89-3586-CP

IMPLEMENTATION OF A JPDAF TRACKER FOR AN ELECTRONICALLY SCANNED RADAR

Gary M. Stuart, Specialist Engineer
Robert I. Odom, Principal Engineer*
Frank D. Gorecki, Technology Manager*
Boeing Aerospace
Seattle, Washington

Abstract

Reliable tracking in a high clutter, dense target environment requires the utilization of a flexible tracking system capable of solving difficult track associations with minimized computational burden. To study this problem we have implemented, through simulation, a Joint Probabilistic Data Association Filter with an Electronically Scanned Array. This application is capable of handling large numbers of targets illuminated at arbitrarily spaced time intervals. We have developed efficient techniques for the implementation of the JPDAF algorithms, and have begun studying optimization techniques for data utilization and shared workload in order to realize the full potential of the ESA/JPDAF symbiosis. Since the actual JPDAF is not capable of track initiation, a separate adaptive, gated, line-fitting algorithm has been developed for this purpose.

Introduction

A classical and computationally simple approach to tracking is to assume that the report nearest to a given track, in some statistical sense, is actually from the source of that track. This "Nearest Neighbor" approach can lead to very poor track information in an environment where spurious reports occur frequently, or there are several targets very close together. The cause of this unreliability results from the failure of this algorithm to account for the stochastic nature of reports and the possibility that correlation with the "nearest neighbor" may not correct.

To more reliably operate in high clutter, dense target environments it is necessary to utilize techniques that are capable of determining the likely probability of a given report being associated with each of the existing tracks. To minimize computational burden, it is advantageous to utilize an algorithmic technique that performs valid track updates after processing each set of reports, instead of hypothesizing multiple possibilities for decision making in later update cycles. A tracking algorithm that meets these criteria is the Joint Probabilistic Data Association Filter (JPDAF)[1], the tracking algorithm utilized in our application. For a given track, each report in the set of possible associations is weighted to the probability of actually being the correct association. The centroidal value of these combined weightings is then used to update track position. This approach does allow the report with the highest probability to have the most impact on track update, but the inclusion of the other reports tends to minimize the impact of a spurious report that may be closest to a predicted track position.

Standard track-while-scan radars, at best, are capable of track update only once per scan. Since multiple scans are required for track initiation, a significant time lag can exist between first detection and track initiation. This time lag can be reduced by placing the detector on a steerable platform, but is still limited by the mechanical motion of the hardware. A significant improvement over these systems can be realized through the use of an Electronically Scanned Array (ESA). This detection system can be rapidly aimed in any direction within its operational field of view, allowing optimization of available resources. A coordinated beam steering algorithm is required to control this detector, but has the benefit of allowing the tracking system to selectively direct more attention to those targets it considers most important. By combining this flexible detector with the JPDAF, it is possible to obtain track initiation in less time, and to maintain more tracks in denser target scenarios, than with the other systems.

This paper presents our implementation of a JPDAF tracker, and its interaction with an ESA detection system. This includes discussions of the techniques utilized in implementing the necessary algorithms, process synchronization between the JPDAF and the ESA, cycle time as a function of the number of reports and cluster size, and indications of overall memory requirements. The theoretical background and performance analysis of this application is presented in a companion paper[2], and we therefore refer the reader to that paper for any detailed discussions of the algorithms.

System Overview

The tracking system consists of two primary functions, the system Controller and ESA, and the JPDAF. The Controller is a knowledge-based system responsible for optimizing the resources of the ESA to best meet the tracking requirements of the current threat environment. This *optimization* process balances detection time between updating existing tracks based on perceived threat levels, and the need to search the entire scan volume for new targets. The JPDAF is responsible for obtaining asynchronous target information from the ESA, filtering out noise and false alarms, initiating and updating target tracks, and providing an indication of track reliability.

The information provided by the JPDAF is used to drive a four-color tracking display, and to indicate to the Controller which tracks are most in need of new measurement data. The Controller balances these requests with its other scan requirements to direct the ESA to the next measurement location. The most effective mix of shared data between the Controller and the JPDAF, and the utilization of ESA resources is a continuing area of research.

The Controller and ESA will not be discussed in further detail except where necessary. The remainder of this paper is dedicated to a detailed description of the JPDAF implementation.

