or the square of the two-norm of V. For a given fixed total number of targets, this norm measures the amount of overall accuracy to which the tracker *claims* to be localizing the targets; e.g., a collection of diffuse track state probability densities will yield a norm near zero, while densities approximating a combination of delta functions will yield a very large two-norm. Along with this inner product of V with itself, one would naturally think of calculating the inner product of V with

a sum of delta functions representing knowledge of the truth. Work at Wagner Associates, (reference 5), combined these two ideas (and tailored them slightly by inserting a weight function representing the limits of the region of interest) to obtain their Weighted Sum of Densities (WSOD) MOE. The WSOD MOE takes this form

$$WSOD = \sum_i f(x_i)V(x_i) - \frac{1}{2}\int_R f(x)V(x)^2 d\,x$$

where x_i is the true position of target i, and $f(x)$ is an appropriately normalized weight function supported on the region of interest. The meaning of the coefficient of one-half is not clear, as it allows the MOE to take on positive and negative values, and the value of zero does not correspond to some distinguished situation such as ignorance or perfect knowledge. Still, Schweiter and Stromquist in reference 5 claim that the WSOD is the "most satisfactory single MOE for evaluating tracker/correlators" of those they were familiar with.

4.4 Search area target value relation

The Search Area Target Value Relation (SATVAR) MOE is also based on a target density function, and its definition can be formulated in highly satisfying operational terms, so that we obtain a clearer picture of just how well the DFS is supporting the human decision-maker. Moreover, SATVAR evaluates the resolution as well as the accuracy of the tracker outputs, because of the oppositely signed payoffs attached to friendlies and hostiles.

Our approach in deriving the SATVAR MOE formula will be to model a greatly simplified combat system in order to count or weigh the number of threat targets that can be successfully attacked because of good tracking. This approach has been implemented in software, using both of the approximation algorithms described below, and it has been used for informal evaluations and for development work.

4.4.1 System concept

At one fixed point in time, we assume, the commander will be presented a tactical picture, consisting of tracks and data association hypotheses, with their probabilities. Attached to each track is current state information in the form of a 2-D positional state density, along with an estimate of the payoff (utility or value) expected for destroying a target represented by that pretrack. The commander then chooses an *optimal* targeting plan based on that picture and fires one salvo. We compare the plan against the true picture and add up the payoffs, positive and negative, attached to the objects the salvo would have actually hit.

If the value or utility measures assigned to the targets are not all equal, e.g., positive for a hostile target, negative for a friendly one, then we must group

the possible targets into classes of targets having equal value. The track-state information must include the probabilities assigned by the tracker (classifier) of the target belonging to each of these classes, so that we can compute as a weighted average the statistically expected payoff for destroying whichever target is associated with that pretrack.

The armament available to the commander can be characterized by a single scalar parameter, A, measured in units of area. We assume the commander is capable of choosing any (topologically open) region of ocean, R, with an area not greater than A, and when he fires the salvo, everything within the region R is surely destroyed. (Allowing for a probability of kill less than one would make the description of an optimal plan much more difficult.) The commander selects his targeting region R so as to destroy the most hostiles and the fewest friendlies possible, i.e., maximize his expected payoff. Usually this region would consist of a certain amount of area arranged around each of the most threat-like tracks.

Localization effectiveness is measured either by the payoff actually achieved by means of a certain area A or by the size of A required to achieve a certain payoff goal. Such measures seem to capture the intuitive concept of "localization" as the process of reducing the areas of uncertainty around tracked objects and of resolving objects of interest among clutter.

