Project Report
ATC-245

ASR-9 Processor Augmentation Card Scan-Scan Correlator Algorithms

J. B. Evans

2 April 1996

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Federal Aviation Administration,
Washington, D.C. 20591

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. ATC-245	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ASR-9 Processor Augmentation Card Scan-Scan Correlator Algorithms		5. Report Date 2 April 1996 6. Performing Organization Code
7. Author(s) Jenifer B. Evans		8. Performing Organization Report No. ATC-245
9. Performing Organization Name and Address Lincoln Laboratory, MIT 244 Wood Street Lexington, MA 02173-9108		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTFA01-93-Z-02012
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington, DC 20591		13. Type of Report and Period Covered Project Report 14. Sponsoring Agency Code

15. Supplementary Notes

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology under Air Force Contract F19628-95-C-0002.

16. Abstract

This report documents the scan-scan correlator algorithms developed for the ASR-9 Processor Augmentation Card (9-PAC) project. The 9-PAC is a processor card that serves as a processing enhancement to the existing ASR-9's post-processor system. It provides increased speed and memory capabilities to the processor, which allows for the introduction of more complex primary and secondary surveillance algorithms. These more complex algorithms improve the ASR-9's system performance by offering decreased false alarms, and increased detection of aircraft.

The 9-PAC scan-scan correlator, also known as the tracker, consists of three basic processing tasks: initialization, input/output, and the actual tracker. The tracker can be broken down further into four main processing functions: report-to-track association, report-to-track correlation, track update, and track initiation. These four 9-PAC functions are the same as in the original ASR-9 processor, but with different algorithms. Each of these functions is addressed individually in this report and is further broken down into sub-functions for more detailed discussion. In addition to the algorithms, the system requirements are reviewed, and the chosen data structures for implementing the algorithms are detailed

17. Key Words Tracker 9-PAC ASR-9	18. Distribution Statement This document is available to the public through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 90	22. Price

FORM DOT F 1700.7 (8-72) Reproduction of completed page authorized

EXECUTIVE SUMMARY

This report documents the Scan-Scan correlator algorithms developed for the ASR-9 Processor Augmentation Card (9-PAC) project. The 9-PAC is a processor card that serves as a processing enhancement to the existing ASR-9's post-processor system. It provides increased speed and memory capabilities to the processor, which allows for the introduction of more complex scan-scan correlator algorithms. These more complex algorithms improve the ASR-9's system performance through decreased false alarms, and increased detection of aircraft.

The 9-PAC Scan-Scan correlator, also known as the Tracker, consists of three basic processing tasks: initialization, input/output, and the actual Tracker. The Tracker can be broken down further into four main processing functions: report-to-track association, report-to-track correlation, track update, and track initiation. These four 9-PAC Tracker functions are the same as in the original ASR-9 processor, but with different algorithms. Each of these functions is addressed individually in this report, and is further broken down into sub-functions for more detailed discussion.

This report is one in a series of reports that document the algorithms implemented in the 9-PAC. "Documentation of 9-PAC Beacon Target Detector Processing Function" [1] and "Description of Radar Correlation and Interpolation Algorithms for the ASR-9 Processor Augmentation Card (9-PAC)" [2] are two other reports in the series.

ACKNOWLEDGMENTS

The author thanks the people on the 9-PAC team who helped in this effort: Oliver Newell and Gabe Elkin for their great assistance in taking the tracker to real-time, Walter Heath for his simulator tools and support, Zahidul Hassan for his programming assistance, John Anderson of the University of Wisconsin and Jeff Gertz for their tracker theory discussions, and Jim Pieronek and Mel Stone for their leadership.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Air traffic surveillance at major airports depends on two types of radar: primary radar surveillance and secondary surveillance radar. Primary radar surveillance is the more traditional radar surveillance; a signal is transmitted by the radar, reflected off an object, and received by the radar. The aircraft do not need any special equipment to be seen by the radar. Unfortunately, neither do the birds, cars, raindrops, and countless other objects. The most frequently used primary radar system at the major U.S. airports today is the ASR-9 (Airport Surveillance Radar.)

In addition to the primary radar system, there is the secondary surveillance radar system, also known as the beacon system. Secondary surveillance radar differs from primary radar surveillance: a signal (interrogation) is transmitted by the radar, the signal is received by a transponder on an aircraft, an answering signal (reply) is transmitted at a different frequency by the transponder, and the answering signal is received by the radar. This surveillance system requires the intended targets be properly equipped with transponders to be detected by the system. Currently, the most common secondary radar system in use at major airports is a Beacon Interrogator in conjunction with an ASR-9. The Beacon Interrogators are often called sliding window beacons, and are only capable of transmitting interrogations and receiving replies. The processing of the received secondary surveillance data is handled by the ASR-9. The Beacon Interrogator systems are being replaced with Mode S secondary surveillance systems. The Mode S system is capable of processing it's own secondary surveillance data.

The two surveillance systems produce two types of reports, typically referred to as radar and beacon reports from the primary and secondary surveillance systems, respectively. Ideally, all aircraft will have transponders and will be seen by both systems. In reality, not all aircraft have transponders, and not all transponders perform perfectly at all times. Additionally, not all aircraft can be detected with the primary system at all times, especially in high clutter regions. Hence the need for both types of surveillance. The ASR-9 is the first terminal radar system to provide both types of surveillance in a digital format to the end user.

The ASR-9 is an advanced radar system, providing significant improvements in aircraft detection in bad weather. However, as the ASR-9 has been deployed around the country, site-specific, environmentally-induced performance problems have been discovered. These problems include false beacon targets due to reflections and processing splits; radar false targets due to weather breakthrough, ground traffic, and other clutter sources; false radar tracks due to false radar targets; and missed radar tracks due to poor track initiation. These problems can all be addressed with improved, more complex processing algorithms. However, the ASR-9's post-processor, the Array Surveillance Processor (ASP), does not have spare processing capacity, and given its architecture and machine language, it is also difficult to modify and support. As a result the ASR-9 Processor Augmentation Card (9-PAC) was developed.

The 9-PAC is a processor card that replaces an ASP memory board and serves as a processing enhancement to the existing ASR-9's post processor system. It provides significantly increased speed and memory capabilities that make it possible to introduce more complex algorithms for handling the aforementioned performance problems. The 9PAC software can be divided into four basic processes: Beacon Target Detector (BTD), Correlation & Interpolation (C&I), Merge, and Scan-Scan Correlator. In addition there is the Operating System which holds it all together. This report focuses solely on the Scan-Scan Correlator, also known as the Tracker.

This report documents the software modifications and algorithms implemented in the 9-PAC Scan-Scan Correlator or Tracker. Section 2 addresses the basics of why the Tracker is necessary and how it is used. Section 3 outlines the Tracker system requirements. The various data structures implemented in the software are addressed in Section 4, and finally, Section 5, provides the actual algorithms.

System performance improvements provided by the new Tracker algorithms will be addressed in a separate report. The purpose of this report is to document the tracker algorithms in enough detail to support implementation by a second party. This report is one in a series of reports documenting the 9-PAC algorithms [1], [2].

2. OVERVIEW

First, a word about terminology. In the ASR-9 the Tracker function is, in reality, a scan-scan correlator. The distinction is that a true tracker generally maintains a track identification number, predicts at least a few scans ahead, and/or smoothes the input data. A scan-scan correlator determines which report most likely belongs to a single track, and then outputs that report. It does not append a track identification number, it does not do long range predicting, and it does not smooth the data. A scan-scan correlator is essentially a false alarm filter. The ASR-9's current post-processor uses a scan-scan correlator, although it is often referred to as a tracker. Staying with ASR-9 terminology, this report will also refer to the Scan-Scan Correlator portion of the post-processor as the Tracker.

When the ASR-9 is operating with a Beacon Interrogator (BI), e.g., the BI-5, or with the Mode S as an Interim Beacon Interrogator (IBI), the data flow is as shown in Figure 1. Beacon replies are grouped together to form beacon targets in the Beacon Target Detector (BTD). Radar primitives are grouped together to form radar targets in the Correlation & Interpolation (C&I) process. These two streams of data are fed to the Merge process which determines which beacon targets and radar targets correspond to the same aircraft. These targets are merged together and called radar-reinforced targets. The radar-reinforced targets and the leftover beacon-only and radar-only targets are the output of Merge. They are simultaneously passed to the end user and to the Tracker. The Tracker determines which leftover radar-only reports most likely belong to real aircraft, then passes these reports to the end user as correlated radar reports. These correlated radar-only reports may actually correspond to radar-only, non-transponder equipped aircraft, or they may correspond to beacon, transponder-equipped aircraft, which did not have an interpretable beacon signal on that scan.

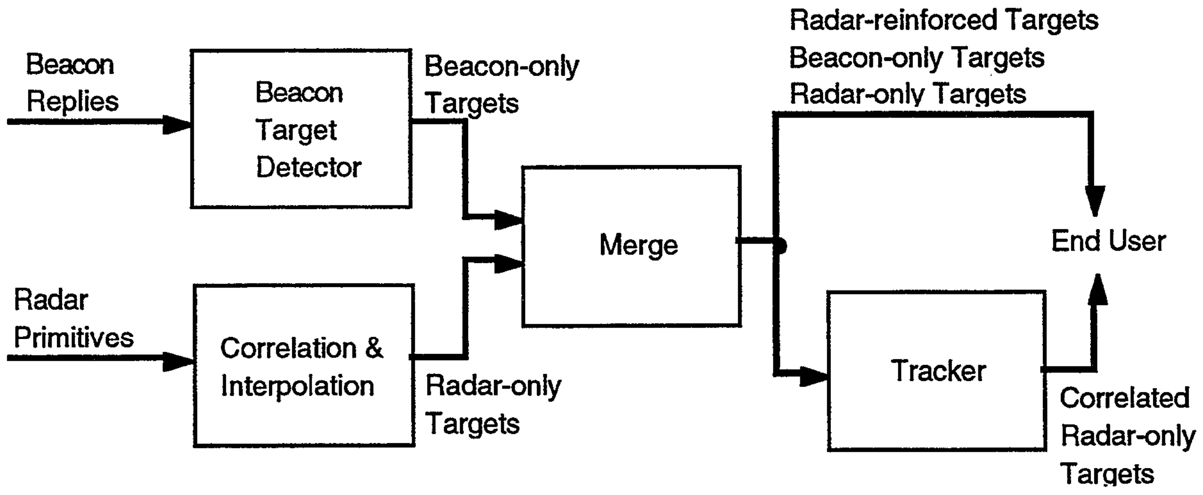


Figure 1. ASR-9 data flow.

When the ASR-9 is operating with a Mode S sensor, the data flow changes slightly. The modified data flow is depicted in Figure 2. The Mode S is responsible for the processing and forming of beacon targets. It receives the radar-only targets formed by the ASR-9's C&I. The Mode S merges these two types of data and outputs radar-reinforced targets, leftover beacon-only targets, and leftover radar-only targets to the ASR-9. The ASR-9 immediately sends the target data to the end user and to the Tracker. The Tracker behaves identically for a BI system and a Mode S system; it acts as a filter for radar-only targets and outputs correlated radar targets.

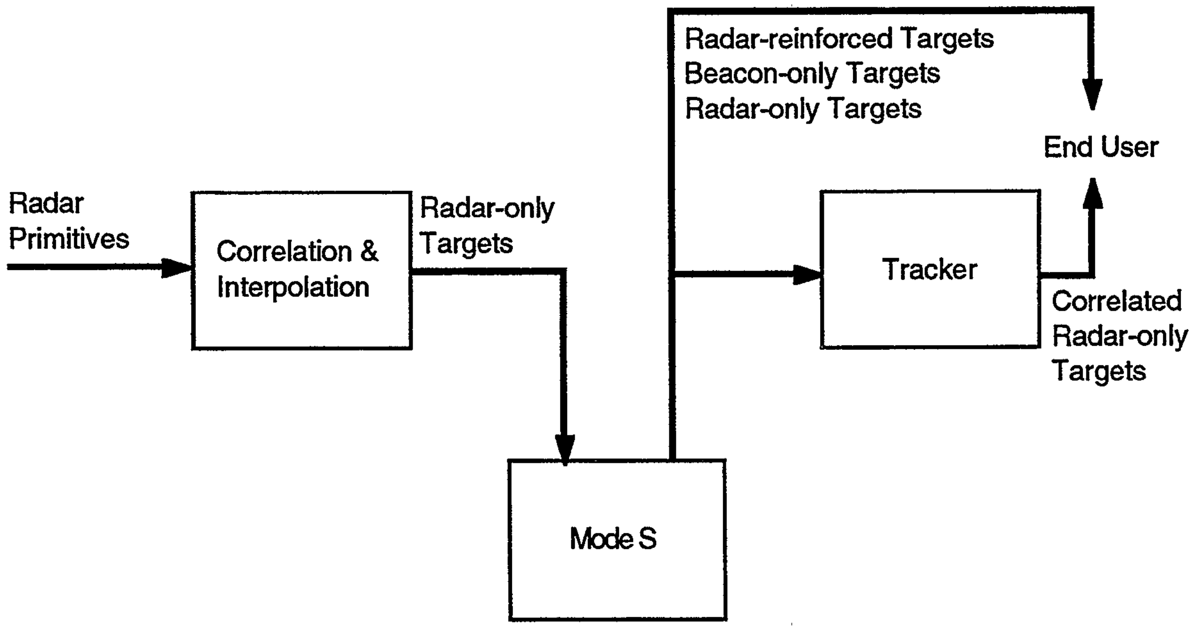


Figure 2. ASR-9 data flow when operating with Mode S.

There are a few interesting points about the data flow and system processing for both configurations. While the Tracker receives radar-reinforced targets and beacon-only targets in addition to the radar-only targets as inputs, it only outputs correlated, radar-only reports. The radar-reinforced targets and beacon-only targets are used to maintain internal tracks. This allows for substitution of a radar-only report in case of a missing beacon report.

Due to timing concerns, the ASR-9 outputs the radar-reinforced targets, the beacon-only targets, and the leftover radar-only targets as soon as possible to the end user. Simultaneously, these same data are input to the Tracker. The Tracker filters the data and subsequently outputs the correlated, radar-only data. These output data are a subset of the leftover, radar-only data that was part of the input to the Tracker and that was previously sent to the end user. Essentially, all of the output of the Tracker is a subset of previously output data, but it has a time delay due to the processing time of the Tracker and a bit set in the header indicating the report is part of a correlated track. Because of the Tracker time delay, the correlated, radar-only data is used solely for display purposes in the ATC system.

3. REQUIREMENTS

The 9-PAC Tracker is designed to meet the same specifications as the ASR-9 Tracker while addressing some performance concerns. These concerns are basically high false alarm rates, particularly in high clutter regions, and an inability to track quickly maneuvering targets. The following are system inputs, outputs, and requirements as defined in the ASR-9 specification FAA-E-2704B [3] and reiterated in the Software System/Subsystem Specification Surveillance Processor for the ASR-9 [4]. The 9-PAC Tracker must satisfy these requirements.

1. Inputs to Surveillance Processor Tracker:
   - Beacon target reports
   - Radar target reports
   - Radar-reinforced target reports
   - Azimuth word
   - Indication of transmitter failure
   - Variable Site Parameters
2. Outputs from Surveillance Processor Tracker:
   - Radar correlated target reports
   - Performance monitor data
   - Tracker overflow alarms
3. Capacity requirement for Surveillance Processor Tracker:

Table 1. Capacity requirement for Tracker.

Capacity Requirement	Number of Targets	FAA-E-2704 Paragraph No.
Peak Aircraft Targets	700	3.4.3.2
Peak Non-aircraft Targets	300	3.4.3.2
Peak targets in 90° scan	250	3.4.3.2
Peak targets across two contiguous 11.25° sectors	100	3.4.3.2
Peak 1.3° azimuth wedge targets	16	3.4.3.2
Sequence of surveillance steps. Minimum of 32 steps per scan.		3.12.5.6.1

4. Target overload conditions shall be handled in an orderly manner; e.g., reduced processing range ([3], Paragraph No. 3.4.3.2).
5. The surveillance processor (tracker) shall output fewer than 1.0 false scan correlated radar target reports per scan averaged over a one hour period, during normal operating conditions. The peak rate of false scan correlated radar target reports shall be fewer than ten per scan averaged over a one hour period, under extreme conditions of "angel" activity or ducting ([3] 3.12.5).

6. The Scan-Scan Correlator shall process the data from Mode S when operating with a Mode S. ([3], 3.12.5).
7. The maximum delay of the scan-scan correlated radar reports to ATC display shall not exceed 2.1 seconds as compared to antenna boresight ([3], 3.12.5).
8. The Scan-Scan Correlator functions shall not drop tracks from the surveillance track list when the POWER-DOWN-INDICATOR = TRUE for less than 15 seconds ([3], 3.3.1).
9. The Scan-Scan Correlator shall not initiate a track by using a target report flagged as an MTI target ([3], 3.12.3.4.12).
10. The Scan-Scan Correlator shall only associate an RTQC target report with a track that was initiated by an RTQC report during a previous scan ([3], 3.13.3.1.1).

4. DATA STRUCTURES

The data structures used in the 9-PAC Tracker are relatively simple. They are described here to help with the understanding of the following discussion of the 9-PAC algorithms. There are basically two unique data types used in the Tracker, reports and tracks, which are usually kept in linked lists. The specific fields that make up the two data types are listed in Appendix A with descriptions. Note that there is plenty of memory in the 9-PAC so it isn't necessary to pack data into these data types; there are numerous fields that only serve as a single bit flag.

4.1 REPORT DATA TYPE

For simplicity, the 9-PAC has only a single report data type. Each report contains a field to differentiate between radar-reinforced, beacon-only, and radar-only reports, but the overall structure is the same. Fields which are meaningless for some report types, e.g., altitude for a radar-only report, are generally set to null and ignored. The report data type has fields for the typical report elements such as range, azimuth, altitude, code, etc., which are generated by BTD and C&I. In addition, it has a number of fields which are used for associating and correlating the report to tracks. These fields count the number of associations, have pointers to associating tracks, and set flags for correlation. In addition, there are a few report fields which are used to reduce the overall system processing load by limiting the need to do a given calculation more than once, e.g., coordinate conversions.

4.2 TRACK DATA TYPE

The 9-PAC has only a single track data type, which has many more fields than the report data type. There are fields containing the positions of the last three reports used to update the track in addition to the predicted position for the next scan. There are also miss counters to determine how many scans occurred between each of the last three reports. There are numerous fields for the actual smoothing and predicting of tracks including the latest alpha and beta gains and track residuals. There are fields for maintaining the association boxes and association and correlation information. There are numerous fields which contain information that can be used for clutter rejection, e.g., velocity, acceleration, minimum distance. Finally, there are fields for statistics, e.g., age count and history count of the track.

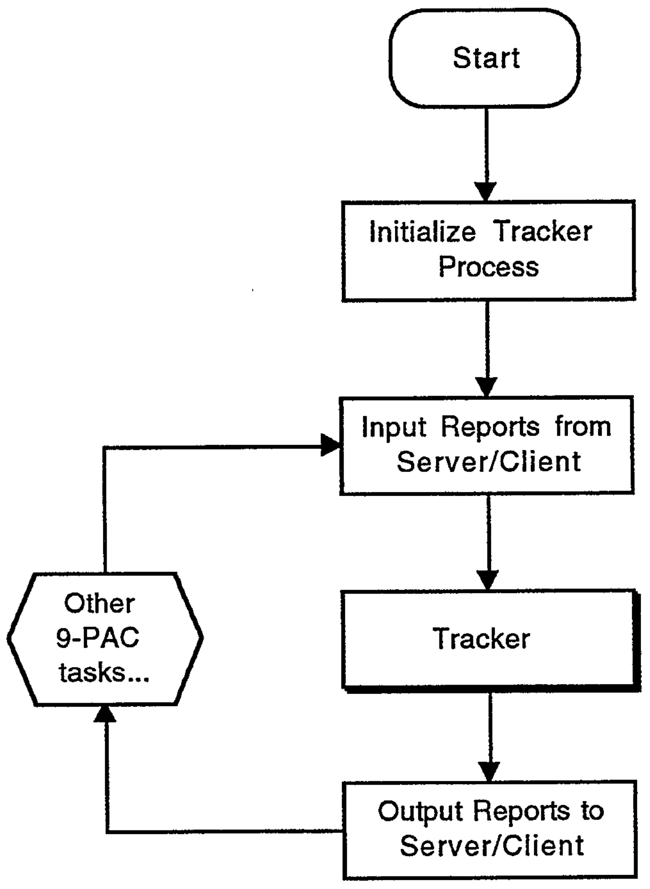
4.3 REPORT AND TRACK LINKED LISTS

Both reports and tracks are maintained in linked lists. The reports are maintained in a single linklist in the order in which they are received. Due to the nature of the 9-PAC processing, the reports are received in clusters, and adjacent reports are rarely more than one sector (128 ACPs) apart. Since the order of the reports is roughly correlated with their azimuths, the processing time required to create an azimuth ordered report list would outweigh the processing benefits achieved by doing so.

Tracks are maintained in two separate linklists: a general list and an active list. The general track list is an azimuth ordered list (by predicted azimuth) for essentially all tracks. The active track list is for tracks which are ready to be updated. When the antenna passes the predicted azimuth position for a track, the track is moved from the general track list to the active track list. After being updated by a report or coasted, the track is returned to the general list. By maintaining two separate lists, update processing is simplified and time is saved by reducing the number of tracks which need to be cross-checked against reports for possible associations, and which need to be monitored for possible correlation.

5. PROCESSING

This section discusses the 9-PAC data processing. This discussion is broken down into three parts: the initialization process, the input/output process, and the actual Tracker process. These processes fit together as shown in Figure 3.



Start
Initialize Tracker Process
Input Reports from Server/Client
Other 9-PAC tasks...
Tracker
Output Reports to Server/Client

Figure 3. Overview of 9-PAC Tracker processes.

5.1 INITIALIZATION PROCESS

When the 9-PAC system is started, each of the main processes needs to be initialized. The Tracker initialization process includes allocating stack space for the maximum number of reports and maximum number of tracks. While the maximum number of tracks should never exceed 1100 (system requirement for 700 beacon targets, 300 primary radar targets, and 100 false alarms per scan), there is more than sufficient memory in the 9-PAC, so the maximum number of tracks has been set to 2000. The maximum number of reports is also oversized and limited to 1000.

After allocating memory for tracks and reports, the data structures that will hold these are initialized. The 9-PAC as implemented uses three linklists: reports, active tracks, and general tracks. Tracks move back and forth between the active list and the general list, but can only be on one list at any given time. The active tracks list contains tracks that have predictions in the vicinity of the radar antenna, and are waiting to be updated or coasted. The general tracks list contains all of the other tracks.

The initialization process also performs one time functions such as generation of a sine and cosine table to reduce future processing.

Finally, the Variable Site Parameters (VSPs) are read for the first time, and the Performance Monitors are reset. (See Appendix B for a listing and definition of VSPs.)

