TEMPORAL CLUSTERING IN THE MULTI-TARGET TRACKING ENVIRONMENT

APPROVED BY

SUPERVISORY COMMITTEE:

[illegible]

Byron D Tapley

Thomas A. Feo

Paul A. Jensen

John B Lundberg

To my daughter, Shannon, who means the world to me.

TEMPORAL CLUSTERING IN THE MULTI-TARGET TRACKING ENVIRONMENT

by

THOMAS SEAN KELSO, B.S., M.B.A., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 1988

Acknowledgments

As the culmination of almost a dozen years of graduate education, the completion of this dissertation marks a major milestone in my life. But while I've invested considerable effort and made many personal sacrifices, this achievement would not have been possible without the help of many others, both old friends and new colleagues. In particular, I'd like to thank Dr. Melba Crawford for the opportunity to work with her on this challenging project and for all her time and encouragement. I'd also like to thank her for persuading me to learn and use TEX and LATEX; their use has been an immense help in simplifying the mechanics of producing this dissertation, allowing me to concentrate more fully on its content. Thanks also go to Stuart H. Smith, with whom I worked very closely on the initial development of the simulation and other support software. I've learned much from our collaboration and know that his personal efforts were instrumental to the overall success of this endeavor. Special thanks are extended to Laura Schofield and Mary Kay Hamill; their constant support and encouragement were a continual source of strength, especially when circumstances seemed their bleakest. And, finally, I'd like to thank the Lord God, without whose love and ever watchful guidance I know I would never have made it so far and accomplished so much.

THOMAS SEAN KELSO

The University of Texas at Austin
August 1988

TEMPORAL CLUSTERING IN THE MULTI-TARGET TRACKING ENVIRONMENT

Publication No.

Thomas Sean Kelso, Ph.D.
The University of Texas at Austin, 1988

Supervising Professors: Melba Crawford
Thomas Feo

In multi-target tracking problems such as those found in high-energy particle physics, fluid mechanics, and ballistic missile defense, the common objective is to separate the data into observations associated with individual targets and to use this data to estimate the targets' trajectories. In defense related applications, it is necessary to have algorithms which are computationally efficient, robust, and minimize data storage requirements. Recently developed approaches in the field of multi-target tracking, however, have been shown to have significant computational disadvantages.

In this study, non-hierarchical clustering methods are combined with computationally efficient algorithms such as those used to solve assignment and quadratic programming problems to provide an integrated procedure which is computationally efficient, minimizes data storage requirements, and gives a reasonable estimate of the number of targets. Combined with a sequential estimation filter such as the extended Kalman filter, the procedure can provide estimates of a target's state and state covariance after three observations and continuously maintain updated target state estimates in real time.

Empirical results based on 100 targets in ballistic trajectories have demonstrated this method's effectiveness by properly clustering data with four measurement attributes (range, range rate, azimuth, and elevation) in over 98 percent of the cases. Its robustness is manifested by the fact that these results apply to scenarios with 20 percent missing data and biases of up to one arc minute in the sensor attitude and 0.5 seconds in the sensor clock. And its capability to track in real time is demonstrated with a duty cycle of less than five percent.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

Today, many applications require the tracking of a large set of unidentified targets. Among these are applications in high-energy particle physics, fluid mechanics, and ballistic missile defense. Crucial to the concept of a space-based ballistic missile defense under the proposed Strategic Defense Initiative (SDI) is the development of a system with the capability to detect, classify, and predict the motion of a large number of unidentified targets. Composed of orbiting sensor platforms and linked through the Command, Control, Communications, and Intelligence (C^3I) element, this system must not only be able to handle multiple targets, but also to integrate the combined data from multiple sensors. This data must be combined in such a way as to present a realistic picture of the scenario underway so that limited resources may be directed in an appropriate response. Because the time for such a response in an Intercontinental Ballistic Missile (ICBM) attack scenario is so short, efficient, robust, and accurate algorithms capable of handling multi-target, multi-sensor data in real time are critical to achieving a successful ICBM defense.

1.2 Problem Statement

Nominally, this *detection-estimation system* will be comprised of two elements. The first element is a constellation of observation satellites placed in orbital configurations which allow suitable coverage of the areas of interest (the ICBMs' trajectories from their launch sites to the anticipated impact points). The satellites' onboard sensors must be capable of providing time-tagged observations of a number of attributes of the targets in its field-of-view. Typically, these attributes might include range and range-rate (for active sensors) and azimuth and elevation (for both active and passive sensors).