JPDAF Model & Implementation

The JPDAF is a sub-optimal "all-neighbors" Bayesian, target-oriented tracking algorithm, which has manageable computational demands, orderly bookkeeping, and is applicable to a dense target environment.

Conceptually, the actual JPDAF consists of only three main parts: 1) cluster determination, 2) hypothesis and probability generation, and 3) the filter itself. In addition to these there is the need for data input and output, and for track initiation and deletion. The combined unit of all these functions is what we loosely refer to as the JPDAF (see Figure 1). A general description of these primary functions is provided in the following sub-sections, in roughly the order that they occur in the application.

*Member AIAA

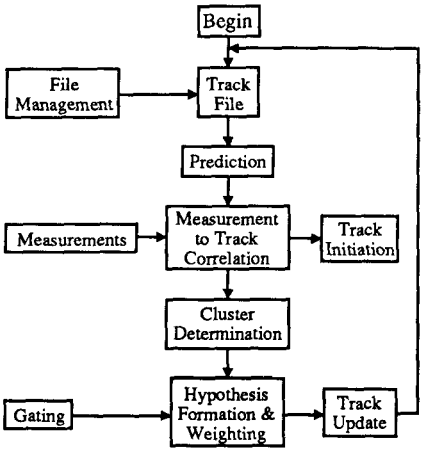


Figure 1. Flow Diagram of JPDAF

Report Input & Track Propagation

The update cycle begins by obtaining a set of reports from the ESA. A *set* of reports is defined to be a group of independent measurements detected within a time period of a few milliseconds, and can simply consist of all targets detected while the boresight of the ESA is momentarily fixed at a specific elevation and azimuth, or can be the combination of multiple boresight *looks*. Since the ESA can be arbitrarily pointed in any direction within its operational field of view, the resulting input is highly asynchronous.

Once the reports have been obtained, the track estimates are propagated to the detection time of the current set of reports. Since the track estimates, in our application, are maintained in the X, Y and Z (XYZ) Cartesian coordinate system, and the reports are in the Azimuth, Elevation, and Range (AER) modified spherical coordinate system, the track estimates are then converted to the AER coordinate system, in preparation for report/track association.

Report-to-Track Association

Report-to-Track association (see Figure 2) begins by using the estimated distance traveled and the expected measurement errors of the report to determine, for each report/track combination, the minimum distance the target must have traveled, in order for an association with the report to be considered valid. If this minimum distance criterion is not met, the measurements cannot be considered to be independent. The result is an unreliable update. Since $\sigma_v = \sigma_p / \Delta t$, where σ_v and σ_p are the velocity and position variances, respectively, and Δt is the sample interval, the minimum distance criterion is really a minimum time criterion.

If the minimum distance criterion is met, it is then necessary to prevent highly improbable report-to-track associations by ensuring that the residual error of the combination is within the measurement error of the ESA. This test is performed by calculating the square of the normalized weighted residual (innovation) vector between the report measurement and track estimate, and comparing it to a Chi Squared (χ^2) distribution gating value, as shown.

$$[\,y_j{}^t\,]^T\, S^{-1}\, y_j{}^t \le \chi^2 \text{ Threshold}$$

where $y_j{}^t$ is the measurement residual vector for measurement j and target t, and S is the measurement covariance. If the test value is less than the χ^2 threshold, the association is accepted, and the memory indexes for both the track and the report are placed in the Association List (AL). Figure 2 lists example values, with the Track indexes in the left column of the AL and report indexes in the right column.

Cluster Determination

If a specific report/track combination does not meet the distance criterion, it is not placed in the AL, and any rows in the AL already containing the corresponding report index are removed. Also, this report is disallowed for any further association testing for this data set. The reason is as follows. The report-to-track spacing was found to be very small, so even though the association *is* possibly valid, its use would result in an unreliable update. If this report was used in other associations, while excluding the close association, the resulting weighting values would incorrectly represent the conditions that exist. Therefore, this report can not be reliably used in any other associations that may exist.

When the association phase is complete, the unassociated reports (not including the disallowed reports) are sent to the track initiation routine, and the AL is utilized for cluster determination.

Since a given report may associate with multiple tracks, it is necessary to locate all interrelationships between the reports and tracks, and to separate these combinations into independent groupings of tracks and reports. These sets of related tracks and reports are generally subsets of all tracks and reports that exist at the time of a given processing cycle, and can be defined as a statistically distinct group of statistically related tracks and reports (see Figure 3). Each of these subsets is referred to as a Cluster, and is processed separately by the update logic.