5.2 INPUT/OUTPUT PROCESS

The 9-PAC Tracker process receives and sends data via server/client channels. The Tracker task is a client on receive from the server Merge task during stand-alone ASR-9 operations. When the 9-PAC is operating with a Mode S, the Tracker task is a client on receive from the server input task. In either mode, the 9-PAC Tracker task receives radar-only, beacon-only, and radar-reinforced reports from the serving task. The reports are mixed together in a single stream.

Whenever the Tracker task is called, the first function implemented is receiving reports from the server/client channel. This occurs approximately once every 16 ACPs (4096 ACPs = 360°.) On receipt of the reports, the Tracker process copies the reports from the received format into the previously defined report data structure. Most fields are copied directly to the corresponding field in the Tracker data structure. The exception is the altitude field which is converted from the gray code to an altitude in feet, if possible. In addition to copying the received report fields, a number of other report fields are initialized, e.g., Cartesian coordinates and various counters.

While the processing of incoming reports is called every time the Tracker task is called, the remainder of the Tracker task is only called once every 64 ACPs. System requirements as delineated in Table 1 require calling the surveillance steps at least 32 times a scan, which is once every sector, or once every 128 ACPs. While it is desirable to increase the output rate by calling the Tracker task more often, doing so increases the processing load. Sixty-four ACPs is a good compromise: calling the task even more often does not significantly improve the output rate of the tracker, or significantly change the output, but it does significantly increase the processing load.

Each time the complete Tracker task is called, the Tracker task acts as a server to the 9-PAC Output task client. The 9-PAC Tracker task buffers all correlated radar-only reports corresponding to stable tracks, and at the end of the task, sends them to the 9-PAC Output task. This occurs approximately once every 64 ACPs. In addition, Performance Monitors which are updated each time the complete Tracker task is called are output in the same manner to the Output task client, but only once per scan.

5.3 TRACKER PROCESS

The discussion of the specific 9-PAC Tracker task or Tracker algorithms follows and is accompanied by flowcharts depicting the main Tracker functions. The functions listed by the flowcharts and referred to in this discussion do not necessarily correspond to actual C program functions or subroutines. They are basic tracking functions necessary to achieve the desired tracker performance. How a user or programmer implements the functions in code is up to the individual.

The 9-PAC Tracker architecture is similar to the original ASR-9 Tracker architecture and most other tracker architectures. The Tracker cycles through four basic processes: report-track association (determining which reports are candidates for updating a track), report-track correlation (determining which of the associating reports is the best report for a track), track update (given the correlated report, predicting where the track will be on the next scan), and track initiation (determining which left-over reports could be new tracks.) This basic flow is depicted in Figure 4.

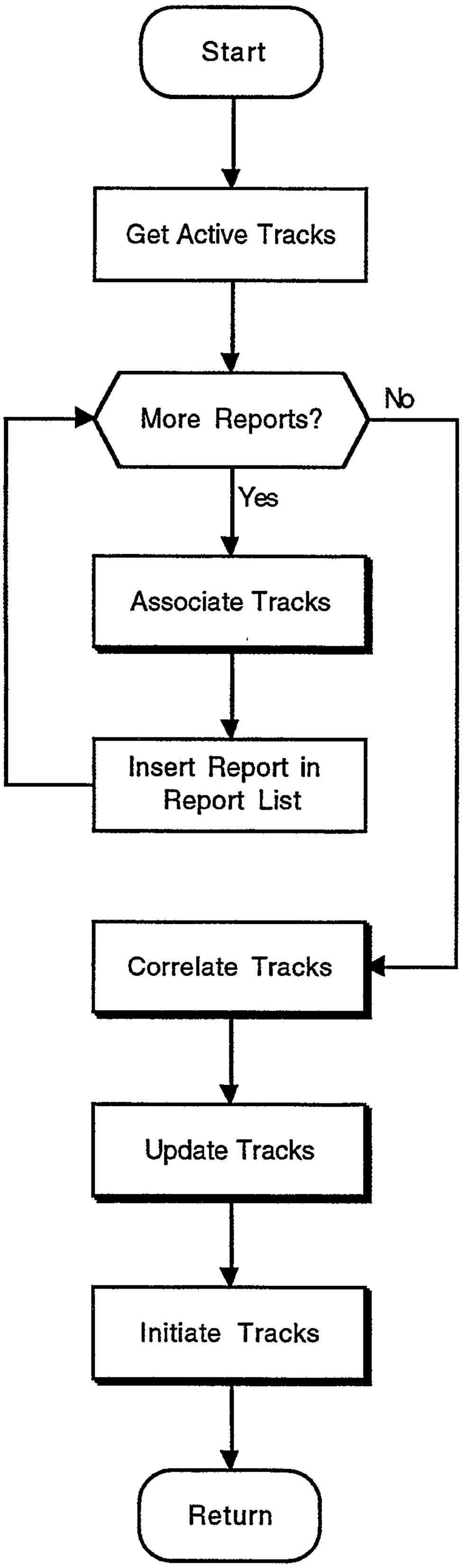


Figure 4. Basic Tracker algorithm flow.

The four basic Tracker functions, associate, correlate, update, and initiate, and their sub-functions are listed below in their respective hierarchy. Each function is accompanied by a flowchart. Some sub-functions have additional flowcharts with further detail; these sub-functions are indicated by a shadowed box in the flowchart. Not surprisingly, some sub-functions are called from more than one function. These sub-functions are only detailed and charted the first time they are called. These are the functions and sub-functions charted in the Sections 5.3.1 through 5.3.4.

9-PAC Tracker

1. Associate Tracks
   — Linearity Test
2. Correlate Tracks
   — Possibly Correlate
   —— Beacon Split
   —— Compute Score
   ——— Doppler Agreement Test
   —— Conflict Resolution
   ——— Compute Score
   ———— Doppler Agreement Test
   ——— Build Association Matrix
   ———— Compute Score
   ————— Doppler Agreement Test
   ——— Modified Munkres Algorithm
   ———— Mark Pair
   ——— Mark Pair
   — Ready to Correlate
   —— Beacon Split
   —— Conflict Resolution
   ——— Compute Score
   ———— Doppler Agreement Test
   ——— Build Association Matrix
   ———— Compute Score
   ————— Doppler Agreement Test
   ——— Modified Munkres Algorithm
   ———— Mark Pair
   ——— Mark Pair
   —— Mark Pair
3. Update Tracks
   — Track Coast
   —— Compute Association Boxes
   — Track Update
   —— Determine Gains
   —— Update Predictions
   —— Compute Velocity
   —— Set Track Type
   —— Check Minimum Distance
   —— Update Track State
   —— Compute Association Boxes
   —— Update Gains
4. Initiate Tracks
   — Compute Association Boxes

5.3.1 Associate Tracks

Called by:	Tracker
Calls:	Linearity Test
Purpose:	The Associate Tracks process determines which reports are candidates for updating which tracks. If a report is a candidate for updating a specific track, the pair are considered associated.

The 9-PAC report-track association process determines which reports are possible candidates for updating which tracks. If a report and a track are considered a possible match, the pair are said to be associated. The best report for a given track is chosen from the associated reports. As reports are received from the server/client channel, they are compared to each track on the active track list, and all associations are marked.

The main criteria for report-track association in the 9-PAC is location. Two association zones are defined for most tracks: zone 0 represents the predicted track position plus room for system errors; zone 1 encompasses zone 0 plus adds room for a maneuvering track. A report can associate with a track via zone 0 or zone 1. In addition to the location criteria defined by zone 0 and zone 1, there are additional criteria for report-track association. These are discussed below.

If the report is a radar-only target report, the report must have a Confidence field greater than 2, or the track must be mature. (Confidence is set in C&I. Confidence 0 corresponds to a geo-censored report. Confidence 1 corresponds to a report from a high clutter region. Confidence 2 corresponds to a report from possible interference, e.g., another radar.) If the track is mature, a low Confidence report is allowed to associate, but only in zone 0. Since low Confidence reports are likely to be clutter as opposed to real aircraft, it is not desirable to initiate tracks with them. The probability of initiating a false track is too high. However, once a track is established and declared mature, the damage due to a clutter report is less severe if the clutter report is near the predicted position. And since not all low Confidence reports are clutter, allowing low Confidence reports in zone 0 to associate to a mature track maximizes the probability of continuing valid tracks through a clutter region without significantly increasing the false alarm rate.

The total number of associations for a single track or a single report is limited to six. In a typical environment, more than 90% of the reports and tracks only have one association. The limit of six is rarely hit, and even then it is due to a system anomaly and a severe clutter breakthrough. If the number of associations limit is hit, no more associations are allowed and it is assumed that with the severe clutter, the chances of a greatly superior association is small and not worth the heavy processing load that would be encumbered.

The final criteria is a linearity test. The purpose of the linearity test is to assure the reports contributing to the track are moving in a manner reasonable for an aircraft. The details of the test are described in Section 5.3.1.1.

If all the criteria for association are met (position, Confidence, number of associations, and linearity), the association is marked: counters in the report and track structures are incremented, and pointers are set from the report to the track and vice versa. In addition, if the report and track have matching discrete codes, a special flag is set indicating a discrete association.

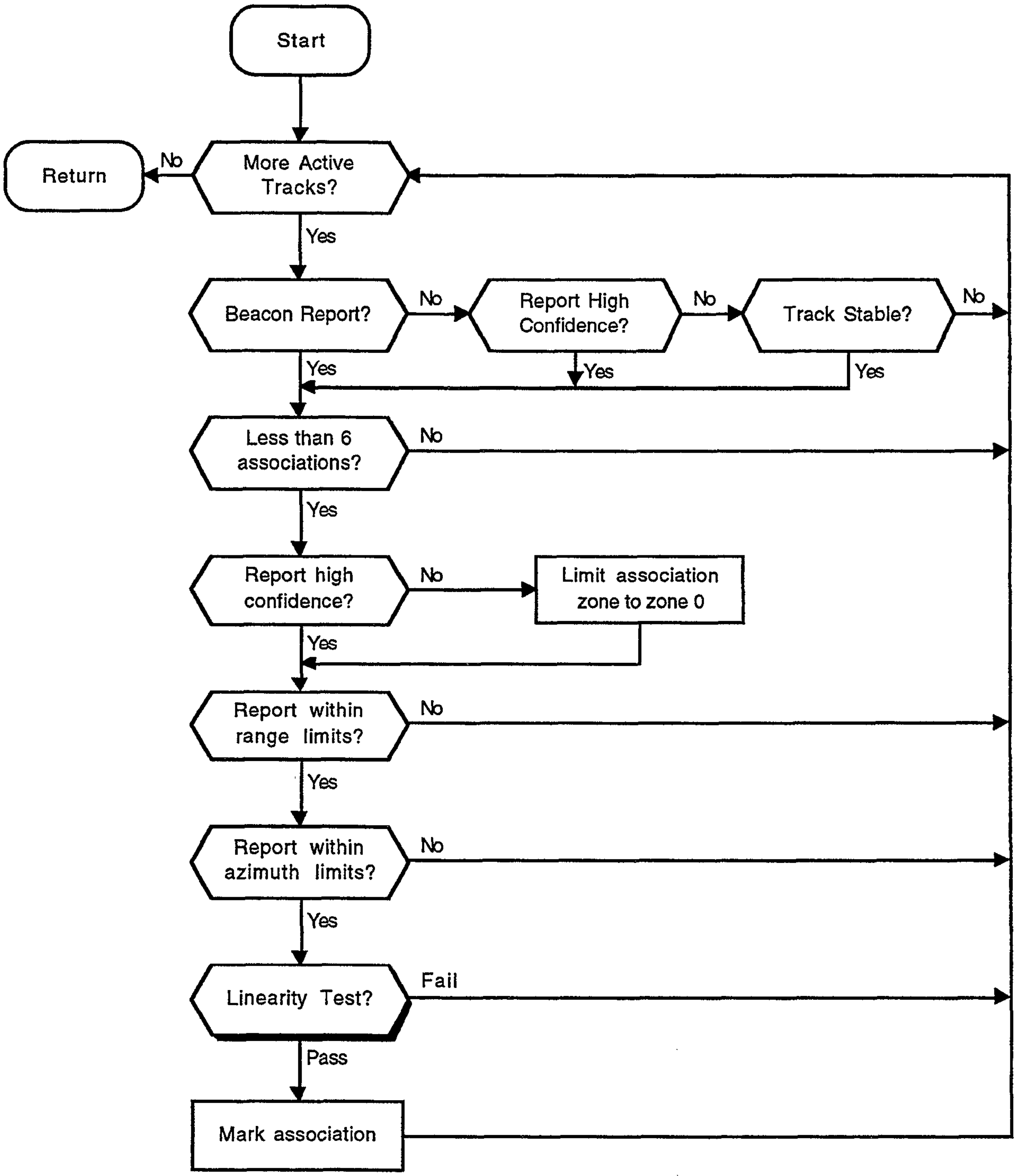


Figure 5. Associate tracks.

5.3.1.1 Linearity

Called by:	Associate Tracks
Calls:	
Purpose:	The linearity test determines whether the reports contributing to a track lie in a path consistent with a real aircraft.

The linearity test uses the track history and computes the deviation of a middle report in a series of reports from the expected position. The deviation from the expected position is compared to the expected system errors. If the measured deviation differs by more than a predetermined number of standard deviations, the report and track fail the linearity test. The number of standard deviations used for the test depends on the maturity of the track: a mature track is allowed a larger deviation than a non-mature track. A more detailed description of the test follows.

The track under consideration has been updated the last three scans by reports received at times *t-3*, *t-2*, and *t-1*. The range and azimuth, (ρ,θ), of these reports has been saved along with the number of misses between these updates. The report under consideration has been received on this scan at time *t*. The linearity test uses the four reports, *t-3*, *t-2*, *t-1*, and *t*, along with the miss history to determine if the fourth report, *t*, really belongs with the track.

The linearity test has two steps to it. For each step the test is first applied to the new report of interest, *t*, and the last two reports to update the track (*t-1*, *t-2*). If the test is not passed, the test is repeated with the new report of interest, *t*, and the reports used to update the track two and three scans ago (*t-2*, *t-3*). By repeating the test with data that is one scan older, it is possible to prevent a single bad update report from preventing future good updates.

The first step of the linearity test is to assure there is a linear progression in range or azimuth of the three reports.

$$\begin{aligned}&\min(\rho(t-2), \rho(t)) < \rho(t-1) < \max(\rho(t-2), \rho(t)) \text{ or}\\&\min(\theta(t-2), \theta(t)) < \theta(t-1) < \max(\theta(t-2), \theta(t)) \qquad (1)\end{aligned}$$

If a progression does not exist for range or azimuth for the last two reports, the test is repeated with one scan older data, and the progression is required for both range and azimuth:

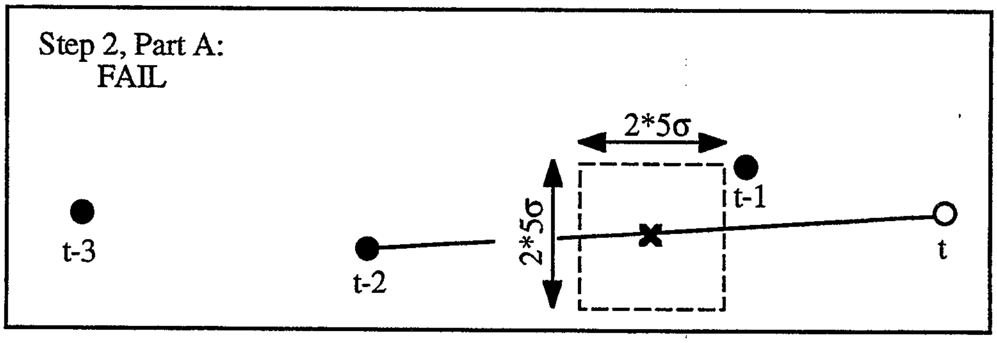
$$\begin{aligned}&\min(\rho(t-3), \rho(t)) < \rho(t-2) < \max(\rho(t-3), \rho(t)) \text{ and}\\&\min(\theta(t-3), \theta(t)) < \theta(t-2) < \max(\theta(t-3), \theta(t)) \qquad (2)\end{aligned}$$

If a progression still does not exist for range and azimuth for the older data, the first step of the linearity test is failed, and the new report is not allowed to associate with the track.

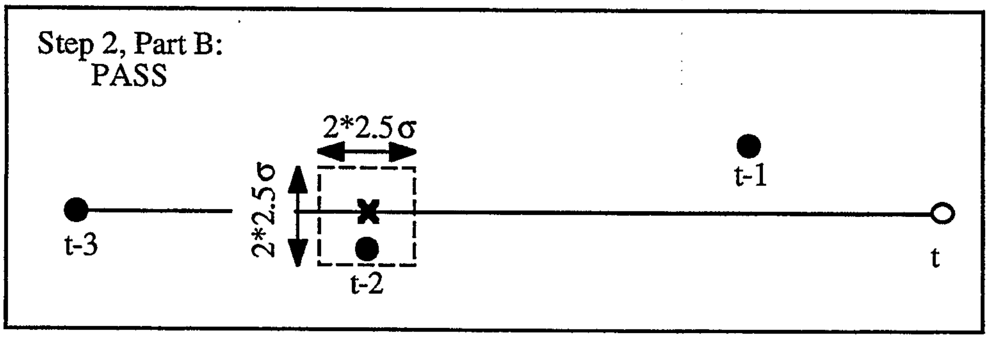
If the first step of the linearity test is passed, a more precise second step is performed. Once a linear progression has been assured by the first step, the second step checks the accuracy of the progression with respect to the system errors. As with the first step, the new report under consideration, *t*, and the last two reports to update the track (*t-2*, *t-1*) form a triplet which is analyzed first. Given the two endpoints, *t-2* and *t*, of the triplet, the midpoint is calculated and deemed the expected value for the triplet's middle report, *t-1*. A check is then made to see if the triplet's middle report is within a few sigmas of the expected value. This part of the test is depicted in Figure 6. As with the first step, if the triplet does not pass this test, the test is repeated with a triplet consisting of the report under consideration, *t*, and older data from the track, (*t-3*, *t-2*). The acceptable variance between the expected midpoint and the triplet's middle report is dependent on the situation and defined in Table 2.

Table 2. Linearity test accuracy limits.

Track State	Triplet	Accuracy
Stable	t, t-1, t-2	5σ
Stable	t, t-2, t-3	2.5σ
Initiating	t, t-1, t-2	3σ
Initiating	t, t-2, t-3	1.5σ

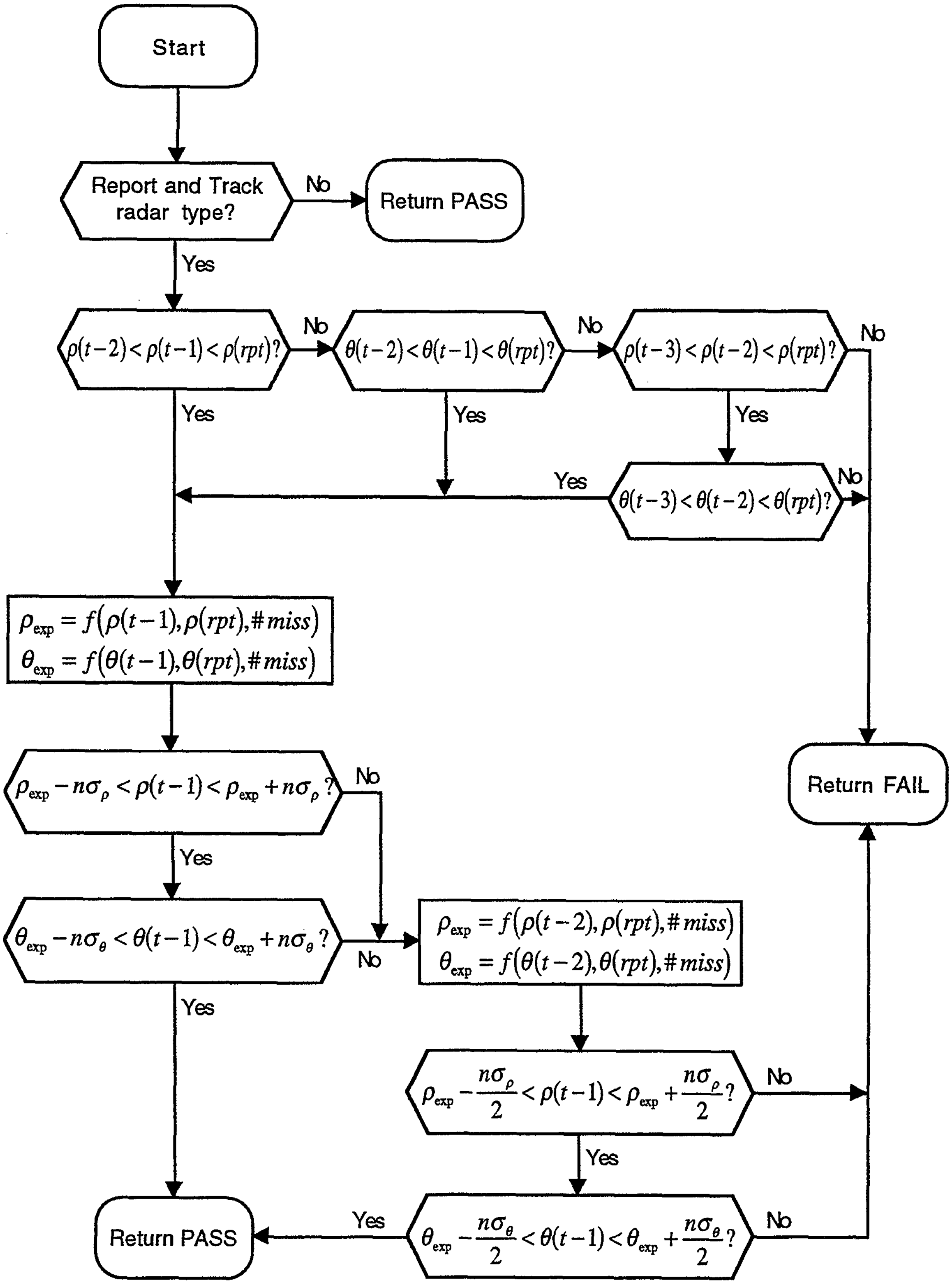


(a)



(b)

Figure 6. Illustration of Step 2 of linearity test, (a) Part A using reports t-2, t-1, and t. Test is failed making (b) Part B necessary using reports t-3, t-2, and t. Test is passed.



Start
Report and Track radar type?
No
Return PASS
Yes
$\rho(t-2)<\rho(t-1)<\rho(rpt)$?
$\theta(t-2)<\theta(t-1)<\theta(rpt)$?
$\rho(t-3)<\rho(t-2)<\rho(rpt)$?
$\theta(t-3)<\theta(t-2)<\theta(rpt)$?
$\rho_{\exp}=f(\rho(t-1),\rho(rpt),\#miss)$
$\theta_{\exp}=f(\theta(t-1),\theta(rpt),\#miss)$
$\rho_{\exp}-n\sigma_\rho<\rho(t-1)<\rho_{\exp}+n\sigma_\rho$?
$\theta_{\exp}-n\sigma_\theta<\theta(t-1)<\theta_{\exp}+n\sigma_\theta$?
$\rho_{\exp}=f(\rho(t-2),\rho(rpt),\#miss)$
$\theta_{\exp}=f(\theta(t-2),\theta(rpt),\#miss)$
$\rho_{\exp}-\frac{n\sigma_\rho}{2}<\rho(t-1)<\rho_{\exp}+\frac{n\sigma_\rho}{2}$?
$\theta_{\exp}-\frac{n\sigma_\theta}{2}<\theta(t-1)<\theta_{\exp}+\frac{n\sigma_\theta}{2}$?
Return FAIL
Return PASS

Figure 7. Linearity test.