In a typical ballistic missile defense scenario, these orbiting satellite sensor platforms will survey the earth from altitudes of several thousand kilometers. Upon launch of an ICBM attack, each sensor may detect as many as 100 targets within its field-of-view, and these targets will likely be closely spaced and have similar attributes. Not only will there be uncertainty associated with the measurements of the targets' attributes due to sensor limitations, but some observations will be lost due to spurious measurements or unobservable conditions relating to the sensor-target geometry. In addition, there will be uncertainties associated with both the sensor attitude and sensor clock (position).

The second element of the detection-estimation system is the C^3I site where the data from various sensors is combined. This site might be a land-based command center or a space-based battle station (perhaps even co-located with one of the observation sensors).

Between these two system elements, four basic tasks must be accomplished:

- Separate the available data into tracks associated with individual targets,
- Correlate/combine tracks from various sensors,
- Estimate the targets' state at some reference epoch, and
- Predict (track) the targets' state at some future epoch.

The order in which these tasks are listed should not be taken to imply a sequential relationship. In fact, how these tasks are performed will determine the overall complexity of the multi-target, multi-sensor tracking problem. In addition, just what processing is done and by which element is a question of distributed estimation.

Further complicating an already difficult problem are uncertainties in the numbers of targets visible at each sensor or jointly visible at any subset of sensors, noisy and spurious measurements, and the nonlinear measurements and nonlinear dynamics of ballistic flight. Additional constraints may arise due to the need to decentralize data processing to enhance survivability.

1.3 Multi-Target Tracking

As a result of the many complexities involved in the multi-target tracking problem, a wide range of methods have been developed in an attempt to handle these four tasks and their associated difficulties. Research in multi-target tracking has concentrated in four primary areas:

- Track initiation,
- Track maintenance,
- Sensor-to-sensor correlation, and
- Improved estimation methods.

These four areas correspond roughly to the four tasks performed by the detection-estimation system.

Methods for handling track initiation and track maintenance often have much in common and, as such, have actually been handled as different manifestations of the same problem. Historically, approaches to this problem can be classed as Bayesian or non-Bayesian.

Initially, research focused on what are now known as non-Bayesian methods. Led by the pioneering work of Sittler in 1964, these methods include (1) tracking via data association, (2) track-split filtering, and (3) the maximum likelihood method. In tracking via data association, Sittler [47] devised a method whereby whenever more than one sensor measurement was observed in the neighborhood of a predicted measurement, the current track was split. Trajectories whose maximum likelihood function fell below a certain threshold were dropped from further consideration. This method handles both track initiation *and* track termination, as well as false alarms and missing measurements.

In 1975, Smith and Buechler [48] expanded on Sittler's approach within the framework of Kalman filtering (which was not in common use in 1964) for an application to radar tracking. In the track-splitting filter, Sittler's concept of a neighborhood was now a validation region which was derived from the innovation covariance matrix obtained from the standard Kalman filter. However, both Sittler's method and that of Smith and Buechler were considered impractical because the exponential growth in the number of trajectories would saturate the memory and computational capability of even the largest computers.

Stein and Blackman [52] further modernized Sittler's work in the development of the maximum likelihood method. They used a suboptimal sequential method which selected only the most likely assignment of targets and measurements from each data set or *scan*, thereby mitigating the trajectory growth problem. Morefield [39] extended this approach by partitioning the data into mutually exclusive and exhaustive sets of feasible tracks and formulating a 0–1 integer program. While Morefield's work put track initiation and maintenance on a more solid theoretical basis, its application still has large computational and memory requirements in a dense target environment.

Initial work with Bayesian approaches began with the nearest neighbor filter. Sea [42], Singer and Stein [44], and Singer and Sea [45] used the nearest neighbor of a predicted measurement and modified the Kalman filter to account for the *a priori* probability that this measurement might be spurious (Bar-Shalom [10]). However, it was discovered that this filter can easily lose the target in a cluttered environment.