Figure 3 shows an example of Clustering. The circles are 2-dimensional, ideal representations of the three-dimensional gating volumes around the estimated position of each track, and are marked by the Track number closest to them by T_i . The R_j values indicate the position of each of the reports obtained this cycle. The dashed lines indicate that there are three Clusters to be processed, and that Tracks T_{12} and T_{13} will not be updated since none of the reports can be

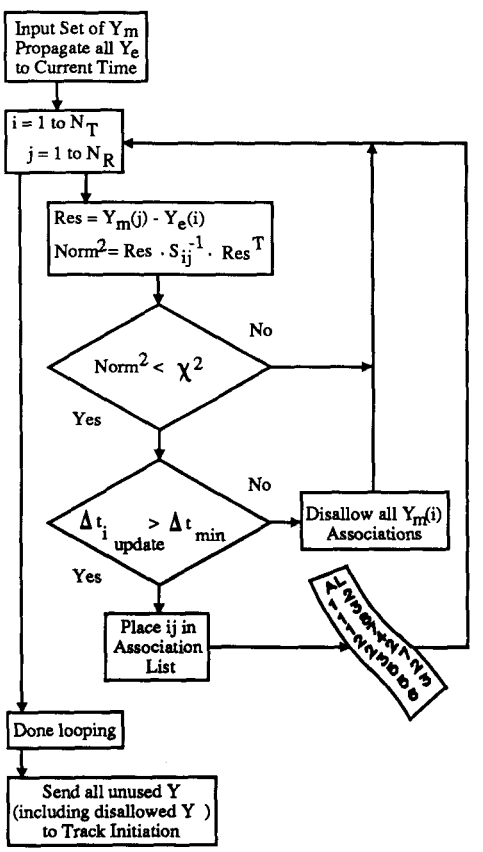


Figure 2. Report-to-Track Association Flow

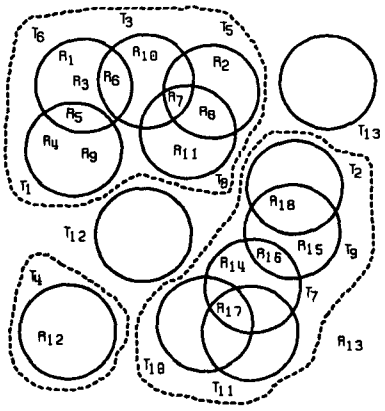


Figure 3. Example Cluster

statistically associated with them. Also, R_{13} will be placed in the track initialization file since it did not associate with any existing tracks.

The search for clusters (see Figure 4, {} will refer to values contained in Figure 4.) begins by obtaining the first track number (left column) {1} in the AL, and placing it in the Cluster Track List (CTL). The associated report (right column) {2} is then obtained and placed in the Cluster Report List (CRL). The AL is then searched for any other occurrences of that report number, and if found {tracks 3 & 5}, a pointer to that association is placed in a Stack.

The AL is then processed in a similar manner for each occurrence of the present track number, except that the track number is not repeatedly added to the CTL. Once all occurrences of the present track number have been processed, a new track number is obtained by referencing the track number in the AL that is indicated by the most recent association pointer on the Stack {track 6}. This entire cycle is then repeated for this new track number, with one exception: if any associated report number is that of one already processed, it is skipped. This then forms yet another larger cycle that is repeated for each pointer on the Stack, with one exception: if any track number is that of one already processed, it is skipped, and the Stack is checked for the next track number. When the Stack is found to be empty, the cluster is complete, and is sent on to be used for track update.