5.3.2 Correlate Tracks

Called by:	Track
Calls:	Ready to Correlate
	Possibly Correlate
Purpose:	Steps through the track list and determines if a track is ready for correlation, possibly ready for correlation, or not ready for correlation.

Correlation of tracks is the determination of which report-track association is the best, and marking that association as a correlated pair. Ideally, the determination would not be made until all possible associations have been made for that track, and all possible associations have been made for all of the associating reports, and all associations have been made for all of the associating reports' associating tracks, and so on. Also, ideally, the determination would be made as soon as the correct report was received. However, these two ideals are counter to each other and tradeoffs must be made. This is accomplished by considering each track as possibly ready for correlation as soon as the position of the expected report is passed. If certain conditions are met, correlation is performed early. If the conditions are not met, correlation is delayed. After all possible associations should have been received for a given track, the track is considered ready for correlation. At this point a correlation is made if possible. The process is explained in greater detail below.

Each track on the active track list is considered one at a time. First, it is confirmed the track has not already been marked for correlation. (It is possible that a track could already have been marked, e.g., if it had multiple associations of which at least one was shared with a track earlier in the list.) If the track has already been marked, it needs no further correlation processing and the next track on the list is considered.

Given that a track is not correlated, a check is made between the antenna position, the predicted azimuth position, and the maximum association box azimuth. If the antenna position has not yet reached the predicted position, no correlation attempt is made; chances are good the best association has yet to be found. If the antenna position has passed the predicted azimuth position, it is considered possibly ready for correlation; there is a good chance the best association has been found, but all associations probably have not been found. If the antenna position is beyond the maximum association box azimuth, or more than 3 sectors (3*128 ACPs) beyond the predicted azimuth, the track is considered ready for correlation. The details for Possibly Correlate and Ready to Correlate are in Sections 5.3.2.1 and 5.3.2.2.

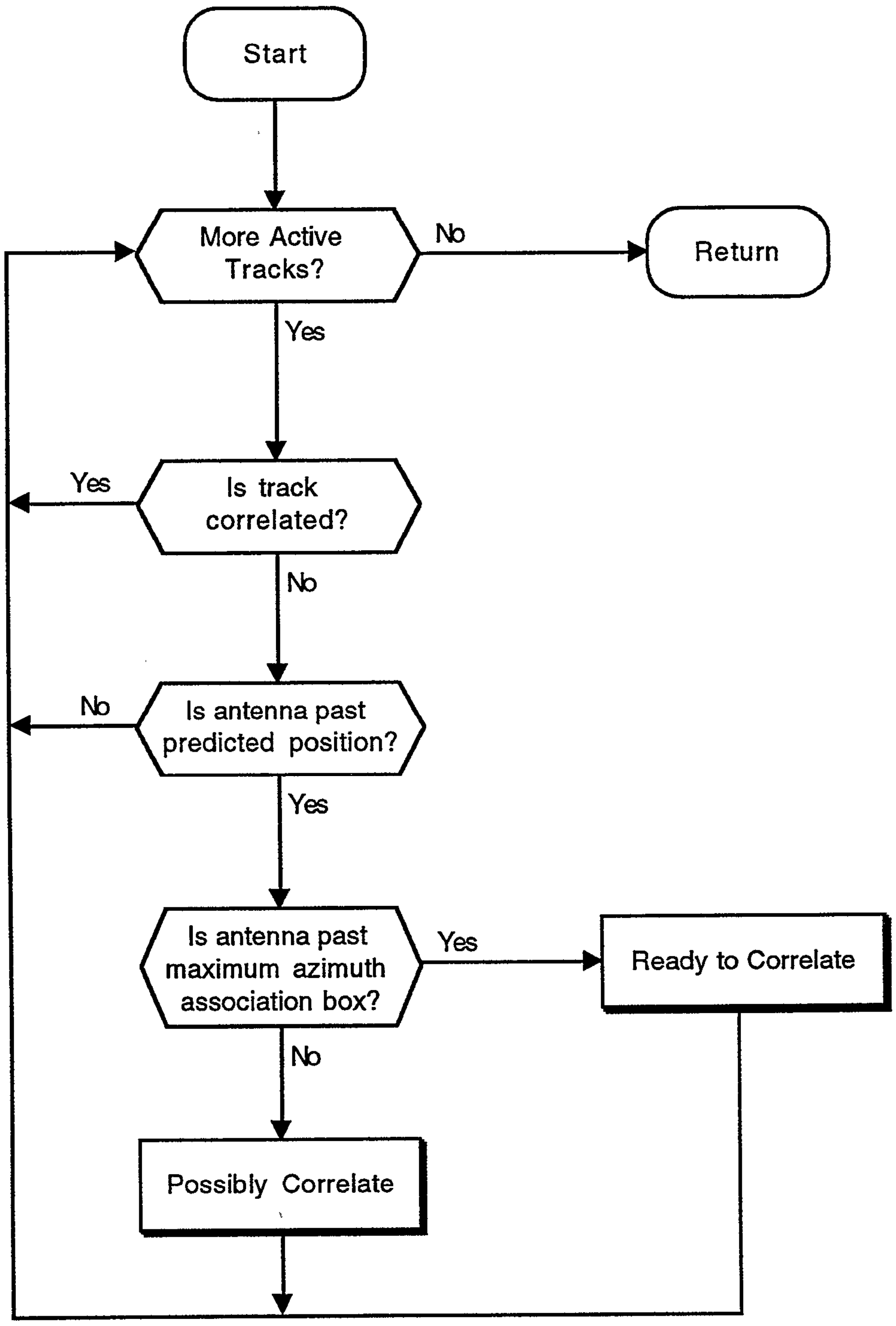


Figure 8. Correlate tracks.

5.3.2.1 Possibly Correlate

Called by:	Correlate Tracks
Calls:	Beacon Split
	Compute Score
	Conflict Resolution
	Mark Pair
Purpose:	Determines if a track should be correlated early. If a single very good association exists, correlation is performed early. If there are multiple associations or no outstanding associations, correlation is delayed.

A track is considered a possible candidate for correlation if the antenna position has passed the predicted azimuth position, but has not passed the maximum association box azimuth. In a typical ASR-9 environment, more than 90% of the reports and tracks have only one association. Given this information, it is likely that the best, and most likely only, association has already been made since the predicted position has been passed. If this is truly the case, then the associated pair should be considered for early correlation as opposed to unnecessarily delaying the correlated output.

The first step before determining whether a track should be correlated early, is to assure there are not multiple discrete associations due to a beacon split. A discrete association is the association of a beacon report with a discrete Mode 3/A code to a track with an identical Mode 3/A code. If there are multiple discrete associations, a special function to handle beacon splits is called. This function is detailed in Section 5.3.2.3.

If the track of interest has a one-to-one association with a report, a measurement of the quality of the association is made. A numeric score, which is essentially a multidimensional distance, is computed. If this score falls below a predetermined threshold, the association is accepted for early report-to-track correlation. If the score exceeds the threshold, a flag is set to avoid re-computing the score in the future. The details for computing this score are discussed in Section 5.3.2.4.

If the track has only one associating report, but the report has more than one associating track, it is assumed the track is near a clutter area, or a crossing aircraft situation is occurring, or some other atypical situation. It is best in this situation to allow all available information to be gathered before making a correlation decision. The track of interest is returned to the list and correlation will be attempted after the maximum association box azimuth has been passed and all possible associating reports have been gathered.

It is possible that a single track has multiple associations, but only one is a discrete association: the track's discrete code matches a report's discrete code. If this is the case, it is most likely that the discrete association is the best association, and the other associations are due to surrounding clutter. To avoid unnecessarily delaying the output due to the non-discrete association(s), this track and its multiple associating reports are immediately sent to Conflict Resolution. Conflict Resolution is a method for determining the best correlations given a number of cross associating tracks and reports. It is detailed in Section 5.3.2.6.

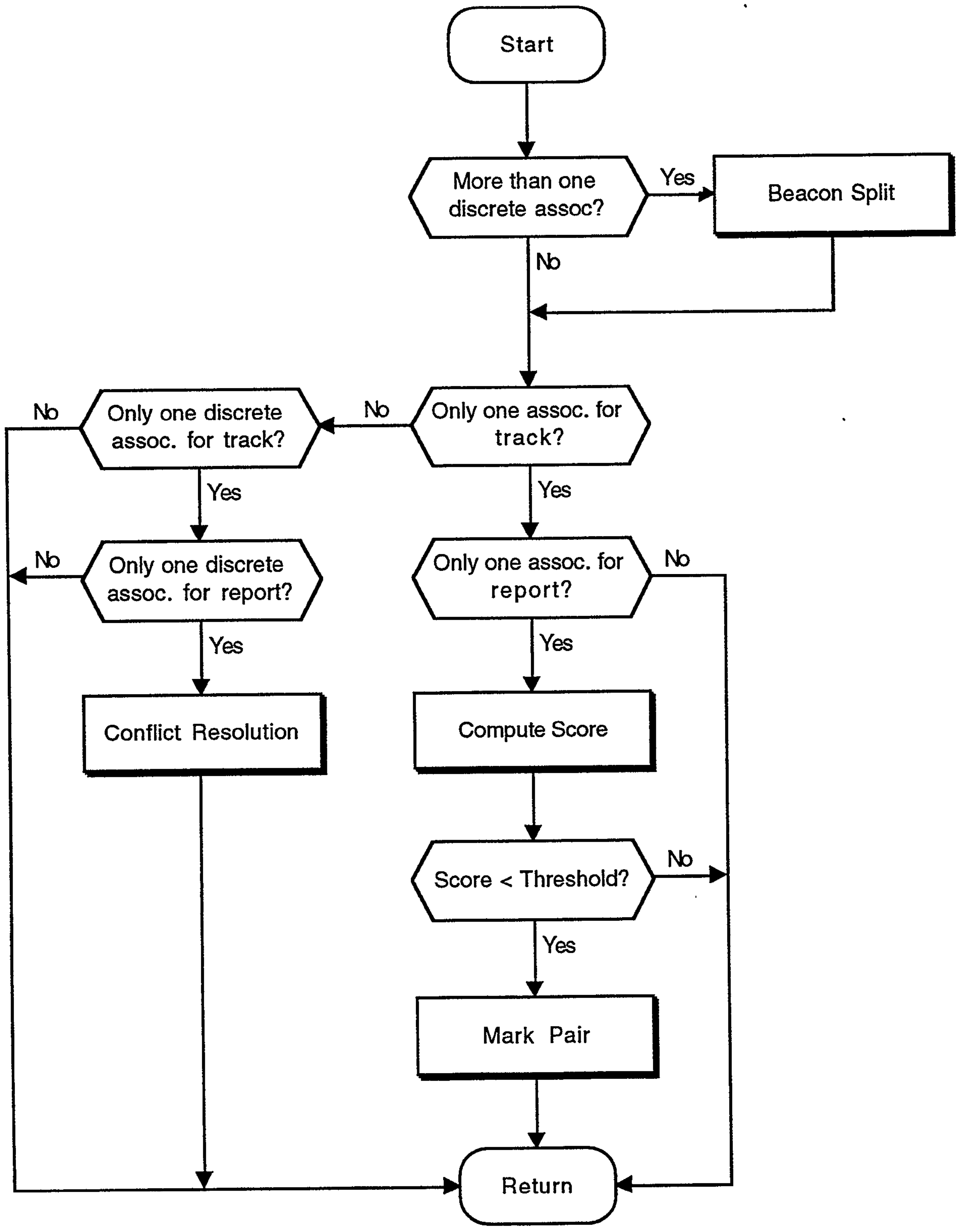

Start
More than one discrete assoc?
Yes
Beacon Split
No
Only one assoc. for track?
No
Only one discrete assoc. for track?
No
Yes
Yes
Only one discrete assoc. for report?
No
Only one assoc. for report?
No
Yes
Yes
Conflict Resolution
Compute Score
Score < Threshold?
No
Yes
Mark Pair
Return

Figure 9. Possibly correlate.

5.3.2.2 *Ready To Correlate*

Called by:	Correlate Tracks
Calls:	Beacon Split
	Conflict Resolution
	Mark Pair
Purpose:	Attempts to correlate a report and track.

If the antenna position has passed the maximum association box azimuth, all candidate reports will have been received and associated (assuming no unusual system delay or overload problem). At this point, it is time to correlate the track to the best associating report.

As with the Possibly Correlate testing, the first step is to assure there are not multiple discrete associations for a discrete track. This would be caused by beacon splits, and a special Beacon Split function is called to handle this situation. The details of the Beacon Split function are discussed in Section 5.3.2.3.

If the track of interest has a one-to-one association with a report, the pair should be correlated. No further testing is necessary.

If the track has no associating reports, the track is marked for coasting. No further testing is necessary.

If the track has more than one associating report, or if there is just one associating report, but it has more than one associating track, the Conflict Resolution function is called. Conflict Resolution is a method for determining the best correlations given a number of cross associating reports and tracks. It is detailed in Section 5.3.2.6.

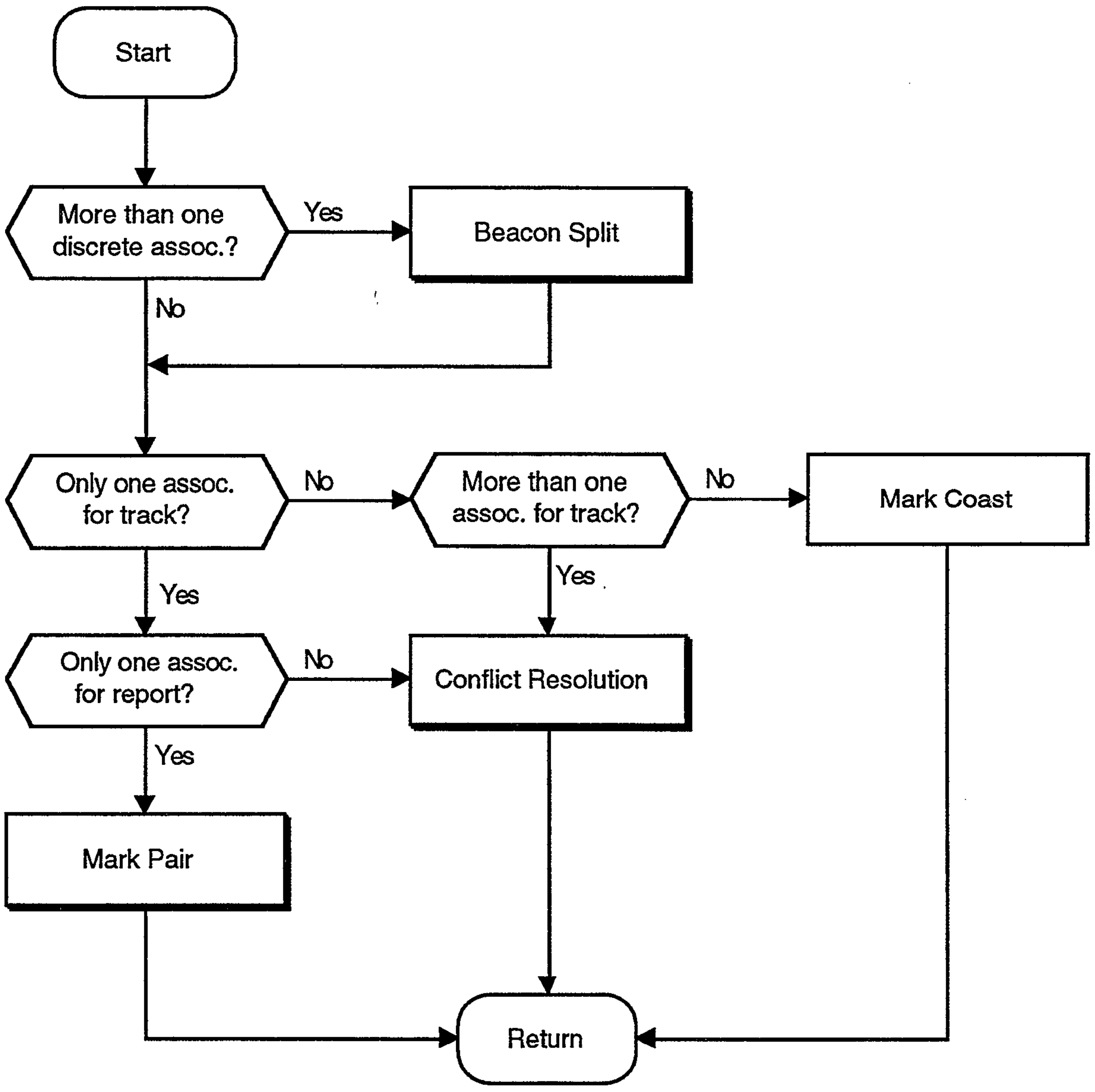

Start
More than one discrete assoc.?
Yes
Beacon Split
No
Only one assoc. for track?
No
More than one assoc. for track?
No
Mark Coast
Yes
Yes
Only one assoc. for report?
No
Conflict Resolution
Yes
Mark Pair
Return

Figure 10. Ready to correlate.

5.3.2.3 *Beacon Split*

Called by:	Possibly Correlate
	Ready to Correlate
Calls:	
Purpose:	Handle beacon splits in a non-detrimental manner for the Tracker.

Beacon splits can be due to azimuth or range splits which are typically environment induced, or due to a faulty transponder. Most beacon splits result in poor azimuth or poor range position for both reports. Regardless of the cause of the split, it is important to update the track in a reasonable manner and to avoid initializing a second discrete track with the extra split beacon report.

The purpose of the ASR-9 Scan-Scan correlator is to output correlated radar-only reports, not beacon reports. The majority of the beacon splits are due to environmental factors, which typically last a scan or two, not for a long period of time for the same target. Given these two pieces of information, the simplest and most efficient manner for handling beacon splits, is to ignore the split reports and coast the track. By doing this, the track is not adversely affected by the poor position accuracy of the reports. In addition to coasting the track, the split reports are flagged to assure they are not used for track initiation.

While coasting the track may be preferred in the majority of cases, it is not the preferred method for long lasting phenomena like a faulty transponder. A faulty transponder may cause beacon splits for the life of the track as opposed to just a scan or two. If the split reports were ignored, and the track coasted, the track would quickly drop. The preferable solution in such a case is to take the average of the split reports and update the track.

To determine which method to use, a count is maintained for the number of consecutive beacon splits seen by a track. For the first three splits, the track is simply coasted and the reports are discarded. After three splits, a transponder problem is suspected and the split reports are averaged and then used to update the track. If a track is updated by a typical beacon report (not a split) the counter is reset to zero. In addition, to prevent the track from coasting out after three splits, a special flag is set in the track report indicating the track was forced to coast. This flag is monitored by the Track Coast function (Section 5.3.3.9) to assure proper handling of the track.

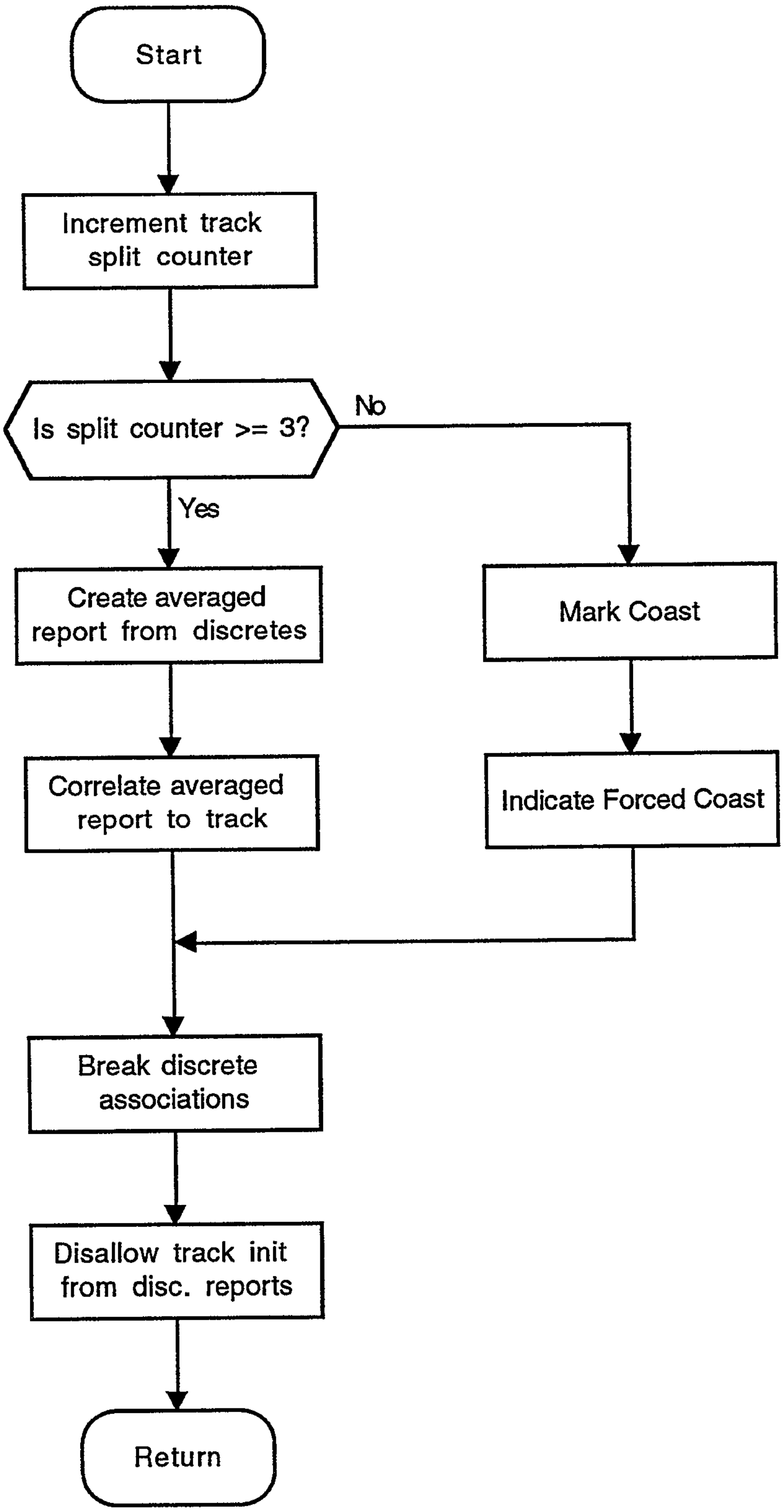

Start
Increment track split counter
Is split counter >= 3?
No
Yes
Create averaged report from discretes
Mark Coast
Correlate averaged report to track
Indicate Forced Coast
Break discrete associations
Disallow track init from disc. reports
Return

Figure 11. Beacon split.