Work by Jaffer and Bar-Shalom [31] and Bar-Shalom and Jaffer [6] led to the development of the probability data association filter (PDAF) by Bar-Shalom [8], Bar-Shalom and Tse [7,9], and Bar-Shalom and Birmiwal [11]. A suboptimal Bayesian approach, the PDAF sequentially incorporates clusters of measurements into a track by attaching to each cluster an *a posteriori* probability of being correct. This is important because the standard formulation of the Kalman filter is optimal only when there is no possibility of incorrect assignments being made to a track. As a result, estimates and covariances in the PDAF account for the measurement origin uncertainty rather than being conditioned on the "accepted" tracks being true. The primary limitation of the PDAF is that it only tracks a single target in a multiple target or cluttered environment.

More recently, the joint PDAF of Fortmann, Bar-Shalom, and Scheffé [27], Chang and Bar-Shalom [18,19], and Chang, Chong, and Bar-Shalom [20] is used to jointly compute the probabilities for all targets and measurements that form a cluster. In it, posterior probabilities of the joint probability distribution function are conditioned on all measurements up to the present, allowing multiple targets (resolved or unresolved) to be tracked in a cluttered environment.

An optimal Bayesian approach was developed by Singer, Sea, and Housewright [46] for a single target in a cluttered environment. The major difference between the optimal Bayesian approach and the PDAF is that decomposition of the state estimate is accomplished in terms of all combinations of measurements from initial to present time rather than in terms of the latest measurements only. That is, the state estimate is determined based upon all possible track histories for a given target. Again, the major difficulty with each of these last two approaches is exponential memory growth.

In a multiple sensor environment, data from the various sensors must somehow be combined to correlate targets sets which may be visible to several sensors. Methods for sensor-to-sensor correlation attributed to Singer and Kanyuck [43], Stein and Blackman [52], Bowman [14], and Chang and Youens [16], are generally extensions of the maximum likelihood approaches used for the track maintenance problem. There are two primary approaches to correlating target sets from multiple sensors. The first is to map all observations from all sensors into a common measurement space and apply any of the tracking methods used for the single-sensor case. However, this approach will work only with measurement sets which permit unambiguous mappings into a common measurement space. The second approach is to form single-sensor tracks and then correlate the tracks from various sensors via pattern recognition or matchings of target state estimates.

Finally, because of the likelihood of imperfect correlation of observations with clusters, it is necessary to develop tracking filters which incorporate these correlation errors in the update of the error covariance matrix. The efforts of Singer and Stein [44], Jaffer and Bar-Shalom [31], Singer and Sea [45], and Singer, Sea, and Housewright [46] resulted in the development of filtering algorithms which use the *a posteriori* probability that an observation originated from a specific target in a dense multi-target environment. In addition, Tse, Larson, and Bar-Shalom [56] and Chang [15] considered the specific problem of estimation with angles-only measurements. The particular set of measurements available can play a significant role in both choosing the most appropriate tracking method and developing the tracking filter due to problems of marginal observability. The angles-only case is particularly difficult in this respect.

1.4 Clustering

In an attempt to mitigate some of the computational complexity of the algorithms discussed in the previous section, recent investigators have examined the application of a broad range of methods from the classification field (Tapley et al. [53,54], Balakrishnan et al. [4]). These methods are collectively known as clustering methods.

The first definitive results in clustering were produced by Sokal and Sneath [50] and refined by Sneath and Sokal [49] in the field of numerical taxonomy. While these references are devoted primarily to the biological sciences, the approach applies to all types of clustering. The clustering algorithms presented in these and subsequent references by Anderberg [2] and Romesburg [41] all share a common framework.

First, the data to be clustered is standardized based on some figure of merit which considers the relative importance of the various types of measurements. Once this standardization is completed, a similarity or dissimilarity coefficient is computed between *each* pair of measurements. The resulting matrix is known as a resemblance matrix. There are many methods for computing its coefficients. Typically, these coefficients are metrics such as Euclidean or average Euclidean distance in the measurement (attribute) space. Although other types of coefficients exist, they are not appropriate to this endeavor because of their inability to discriminate the types of clusters encountered in the multi-target tracking problem (those with additive or proportional translations).

Given the resemblance matrix, there are two approaches to forming clusters: hierarchical and non-hierarchical. The earliest approaches to clustering involved hierarchical clustering, wherein clusters are "built up" from the data until one large cluster is formed. The most common of the many methods available involve linking clusters through the use of either weighted or unweighted pair-group procedures using arithmetic averages (WPGMA or UPGMA). In these methods, links are developed based on the smallest (weighted or unweighted) average of all the distances[1] between the points in a pair of clusters.