4.4.2 Notations and definition

Let $T_i, i = 1, \ldots, N$ denote all the tracks found in all the hypotheses present in the tactical picture output by the DFS. For each pretrack there is a probability q_i, which is the sum of the probabilities of the association hypotheses that contain T_i, as well as a state density function $p_i(x)$, which is at least upper semicontinuous, and an expected payoff v_i, representing the utility of destroying the target represented by T_i. If only one association hypothesis is considered, then $q_i = 1$ for all i, and N is the number of tracks in that hypothesis. More generally, $\sum_{i=1}^{N} q_i \cdot 1$ would be the expected number of targets detected. The $p_i(x)$ might commonly be 2D Gaussian densities or mixtures of the same. If the v_i are not all set to 1, then they should be scaled so that the commander would prefer not to attack a target for which $v_i < 0$.

Similarly, let $\tau_j, j = 1, \ldots, M$ be the true targets, with true positions x_j, and true payoffs u_j for destroying τ_j. A rational commander would have v_i and u_j consistent with each other in the obvious way.

The commander's targeting strategy depends entirely on the function

$$V(x) = \sum_{i=1}^{N} v_i q_i p_i(x)$$

which describes the expected payoff for sanitizing a unit area of ocean around x. The optimal targeting strategy will always be to choose a targeting region

R that blankets all the places where V takes on its highest values, i.e., he need only choose a nonnegative number r and define R thus.

$$R = \{x : V(x) > r\}$$

The value of r is chosen to be as low as possible, provided only that the area of R does not exceed A, the resource constraint on armament.

Given the strategy just outlined, we can define the localization MOE L to be this step function:

$$L(A(r)) = \sum_{x_j \in R(r)} u_j$$

for $A > 0$. This is the total payoff for destroying those targets found within the optimal targeting region R of area A. The function $L(A)$ will often be nondecreasing, but this property will fail when the sensor data are poor enough that for some j, $V(x_j) > 0$ while $u_j < 0$, i.e., a friendly sits in some targeting region.

Various scalar statistics could be extracted from this basic MOE. An example might be: the smallest value of A for which $L(A)$ is better than the payoff given for destroying 75% of the hostiles and 5% of the friendlies.

It should be admitted that an ambiguity arises with this definition if V has flat spots, i.e., if, for some $r > 0$, the region $\{x : V(x) = r\}$ has positive area. If this is the case, the optimal targeting strategy will not be uniquely determined for A in some intervals without making additional assumptions. This problem never occurs if only Gaussian or other analytic density functions are involved, which is the case we are interested in, but it would arise when evaluating a quadrilateral (or polytope) tracker.

4.4.3 Lake bottom analogy

One of the best ways to visualize the search payoff distribution, V, is to make an analogy to topographic maps. Suppose we imagine a lake that has interesting topographical features such as valleys and peaks beneath its surface. Let us further suppose that it is possible to drain water from the lake, the shape of the shoreline revealing these features at various water levels. If a target density function is plotted as the z coordinate on an x-y surface map, then the resulting three-dimensional surface is analogous to the lake bottom described above.

If a target's position is accurately known, then the corresponding peak in the probability density function is very steep and narrow. Since for one given target the total probability is unity, such a narrow peak is high. If the measurements pertaining to a given track are imprecise or the measurements from one target are confused with measurements from another, then the corresponding peak would have a gentle slope and would cover a large region of the lake bottom. Some trackers produce multi-modal estimates of target positions. For such position estimates, the probability of finding a target is not localized to a single

peak, but multiple peaks represent the likelihood of finding the target in various regions.

If the true target positions are known, these locations can be projected onto the lake bottom. First, suppose the lake is filled until all the peaks are covered with water. Now, when the lake is being drained, eventually the highest part of the highest peak will be uncovered. This is approximately the maximum likelihood estimate of the position of the DFS's best localized track. In general, this point will not correspond exactly with the actual position of a target due to measurement errors and the effect of the tails of the distributions around the other targets. When the lake has just been drained to the point of first uncovering a target position, then the two-dimensional area of the exposed islands represents the search area required to find that first target. (There is no guarantee that the detections primarily responsible for this peak in the surface actually originated from the first target uncovered, although that is highly likely.) If the level of the lake is lowered further, other peaks will become exposed and other targets will be revealed. As each target is exposed, the two-dimensional area of the exposed land represents the region that must be searched to find that target.