5.3.2.4 Compute Score

Called by:	Possibly Correlate
	Ready to Correlate
	Conflict Resolution
	Build Association Matrix
Calls:	Doppler Agreement Test
Purpose:	Compute a numerical association measure relating a given report to a given track.

The association measure score is a measure of the quality of the association between a track and report. It is a function of many different report and track fields in addition to the true distance between the report's position and the track's predicted position. It is essentially a multi-dimensional distance; the lower the score, the better the association. The score is actually computed by following a series of tests and assigning penalties for less than ideal results. The fewer penalties, the lower the score, the shorter the multi-dimensional distance.

The first test checks for a matching discrete report and track. If either the report or track is radar-only, this test is failed and a penalty of 7 is assigned. If the report and track have matching discrete codes, their altitudes are checked. If the altitudes match, no penalty is assessed. The in-between case is where the report and track are both discrete but do not match. If both the code and altitude are significantly different, a penalty of 7 is desired to match the radar-only case. However, it could just be a case of a dropped bit or two, so the penalty should not be so severe. The algorithm used for assigning the penalty is as follows:

$$\text{Penalty} = ((2/3)*(\text{number of code bit differences}) + (1/3)*(\text{number of altitude bit differences}))*(7/12) \qquad (3)$$

The maximum number of bit differences is 12. If more than half of the bits are different, the difference is set to the maximum. While a matching altitude says something about the association, it is not as important as the discrete code, so the code aspect of the penalty is weighted more than the altitude aspect. The net effect is a penalty between 0 and 7 which is dependent on the accuracy of the discrete codes and altitude.

The second test checks the association zone. If the report falls in association zone 0 (the zone corresponding to the track's predicted position plus system errors), no penalty is assessed. If the report falls in zone 1 (the zone encompassing zone 0 plus a margin for a maneuvering target), a penalty is assigned. This penalty is 5.

This assignment of penalties allows a radar-only report located near the predicted position in zone 0 to update a beacon track instead of a poor code/altitude matching beacon report further from the predicted position in zone 1. If the code and altitude do match for the beacon report and track, the beacon report will be chosen over any radar report regardless of the association zone. This mode of logic for relating the importance of various report and track fields was used in setting the penalty levels for all the tests.

The third test considers track and report types. If the report type (radar-only, beacon-only, radar-reinforced) matches the track type, no penalty is assigned. If the track is radar-only and the report beacon-only or vice versa, a mismatch penalty of 7 is assigned. If either the report or the track is radar-reinforced, but not both, half of the mismatch penalty is assigned.

The fourth test is only applied to radar-only reports since it uses the interpolated Doppler field which is only available for radar reports. If the Doppler test (Section 5.3.2.5) is passed, no penalty is assessed. If the Doppler test fails, a penalty of 5 is assessed.

The fifth test is also only applied to radar-only reports since it uses the Confidence field, another field only available for radar reports. Confidence is set by C&I, ranges from 0 to 5, and is a measure of the credibility of the report. Confidence and the applicable penalty are described in Table 3.

Table 3. Confidence field description and assigned penalties.

Confidence	Meaning	Penalty
0	Target reports from roads	5
1	Target reports from heavy clutter	5
2	Single CPI reports from interference	2
3	Target reports from aircraft, thermal false alarms, angels, etc. and range less than R.	0
4	Target reports whose maximum Doppler response is from the ZVF	2
5	Target reports from aircraft, thermal false alarms, angels, etc. and range greater than R.	0

The sixth test is similar to the Confidence test, but is applied to all reports and tracks. It uses the Quality field that is set by C&I. Quality is a measure of the azimuth accuracy and strength of the target return. As set by C&I, Quality ranges from 0 to 3. The 9-PAC adds Quality 4 for all beacon reports. This allows the application of the 9-PAC algorithms to all reports, regardless of report type. This can be done for the Quality field because the azimuth accuracy of the beacon-only reports is known along with the azimuth accuracy of the various Quality radar reports. Quality and the applicable penalty are described in Table 4.

Table 4. Quality field description and assigned penalties.

Quality	Meaning	Penalty
0	One CPI report	5
1	Two CPI report, both PRFs	3
2	Two or more CPI report, one PRF	1
3	Three or more CPI report, both PRFs	0
4	Beacon report	1

The seventh test checks the age of the track. If two tracks associate with the same report, and all other factors are equal (association zone, Doppler, etc.), the more established track should take precedence. This rule should only apply in the early stages of tracking, and therefore only has an effect on a track which has been around for seven or fewer scans. If the track has been around for at least seven scans, no penalty is assigned. If less than seven scans, the penalty is the difference between seven and the age. If the association is in zone 1 (for maneuvering targets), the age penalty is doubled.

The eighth and final test checks the actual distance between the track's predicted position and the report. To keep the distance penalty on a similar scale as the other penalties, the actual distance squared is multiplied by a factor of 100. For a report off by 3 sigma in range and perfect in azimuth, a penalty of 1 would result. For a report off by 3 sigma in azimuth and perfect in range at 30 nmi, a penalty of 4.8 would result. There is a point where the distance error should just be considered quite large and the actual magnitude of the error should not be considered. Whether the error is 20 sigmas or 22 sigmas is not relevant - both are bad. To handle this situation, the distance penalty is top limited to 10.

The final association measure score is obtained by adding all of the penalties accrued for the eight possible tests. The best association measure possible would be 0 for a perfect match for a beacon report. The larger the score, the less desirable or less perfect the association. Essentially, the association measure score is a multi-dimensional distance computed as a function of position, report type, Quality, Confidence, and Doppler.

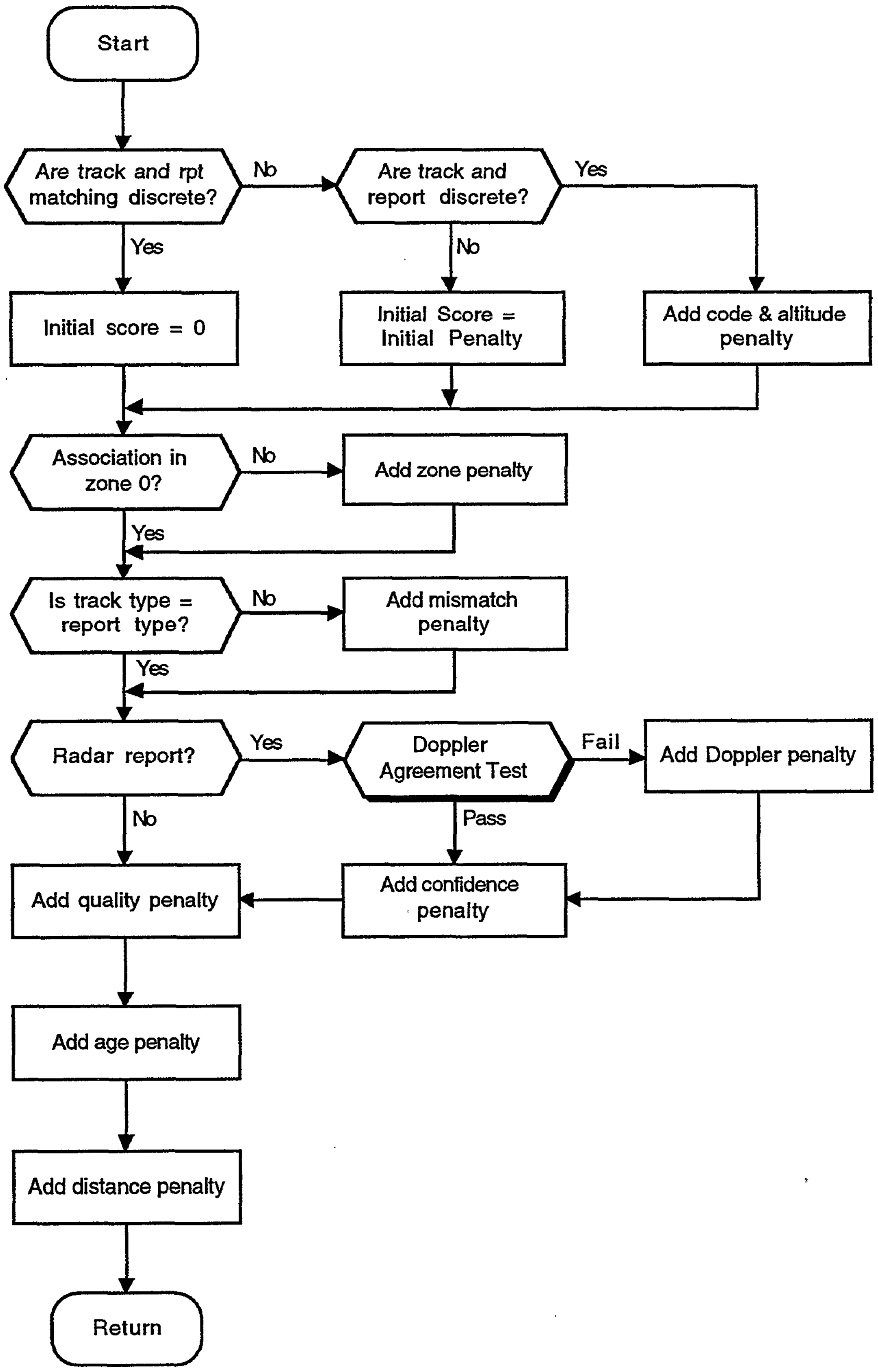


Figure 12. Compute score.

5.3.2.5 Doppler Agreement Test

Called by:	Compute Score
Calls:	
Purpose:	Determines if the interpolated Doppler numbers for a report are correct if the report were to update a certain track.

All radar reports have one or two interpolated Doppler numbers. Studies have shown that these Doppler numbers agree with the radial velocity of the target more than 85% of the time [5]. This test assumes a report is used to update an existing track and computes the estimated radial velocity of the track given the new report. A comparison is then made between the new radial velocity estimate and the report's interpolated Doppler numbers. If the numbers agree, the test is passed. The details and justification for the Doppler Agreement test are found in [5]. The high level basics of the test follows.

An estimate of the range rate of the track is made using the new report as the most recent update. This estimated range rate is used to compute the expected interpolated Doppler numbers for the report. If the actual interpolated Doppler numbers for the new report fall within a defined area around the expected interpolated Doppler numbers, the test is passed. This area is shown in Figure 13. Numerically, the test is implemented by computing the differences between the actual and expected interpolated Doppler numbers for both PRFs. Let these differences be dl and dh for the low and high PRFs, respectively. If the differences satisfy the following equations, the test is passed.

$$0.83dh + 14.0 \le dl \le 0.83dh - 3.0 \tag{4}$$

$$2.00dh - 18.75 \le dl \le 2.00dh - 45.25 \tag{5}$$

If the report only has one interpolated Doppler number (Quality = 0 or 2), the test must be simplified. An area can no longer be defined since the test must be reduced to one dimension. The test checks to see if the actual interpolated Doppler number is within 6 units of the corresponding expected interpolated Doppler number. If so, the test is passed.

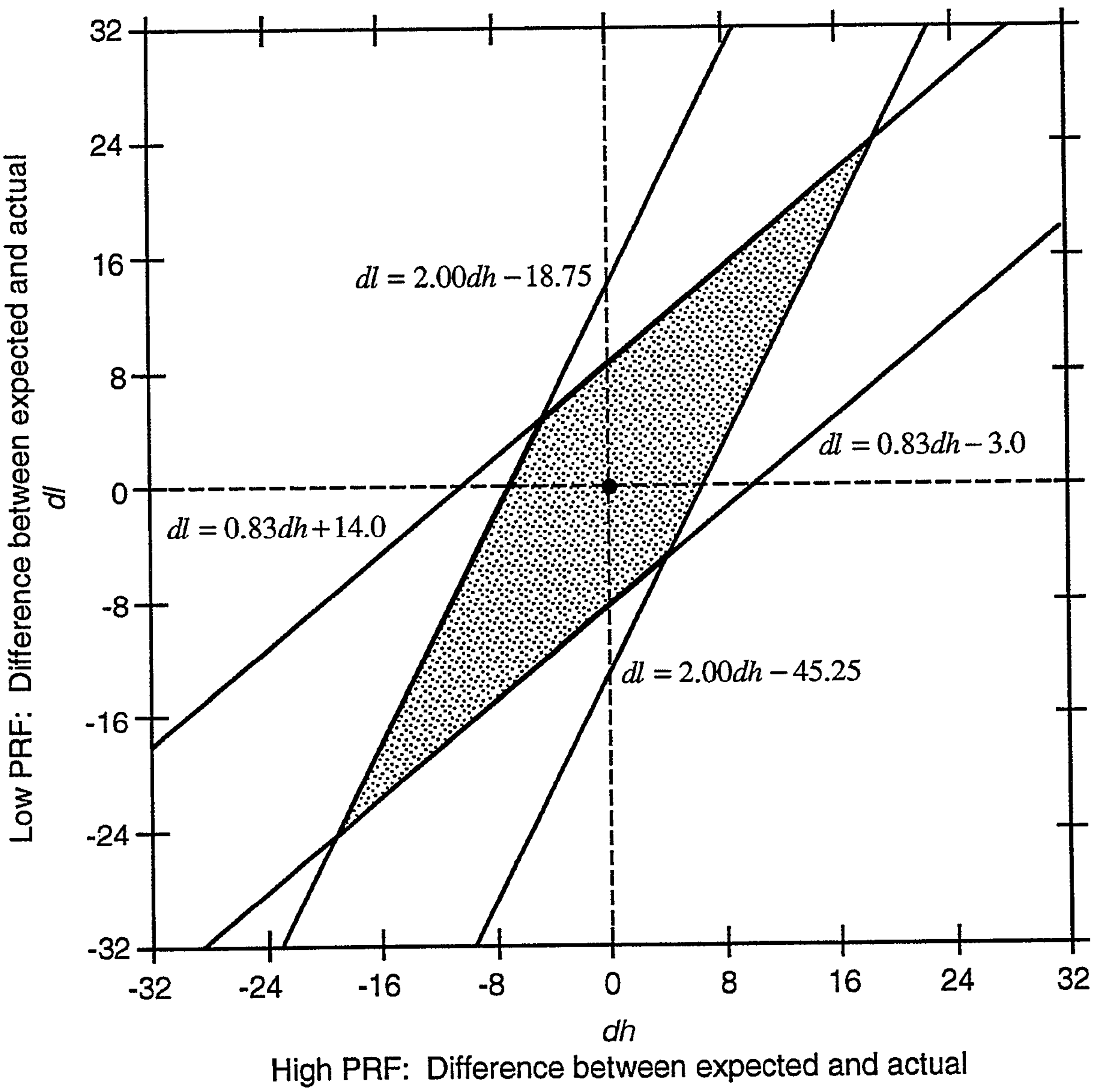


Figure 13. Graphical Doppler Agreement Test. If the differences between the high and low, expected and measured interpolated Doppler numbers falls in the shaded region, the test is passed.

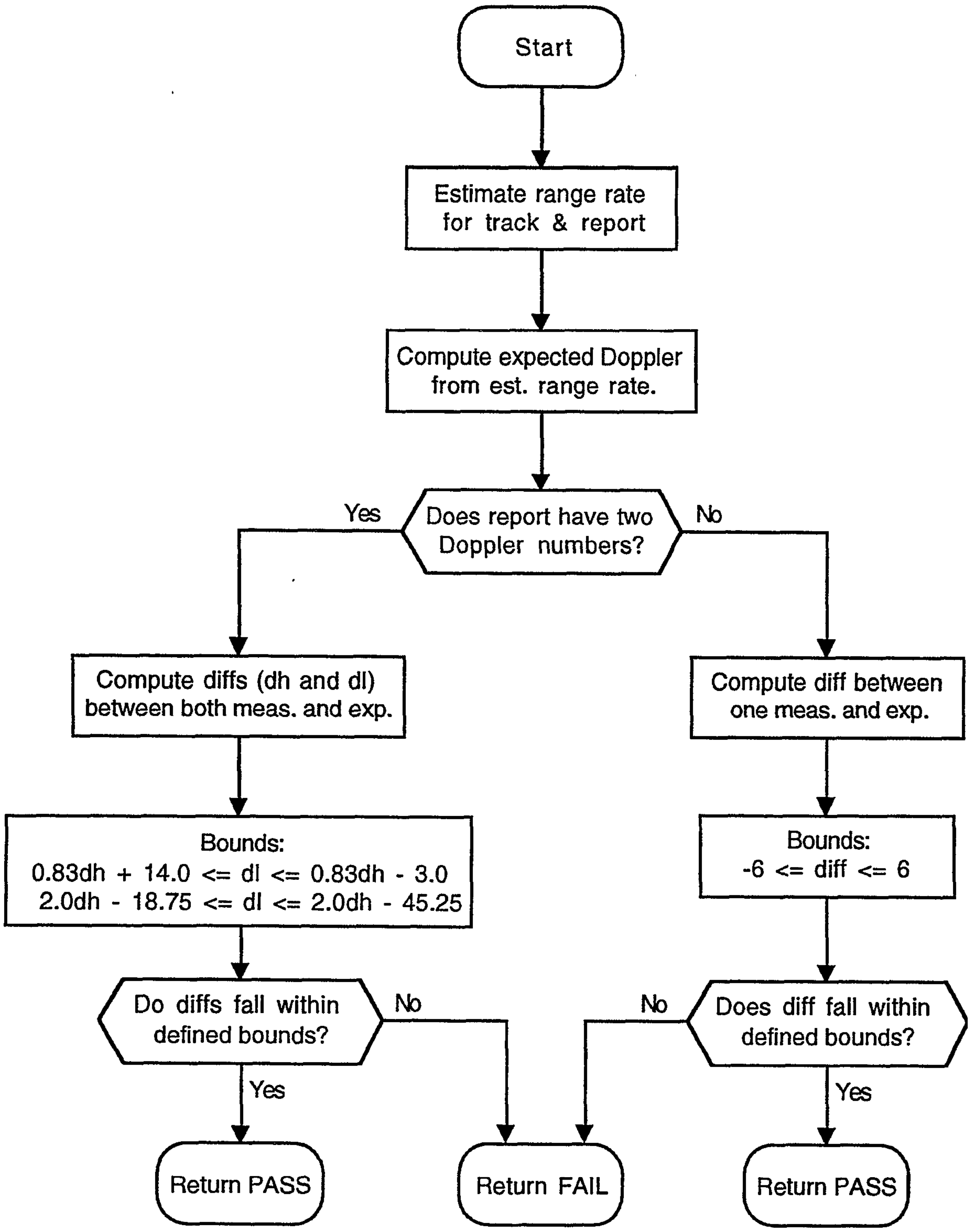

Start
Estimate range rate for track & report
Compute expected Doppler from est. range rate.
Does report have two Doppler numbers?
Yes
No
Compute diffs (dh and dl) between both meas. and exp.
Compute diff between one meas. and exp.
Bounds:
0.83dh + 14.0 <= dl <= 0.83dh - 3.0
2.0dh - 18.75 <= dl <= 2.0dh - 45.25
Bounds:
-6 <= diff <= 6
Do diffs fall within defined bounds?
No
Yes
No
Does diff fall within defined bounds?
Yes
Return PASS
Return FAIL
Return PASS

Figure 14. Doppler agreement test.

5.3.2.6 *Conflict Resolution*

Called by:	Possibly Correlate
	Ready to Correlate
Calls:	Build Association Matrix
	Modified Munkres Algorithm
	Compute Score
	Mark Pair
Purpose:	Determine the best set of correlations given a number of cross associating tracks and reports.

More than 90% of tracks and reports have a one-to-one association (a track associates to just one report, and that one report associates only to that one track.) In such simple cases, the best correlation is the same as the only association. However, when any other situation occurs, Conflict Resolution is called. These other cases can be broken down into three basic groups: one track and many reports, many tracks and one report, and many tracks and many reports.

Two of the cases are quite similar and have a fairly simple resolution. For one track and many reports, the association measure is computed for each possible pair, and the report with the lowest association measure score is chosen to correlate with the track. For many reports and one track, the same basic process is used. The association measure is computed for each possible pair, and the track with the lowest association measure score is chosen to correlate with the report. (The association measure score is essentially a multi-dimensional distance; the smaller the score, the closer the association.)

The third case with many tracks and many reports is more complex. It involves building an association matrix which details the cross associations between the tracks and reports. After the matrix is formed, it needs to be reduced in such a manner as to determine which set of associations are the best and should be marked for correlation. A Modified Munkres Algorithm is used for this matrix reduction. The building and solving of the matrix is detailed in Sections 5.3.2.7 and 5.3.2.8.

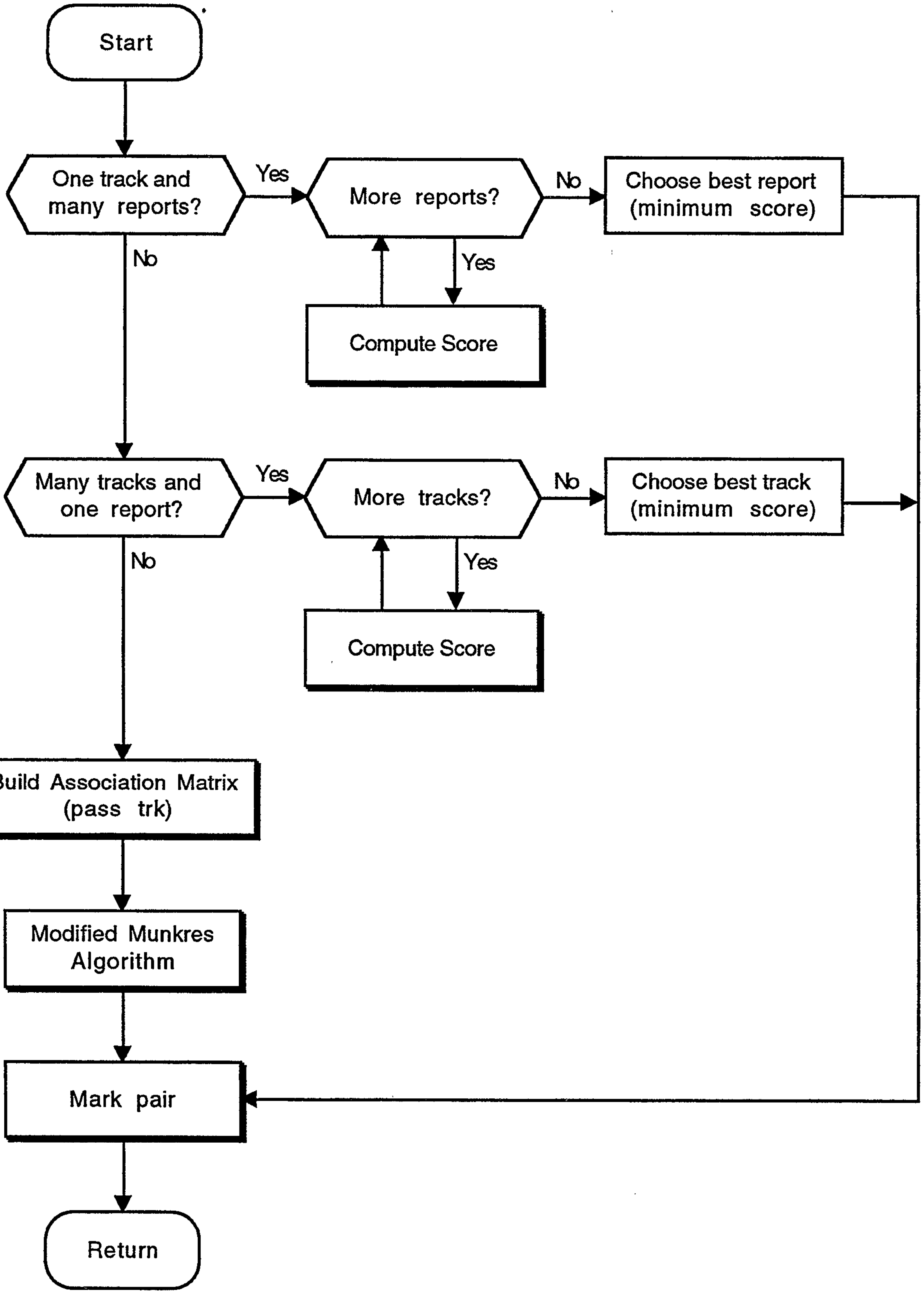

Start
One track and many reports?
Yes
More reports?
No
Choose best report (minimum score)
No
Yes
Compute Score
Many tracks and one report?
Yes
More tracks?
No
Choose best track (minimum score)
No
Yes
Compute Score
Build Association Matrix (pass trk)
Modified Munkres Algorithm
Mark pair
Return

Figure 15. Conflict resolution.