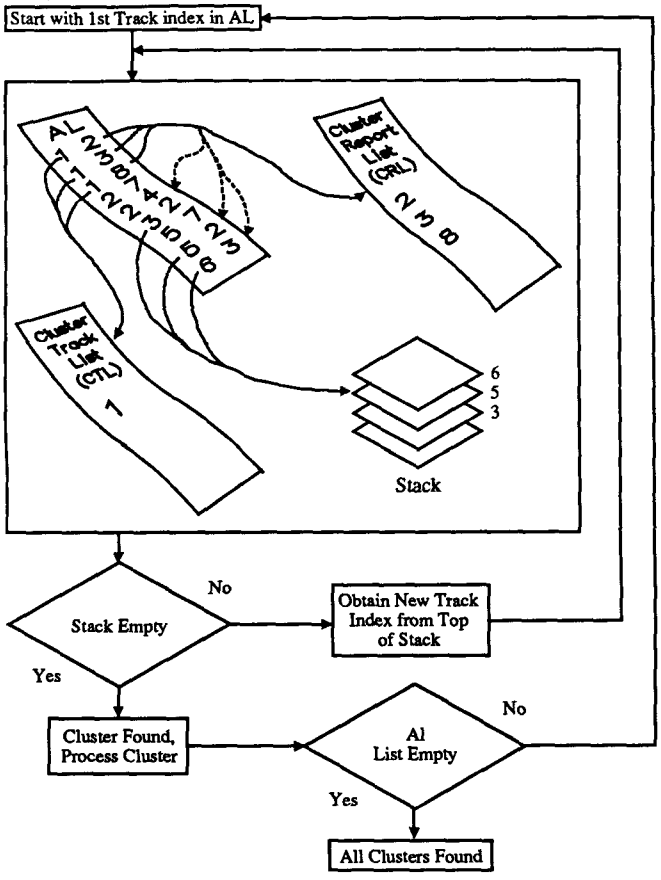


Figure 4. Cluster Determination Flow Diagram Using Association List of Figure 1.

After the cluster is processed by the track update routine, the AL is searched for any unused associations, and if one is found, the track number is obtained, the CTL and CRL are cleared, and the complete cluster determination cycle is repeated. It should be noted that the actual implementation of this technique includes the use of control flags that indicate a particular item has been processed, and a link-list containing information about the specific associations of the reports. Through the use of this information the actual amount of searching of the AL is substantially less than might be implied by the above functional description.

Once a Cluster has been determined, the CTL and CRL are cross-referenced to generate a gating matrix reflecting the valid Report/Track associations within the cluster. This information is utilized in the generation of the Hypotheses.

Hypothesis Generation

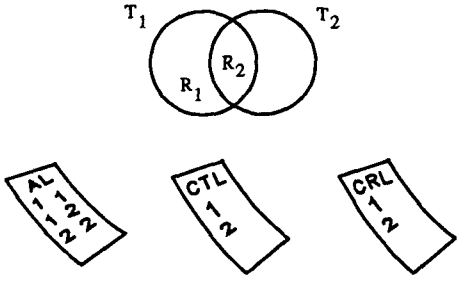
Figure 5 shows a simple cluster in which Report R_1 lies within the correlation gate for the predicted position of Track T_1, while Report R_2 lies within the correlation gates of both Tracks T_1 and T_2. This example decomposes into the three associations shown in the AL, which results in the creation of the Hypothesis Matrix shown below it. This matrix is based on the following two rules:

1) a report is the result of only one source

2) a target will create only one report

As a direct result of these rules, if a single target has more than one report associated with it, all but one must be a false alarm or new target.

Each row in the Hypothesis Matrix represents a possible hypothesis. The numbers below the 'Track' titles are the indexes of the reports, and indicate the valid associations between the reports and tracks. The '0' report index is used to represent the possibility that neither report



Hypothesis	Track 1	Track 2
1	0	0
2	0	2
3	1	0
4	1	2
5	2	0

Figure 5. Simple Cluster with Association Lists and Hypothesis Matrix

associates with that track, i.e. that the target may not have been actually detected, or that these reports did not originate from the previously detected target.

In the application, the actual generation of hypotheses (see Figure 6) is performed by a simple counting technique. If we were going to count from zero to 99, by one, we would set the counter to 00, and increment the right digit by one. When its value reached nine, the next increment would set this digit to zero, and the left digit would be incremented by one to the value of one. This process would continue until we reached 99. This example would be representative of a case where we had two tracks (two digits) and nine reports (counting in base ten), where the value of each digit represents the index of a report. As can be easily seen, counting from 0 to 99 would produce every combination of two tracks and nine reports, resulting in the generation of 100 hypotheses. However, since a single report can only be associated with one track, any combinations where the two digits are the same, such as 33, would be disallowed, since this would indicate that Report 3 was associated with both tracks. 00 is the only exception, since this indicates that none of the reports associate with any of the selected tracks, which is possible.