4.4.4 Graphic example

Figure 2 illustrates two level curves of V, which is a linear combination of three Gaussian density functions, with means located at points m_1, m_2, and m_3. The two levels correspond to two different values of the altitude, r and r'. Notice that m_1 lies only in the outermost region R above the altitude r', but not the region above the altitude r. The long triangle anchored to the point m is suggestive of an increment of area dA, which results from tracing along the level curve a distance dt.

4.4.5 Computation of R and $L(A)$

Picture for a moment the situation where V is a weighted sum of K Gaussian densities. Assume that V is positive somewhere; otherwise $L(A) = 0$ for all A and no further computation is needed. Now, if K were 1, then the payoff distribution V would be a simple Gaussian and R an elliptical region with area A bounded by a level curve of V. If V were a combination of Gaussians which were *well separated*, then R would be the union of no more than K approximately elliptical regions, and their total area corresponding to any given threshold r is easily computed in this case. However, if the Gaussians are not all well separated, problems arise in dealing with their interactions. How might this area computation be handled in the presence of more general interactions?

First, we *do* know the exact values of r connected with each of the jumps in the step function $L(A)$. L changes value instantaneously when the region R expands so as to include one of the target positions x_j. This occurs when

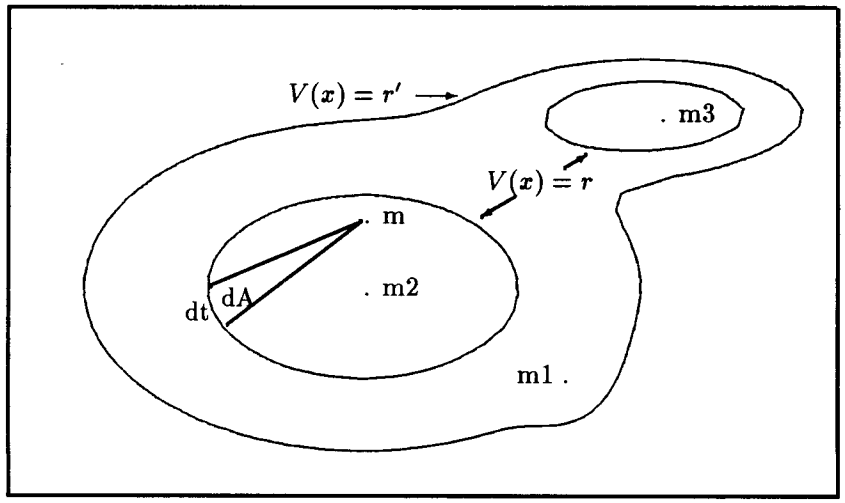

Figure 2: Payoff function level curves.

$r = V(x_j)$. So, evaluate $\phi_j = V(x_j)$ for each of the targets $j = 1, \ldots, M$. Sort the values $\{\phi_j\}$ into some nonincreasing order, i.e., obtain a permutation k of the set $\{1, \ldots, M\}$ such that $\phi_{k(1)} \geq \phi_{k(2)} \geq \cdots \geq \phi_{k(M)}$. Let $k(K')$ be the last subscript for which $\phi_{k(K')} > 0$. Notice that the targets $\tau_{k(j)}$ for $j > K'$ will never be attacked, because $V(x_{k(j)}) \leq 0$. We conclude that as A increases, $L(A)$ will take on the values $U_m = \sum_{j=1}^{m} u_{k(j)}$ for $m = 0, \ldots, K'$, in that order.

The remaining problem is to compute approximations to the values of A at which $L(A)$ jumps from one step to the next. Two approaches have been implemented in software to date: one intuitive and accurate, but potentially unstable, the other more brute-force and reliable, although slower to attain a given accuracy.