5.3.2.7 Build Association Matrix

Called by:	Conflict Resolution
Calls:	Compute Score
Purpose:	Maps out cross associations between tracks and reports and determines the association measure score for each.

The association matrix is a matrix of cross associating tracks and reports. The matrix rows correspond to tracks and the matrix columns correspond to reports. The matrix elements are the association measure scores for the corresponding tracks and reports.

The first step in building the association matrix is to initialize the matrix elements. Since the association measure is essentially a multi-dimensional distance, a low association measure score corresponds to a close association. Reports and tracks that do not associate with each other but are both in the matrix, need a high number in the association matrix to assure there is no confusion between a null and a perfect association. Hence, the association matrix is initialized with a high number, e.g., 9999.

After the association matrix is initialized, the matrix is started with a single track passed from Conflict Resolution. The track represents a row in the matrix. All reports that associate with this track are then added to the matrix and represent individual columns. The actual elements of the matrix are the association measure scores relating the specific tracks and reports. These scores are computed and entered in the matrix. An iterative process is then followed to complete the matrix. For each r[illegible]t that was added to the matrix, a search is made for more associating tracks (it obviously h[illegible] least one associating track, or the report would not have been added to the matrix in the first pl[illegible].) If there are additional associating tracks, it is confirmed that they are not already part of the m[illegible]ix, and then added as new rows. After all the new reports have been checked for additional ass[illegible]iating tracks, the new tracks are checked for additional associating reports. As new reports and tracks are added to the matrix, the corresponding association measure scores are computed and entered. This process continues until a closed set is formed, that is, there are no more associating tracks or reports to be added, or until the matrix has reached the maximum allowable si[illegible]. This maximum size is currently set to 10 x 10. An example of the process follows.

Tab[illegible] contains a list of tracks, the number of associations for each track , and a list of the associating reports for each track. Table 6 is basically the same but for reports. It has a list of reports, the number of associating tracks for each report, and a list of the associating tracks for each report.

Table 5. Example track and associating reports.

Track List	# Assocs.	Assoc. 1	Assoc. 2	Assoc. 3	Assoc. 4
Track A	2	Report A	Report B	-	-
Track B	3	Report A	Report B	Report C	-
Track C	1	Report C	-	-	-

Table 6. Example report and associating tracks.

Report List	# Assocs.	Assoc. 1	Assoc. 2	Assoc. 3	Assoc. 4
Report A	2	Track A	Track B	-	-
Report B	2	Track A	Track B	-	-
Report C	2	Track B	Track C	-	-

The first step in building the association matrix is to initialize all of the potential matrix elements to 9999. However, to reduce clutter and increase readability, this step is not shown here.

(1) Begin with Track A, the track received from the calling routine Conflict Resolution. Track A is the first row in the matrix.

Track A			

(2) Using Table 5, add columns for each associating report: Report A and Report B. Compute the association measure score as the matrix element. (The association measure scores shown are representative scores.)

	Report A	Report B	
Track A	7	3	

(3) Using Table 6, find all tracks which associate with Report A in column 1: Track A and Track B. Since Track A is already part of the matrix, only add Track B. Compute the association measure score.

	Report A	Report B	
Track A	7	3	
Track B	6		

(4) Using Table 6, find all tracks which associate with Report B in column 2: Track A and Track B. Both Track A and Track B are part of the matrix so no new rows need to be added. The association measure score for Report B and Track B needs to be computed, however.

	Report A	Report B	
Track A	7	3	
Track B	6	3	

(5) All new columns have been checked. Start checking new rows. Using Table 5, find all reports which associate with Track B in row 2: Report A, Report B, and Report C. Since Report A and Report B are already in the matrix, only one new column for Report C needs to be added. Compute the association measure score.

	Report A	Report B	Report C
Track A	7	3	
Track B	6	3	2

(6) All new rows have been checked. Back to checking new columns (this is an iterative process.) Using Table 6, find all tracks which associate with Report C: Track B and Track C. One new row needs to be added for Track C. Compute the association measure score.

	Report A	Report B	Report C
Track A	7	3	
Track B	6	3	2
Track C			4

(7) All new columns have been checked. Go back to check new rows. Using Table 5, find all reports which associate with Track C: Report C. Since this is already part of the matrix, no new columns are added. Since all new rows have been checked and no new columns were added, the iterative process is complete, a closed set has been found, and the matrix is finished.

Sometimes a large number of cross associations exist and it is not computationally practical to allow the association matrix to grow without limits. For this reason, the size of the association matrix is limited by a VSP constant, typically 10x10. If the number of cross-associations would result in a bigger matrix, there is an unusual phenomenon occurring or there is a very high clutter situation. Regardless, the chances of finding a significantly better set of correlations is not large enough to compensate for the increased processing that would result.

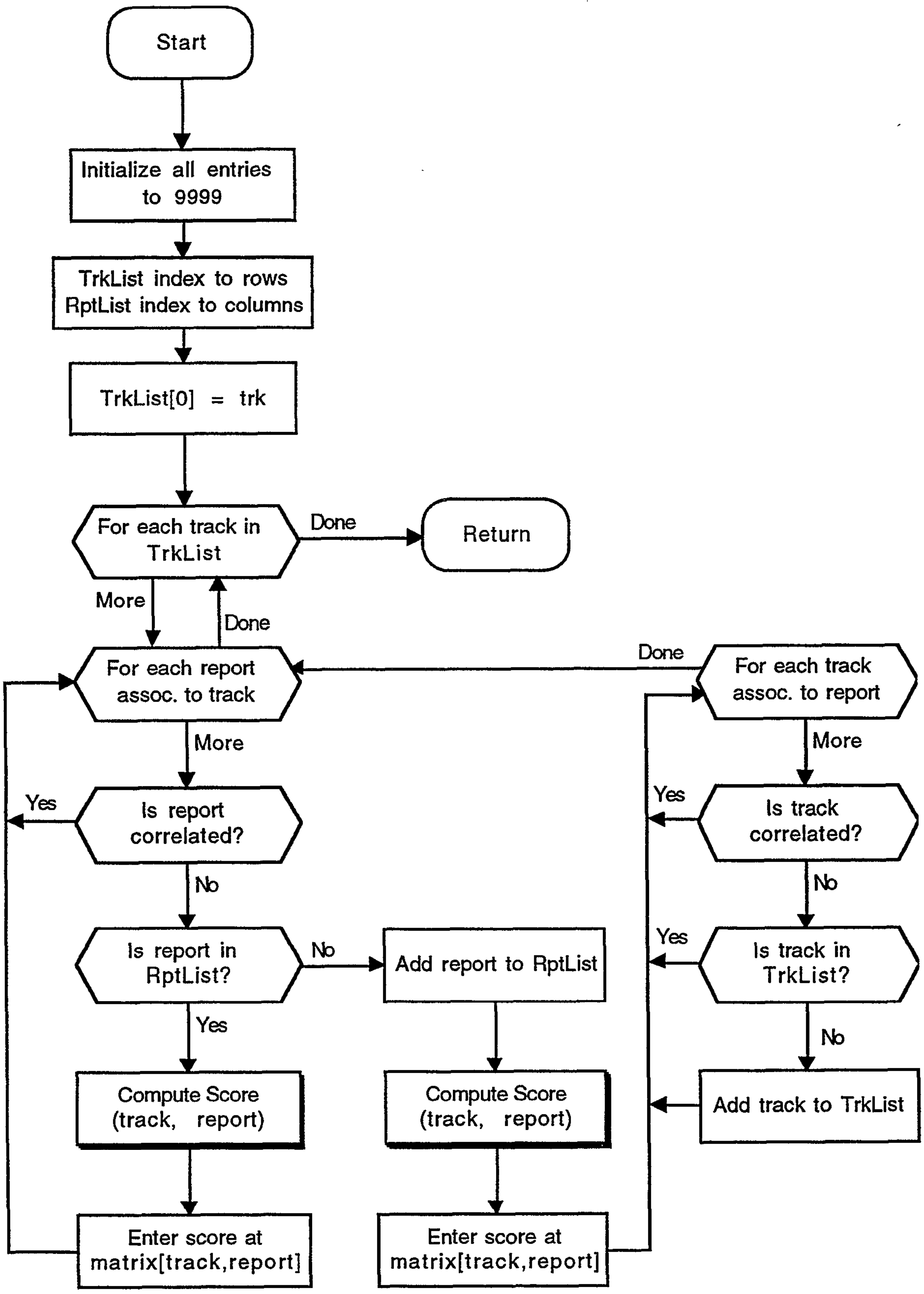

Start
Initialize all entries to 9999
TrkList index to rows RptList index to columns
TrkList[0] = trk
For each track in TrkList
Done
Return
More
Done
For each report assoc. to track
More
Yes
Is report correlated?
No
Is report in RptList?
No
Add report to RptList
Yes
Compute Score (track, report)
Compute Score (track, report)
Enter score at matrix[track,report]
Enter score at matrix[track,report]
Done
For each track assoc. to report
More
Yes
Is track correlated?
No
Yes
Is track in TrkList?
No
Add track to TrkList

Figure 16. Build association matrix.

5.3.2.8 Modified Munkres Algorithm

Called by:	Conflict Resolution
Calls:	Mark Pair
Purpose:	Reduce the association matrix to a set of optimal report-track correlations.

After the association matrix is defined, it is necessary to reduce the matrix in some manner to define the best set of correlations from the closed set of cross associations. For the 9-PAC, the optimal solution has been defined to be the set of correlations with the minimum total association measure scores. If the association measure score is considered to be a multi- dimension distance, the desire is to minimize the total distance for a given set of cross associations.

The method for minimizing the total distance is the Modified Munkres Algorithm. This method gives the same result as the Munkres Algorithm [6] in almost all situations, however, it is computationally much simpler. For an *mxn* rectangular matrix, the process is started with the lesser dimension. If there are *m* rows and *m<n*, rows are addressed first. For an *nxn* square matrix, tracks (which are rows) are addressed first. For each row the difference is computed between the lowest and the second lowest score. Boxes with no association measure score are considered to be a very large number like 9999.

Consider the matrix developed in Section 5.3.2.7 and shown again here as part of Table 7. The last three columns of the table are the second lowest score, the lowest score, and the difference between the two for each row of the association matrix. The computed differences for this matrix are 4, 1, and 9995. The row with the largest difference is correlated first with it's best associating column. In this case row 3 has the largest difference, 9995, so Track C is correlated with Report C. Note that since Track C only has one associating report, by assigning 9999 to all of the other boxes, it assured Track C would be correlated first to its one and only associating report, assuming all other tracks have more than one association. If more than one track shared a single report as their only association, this method assures that the best track is correlated first with that report. Track C and Report C are removed from the matrix after being correlated, and the process is repeated with the remaining reports and tracks.

Table 7. Solving an association matrix.

	Report A	Report B	Report C	2nd Low	Low	Diff
Track A	7	3	9999	7	3	4
Track B	6	3	2	3	2	1
Track C	9999	9999	4	9999	4	9995

Table 8 shows the reduced matrix. Again the last three columns are the second lowest score, the lowest score, and the difference between them for each row. The differences are 4 and 3. The row with the largest difference is row 1, so Track A is correlated with its best report, Report B. Track A and Report B are removed from the matrix which leaves Track B and Report A as shown in Table 9. They are correlated with each other.

Table 8. Solving an association matrix.

	Report A	Report B	2nd Low	Low	Diff
Track A	7	3	7	3	4
Track B	6	3	6	3	3

Table 9. Solving an association matrix.

	Report A	2nd Low	Low	Diff
Track B	6	-	-	-

The net effect of this process leaves a total association measure score of (4 for Track C::Report C) + (3 for Track A::Report B) + (6 for Track B::Report A) = (4 + 3 + 6) = 13. Inspection shows this is the minimum possible score possible given this set of cross associations. It is the same answer obtained with the more complex Munkres Algorithm. Study has shown that for more complex cross associations, the same generally holds true.

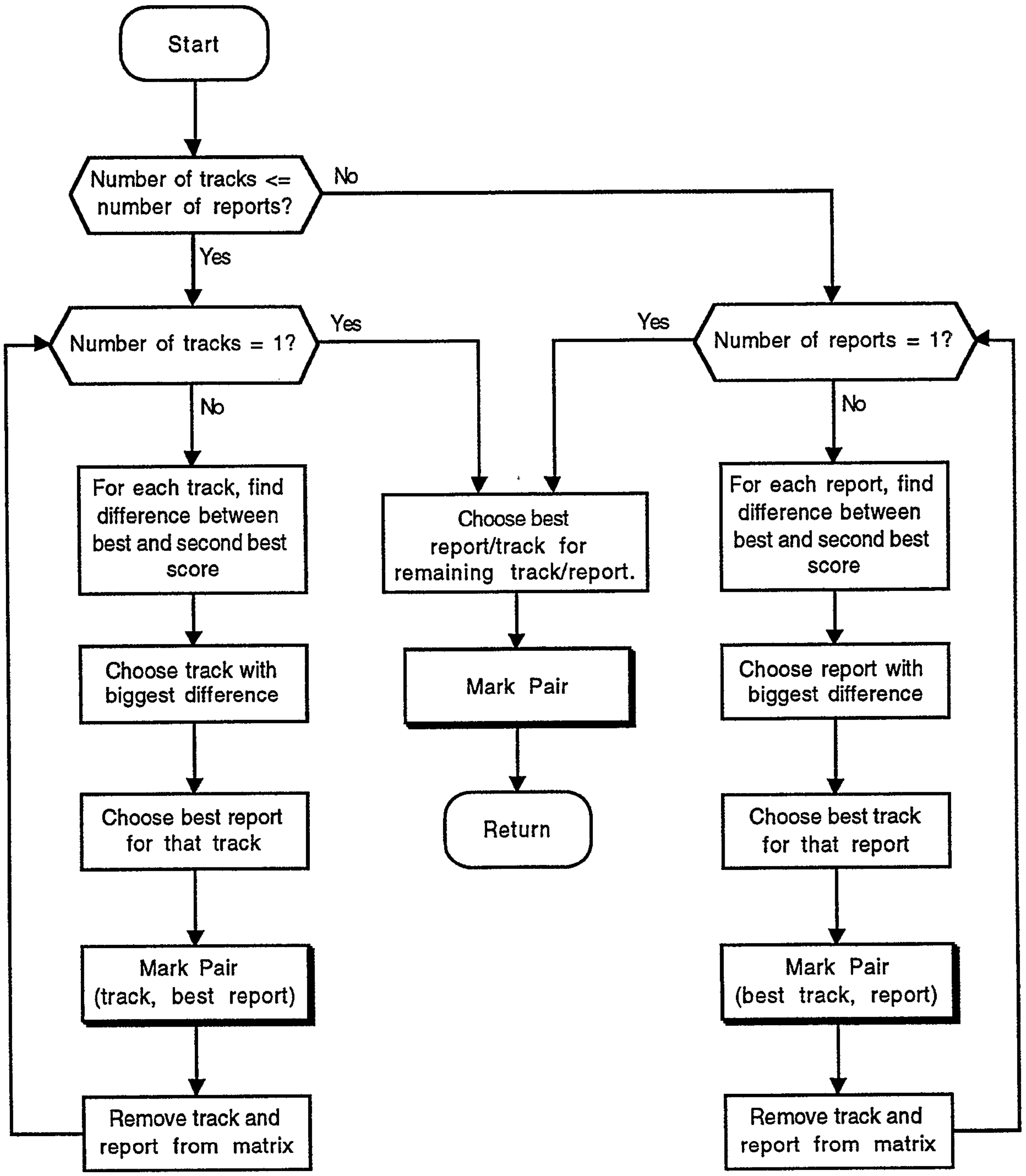

Start
Number of tracks <= number of reports?
No
Yes
Number of tracks = 1?
Yes
No
For each track, find difference between best and second best score
Choose track with biggest difference
Choose best report for that track
Mark Pair (track, best report)
Remove track and report from matrix
Number of reports = 1?
Yes
No
For each report, find difference between best and second best score
Choose report with biggest difference
Choose best track for that report
Mark Pair (best track, report)
Remove track and report from matrix
Choose best report/track for remaining track/report.
Mark Pair
Return

Figure 17. Modified Munkres Algorithm.

5.3.2.9 *Mark Pair*

Called by:	Possibly Correlate
	Ready to Correlate
	Beacon Split
	Conflict Resolution
	Modified Munkres Algorithm
Calls:	
Purpose:	Marks correlation between a report and a track, and removes all other associations.

When a report and track are correlated to each other, the correlation needs to be flagged and all associations to other unused reports and tracks should be broken. This is accomplished in the following manner. Each report, which is associated to the track of interest, has a pointer to the track. These pointers are removed. Next the pointers from the track of interest to these reports are removed. Note, the reports had to be handled first or else there would have been no way to find the reports once their pointers were removed from the track.

The process is repeated for the tracks which associated to the report of interest. After all of the pointers are removed from the tracks to the report, the pointers from the report to the track are removed.

Flags are set for the report and the track indicating they have been correlated. Finally, a special pointer is set from the track to the report to be used when updating the track predictions.

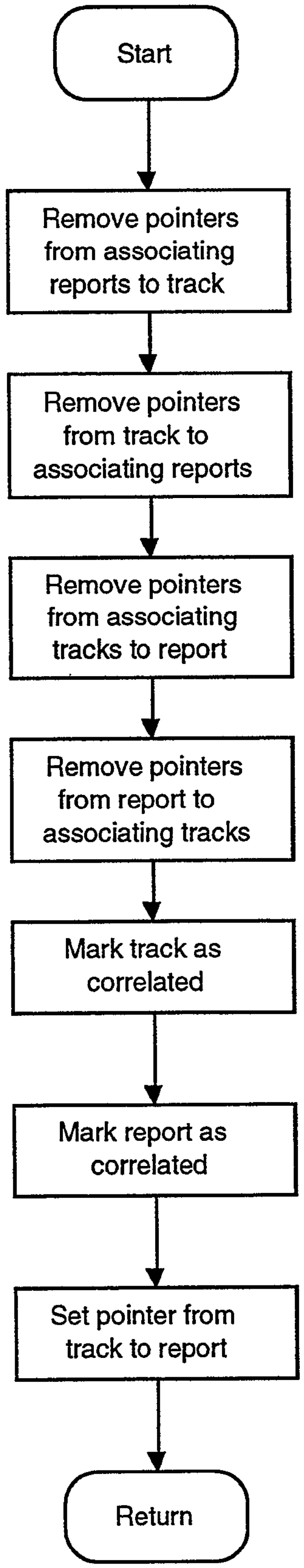

Start
Remove pointers from associating reports to track
Remove pointers from track to associating reports
Remove pointers from associating tracks to report
Remove pointers from report to associating tracks
Mark track as correlated
Mark report as correlated
Set pointer from track to report
Return

Figure 18. Mark pair.

5.3.3 Update Tracks

Called by:	Track
Calls:	Track Coast
	Track Update
Purpose:	Determines if a report should be updated by a correlating report or coasted.

The Update process steps through all of the Active Tracks. If a track has been marked for coasting, the Track Coast routine is called. If a track has been flagged as correlated, the Track Update routine is called. If neither has occurred, the track must not be old enough for update and the next track on the list is considered.

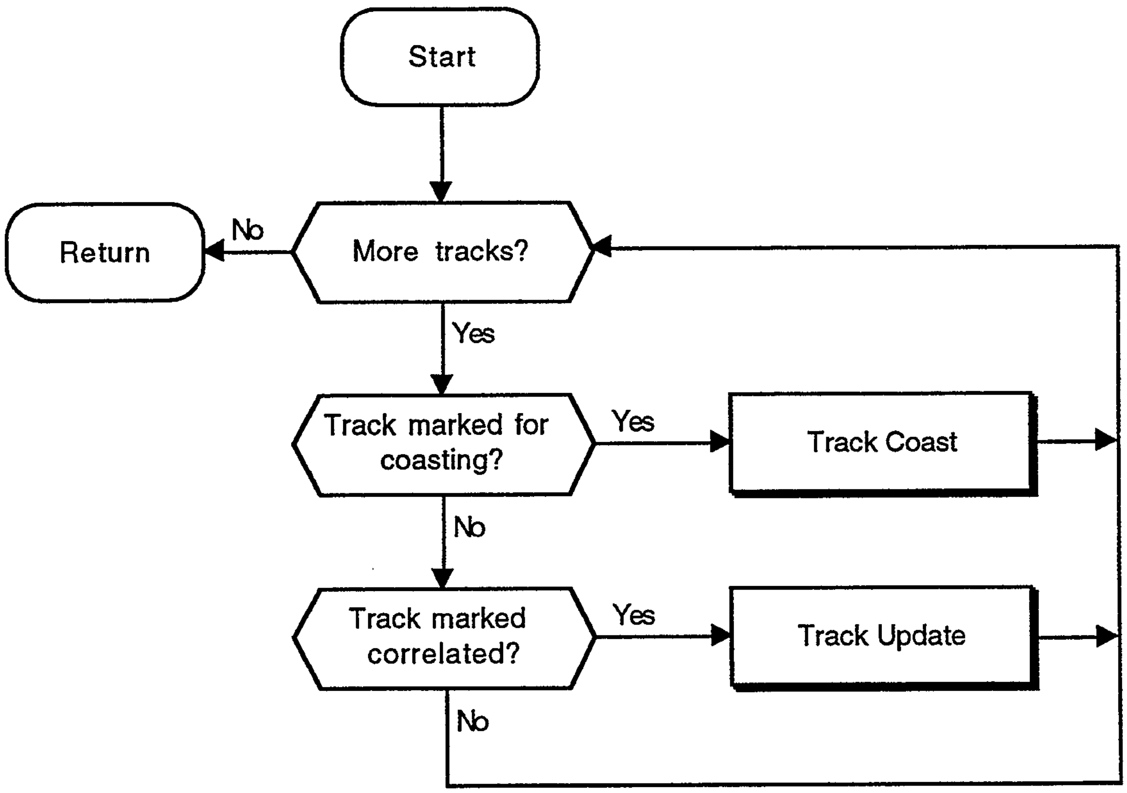


Figure 19. Update tracks.