In general, the technique can be stated as follows: count by 1, starting at zero, with the numerical base being one greater than the number of reports. The right column starts at zero, increments by 1, and when it reaches the base value (three, for the example in Figure 5), increments the column to the left by 1, and starts over. When invalid combinations (this includes both rules stated earlier) are found, they are simply skipped. This technique continues until the time at which the next increment would cause the left-most column to be reset to zero.

Probability & Weighting Calculations

After determination of the valid hypotheses, the probability of occurrence for each hypothesis is computed, and normalized. Figure 7 presents representative normalized probability values for the example in Figure 5.

These hypotheses and corresponding weighting values are then summed in order to determine the probable relationship between each report and each track. Figure 8 shows the final normalized weights resulting from summing the probabilities for each valid report-to-track hypothesis for the example in Figure 7. It is this information that is utilized in the probabilistic update of each track. Also, it is important to remember that the hypothesis generation, probability weighting, and track updates are performed on a per cluster basis. This entire cycle is repeated for each cluster in the current process cycle before obtaining a new set of reports.

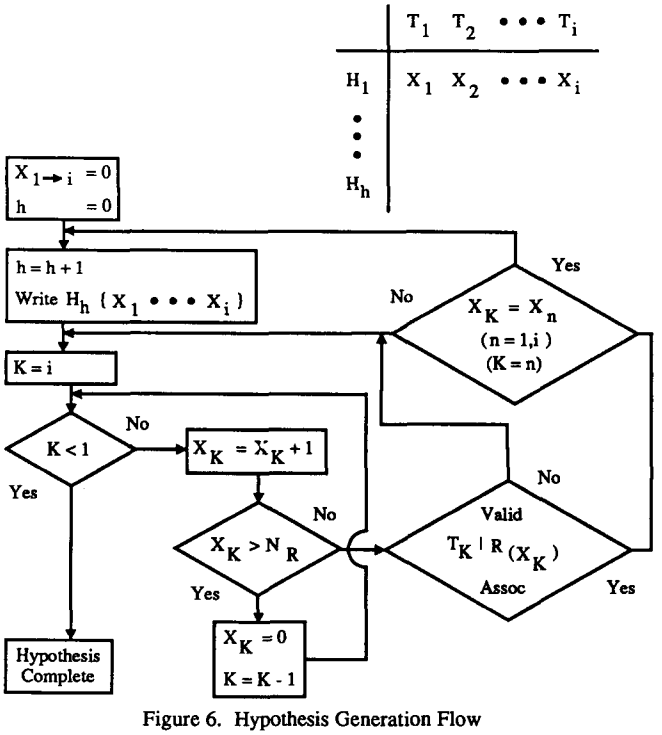


Figure 6. Hypothesis Generation Flow

The details of the probability and weight calculations are described in the companion paper (Odom, Stuart, & Gorecki).

Filter

The JPDAF is basically a Kalman filter with two differences: 1) A conditional mean is used for both the state estimate and the innovations vector in the state update equation, and 2) there are data dependent terms in the covariance update equation. These data dependent terms transform the deterministic Riccati equation for the covariance update of the standard Kalman filter into a stochastic differential equation. Also, the covariance equation depends on the reports, and therefore the estimation accuracy depends on the data that are actually encountered. What this means is that the JPDAF is a nonlinear filter. A detailed description of the primary equations describing the JPDAF is avialable in the references[1, 2, 3].

Track Initiation

Since the JPDAF algorithm requires a separate algorithm to provide track initiation, all unassociated reports (except those disallowed) are sent to this routine. The track initiation approach implemented is a multi-phase process, made up of the following steps: 1) Obtain and associate two reports with different report times (this is referred to as a potential track); 2) over multiple input cycles associate more reports with each potential track; 3) when it is determined that enough reports have been found, perform line fit of reports and specify heading and velocity (this is referred to as a tentative track); 4) when the tentative track has been updated n times through normal track update procedures, it is referred to as a confirmed track.

When a new report is obtained by this logic, one of three things will be done with it: 1) it will not associate with any existing reports stored within this track initiation logic, and will therefore be stored for comparison with reports of later input times, 2) it will associate and will be made part of a potential track, or 3) it will be found to be too closely spaced with a report in an existing potential track, and as was done in the association logic, this report will be ignored. For purposes of simple explanation, track initiation will be discussed in terms of a single track being detected and initiated.