Other common methods include the single linkage, complete linkage, and centroid methods. Single linkage builds clusters through association of the shortest distance between the closest points in any two clusters while complete linkage builds clusters based on association of the shortest distance between the farthest points in any two clusters. Single linkage may be thought of as a "nearest neighbor" approach whereas complete linkage is a "strongest association" approach. Centroid methods offer a compromise between these two extremes, building linkages based on the shortest distance between the centroids of existing clusters. Of the hierarchical methods discussed, the single linkage method is the most appropriate for the multi-target tracking problem because of its tendency to form clusters which are chains of data points (a feature which is normally considered a drawback to this method).

The tree formed by any of these clustering methods, which shows how the clusters are linked and at what level, is known as a *phenogram* or *dendrogram*. In Figure 1.1, the phenogram shows four objects and how they are related. The level at which objects are linked together represents their similarity, with the lowest links indicating the strongest similarities. To complete the hierarchical approach the phenogram must be "split" at some level to decide how many clusters exist, a requirement key to the multi-target tracking problem where the number of targets is unknown. Depending on what level the phenogram in Figure 1.1 is split, will determine how many distinct clusters exist. Splitting at Level 1 yields four clusters while splitting at Level 2 yields only two.

In hierarchical clustering methods, a data set of n observations yields n nested classifications ranging from one cluster with n observations to n clusters with one observation each. Non-hierarchical methods, on the other hand, cluster

[1]The term *distance* is used in this discussion to imply a similarity or dissimilarity coefficient. Smaller distances refer to similar coefficients while larger distances refer to the opposite.

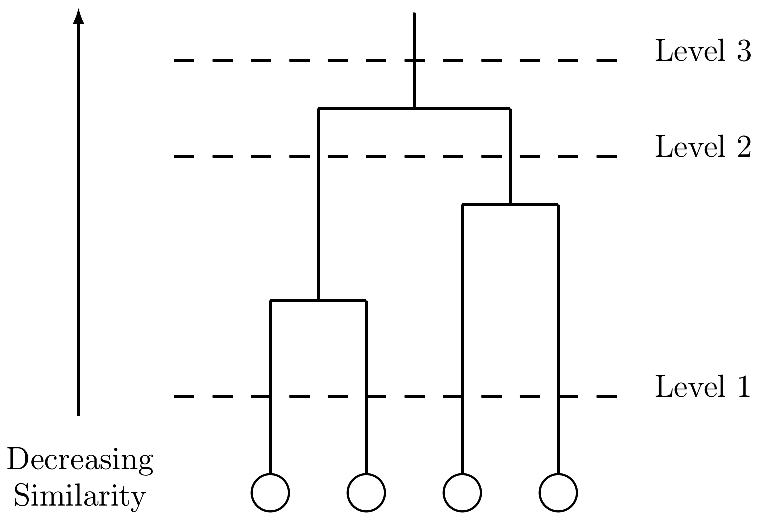

Figure 1.1: Sample Phenogram

observations into k clusters, where k is either specified *a priori* or is determined as part of the method used (Anderberg [2]). These methods enjoy an advantage over hierarchical methods since it is not necessary for them to store the similarity matrix or even the data set since the data is typically processed serially. It is, therefore, possible to cluster much larger data sets with non-hierarchical methods.

The majority of non-hierarchical clustering methods involve the use of seed points. There are many ways of seeding clusters. MacQueen [36] suggested choosing the first k observations as seeds while McRae [37] chose k random observations. Forgy [26] partitioned the data into k mutually exclusive and exhaustive sets and used the set centroids as the seed points. A more intuitively appealing approach was used by Astrahan [3] wherein "densities" were calculated for each data point and the points with the k highest "densities" were selected as seeds. In a similar approach, Ball and Hall [5] chose the first seed as the centroid of the data set. Additional seed points were added while processing the data if they were more than some set distance from all existing seed points.

Once the seed points have been determined, clusters are built around them. Forgy's method [26] and Jancey's variant [32] assign observations to the closest seed while MacQueen's k-means method [36] assigns observations to the cluster with the nearest centroid. Usually the clustering is reapplied after generating new seed points based on the current partition, such as the new cluster centroids, and repeated until some convergence criterion is satisfied.