5.3.3.1 *Track Update*

Called by:	Update Tracks
Calls:	Determine Gains
	Update Predictions
	Compute Velocity
	Set Track Type
	Check Minimum Distance
	Update Track State
	Compute Association Boxes
	Update Gains
Purpose:	Step through all of the processes necessary to determine the best filter gains and predict where the track will be on the next scan.

Given a track and a correlating report, a series of steps are followed to assure the report really belongs to the track, the degree to which the report should affect the track, and where the track is likely to be on the next scan. First, the new gains are determined as a function of the report Quality. The new gains are used to update the track predictions. A velocity check is made to assure the track is behaving as a real aircraft could. If the velocity is not reasonable, the correlation is broken and the track is coasted.

Finally, some bookkeeping is done. The track type is compared to the report type, and changes are made if appropriate. The track state is updated according to the state diagram. The filter gains are adjusted based on the track residuals and the report Quality. And the new association boxes are drawn based on the new predictions and the track state.

After all of the update processing has been completed, the track is moved back to the azimuth-ordered General Track list.

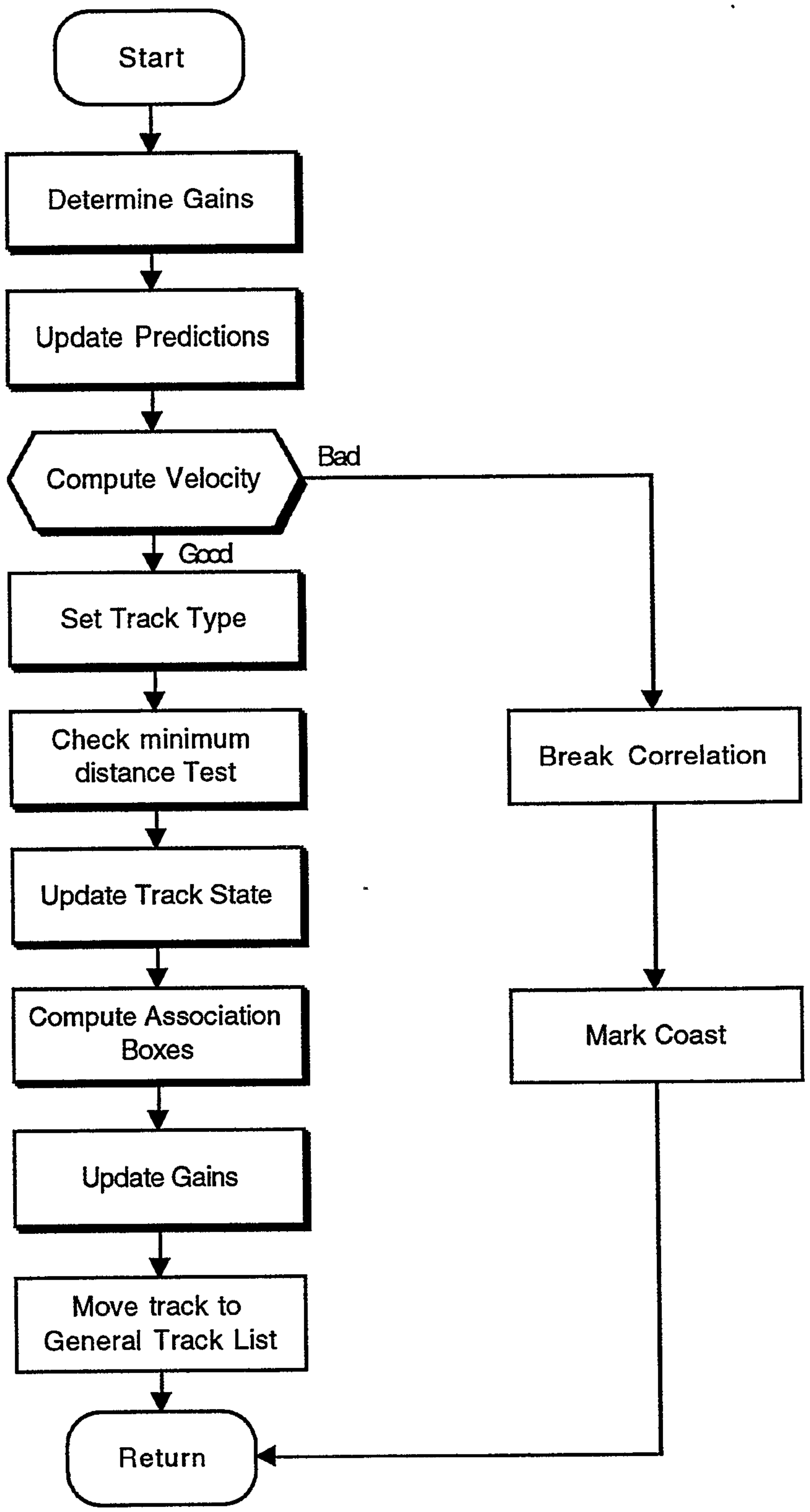


Figure 20. Track update.

5.3.3.2 Determine Gains

Called by:	Track Update
Calls:	
Purpose:	Determine the filter gains, αx, αy, βx, and βy, to be used for the alpha-beta track prediction.

The Tracker is an alpha-beta tracker in Cartesian coordinates. The gains in *x* and *y* are maintained independently from each other.

Beta is a simple function of alpha as shown in equation 6. It is bottom limited to 0.10 to prevent it from becoming too small and losing all effect on the velocity errors.

$$\beta_x = \frac{2}{3}\left(\frac{\alpha_x^2}{(2-\alpha_x)}\right) \quad \text{and} \quad \beta_y = \frac{2}{3}\left(\frac{\alpha_y^2}{(2-\alpha_y)}\right) \tag{6}$$

Alpha is updated every scan. In general, if alpha is large, the tracker is considered to be loose and will quickly follow the new data. If alpha is small, the tracker is considered to be stiff, and will maintain the existing trajectory to a higher degree. The challenge is to set alpha at a level where the predictions are not too loose and jump wildly with bad data, but does not miss maneuvering targets because of stiffness.

The theory used to set alpha in this tracker is as follows. If the reports used to update a track are considered to be accurate, the track should become more stable and trustworthy. As the track becomes more stable and trustworthy, it is beneficial to keep alpha low and the track stiff. However, if a track is too stiff, it may lose a maneuvering or drifting target. If this happens, it is beneficial to increase alpha and follow the measured data more closely.

These two goals are achieved in the following manner. The best measure of report accuracy is in the Quality field. This field is directly related to the azimuth accuracy of the report as is shown in Table 10. If the report has Quality = 3, the report is considered accurate. If an accurate report is used to update a track, the track becomes more trustworthy and alpha should be lowered. If a report with Quality = 3 is used to update the track, alpha is decremented by 0.1. To avoid having the track become too stiff, alpha is bottom limited to 0.25.

Table 10. Quality field and azimuth accuracy.

Quality	0	1	2	3	4
Azimuth Accuracy (mrad.)	7.0	2.9	2.4	1.5	2.0

If the track is too stiff, or if there is a maneuvering target, the position predictions will most likely begin falling consistently to one side of the measured data. By maintaining a low pass filtered measurement of the accumulated errors between the predicted position and the measured position, it is possible to detect if this problem is occurring. This low pass filtered measurement is the residual. If the residuals exceed a certain level, the track is too stiff and alpha needs to be increased. If the residuals exceed 0.05, alpha is increased by 0.15. If the residual exceeds 0.10, alpha is increased by 0.25. To avoid having the track become too loose, alpha is top limited to 0.90.

In addition to varying alpha based on the track history (Quality of reports used to form the track, and track residuals), alpha is also temporarily modified on a scan-scan basis depending on the individual report used to update the track. As shown in Table 10, the accuracy of the report is directly related to the Quality field of the report. If a very poor Quality report is used to update a track, it is desirable to further stiffen the track to avoid being overly influenced by a bad report. If a very good report is used to update a track, it is desirable to loosen the track to assure the track is sufficiently influenced by the good report. These modifications to alpha are determined each scan, are dependent solely on the Quality of the report, and are not maintained for further influencing of alpha or the track history. The actual temporary modifications are shown in Table 11. If Quality = 0, temporarily reduce alpha by 0.1 and reduce the report's influence. If Quality = 3, temporarily increase alpha by 0.2 and increase the report's influence.

Table 11. Quality field and effect on Tracker gains.

Quality	0	1	2	3	4
Alpha mod.	-0.1	0.0	0.1	0.2	0.0

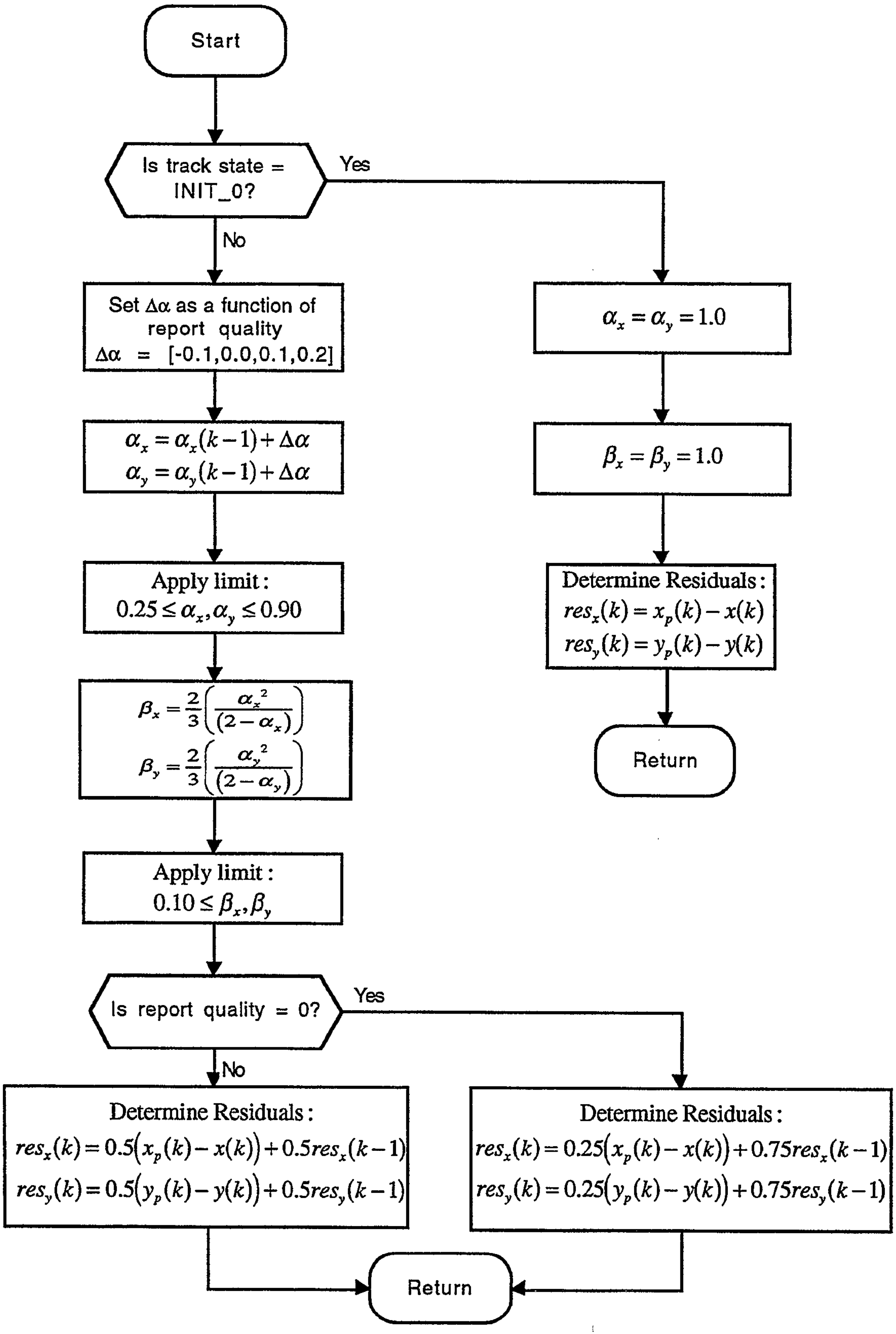

Start
Is track state = INIT_0?
Yes
No
Set Δα as a function of report quality
Δα = [-0.1,0.0,0.1,0.2]
$\alpha_x = \alpha_x(k-1) + \Delta\alpha$
$\alpha_y = \alpha_y(k-1) + \Delta\alpha$
Apply limit:
$0.25 \le \alpha_x, \alpha_y \le 0.90$
$\beta_x = \frac{2}{3}\left(\frac{\alpha_x^2}{(2-\alpha_x)}\right)$
$\beta_y = \frac{2}{3}\left(\frac{\alpha_y^2}{(2-\alpha_y)}\right)$
Apply limit:
$0.10 \le \beta_x, \beta_y$
Is report quality = 0?
Yes
No
Determine Residuals:
$res_x(k) = 0.5(x_p(k) - x(k)) + 0.5res_x(k-1)$
$res_y(k) = 0.5(y_p(k) - y(k)) + 0.5res_y(k-1)$
Determine Residuals:
$res_x(k) = 0.25(x_p(k) - x(k)) + 0.75res_x(k-1)$
$res_y(k) = 0.25(y_p(k) - y(k)) + 0.75res_y(k-1)$
Return
$\alpha_x = \alpha_y = 1.0$
$\beta_x = \beta_y = 1.0$
Determine Residuals:
$res_x(k) = x_p(k) - x(k)$
$res_y(k) = y_p(k) - y(k)$
Return

Figure 21. Determine gains.

5.3.3.3 Update Predictions

Called by:	Track Update
Calls:	
Purpose:	Given filter gains, αx, αy, βx, and βy, update the track predictions.

The Tracker is an alpha-beta tracker in Cartesian coordinates. The alpha-beta filter gains are determined as described in Section 5.3.3.2. The track predictions are updated with Equations 3, 4 and 5.

$$x_s(k) = x_p(k) - \alpha_x\left[x_p(k) - x_o(k)\right] \qquad (7)$$

$$\dot{x}_s(k) = \dot{x}_s(k-1) - \frac{\beta_x}{T}\left[\left[x_p(k) - x_o(k)\right]\right] \qquad (8)$$

$$x_p(k+1) = x_s(k) + \dot{x}_s(k)T \qquad (9)$$

$$y_s(k) = y_p(k) - \alpha_y\left[y_p(k) - y_o(k)\right] \qquad (10)$$

$$\dot{y}_s(k) = \dot{y}_s(k-1) - \frac{\beta_y}{T}\left[\left[y_p(k) - y_o(k)\right]\right] \qquad (11)$$

$$y_p(k+1) = y_s(k) + \dot{y}_s(k)T \qquad (12)$$

where

$\alpha_x, \alpha_y, \beta_x, \beta_y$: filter gains

x_p, y_p: predicted positions

x_0, y_0: observed positions

x_s, y_s: smoothed positions

$\dot{x}_s, \dot{y}_s$: smoothed velocities

T: time between updates

A comparison was made to a Kalman filter tracker. The alpha-beta tracker with variable gains proved to be more robust and less computationally intensive than the Kalman filter. The Kalman filter was a better long-term predictor. But the 9-PAC Tracker is a scan-scan correlator and is only looking one scan ahead. The improved long-term predictions were not a significant benefit.

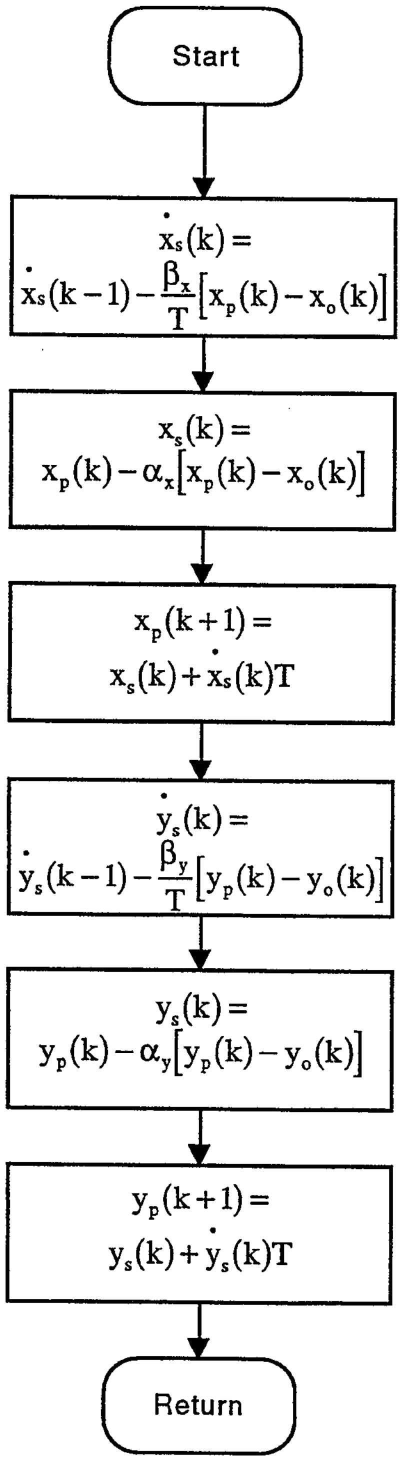

Start
$\dot{x}_s(k) = \dot{x}_s(k-1) - \frac{\beta_x}{T}\left[x_p(k) - x_o(k)\right]$
$x_s(k) = x_p(k) - \alpha_x\left[x_p(k) - x_o(k)\right]$
$x_p(k+1) = x_s(k) + \dot{x}_s(k)T$
$\dot{y}_s(k) = \dot{y}_s(k-1) - \frac{\beta_y}{T}\left[y_p(k) - y_o(k)\right]$
$y_s(k) = y_p(k) - \alpha_y\left[y_p(k) - y_o(k)\right]$
$y_p(k+1) = y_s(k) + \dot{y}_s(k)T$
Return

Figure 22. Update predictions.

5.3.3.4 Compute Velocity

Called by:	Track Update
Calls:	
Purpose:	Determine various track velocities and check for reasonableness. If the velocities or acceleration are not reasonable, the test is failed.

Instantaneous and average velocities and instantaneous and average radial velocities are computed and compared. The instantaneous radial velocity is based on the updating report and the last report to update the track. The instantaneous velocity is based on the smoothed Cartesian velocities determined by the alpha-beta predictor, $\dot{x}_{sm}$ and $\dot{y}_{sm}$. The average radial velocity and average velocity are computed over time with a simple, low pass filter. An instantaneous acceleration is also computed based on the new and previous average velocity.

Comparisons are then made between the various computed velocities and acceleration to determine if the track is behaving reasonably for an aircraft. If the difference between the average velocity and instantaneous velocity is greater than 40 knots, and the difference between the average radial velocity and the instantaneous radial velocity is greater than 40 knots, the velocity reasonableness test fails. Such significant differences between the instantaneous and average velocities indicates an anomaly with the updating report.

If the average radial velocity is more than 1.2 times the average velocity, the velocity reasonableness test fails. The average velocity should always be greater than the average radial velocity since the radial velocity is a component of the velocity. However, a report way out of line could result in the radial velocity temporarily appearing larger than the velocity. If this should occur, the test is failed. The factor of 1.2 compensates for the occasionally noisy data.

Finally, the acceleration is examined. If the acceleration is greater than 1/10 the average velocity or 0.33g, whichever is greater, the velocity reasonableness test fails. The exception to the this rule is if the report under consideration, or the previous report, was a Quality = 0 report. A low Quality report has poor azimuth accuracy and may make the instantaneous acceleration appear large. For this case, if the acceleration is greater than 1/10 the average velocity or 0.5g, whichever is greater, the velocity reasonableness test fails.

If the velocity reasonableness test is failed, the correlation is broken and the track is coasted.

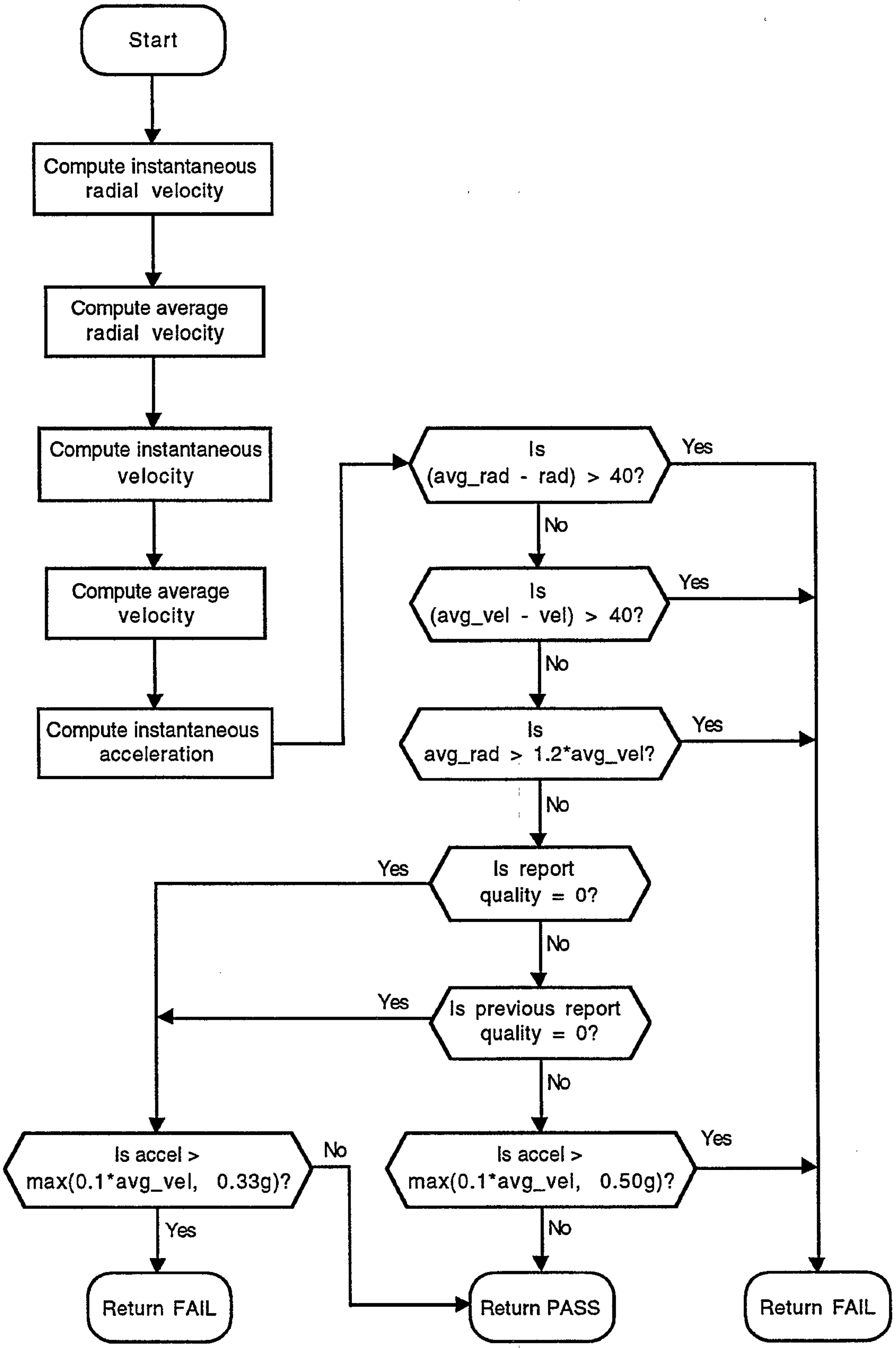

Start
Compute instantaneous radial velocity
Compute average radial velocity
Compute instantaneous velocity
Compute average velocity
Compute instantaneous acceleration
Is (avg_rad - rad) > 40?
Yes
No
Is (avg_vel - vel) > 40?
Yes
No
Is avg_rad > 1.2*avg_vel?
Yes
No
Is report quality = 0?
Yes
No
Is previous report quality = 0?
Yes
No
Is accel > max(0.1*avg_vel, 0.33g)?
No
Yes
Is accel > max(0.1*avg_vel, 0.50g)?
Yes
No
Return FAIL
Return PASS
Return FAIL

Figure 23. Compute velocity.