Hypothesis	Track 1	Track 2	Weights
1	0	0	0.001
2	0	2	0.010
3	1	0	0.101
4	1	2	0.880
5	2	0	0.008

Figure 7. Hypothesis Matrix & Example Normalized Weighting Values for Figure 5.

Report	Track 1	Track 2
0	0.011	0.110
1	0.981	0.000
2	0.008	0.890

Figure 8. Weighting Matrix based on Figure 7.

The process begins by a new report (NR) being sent to this routine (see Figure 9). Since this is the first report, it is simply stored. When the second report is obtained, it compared to the stored report (SR), where it is first checked to verify that the two reports are not from the same time slice, since this invalidates them from being from the same target source. Following this, the spacing between the two reports is determined. If the spacing is too small, $| R_{New} - R_{Stored} | < \Delta D_{Space_{min}}$, the report is ignored.

When a report associates and passes the minimum spacing test, the pair are then checked to ensure that the proposed association is reasonable, based on the likely maximum and minimum velocities of the targets. Utilizing these velocity limits and the sample interval, Δt, a distance toroid (see Figure 10) is constructed about the stored report, defining the maximum and minimum reasonable separations between the new and stored report.

The toroidal gating regions are defined by

$$d_{i\,j} = \max\{ [Z_j(k+1) - Z_i(k) - V_{max}(\Delta t)] , 0 \}$$

$$+ \max\{ [-Z_j(k+1) + Z_i(k) + V_{min}(\Delta t)] , 0 \}$$

where $\mathbf{Z}$ is the AER measurement vector, V_{max} and V_{min} are the maximum and minimum allowable velocity vectors, and max{[ ],0} indicates that the greater of the quantity in the brackets, or 0 is to be taken[3].

When the two reports pass this test they are combined into a Potential Track (PT). The reports are stored in a stack, with the most recent report being added to the top of the stack, such that when a new report arrives for association, it is only compared with the most recent report of the potential track. At the end of each cycle, the spacing is also checked between the first and most recent reports of the potential track. This *spacing* gate, $| R_{New} - R_{First} | < \Delta D_{Init_{min}}$, where $\Delta D_{Init_{min}} >> \Delta D_{Space_{min}}$, is much larger than the previous gate. Presently, this gating value is a user-selectable input value, set large enough to provide accurate track initiation during all tracking scenarios. Eventually, it is intended that an algorithm be developed to adjust this gating value as a function of the current tracking conditions.

When the track initiation spacing gate is passed, and a minimum number of reports are contained in the potential track (this is an adaptive quantity), a least squares line fit is performed on the set of reports. This line fit is used to determine the most likely heading and velocity of the track, and is referred to as a Tentative track (TT). This information is handed to the normal track update algorithms which attempt track updates. If the tentative track is updated the minimum number of times, it is upgraded to a Confirmed track.

Actual track initiation is performed with the following steps. The covariance matrix is initialized using estimates of the variance in the reports, and the position is determined by projecting the last report onto the fitted line and using the position found (see Figure 11). The velocity is determined by calculating the distance between the projections of the first and last reports, and dividing by the corresponding time interval.

Due to false alarms and report noise, there will be stored reports that never associate, and potential tracks that fail to become tentative tracks. To prevent the possibility of invalid associations with reports from much later frame times and to conserve on memory space, these types of reports are deleted. The deletion of single reports is based on a fixed