4.4.6 Recursive triangularization approach

The slower and more reliable algorithm for measuring the areas requires only that V be continuous instead of requiring continuous first partial derivatives. In this first approach, one merely partitions the surveillance region into small triangles. Then, when V is approximated by a linear function on each triangle, the level contours of V are straight lines, and the fraction of the triangle's area found above the level of each contour can be easily calculated by interpolation of the values of V at the vertices of the triangle. All the areas obtained from each small subtriangle are added up to obtain the total areas desired, without

any need to reconstruct the connections between the contour segments.

It is hard to estimate in advance the number of triangle subdivisions needed to ensure that the sum of the errors caused by the linear approximations will be acceptably small. We desire flat regions of V to be surveyed with large triangles, and regions of detailed variation in V to be finely subdivided into many triangles. Accordingly, we begin with two large triangles covering a rectangular surveillance region, and we recursively subdivide each triangle into four subtriangles by cuts that connect the midpoints of the sides of the triangle. We adaptively quit subdividing whenever a measure of nonflatness of V (approximating an average second derivative) falls below a certain threshold.

This approach has been implemented in software in Ada, by means of a set of packages that can easily be inserted into the source code of BayR-2 or the NRaD ADM. The heuristics controlling the depth of recursion succeed in keeping the amount of computation required within reasonable bounds.

4.4.7 Approach of tracing enclosing contours

The other area approximation method computes the sum of the areas of the connected components of the region R. The algorithm approaches each component by starting near a local maximum of V, e.g., at the mean m' of one of the Gaussian components with positive weight. If $V(m') <= r$, i.e., we are outside of R, then we search uphill until we either obtain a point m for which $V(m) > r$ or we discard the peak for lack of sufficient altitude. Now do a line search in any direction to find any point on the contour where $V(x) = r$. By integrating the following autonomous ordinary differential equations:

$$
\begin{aligned}
\frac{dx_1}{dt} &= \frac{V_{x_2}}{\sqrt{V_{x_1}^2 + V_{x_2}^2}} \\
\frac{dx_2}{dt} &= \frac{-V_{x_1}}{\sqrt{V_{x_1}^2 + V_{x_2}^2}} \\
\frac{dA}{dt} &= \frac{1}{2}\left(\frac{dx_1}{dt}\Delta x_2 - \frac{dx_2}{dt}\Delta x_1\right) \\
\frac{d\theta}{dt} &= \frac{2\,\frac{dA}{dt}}{\Delta x_1{}^2 + \Delta x_2{}^2}
\end{aligned}
$$

where

$$
\begin{aligned}
V_{x_1} &= \frac{\partial V}{\partial x_1} \\
V_{x_2} &= \frac{\partial V}{\partial x_2} \\
\Delta x_1 &= x_1 - m_1 \\
\Delta x_2 &= x_2 - m_2
\end{aligned}
$$

one can trace the contour circling m a single revolution and compute the area inside the region. The equation for θ may not be absolutely necessary, but it seems useful for checking the winding number of the curve traced. Finally, taking care not to duplicate regions, we repeat this area computation for each potential peak of V.

The practical problem with this approach lies in the numerical trouble that arises in the vicinity of a saddle point of V, because of the zero gradient there. A condition called "stiffness" arises in integrating the Ordinary Differential Equation (ODE), which is graphically discernible in that the contours converge into a bottleneck and fan out again as they approach and leave the neighborhood of the saddle point. This stiffness is occasionally so severe that the standard variable-order, variable-step predictor-corrector algorithm being used to integrate the ODE fails. In any case, stiffness greatly increases the computational effort required to retain stability in the integration algorithm. One possible work-around not attempted would be to approximate the location of the saddle point itself and then explicitly generate a Taylor expansion of V about that point, in order to approximate the contour with a hyperbola in these difficult regions. The software implementing this approach is sketched in section 6, which describes the EVAL program and its usage.