5.3.3.5 Set Track Type

Called by:	Track Update
Calls:	
Purpose:	Updates the track type, e.g., beacon-only, radar-only, radar-reinforced, given the report type used for the track update. Also updates the discrete code for beacon tracks.

The Tracker monitors the basic type of each track: beacon-only, radar-only, and radar-reinforced. It is necessary to know the track type because the track type is compared to the report type when computing the association measure score. And track types do change for various reasons: faulty transponders, blocking of the transponder antenna, and environmental issues. It is also possible for tracks to change their Mode 3/A code and to change from discrete to non-discrete and vice versa. All of these changes are monitored and recorded by this function.

To allow all tracks to be treated similarly, all radar tracks and reports are assigned a Mode 3/A code of 0000. If a track is updated by a report with a different Mode 3/A code, a counter is incremented. If this happens three updates in a row, the track's Mode 3/A code is set to match the last report's Mode 3/A code and the track type is set to match the report type. If the track and report code match on any update, the counter is reset to zero. This algorithm detects and handles all possible transitions: radar to beacon and vice versa, discrete to non-discrete and vice versa, and change of discrete code.

In addition to monitoring the Mode 3/A code, it is also necessary to monitor the Mode C altitude code. If the track is a radar-only track, altitude information is not available so the field is set to zero. If the track is a beacon-only or radar-reinforced track, it is likely to have altitude information. Since altitude information is not critical to the overall performance of the Tracker, a simple algorithm is used to update the altitude of the track. If the altitude of the report differs from the altitude of the track, the validity of the report's altitude field is checked. If the validity is equal to 3, the highest possible validity, the track altitude is set equal to the report altitude. Otherwise, the track altitude is left unchanged.

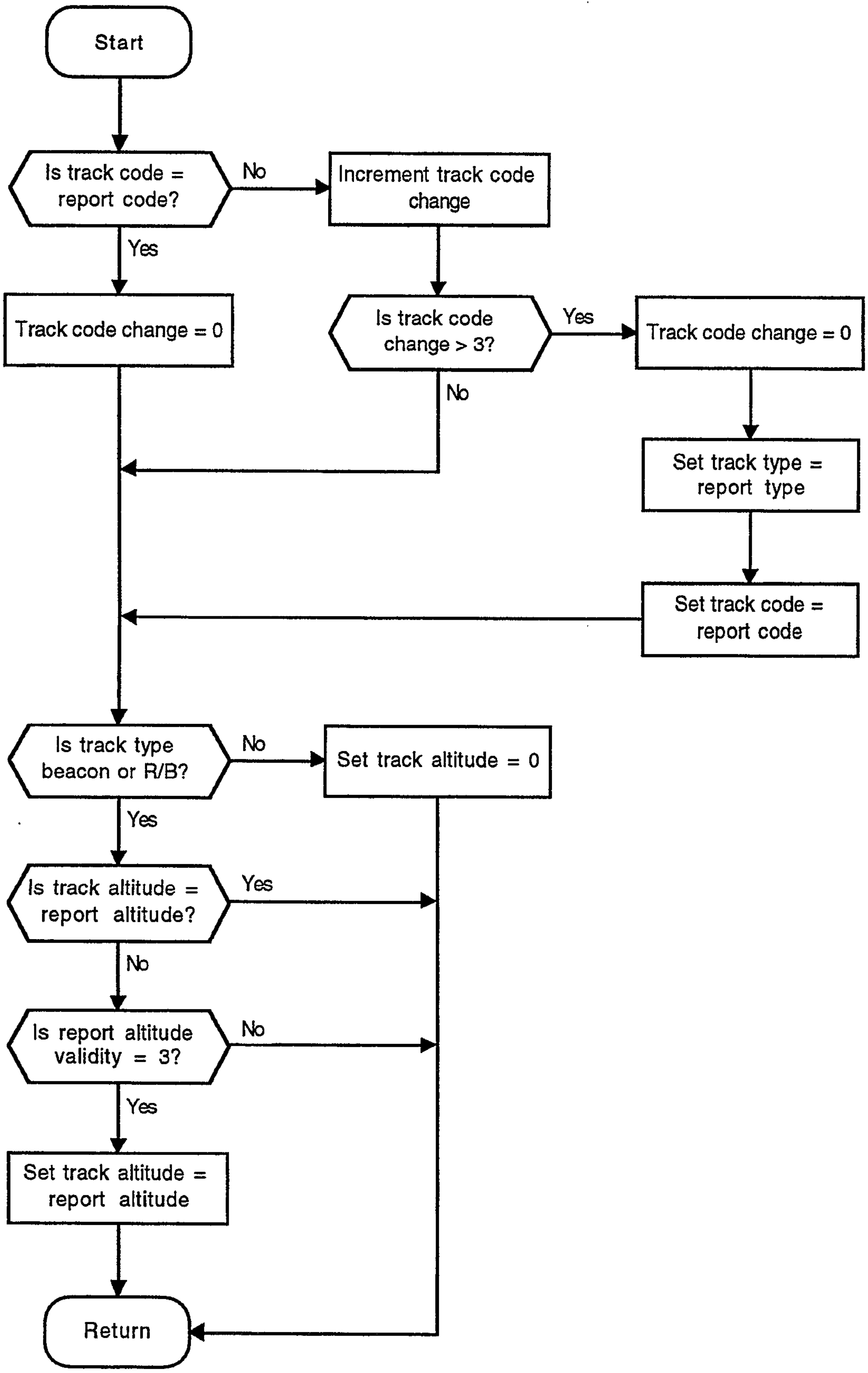

Start
Is track code = report code?
No
Increment track code change
Yes
Track code change = 0
Is track code change > 3?
Yes
Track code change = 0
No
Set track type = report type
Set track code = report code
Is track type beacon or R/B?
No
Set track altitude = 0
Yes
Is track altitude = report altitude?
Yes
No
Is report altitude validity = 3?
No
Yes
Set track altitude = report altitude
Return

Figure 24. Set track type.

5.3.3.6 *Minimum Distance Test*

Called by:	Track Update
Calls:	
Purpose:	Determines if a track has passed a minimum distance criteria. This test effects the track's path through the state diagram.

The purpose of the minimum distance test is to reduce false tracks due to radar-only clutter. The most serious breakthrough clutter tends to occur close to the sensor. For this reason, the minimum distance test is not applied to tracks beyond a certain distance (VSP dependent, typically 20 nmi.) or to beacon tracks.

For processing purposes, the first check made is to see if the test has already been passed which is indicated by a flag. If the flag is set, the test is complete. Tracks which are not subject to the test, automatically have this flag set.

The test is applied in Cartesian coordinates and independently for the *x* and *y* dimensions. If the minimum distance is moved in either dimension, the test is passed. The size of the distance is determined by a VSP (typically 0.25 nmi.) If the test is not passed, the flag is not set. The result of the test does not affect the report-track correlation. Its only effect is on the path through the state diagram.

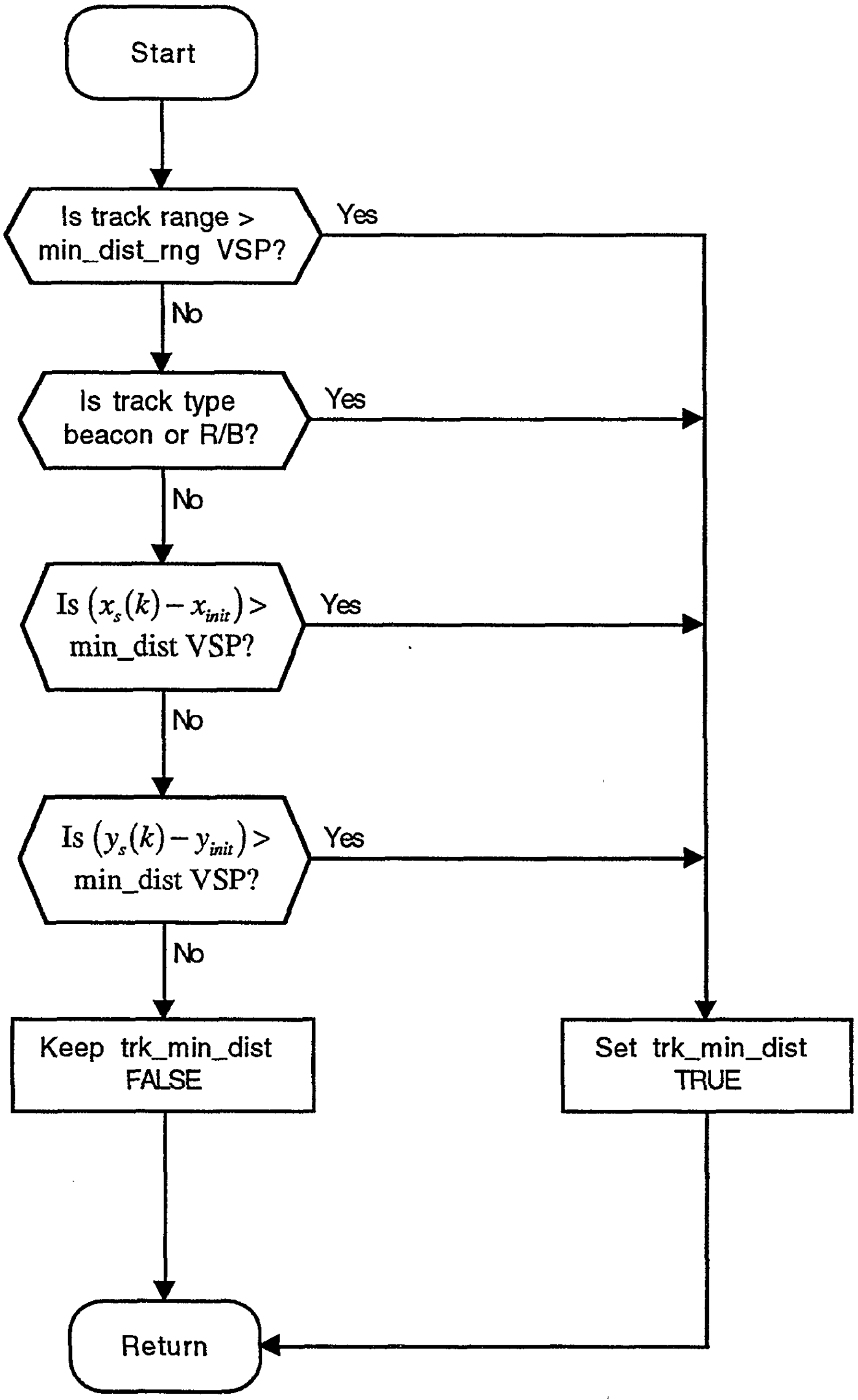
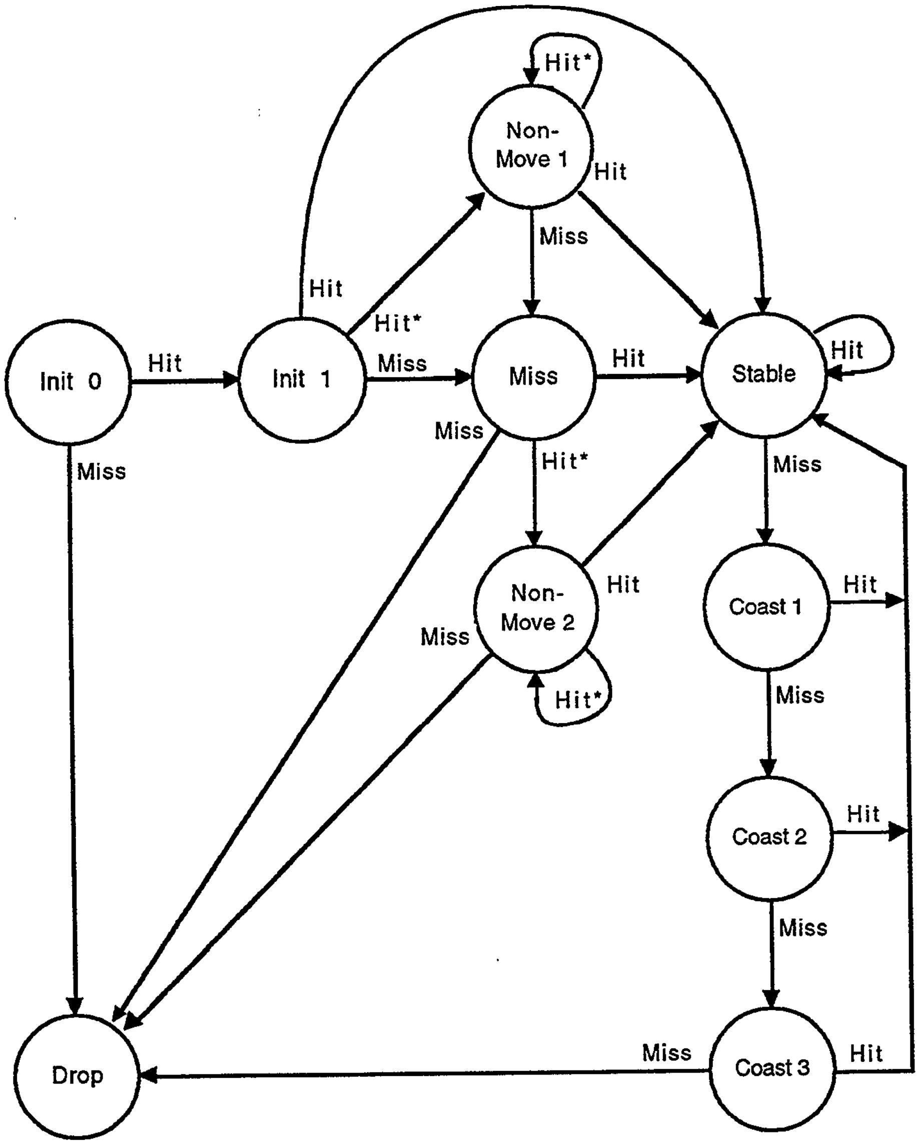

Start
Is track range > min_dist_rng VSP?
Yes
No
Is track type beacon or R/B?
Yes
No
Is $(x_s(k) - x_{init}) >$ min_dist VSP?
Yes
No
Is $(y_s(k) - y_{init}) >$ min_dist VSP?
Yes
No
Keep trk_min_dist FALSE
Set trk_min_dist TRUE
Return

Figure 25. Minimum distance test.

5.3.3.7 Update Track State

Called by:	Track Update
Calls:	
Purpose:	Guides the Tracker state through the state diagram. The Tracker state affects association and update decisions.

The Tracker state diagram is shown in Figure 26. The track state is updated according to the diagram. The decision branches in the diagram are based on the following categories: hit or miss, minimum distance pass or fail, and minimum velocity pass or fail.



Miss: No correlating report
Hit: Correlating report
Hit*: Correlating report, but failed Min. Distance test or Min. Velocity test.

Figure 26. State diagram.

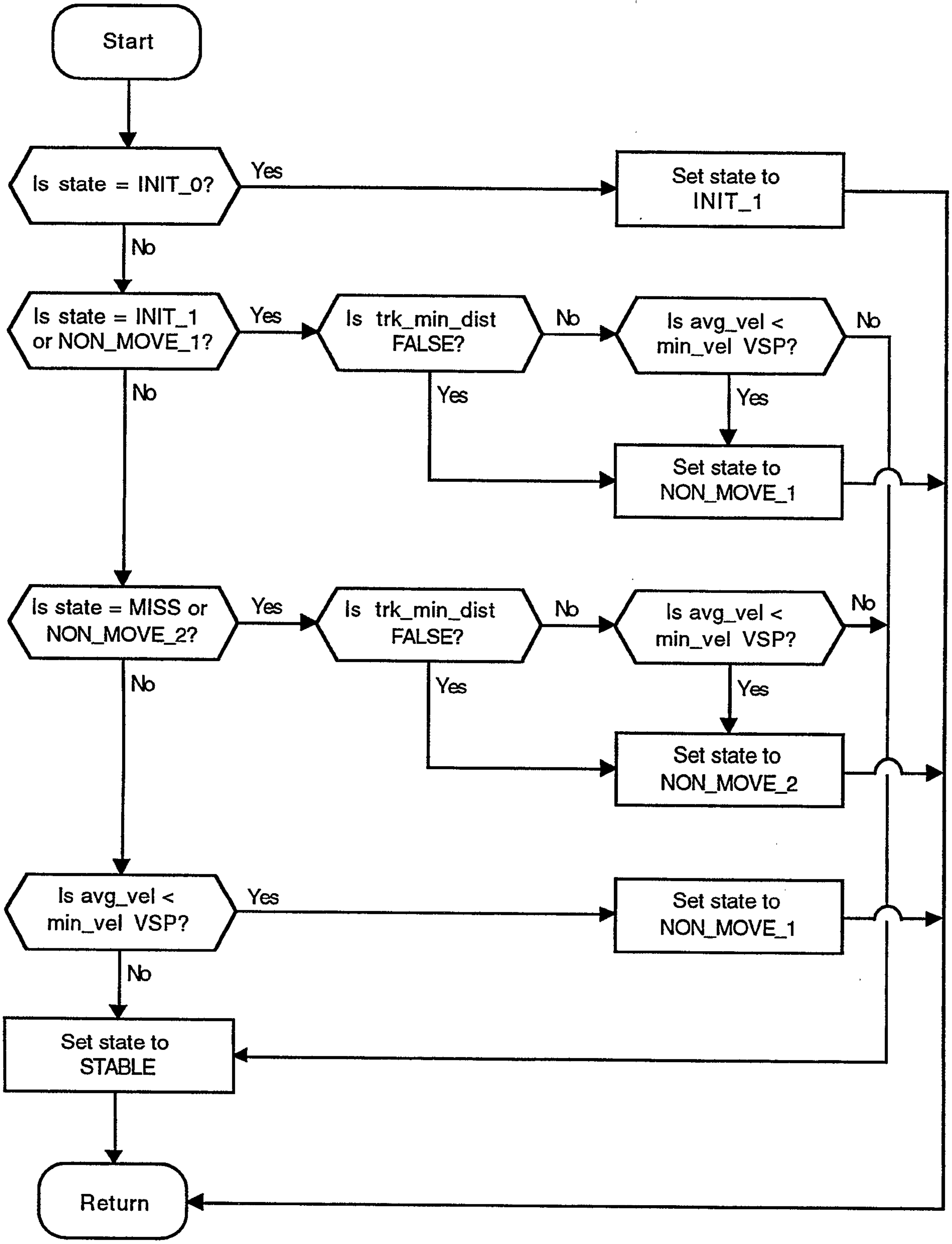


Figure 27. Update track state.

5.3.3.8 *Compute Association Boxes*

Called by:	Track Update
Calls:	
Purpose:	Set the limits for the association boxes.

The association boxes define the geographic region where candidate reports for correlation will be found. The size of the association boxes is dependent on the track state. The association boxes are drawn in the radar system coordinates since the radar system errors occur in the system coordinates. There are a maximum of two association boxes for each track: a zone 0 box corresponding to the system errors, and a zone 1 box corresponding to the system errors of zone 0 plus target maneuvering. For association purposes as discussed in Associate Tracks, a report need only to be in zone 1 to associate with the track. When correlation is performed with otherwise identical reports in zone 0 and zone 1, the report in zone 0 is given preference.

Initiating tracks only have one association box. For programming simplicity, however, two zones exist and zone 1 is set equal to zone 0. For initiating tracks, the original box allows for a 600 knot target traveling in any direction. After two points have been received, the box for initiating tracks is based on a two point interpolation with a margin for a 3 sigma error. (The sigmas are the system accuracies and have been computed to be 0.03125 nmi in range and 2.5 ACPs in azimuth)

As the track matures, the prediction is determined by the alpha-beta filter and two association boxes are used. The smaller box for zone 0 has margin for a 3 sigma error. The larger box for zone 1 encompasses the smaller box, plus adds margin for a maneuvering target. The maneuvering margin allows for a maximum g turn (a VSP, typically 1 G) or a 50% increase of zone 0 margins, whichever is larger. Both boxes also account for the track history and missed updates.