MacQueen [36] and Wishart (Anderberg [2]) developed methods which permit variable numbers of clusters. In these methods, clusters are merged if their seeds are within some pre-specified distance of each other. New clusters are formed when observations are found to be beyond some (usually different) distance from the existing cluster seed points or centroids. As with the fixed number of clusters methods, the process is repeated until convergence.

Finally, many authors have developed methods which propose criteria for evaluating whether movements of individual observations result in an overall improvement of a partition (Anderberg [2], Späth [51]). These criteria are based on multivariate statistical analysis techniques, such as linear discriminant analysis and multivariate analysis of variance. The principal criteria used are:

- Minimize trace W,
- Minimize $|W|/|T|$ or maximize $|T|/|W|$,
- Maximize the largest eigenvalue of $W^{-1}B$, and
- Maximize the trace of $W^{-1}B$,

where T is the total scatter matrix, W is the within cluster scatter matrix, and B is the between cluster scatter matrix. It can be shown that the three matrices satisfy the relation $T = B + W$ (Anderberg [2]). These criteria are generally applied to the non-hierarchical methods discussed above as tests of convergence.

None of the currently employed clustering methods are designed to explicitly handle temporal data. Historically, clustering methods were developed to segregate data into distinct classifications. Because the desire is to group those observations which are most similar, the algorithms in use tend to generate hyperspherical clusters in the attribute space. As applied to the multi-target tracking problem, however, this tendency to form hyperspherical clusters is a major limitation. Due to the temporal dimension, observations associated with the correct cluster will form tracks, not hyperspheres. This result, therefore, necessitates the development of new clustering methods which explicitly account for the temporal dimension in order to be useful in the multi-target tracking problem.

1.5 Current Approach

Due to the inherent complexities of the multi-target tracking problem, it is a formidable task to develop an algorithm which can be shown to be optimal and possess the following characteristics,

- Perform both track initiation *and* track maintenance and
- Permit processing of data in real time while minimizing
  - Computational complexity and
  - Data storage requirements.

The last two sub-objectives are important not only in achieving real time performance but also in simplifying the processing component of the space-based sensor.

A heuristic method is developed which combines the most attractive features of the non-hierarchical clustering approaches with the track initiation and track maintenance approaches suggested in references [38,39] (Morefield), [16] (Chang and Youens), [17] (Chang and Tabaczynski), and [13] (Blackman). In fact, Blackman [13] addresses the idea of combining these features in the track maintenance phase, but does not provide a workable track initiation process capable of handling large numbers of targets in ballistic trajectories. Without track initiation, track maintenance cannot be performed.

To demonstrate the effectiveness of this heuristic approach, the method developed is specifically tailored to the ballistic missile defense problem with targets (ICBMs) in flight above a spherical earth with no atmosphere. Orbiting satellite sensors surveying the ICBM attack provide time-tagged observations of each target's range, range rate, azimuth, and elevation. Additional details regarding the simulation design and modeling assumptions are provided in Chapter 4 and Appendix A.

While the track initiation process marks the beginning of a track's life cycle, the temporal clustering process itself begins with the track maintenance phase. As seen in Figure 1.2, the temporal clustering process begins by reading