Under either of the approaches just sketched, the final output of the algorithm measuring localization effectiveness will then be the function L, as represented by the areas A_j for $j = 1, \ldots, K'$ and the payoffs U_m for $m = 0, \ldots, K'$. This function is computed from the tracking results available at the end of the scenario, at least, and in many cases is repeatedly computed at several intermediate time points of interest during the evolution of the scenario, in order to see when it is that problems arise or to estimate the length of tracking time needed to attain a certain targeting accuracy.

4.5 Entropy measures

In reference 2 and elsewhere, one takes the viewpoint that a DFS is like a communication channel and asks how much information is present in the output of the DFS, or else what is the difference in the amount of information present in the DFS output compared to the DFS input, or compared to a sure knowledge of the ground truth. Such questions are addressed by defining various entropy and cross-entropy measures.

For example, the Paramax information scoring metric in section 3.4.3 of reference 2 is essentially the probability density assigned by the DFS in its output to the ground truth state of the world. This is conceptually simple for a single-target tracker, say, that might only involve evaluating a Gaussian probability density function at a single point. The complexity arises in trying to define a distribution on whole tactical scenes, involving sets of an unknown number of tracks, each having various discrete and continuous state variables depending on each other. In full generality, Goodman's theory of random sets is required,

as developed in reference 6. Instead, Paramax's working implementation uses some pragmatic approximations, because of difficulties in treating the number of targets as a random variable, an area where the analysis in reference 7 might be of benefit.

While entropy methods seem to have great generality and wide applicability, they are all arbitrary or artificial in one certain respect: their value depends on the coordinate system used for describing the output of the DFS. The cause of this dependency lies essentially in the derivation of the entropy measures from probability arguments applied to a finite, discrete state space. The continuous state space actually used by the DFS gets discretized in a seemingly arbitrary way that depends on a coordinate system. It seems rather difficult to define or even imagine what coordinate system *most* naturally reflects the geometry of the data being handled, so that no objection could be raised that the coordinate system is producing artifacts in the results, e.g., the singularity at the origin of a polar coordinate system distorting the significance of results at short ranges.

5 Distributed Data Fusion MOEs

"Distributed" data fusion generally refers to data fusion processing in an architecture where there is no single central processor responsible for outputting the global scene, but rather the sensor data are processed at two or more nodes, no one of which processes all the data, with no single point of failure and no single communications bottleneck. Any one of the processing nodes might have its performance evaluated by means of the MOEs discussed above, just as if its fellow processors did not exist. The performance observed may or may not equal what would be expected from essentially the same algorithm being applied in a nondistributed architecture, because of how well or how poorly the algorithm might be dealing with problems of information feedback and errors introduced by communication. Besides some sort of Total Effectiveness, which is to be discussed last, there are two other main aspects of DFS performance that are peculiar to a distributed architecture: (1) communication requirements and (2) scene mismatch between processors.

The communication bandwidth required in a distributed architecture may well be measured in units like bits-per-second, but actually thinking in terms of bits may confuse the issue somewhat, if it leads to trading off complicated bit-packing schemes and compression methods and weighing whether the third decimal place is significant enough to be worth transmitting or not. This confusion is avoided by fixing some simple uncompressed format for communicated data and then measuring communications requirements in terms like tracks-per-hour or scenes-per-hour.

The main factors determining the communications requirements are the network topology and the reporting rates, which are interrelated of course. The topology determines, first of all, whether any feedback loops are potentially

present in the flow of sensor data through the network. If there are, then not only must the fusion algorithms be ready to take special account of these feedback effects, but the tracks transmitted from one node to another must be accompanied by a substantial amount of tracing information globally identifying the sources of all information, so that re-reported data can be effectively subtracted out. The network topology also determines such factors as the number of nodes that must produce output and the number of source nodes multiplexed together into each processor node receiving data. The reporting rates of the processing nodes depend on external interface requirements as well as on the amount of latency and scene mismatch between nodes that can be tolerated. Maurer, Oates, and Chrysostomou in reference 8 analyze a DFS that continually adjusts reporting rates in real time to meet some *a priori* scene consistency criteria. Two measures of consistency between a sensor's local scene and the DFS global scene are monitored, and when a threshold is crossed, consistency is improved by the sensor broadcasting its current local scene. In operational systems, though, it is more realistic to expect that the communications resources available will be fixed in advance and the DFS will make the best use it can of the bandwidth allotted to its use.