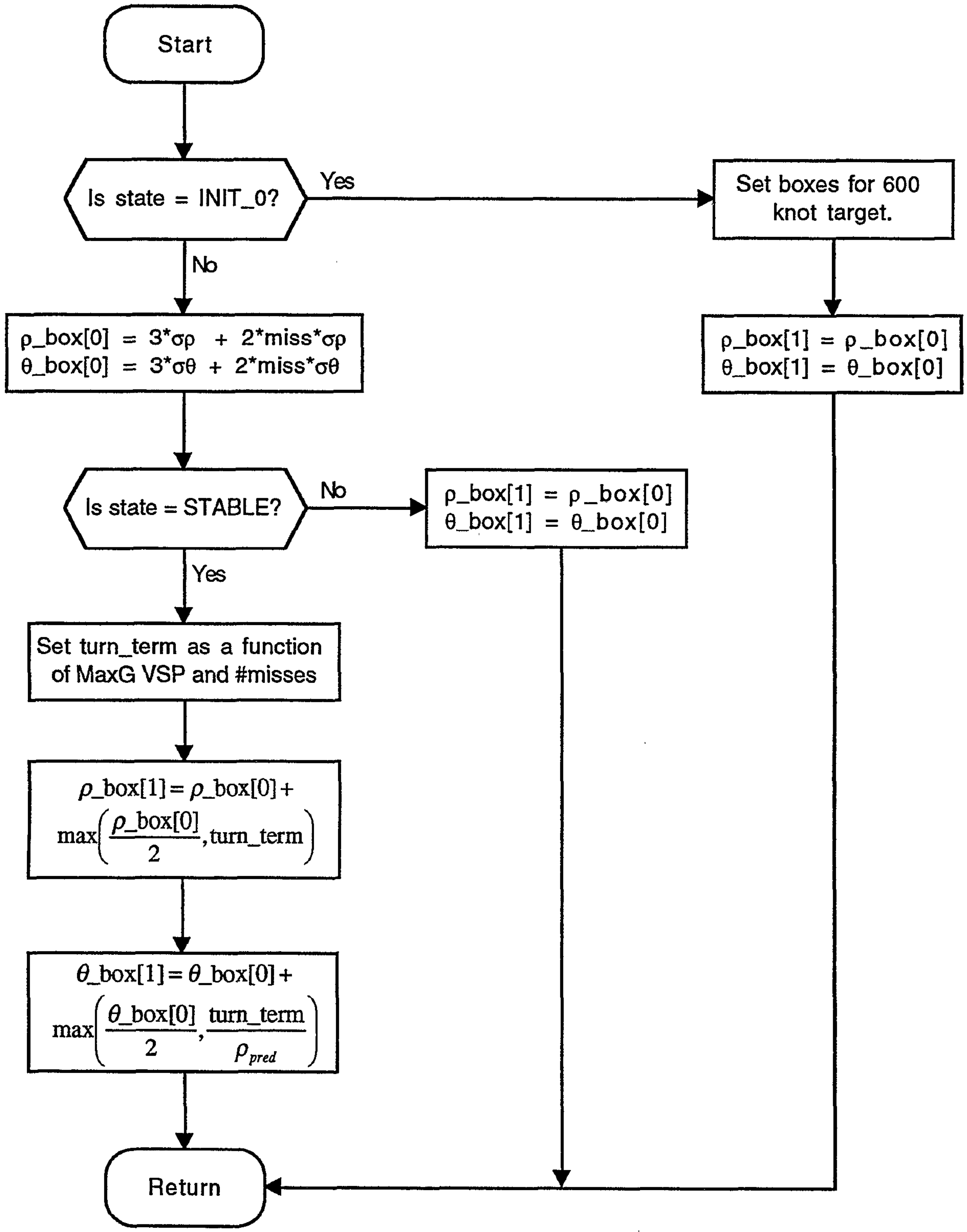

Start
Is state = INIT_0?
Yes
No
Set boxes for 600 knot target.
ρ_box[0] = 3*σρ + 2*miss*σρ
θ_box[0] = 3*σθ + 2*miss*σθ
ρ_box[1] = ρ_box[0]
θ_box[1] = θ_box[0]
Is state = STABLE?
No
Yes
ρ_box[1] = ρ_box[0]
θ_box[1] = θ_box[0]
Set turn_term as a function of MaxG VSP and #misses
$\rho\_box[1] = \rho\_box[0] + \max\left(\frac{\rho\_box[0]}{2}, turn\_term\right)$
$\theta\_box[1] = \theta\_box[0] + \max\left(\frac{\theta\_box[0]}{2}, \frac{turn\_term}{\rho_{pred}}\right)$
Return

Figure 28. Compute association boxes.

5.3.3.9 *Track Coast*

Called by:	Update Tracks
Calls:	Compute Association Boxes
Purpose:	Update the track prediction and perform bookkeeping when a track is coasted due to a lack of a satisfactory report for updating.

If no satisfactory report is found to update a track, the track is coasted. The track prediction is updated based on previous track history. The track state is updated according to the state diagram shown previously in Figure 26.

The track prediction is the previous prediction plus the estimated velocity. The prediction comes from the following equations:

$$x_s(k) = x_p(k) \qquad (13)$$

$$\dot{x}_s(k) = \dot{x}_s(k-1) \qquad (14)$$

$$x_p(k+1) = x_s(k) + \dot{x}_s(k)T \qquad (15)$$

$$y_s(k) = y_p(k) \qquad (16)$$

$$\dot{y}_s(k) = \dot{y}_s(k-1) \qquad (17)$$

$$y_p(k+1) = y_s(k) + \dot{y}_s(k)T \qquad (18)$$

Since the track is coasting, the confidence in the track validity decreases. The next report to update the track should be given an increased weighting. To accomplish this, the filter gain alpha is increased by half the margin to the maximum gain of 0.9.

If the track has coasted for three scans (state = COAST3), or is just initiating (state = INIT_0), or has missed for a second time without becoming stable (state = MISS or state = NON_MOVE_2), the track is dropped. The track is removed from the Active and General track lists and the allocated memory is returned to the stack.

If the track is initiating, has received at least two points, but has not become stable whether due to a lack of points or a lack of distance moved, the track state is set to MISS. Finally, if the track has been stable at anytime, the track state is incremented through the coasting states allowing for three misses before the track is dropped.

After the coasting track's prediction has been updated and its state updated, the track is removed from the Active Track list and placed on the azimuth-ordered General Track list.

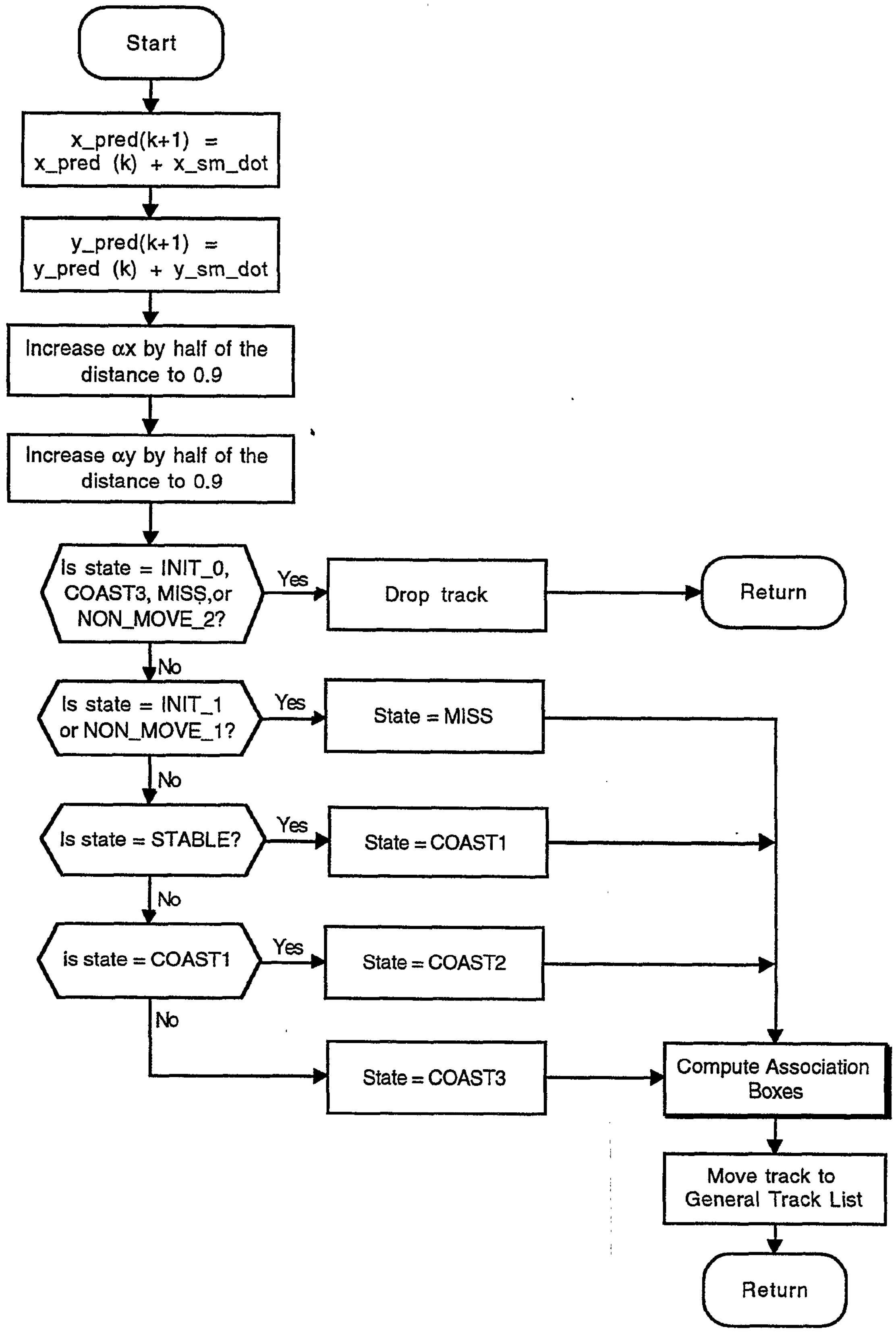

Start
x_pred(k+1) = x_pred (k) + x_sm_dot
y_pred(k+1) = y_pred (k) + y_sm_dot
Increase αx by half of the distance to 0.9
Increase αy by half of the distance to 0.9
Is state = INIT_0, COAST3, MISS,or NON_MOVE_2?
Yes
Drop track
Return
No
Is state = INIT_1 or NON_MOVE_1?
Yes
State = MISS
No
Is state = STABLE?
Yes
State = COAST1
No
is state = COAST1
Yes
State = COAST2
No
State = COAST3
Compute Association Boxes
Move track to General Track List
Return

Figure 29. Track coast.

5.3.4 Initiate Tracks

Called by:	Track
Calls:	Compute Association Boxes
Purpose:	Initialize all fields for an initiating track given the initial report.

A track can be initialized with a single unused report. If the report is a radar-only report, it must not be of Quality = 0 or Confidence <= 1 (see Section 5.3.2.4, Tables 3 and 4 for explanation of Quality and Confidence.) All other reports initiate tracks.

Since only a single report has been received, the track prediction for the next scan is the same as the received report. The smoothed velocity for the alpha-beta filter is zero.

The residuals are initialized to 0. The filter gains α_x and α_y are initialized to 1.0 to assure the track immediately follows the next received report. A track number is assigned and a counter is incremented even though track numbers or surveillance file numbers are not used by the ASR-9 at this time. Numerous other bookkeeping fields are initialized at this time.

After all the fields have been initialized, Compute Association Boxes is called to define where the next report to update the track is most likely to be found. After this is complete, the new track is entered in the azimuth-ordered General Track list. The new track is not output to the ATC user, however. Only stable tracks are output to the ATC user.

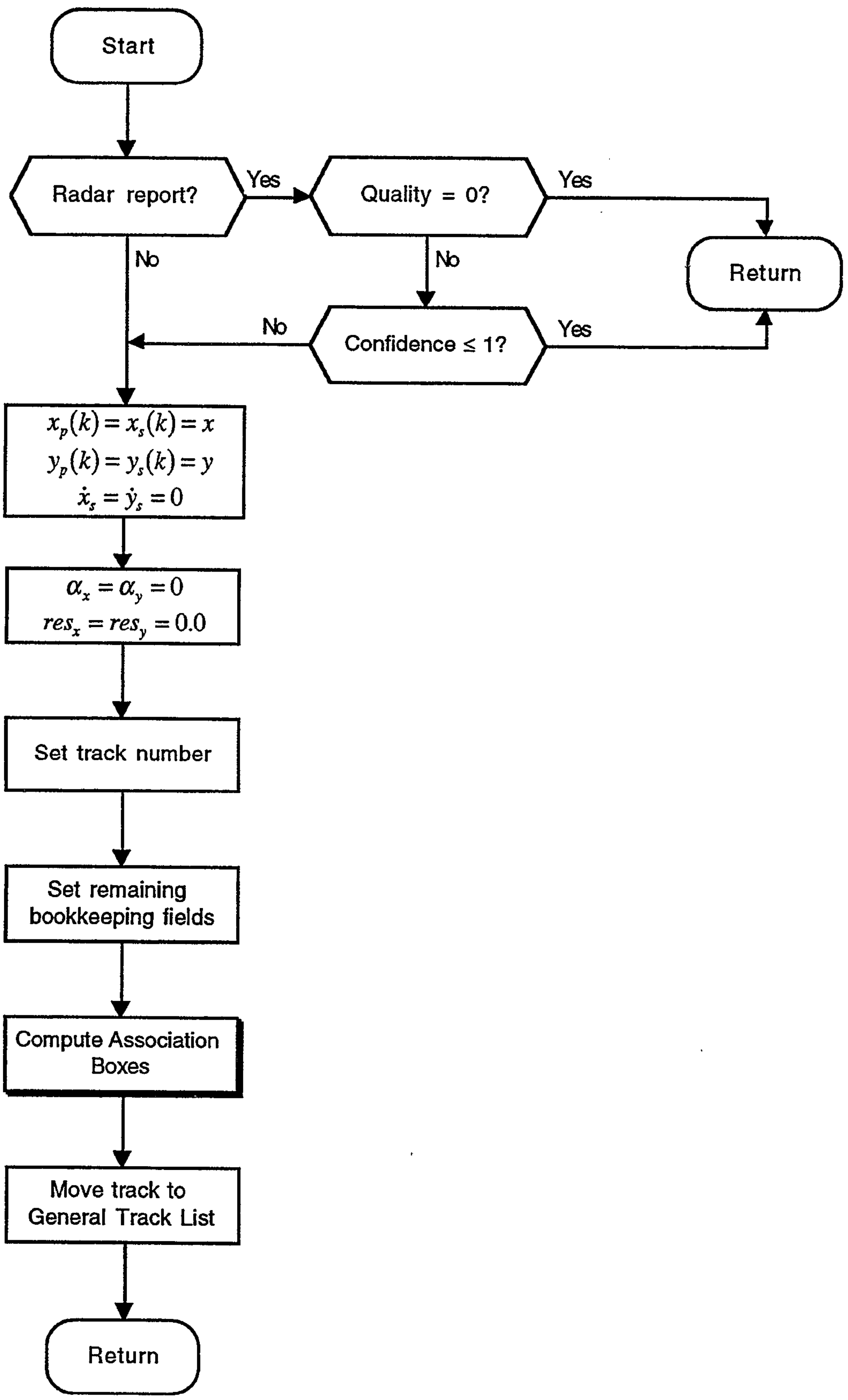


Figure 30. Initiate tracks.

AFTERWORD

This report has covered in detail the algorithms that are in the 9-PAC Scan-Scan Correlator or Tracker. The performance improvements will be addressed in a future document. This report is one in a series of reports that document the 9-PAC algorithms in enough detail to support implementation by a second party.

The Tracker algorithms described herein have been tested with recorded field data in a real-time test facility, are now undergoing field evaluation in ASR-9s at Lincoln Laboratory and Albuquerque. As these tests proceed algorithm refinements may occur. Refinements will be documented in a future report or issued as an addendum to this report.

APPENDIX A. DATA STRUCTURES

Tables 12 and 13 are the two main data structures used in the 9-PAC Tracker: reports (trkrptype) and tracks (trktype). Note that the structures are not dependent on the type of report or track. Beacon and radar reports are the same, and beacon and radar tracks are the same.

Table 12. Report Data Structure.

Field Name	Description
Node_header node_hdr;	Pointer;
/* common fields */	
int rpt_type;	Radar, Beacon, or RB
int scan;	For off-line analysis
int rtqc;	Set by BTD or C&I; Flag: 0 - no 1 - yes
int acp;	Set by BTD or C&I; Azimuth, 4096 ACPs = 360°
int run;	Set by BTD or C&I; Runlength
float range;	Set by BTD or C&I; range in nmi
float x;	Cartesian coordinate, nmi; uses alt. if available
float y;	Cartesian coordinate, nmi; uses alt. if available
/* beacon fields */	
int mode_a;	Set by BTD; Mode 3/A code; set to 0 for radar
int mode_2;	Set by BTD; Mode 2 code, typically 0
int alt;	Mode C code converted to feet; set to 0 for radar
int val_flags;	Set by BTD
int arts_qual;	Set by BTD
int false;	Set by BTD
/* radar fields */	
int quality;	Set by C&I; [0..3]; set to 4 for beacon
int conf;	Set by C&I; [0..5];
int elig;	Set by C&I; [0..3]; not used
float amp;	Set by C&I;
int lo_filt;	Set by C&I; [0..8] maps to [-3..+3]
int hi_filt;	Set by C&I; [0..10] maps to [-4..+4]
int lo_dopp;	Set by C&I; [0..63]
int hi_dopp;	Set by C&I; [0..63]

Table 12. Report Data Structure. (continued)

Field Name	Description
Node_header node_hdr;	Pointer;
int max_filt;	Set by C&I;
int flag1;	Set by C&I;
int flag2;	Set by C&I;
int adapt_thresh_info;	Set by C&I; Two words: flag plus threshold
	amplitude information; 0 - unflagged, 1- flagged
/* working fields */	
int no_init;	Flag: 0 - OK for initiation, 1 - not for initiation
int num_assoc;	Counter
int num_disc_assoc;	Counter
int corr;	Flag: 0 - not correlated, 1 - correlated
int output;	Flag: 0 - not output, 1 - output
TRK_TRK_T *assoc_trk[11];	Array of pointers to associating tracks

Table 13. Track Data Structure.

Field Name	Description
Node_header node_hdr;	Pointer
int trk_type;	Beacon, radar, or radar-reinforced; Set by initial report type;
int old_acp_1;	Last azimuth update (ACP)
float old_range_1;	Last range update (nmi)
int old_acp_2;	2nd to last azimuth update (ACP)
float old_range_2;	2nd to last range update (nmi)
int old_acp_3;	3rd to last azimuth update (ACP)
float old_range_3;	3rd to last range update (nmi)
int az_pred;	Predicted azimuth (ACP); from (x,y) prediction
float range_pred;	Predicted range (nmi); from (x,y) prediction
float x_pred;	Alpha-Beta filter output
float x_smooth;	Alpha-Beta filter output
float x_sm_dot;	Alpha-Beta filter output
float y_pred;	Alpha-Beta filter output
float y_smooth;	Alpha-Beta filter output
float y_sm_dot;	Alpha-Beta filter output
float x_init;	Position of first report; for min_dist test
float y_init;	Position of first report; for min_dist test
float alpha_x;	Alpha-Beta filter gain
float alpha_y;	Alpha-Beta filter gain
float beta_x;	Alpha-Beta filter gain
float beta_y;	Alpha-Beta filter gain
float resid_x;	Residual; filtered diff. between x_pred & report
float resid_y;	Residual; filtered diff. between y_pred & report
float range_box_max[2];	Association box upper limit; zone 0 and 1
float range_box_min[2];	Association box lower limit; zone 0 and 1
int az_box_max[2];	Association box upper limit; zone 0 and 1
int az_box_min[2];	Association box lower limit; zone 0 and 1

Table 13. Track Data Structure.
(continued)

Field Name	Description
float vel;	instantaneous velocity
float avg_vel;	filtered velocity
float rad_vel;	instantaneous radial velocity
float avg_rad_vel;	filtered radial velocity
float accel;	instantaneous acceleration
/* beacon specific fields */	
int code;	Mode 3/A code; set to 0 for radar
int discrete;	Flag: 0 - non-discrete, 1 - discrete
int code_change;	Counter
int alt;	Mode C converted to feet; set to 0 for radar
int split;	Counter
/* track status fields */	
int trk_num;	Track file number
int state;	Track state - see state diagram
int qual_old;	Last update's Quality; used for velocity test
int miss_1;	# of misses since last update
int miss_2;	# of misses between last and 2nd to last update
int miss_3;	# of misses between 2nd to last and 3rd to last update.
int hit;	Total # of hits on track; for statistics
int age;	Total # of possible hits on track; for statistics
int min_dist;	Flag: 0 - not passed test, 1 - passed test;
int force_coast;	Flag: 0 - not forced, 1 - forced; for beacon split
int coast;	Flag: 0 - not coasted, 1 - coasted;
int corr;	Flag: 0 - not correlated, 1 - correlated;
int available;	Flag: 0 - not avail, 1 - avail; for Gen. to Act. list
int high_score;	Flag: 0 - not too high, 1 - too high; for fast corr.
int num_assoc;	Counter
int num_disc_assoc;	Counter
TRK_RPT_T *assoc_rpt[11];	Array of pointers to associating reports
TRK_RPT_T *corr_rpt;	Pointer to correlating report

APPENDIX B. VARIABLE SITE PARAMETERS

The following are the Variable Site Parameters (VSPs) used by the 9-PAC Tracker.

Table 14. Variable System Parameters.

VSP Name	Description	Nominal Value
RANGE	Maximum range for which minimum distance test applies.	20 nmi
SCAN RATE	Antenna scan rate measured in seconds. Required for Doppler and velocity calculations.	4.8 seconds
MIN DIST	Minimum distance a track must move to become established.	0.25 nmi
MIN VEL	Minimum velocity a track must achieve to become established.	0 knots
PRF L	Low Pulse Repetition Frequency measured in Hz. Required for Doppler calculations.	Site parameter
PRF H	High Pulse Repetition Frequency measured in Hz. Required for Doppler calculations.	Site parameter
FREQ	System operating frequency measured in MHz. Required for Doppler calculations.	Site parameter
NG	Number of g's. Maximum tracked acceleration.	1.0 g
SIGMA AZ	Azimuth accuracy measured in ACPs.	2.5 ACP
SIGMA RANGE	Range accuracy measured in nmi.	0.03125 nmi

APPENDIX C. PERFORMANCE MONITORS

The following are the Performance Monitors output by the 9-PAC Tracker.

Table 15. Performance Monitors.

Name	Description
Total_tracks	Total number of tracks.
BO_tracks	Number of beacon-only tracks.
RO_tracks	Number of radar-only tracks.
RB_tracks	Number of radar-reinforced tracks.
Init_tracks	Number of initiating tracks (Track State = Init 0, Init 1, Miss, Non-Move 1, Non-Move 2).
NonMoving_tracks	Number of non-moving tracks (Track State = Non-Move 1, Non-Move 2).
Coasting_tracks	Number of coasting tracks (Track State = Coast 1, Coast 2, Coast 3).
Stable_tracks	Number of stable tracks (Track State = Stable).
OneToOne_corr	Number of one-to-one correlations.
ManyToOne_corr	Number of many-to-one correlations.
ManyToMany_corr	Number of many-to-many correlations.
Radar_hit_ratio	Hit ratio for radar-only tracks.
Beacon_hit_ratio	Hit ratio for beacon and radar-reinforced tracks.

REFERENCES

[1] J. Gertz and G. Elkin, "Documentation of 9-PAC Beacon Target Detector Processing Function," Lexington, MA, M.I.T. Lincoln Laboratory Project Report ATC-220, 26 July 1994.

[2] O.J. Newell and J.R. Anderson, "Description of Radar Correlation and Interpolation Algorithms for the ASR-9 Processor Augmentation Card (9-PAC)," Lexington, MA, M.I.T. Lincoln Laboratory Project Report ATC-236, 27 October 1995.

[3] Department of Transportation Federal Aviation Administration "Specification for the Airport Surveillance Radar (ASR-9)," FAA-E-2704B, 1 October 1986.

[4] "Software System/Subsystem Specification Surveillance Processor for the ASR-9 Airport Surveillance Radar," Westinghouse Electric Corporation, Baltimore, Maryland, 1989.

[5] J. B. Evans, "Application of ASR-9 Interpolated Doppler for Tracking," Lexington, MA, M.I.T. Lincoln Laboratory Project Report, to be published in 1996.

[6] Burgiois, F., and J.-C. Lassalle, "An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices." *Communication of the ACM*, Vol. 14, Dec. 1971, pp. 802-806.