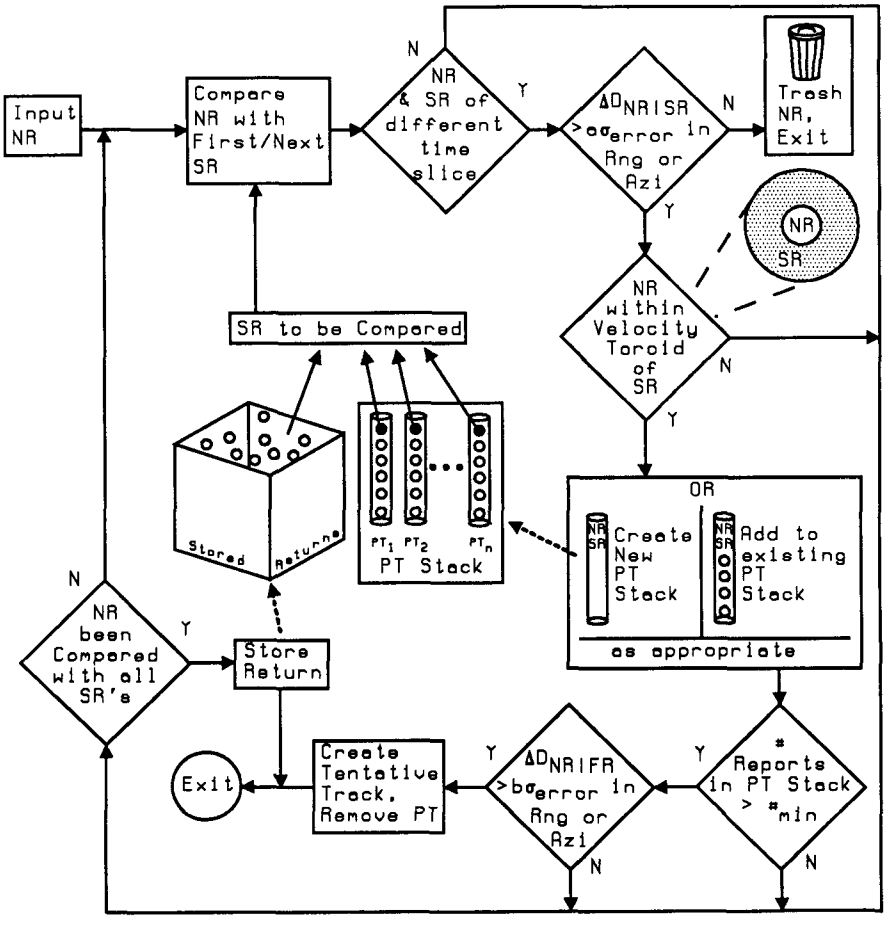


Figure 9. Track Initiation Flow

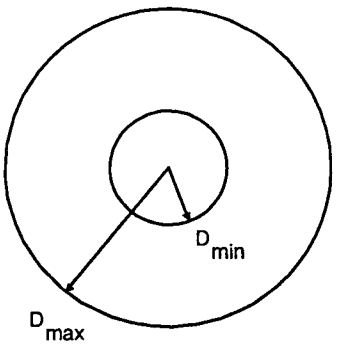


Figure 10. Velocity/Distance Toroid

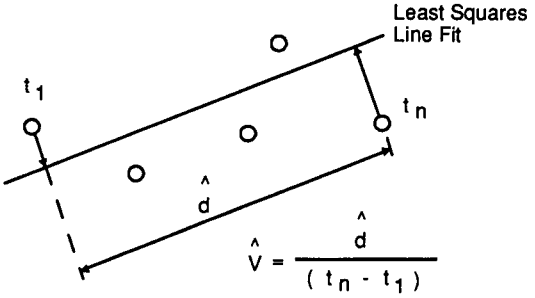


Figure 11. Heading & Velocity Determination
Using Least Squares Line Fit

time since storage, and the potential tracks use this fixed time in reference to the last report stored. Each time a report is added to the potential track set, the deletion clock is reset for that potential track.

Track Deletion

Once tracks exist (Tentative or Confirmed) they must be maintained. However, due to report noise, loss of target, bad track, etc., a few of these tracks will cease to be updated, and need to be removed to prevent unreliable information from being given to the user. When a track is initiated, it is given a Track Identification (TID) number (an external character ID, not to be confused with the internal integer index), and a creation time. The deletion logic then utilizes an M-out-of-N scheme to delete unreliable tracks. This technique specifies that if a track is not updated M out of the last N looks at its predicted position, it is deleted. (The values for M and N are adjustable.) Determination of whether a track *should* have been updated is performed in the following two ways. When the ESA controller instructs the ESA to specifically look for this track, the TID is sent, along with all the reports that were found, to the tracker. Then, if that track is not updated, that information is noted. The second method requires that each track be updated at least once each time the ESA performs a surveillance scan of the entire search volume. Therefore, if it has been longer than one surveillance scan cycle since a track has been updated, that track is marked as having been looked for and not found. Also, each time a track *is* updated, it is marked as having been looked for and updated, regardless of whether or not it was actually being specifically looked for.

The actual implementation requires three memory locations for each track. The first location is used to store the last time that the track was updated so that it can be determined if the track was *missed* during a scan cycle. The second location is used to indicate if the track has had N attempts at being updated yet. If this is not the case, then it cannot be deleted.