This view of communications bandwidth as a cost factor instead of an MOE is highly analogous to an attitude commonly assumed toward DFS requirements of computer memory and processing time. The computing resources available to an operational DFS are not flexible as in a big time-sharing system, with the option of buying more time on days with hard problems and saving one's money instead when there is nothing challenging to do. In an embedded DFS, the resources are fixed when the system is built, and the DFS must make the best use of the processor that is available. There is no benefit gained by leaving processor cycles unused.

Any mismatch between the scenes output by two processors in a distributed DFS could be an occasion for dissatisfaction in the human operators that monitor such outputs, even if there is no theoretical basis for complaint. Complaint *is* called for if two processors too often track the same target but output track states that differ by a statistically significant amount. It would be no surprise at all that the scene output by one processor contains tracks not present in the scene output by another processor, or vice versa. Since any two processors generally operate on two overlapping but unequal bodies of sensor data, they cannot be expected to produce identical outputs, only consistent outputs. Very possibly, each processor has its own defined region of interest, and it discards, or does not request, data not relevant to that region. If so, close comparisons between output scenes ought to be considered only inside the region that is of interest to both processors, if any.

The analysis in reference 8 examines two measures of scene dissimilarity, one comparing the data association hypotheses by means of a graph-theoretic metric, the other comparing the track state densities by means of a Kullback-Leibler measure, introduced in reference 9. Unfortunately, the Kullback-Leibler

measure has two major problems relative to distributed DFSs: (1) it is not a metric, i.e., it does not satisfy the triangle inequality, and (2) applying it to track state probability density functions or target density functions requires identifying a one-to-one correspondence between the tracks present in each output scene being compared. As already discussed above in section 4, there may not be any reasonable mapping that matches up a set of tracks against the set of true targets, much less one that matches up two sets of tracks that may or may not even be assembled out of the same pool of sensor data. The graph-theoretic metric discussed in the same reference does not have either of these problems, and it appears promising in general, especially since its value can be computed and updated recursively as the scenes being compared are updated. Nevertheless, it is unclear how well the value of this metric, which describes association hypothesis dissimilarity, can be related to more visible issues like track state inconsistency.

While some kind of Grand Total Measure of Effectiveness of an entire distributed DFS might seem desirable, it is questionable whether some such number as "average average track purity" would convey much meaningful information to the user. One approach combining MOEs over the nodes of a distributed DFS that could have promise is based on the idea of "regions of interest." One common reason for choosing a distributed DFS architecture might be that none of the processors in the system actually needs a global tactical picture; each is mainly concerned with some more local region assigned to it. These regions may overlap or be fuzzily bounded, but they serve as a means of focusing processing resources and of partitioning a huge global fusion problem into pieces of manageable size.

To make this concept more precise, define interest functions $f_j(x)$ for each DFS processing node j over all locations x in the entire surveillance region. These interest functions might be normalized so that $\sum_j f_j(x) = 1$ for all x, or else they could take nonnegative values measured in units like dollars per square kilometer. In any case, $f_j(x) = 0$ if node j is uninterested in location x. Assume that we have interest functions that are normalized and each processor evaluates the targeting payoff function $V_j(x)$ as in section 4.4.2 on the SATVAR. Then

$$V(x) = \sum_j f_j(x) V_j(x)$$

could usefully serve as a global search payoff function from which a global SATVAR MOE could be computed. This approach would not penalize a node for localizing poorly in areas it was not responsible for, but it could attach greater weight to the localization results obtained in areas of higher strategic importance or greater difficulty of search.