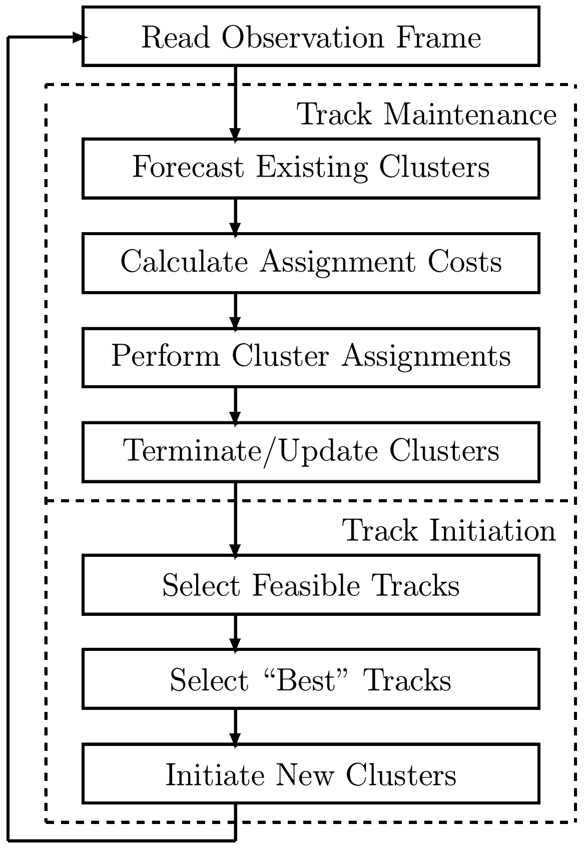

Figure 1.2: Temporal Clustering Process

in the data from the current observation frame and then forecasting all existing clusters to the current observation time. The costs of associating each new observation with a predicted observation corresponding to each existing cluster are calculated and observations satisfying the gating criteria are assigned to clusters to minimize the total overall association costs. Each cluster receiving a new observation has its state estimate updated while clusters not receiving observations are considered for termination. Finally, all remaining observations are passed to the track initiation procedure for evaluation in determining feasible sets of observations to initiate new tracks.

In Chapter 2, the various components making up the track maintenance process of assigning observations from the current observation frame to existing clusters will be discussed. The process of deciding how and when to terminate a track is also addressed here.

Then, in Chapter 3, an effective means for performing track initiation is developed. Chapter 4 contains a discussion of the specific simulation scenarios examined as well as an analysis of the results. Finally, conclusions are presented in Chapter 5 together with a discussion of proposed extensions to the current research.

Chapter 2

Track Maintenance

Each observation frame read by the sensor, yields a set of observation vectors consisting of the observation time and each object's range, range rate, azimuth, and elevation relative to the sensor. This set of vectors does not necessarily contain observations of all the targets in the sensor's field-of-view. This is due to observability problems arising from sensor characteristics and/or defects or as a result of the sensor-target geometry. In addition, there may be observations which do not correspond to any physical target, but are again the result of sensor characteristics and the observation environment. Many of these spurious measurements can be eliminated by pre-processing the data to remove inconsistent observations in light of the sensor characteristics.

Since the track maintenance process is restricted to dealing with only those observations recorded by the sensor, it must be capable of assigning observations to existing clusters so that inappropriate assignments are disallowed without eliminating correct assignments. This capability is necessary to prevent making assignments to a track when observations are missing from that track. It must also be able to continue existing tracks which do not receive an assignment until such time as track termination is deemed appropriate.

2.1 Forecasting

To decide whether an observation should be considered for assignment to an existing cluster, the "closeness" of each observation to each cluster must first be determined. To do so, however, both the observation and the cluster must be evaluated in the same space and at the same epoch. In the problem under consideration, two spaces are used: the attribute space and the state space.

In this study, the attribute space is a four-dimensional spherical coordinate space centered on the satellite sensor. The direction of its primary axis is fixed and points toward the vernal equinox. The four dimensions are the four attributes measured for each target: range, range rate, azimuth, and elevation.

An object's state is some set of physical characteristics which, together with a knowledge of the state transition rules, allows predictions of the state at any future time. For this study, the state space is a six-dimensional inertial Cartesian coordinate space centered at the center of mass of the Earth. The direction of its primary axis also points toward the vernal equinox. This coordinate system is referred to as the Earth Centered Inertial (ECI) coordinate system. The six dimensions of this space consist of three rectangular position components and three rectangular velocity components.

The choice of this state space is possible because the motion of a target in a conservative force field (one which can be described completely by a potential) can be fully described given that target's position and velocity. The case under examination includes only gravitational effects and excludes non-conservative forces such as thrust and drag, and is, therefore, a conservative system.

The use of the state space has several advantages over that of the attribute space in the temporal clustering process. Data storage is minimized because knowledge of a target can be maintained in a single state vector rather than a track of observations. Methods for efficiently tracking targets in the state space, such as the Kalman filter, are readily available. And, for this study, estimates in the state space can be easily and unambiguously mapped into the attribute space while the converse is not true.