The third location is used for the M-out-of-N alignment, and operates at the bit level in which each bit indicates one update status; 1 indicates Yes, 0 indicates No. The right most bit is the most current. Therefore, if the update status for the last four updates has been Yes, Yes, No, Yes, the four right-most bits would be 1101. Each time a track is updated, the bits in the location are shifted one bit location to the left, and the right-most bit, bit zero, is given the current status. This allows for an easy method of always knowing the status of last X updates. Using a short integer storage location, this technique can maintain the status of the last 14 updates.

In order to efficiently maintain an ever changing list of current tracks, when a track is deleted the last current track is moved into that slot and the index to that track is changed to reflect the move.

Application Structure & Design

This tracking application, which does not include the ESA and controller, is designed to operate independently of the input source and consists of three independent pieces of FORTRAN software, each running as a separate process on a VAX computer. These three pieces of software are the Tracker Software, the Display Software, and the User Interface Software. The Tracker Software is responsible for handling the JPDAF functions and providing the necessary data to the Display Software. The Display Software is responsible for displaying track data on a graphic display screen in a meaningful manner. Presently, this includes displaying the tracks' position relative to the sensor, and its velocity and heading through the use of a velocity vector. The User Interface Software is responsible for providing the user with easy control of the tracking system, along with being able to display a variety of information about both the tracks and the tracking system.

In the nominal configuration, this tracking application utilizes only two terminals that provide special graphics capabilities necessary for the Display process and the User Interface process; the Tracker Software runs as a hidden process. When connected to the ESA system, the User Interface is not utilized.

To provide efficient and coordinated communication between the three processes, the tracking system uses a global Common Block area that is installed into the computer's memory and is directly accessible by each of the processes. Since these three processes are operating in a pseudo-parallel environment, and are providing independent functions, they operate at different cycle speeds at different times. To ensure proper synchronization of data utilization within the global common at all times, system control flags are used to oversee these functions. This capability is provided through the use of VAX control flags, called Event Flags. These Event Flags provide the capability to place an idle process into a sleep state until it is needed, which saves valuable computer time for the processes that need it. As soon as the sleeping process is needed, it is given its share of the time. For communication with the ESA, another, very specialized, common block is used to pass the necessary information back and forth.

As a background feature of this system, a complete log can be made of the track update positions so that the Display can replay a scenario without the need for the tracker to be running. This provides quick replay capability, and operates at a faster frame time.

Performance Analysis

We have had only limited time to study the operational speed of this system, and as yet have not performed any benchmarks. However, due to the cluster/hypothesis feature of this system, the run time of the algorithms increases roughly factorially as the size of the cluster increases. It is therefore very advantageous to keep the cluster size as small as possible without losing significant track information. For this reason, our test scenarios are designed to provide cluster sizes of up to four tracks and four associating reports, but due to false alarms and measurement noise they often grow larger. Throughput could be considerably enhanced by conversion to a parallel processing structure. The JPDAF is highly adaptable to this environment. By making one

process an overseer, each cluster can be doled out as a separate process, and the track initiation and deletion operations can be split into separate processes. This would result in significantly faster operation.

Summary & Conclusions

We have implemented the JPDAF for an ESA simulation for tracking large numbers of targets. Our implementation is capable of dealing with asynchronous as well as synchronous data, and includes all features of a complete multi-target tracking system. The track initiation, confirmation and deletion features are all data adaptive depending on measurement accuracy and revisit time. We have developed efficient algorithms for the report-to-track association, clustering and hypothesis generation required for a practical implementation of the JPDAF. The entire tracker has been developed in modular form, and constitutes a complete and flexible testbed for examining a variety of tracking problems.

Overall response time is highly related to cluster size, but could be significantly decreased by restructuring and placing the application on a parallel-processing system.

References

1. Fortmann, T.E., Bar-Shalom, Y., and Scheffe, M., "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," IEEE J. Ocean. Eng., OE-8, No. 3, pp. 173-184, 1983

2. Odom, R.I., Stuart, G.M., and Gorecki, F.D., "Design and Performance Analysis of a JPDAF Tracker for an Electronically Scanned Radar," Proceedings of AIAA Guidance, Navigation & Control Conference, 1989

3. Bar-Shalom, Y., Fortmann, T.E., "Tracking and Data Association," Math. in Science and Eng., Vol. 179, Academic Press, 1988