Partitioning the world up geographically is not the only feasible way to define an interest function. Some nodes could interest themselves only in aircraft, others only in neutral shipping, others only in ESM emitters. Any state variables

that are tracked by the trackers could be used to delineate regions of interest. The main requirement for this approach to be effective is that the nodes all be logically on the same level of the processing hierarchy. One node in the list, for instance, should not obtain all its inputs from some other nodes in the list, since the first node should be thought of as higher in the processing hierarchy than the other nodes it feeds off of. It should be possible to view the nodes being combined as peers.

6 EVAL Program

EVAL is a software program implemented in Ada that measures DFS effectiveness in two respects. It evaluates the statistics discussed in the section of this paper on the "confusion matrix," and it evaluates the SATVAR MOE described in the previous section using the contour tracing approach. The integration of the ODEs defining the contours and the areas employs an Ada implementation of the Shampine-Gordon algorithm, which is based on variable-step, variable-order Adams-Bashforth-Moulton integration formulas. Error returns from the Shampine-Gordon package that signal a problem with stiffness of the ODE system are simply resumed.

The target scenes evaluated by the software are obtained by parsing the output of the DFS in the "hyptrks" format defined in the TACSIM User's Guide, (reference 10). This ASCII DFS output format is sufficiently standardized that it is supported by the BayR2, ORCA, Lira, CMIDS, TRAPPR, MHT, and NRaD ADM systems.

6.1 Computational strategy

The following is an outline of the computational procedure:

```
Select a desired time for MOE calculation
For each true target position at the desired time
        Generate a contour line passing through this point.
        Check to see which peaks are encircled by this contour
        Store this information in an array
Arrange the targets in order of decreasing heights
For each ordered target
        For each peak not encircled by a target contour
                Generate a contour encircling the peak at this target height
                Compute the total area enclosed by all contours at this level
```

6.2 Instructions for using EVAL

EVAL must be executed in a directory containing the required input files. The following subsections contain descriptions of the input files, output files, and

interactively supplied input parameters.

6.2.1 Input files

The EVAL program uses three files as input. The following is a brief description of these files:

eval.dat This file contains the tracks, hypotheses, and clusters arranged to conform with the "hyptrks" format documented in reference 10.

true_data.dat The second file contains a description of the true scenario. It includes information on the positions of all the vessels in the scenario at the times of interest. Its format is that of the "auxstream" file in reference 10.

geo_input.dat Contains information pertaining to the placement and scaling of the graphical output.

Having data from both the truth and reports files, EVAL is able to provide an assessment of algorithm performance.

6.2.2 Output files

EVAL produces several output files. Some of these files can be printed directly, and some are designed to be processed by various graphing and plotting programs. The following is a brief summary of these files:

tek.dat This file contains commands for a Tektronix 4010 terminal or emulation program. The resulting plot is a contour plot of search regions required to find each of the targets present in the scenario.

gnu.dat This file is a list of x-y coordinate pairs of points in all the contours. It can be processed by Gnuplot to produce an on-screen plot or a printed plot, similar to the above mentioned Tektronix output, but realized on any of the wide variety of devices supported by Gnuplot.

gnu.cmd This file contains a script of Gnuplot commands to produce a PostScript plot very similar to that written to the tek.dat file. An example below demonstrates how to run Gnuplot to generate these plots.

map.dat This file contains a time history of the weighted track purity and the weighted correct assignment ratio parameters derived from the confusion matrices determined at each specified time.

brief.dat This is a text file summarizing the results of the EVAL run.

cfm.dat A text file containing a time history of confusion matrices and their associated parameters at each desired time.

loc_moe.dat This file contains a summary of the value of the SATVAR MOE computed by EVAL. This information can be fed into Quattro or a similar spread-sheet program to produce a graphical display of the search area MOE.