To begin the process of assigning observations to clusters, therefore, each cluster's state vector is projected to some common observation epoch and then mapped into the attribute space. Not only is it necessary to forecast the state vector to the observation epoch, however, but the associated state covariance matrix to be used to *gate* the observations must also be forecast to that epoch and both the state and state covariance matrix must be mapped into the attribute space for direct comparison with the observations. As seen in Figure 2.1, the mapped state covariance can be used to form a confidence interval around the projected state estimate. Two tracks are shown together with their estimated states and gates. Only observations (hollow circles) falling within a gate are considered for possible assignment to an existing track. In this example, one observation could be assigned to either Track A or B, one observation could be assigned only to Track B, and one observation could be assigned to neither.

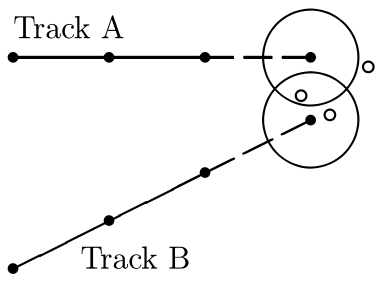

Figure 2.1: Gating Process

The original state covariance matrix is determined when the initial state estimate is formed and is described in detail in Section 3.3.1.

2.1.1 Kalman Filter

A natural method for forecasting the cluster state vector is provided by the Kalman filter. Not only does the Kalman filter provide a recursive means of propagating the state estimate and state covariance matrix but it also provides an optimal means for updating the same with the current observation. The Kalman filter, as developed in Kalman [33] and Kalman and Bucy [34], assumes that discrete states are linearly related via a state transition matrix and that discrete observations are linearly related to the current state. That is

$$\mathbf{S}_{k+1} = \mathbf{\Phi}(t_{k+1}, t_k)\mathbf{S}_k + \mathbf{u}_k \tag{2.1}$$

$$\mathbf{O}_k = \mathbf{H}_k\mathbf{S}_k + \mathbf{w}_k, \tag{2.2}$$

where

$$\begin{array}{ll} E[\mathbf{u}_k] = \mathbf{0} & E[\mathbf{u}_j\mathbf{u}_k^T] = \mathbf{Q}_k\delta_{jk} \\ E[\mathbf{w}_k] = \mathbf{0} & E[\mathbf{w}_j\mathbf{w}_k^T] = \mathbf{R}_k\delta_{jk} \end{array} \quad E[\mathbf{u}_j\mathbf{w}_k^T] = \mathbf{0}. \tag{2.3}$$

Here $\mathbf{S}_k$ is the target state at time t_k, $\mathbf{\Phi}(t_{k+1}, t_k)$ is the state transition matrix which governs how $\mathbf{S}$ changes from time t_k to t_{k+1}, and $\mathbf{u}_k$ is a white noise process. $\mathbf{O}_k$ is the measurement for the target at time t_k, $\mathbf{H}_k$ is the measurement transform matrix which governs how the state $\mathbf{S}_k$ and the measurement $\mathbf{O}_k$ are related, and $\mathbf{w}_k$ is another, independent, white noise process. Both $\mathbf{Q}_k$ and $\mathbf{R}_k$ are assumed diagonal matrices and δ_{jk} is the Kronecker delta.

The optimality criterion is that the estimate be a minimum variance unbiased estimate of the true state. That is

$$\begin{aligned} &\text{Minimize} \qquad E[\hat{\mathbf{e}}_k^T \mathbf{R}_k^{-1} \hat{\mathbf{e}}_k] && (2.4)\\ &\text{subject to} \qquad E[\hat{\mathbf{S}}_k] = \mathbf{S}_k, && (2.5) \end{aligned}$$

where

$$\hat{\mathbf{e}}_k = (\mathbf{O}_k - \mathbf{H}_k \hat{\mathbf{S}}_k). \tag{2.6}$$

Assuming prior estimates of the state $\bar{\mathbf{S}}_k$ and the state error covariance $\bar{\mathbf{P}}_k$, the resulting minimum variance unbiased estimate is

$$\hat{\mathbf{S}}_k = \bar{\mathbf{S}}_k + \mathbf{K}_k(\mathbf{O}_k - \mathbf{H}_k \bar{\mathbf{S}}_k), \tag{2.7}$$

where

$$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}_k^T (\mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1}. \tag{2.8}$$

The updated state error covariance then becomes

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \bar{\mathbf{P}}_k \tag{2.9}$$

and the state and state error covariance matrix are propagated according to

$$\begin{aligned} \bar{\mathbf{S}}_{k+1} &= \mathbf{\Phi}(t_{k+1}, t_k) \hat{\mathbf{S}}_k && (2.10)\\ \bar{\mathbf{P}}_{k+1} &= \mathbf{\Phi}(t_{k+1}, t_k) \hat{\mathbf{P}}_k \mathbf{\Phi}(t_{k+1}, t_k)^T + \mathbf{Q}_k. && (2.11) \end{aligned}$$

Unfortunately, for a target in a ballistic trajectory, neither the dynamics nor the measurements are linear. However, the system can be linearized through the application of Taylor series expansions.