Two ways to generate a PostScript graphic of the search regions using the Gnuplot software are illustrated by the following terminal transcript.

```
% gnuplot
set terminal postscript
set output "gnu.ps"
plot "gnu.dat" with lines
quit
```

```
% gnuplot
set terminal postscript
load "gnu.cmd"
quit
```

6.2.3 Input parameters

Upon running EVAL, the user will be prompted to enter various parameters interactively. The following is a list of the input parameters:

loc_plot If loc_plot is 1, then a list of localization summary areas is produced.

plot_desired If plot_desired is 1, then a plot is produced.

merged_ellipses If merged_ellipses is 1, then the resulting plots will reflect the interaction of all the Gaussian peaks. If it is not 1, then each peak will be plotted as if it were the only peak in the scenario. In this mode it is possible that the displayed elliptical contours might overlap. If the truth file (true_data.dat) is not available, then this is a useful option.

tek_file If tek_file is 1, then the Tektronix plotting commands are stored in the file "tek.dat" for later processing. If the program is running in a tektool environment and tek_file is not 1, then the contours are plotted in the window in real-time. In this mode, interactive commands can be given to zoom in and out and change the map center.

first_scan The number of the first scan to be processed.

last_scan The number of the last scan to be processed.

pv_wave If pv_wave is 1, then a three-dimensional display generated by PV WAVE is sent to a window for viewing.

verbosity index This parameter controls the amount of data to be sent to the screen. The larger this integer, the more output is produced.

reporting_interval Output to the files cfm.dat and brief.dat will be generated for each scan that is a multiple of reporting_interval.

algorithm_code Gives a choice of algorithms. The chief purpose of this parameter is to ensure that the proper notation is affixed to the output files and plots.

Scenario_name The scenario abbreviation (one to three characters) is attached to various outputs as a means of identification.

7 References

[1] JDL Data Fusion Sub-panel. 1987. "Data Fusion Sub-Panel Charter and Functions," *Proceedings of the 1987 Tri-Service Data Fusion Symposium,* vol. I, Appendix A. Warminster, PA: Naval Air Development Center.

[2] Paramax Systems Corporation. 1992. "The Metrics Problem in Data Fusion," White Paper of 17 November 1992.

[3] Shafer, G. 1976. "A Mathematical Theory of Evidence," Princeton, N.J.: Princeton University Press.

[4] Hall, D. L. and R. J. Linn. 1992. "A Comparison of Association Metrics for Emitter Classification," pp. 447-461 in *Proceedings of the 1991 Tri-Service Data Fusion Symposium,* vol. I. Warminster, PA: Naval Air Development Center.

[5] Schweiter, G. A. and W. R. Stromquist. No date. "The Effect of Sensor Quality on Tracker/Correlator Performance," Daniel H. Wagner Associates report.

[6] Goodman, I. R. and H. T. Nguyen. 1985. "Uncertainty Models for Knowledge-Based Systems," North-Holland/Elsevier.

[7] Broman, V. 1989. "Counting Tracks and Counting Targets," *Technical Proceedings 1989 Tri-Service Data Fusion Symposium,* Warminster, PA: Naval Air Development Center.

[8] Maurer, D. E., K. L. Oates, and A. K. Chrysostomou. To appear. "An Architecture for Distributed Data Fusion: Measuring Tactical Picture Dissimilarity," *Proceedings of the 1993 Joint Service Data Fusion Symposium,* Warminster, PA: Naval Air Development Center.

[9] Kullback, S. 1978. "Information Theory and Statistics," Gloucester, MA: Peter Smith.

[10] Klausen, M. B. 1988. "A User's Guide to TACSIM – An ASW Tracking/Targeting Tactical Simulator," Draft of 4 December 1988 or later. San Diego, CA: Naval Command, Control, and Ocean Surveillance Center, RDT&E Div., code 572.

[11] Chong, C-Y., K-C. Chang, S. Mori, and D. S. Spain. 1987. "Tracking Air Targets by a Distributed Network of Acoustic Sensors," pp. 315-325 in *Technical Proceedings 1987 Tri-Service Data Fusion Symposium,* vol. 1. Warminster, PA: Naval Air Development Center.