2.1.2 Extended Kalman Filter

For a system with nonlinear dynamics and nonlinear measurements, the standard model of Equations 2.1 and 2.2 becomes

$$\dot{\mathbf{S}} = \mathbf{f}(\mathbf{S}, t) + \mathbf{u}(t) \tag{2.12}$$

$$\mathbf{O} = \mathbf{h}(\mathbf{S}, t) + \mathbf{w}(t) \tag{2.13}$$

with the covariance properties of $\mathbf{u}(t)$ and $\mathbf{w}(t)$ unchanged. Equation 2.12 is the differential equation describing how the system dynamics affect the state, $\mathbf{S}$, and Equation 2.13 describes the specific relationship between the observation and the state over time.

Given some nominal reference trajectory $\mathbf{S}^*(t)$, the true trajectory may be written as

$$\mathbf{S}(t) = \mathbf{S}^*(t) + \mathbf{s}(t) \tag{2.14}$$

so that Equations 2.12 and 2.13 become

$$\dot{\mathbf{S}}^* + \dot{\mathbf{s}} = \mathbf{f}(\mathbf{S}^* + \mathbf{s}, t) + \mathbf{u}(t) \tag{2.15}$$

$$\mathbf{O} = \mathbf{h}(\mathbf{S}^* + \mathbf{s}, t) + \mathbf{w}(t). \tag{2.16}$$

Assuming $\mathbf{s}$ to be small, $\mathbf{f}$ and $\mathbf{h}$ may be approximated with Taylor series expansions, so

$$\dot{\mathbf{S}}^* + \dot{\mathbf{s}} = \mathbf{f}(\mathbf{S}^*, t) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{S}}\right)^* \mathbf{s} + \cdots + \mathbf{u}(t) \tag{2.17}$$

$$\mathbf{O} = \mathbf{h}(\mathbf{S}^*, t) + \left(\frac{\partial \mathbf{h}}{\partial \mathbf{S}}\right)^* \mathbf{s} + \cdots + \mathbf{w}(t), \tag{2.18}$$

where

$$\dot{\mathbf{S}}^* = \mathbf{f}(\mathbf{S}^*, t), \tag{2.19}$$

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{S}}\right)^* = \frac{\partial \mathbf{f}}{\partial \mathbf{S}}\bigg|_{\mathbf{S}=\mathbf{S}^*}, \quad \text{and} \quad \left(\frac{\partial \mathbf{h}}{\partial \mathbf{S}}\right)^* = \frac{\partial \mathbf{h}}{\partial \mathbf{S}}\bigg|_{\mathbf{S}=\mathbf{S}^*}. \tag{2.20}$$

Retaining only first-order terms, a linearized model in $\mathbf{s}$ results as

$$\dot{\mathbf{s}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{S}}\right)^* \mathbf{s} + \mathbf{u}(t) \tag{2.21}$$

$$\mathbf{o} = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{S}}\right)^* \mathbf{s} + \mathbf{w}(t) = \mathbf{O} - \mathbf{h}(\mathbf{S}^*, t) \tag{2.22}$$

or, defining

$$\mathbf{A}(t) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{S}}\right)^* \quad \text{and} \quad \mathbf{H} = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{S}}\right)^*, \tag{2.23}$$

then

$$\dot{\mathbf{s}} = \mathbf{A}(t)\mathbf{s} + \mathbf{u}(t) \tag{2.24}$$

$$\mathbf{o} = \mathbf{H}\mathbf{s} + \mathbf{w}(t). \tag{2.25}$$

Standard algorithms for implementing the Kalman filter can be used, with the only difference being that the nominal state vector, $\mathbf{S}^*$, and the state covariance matrix, $\mathbf{\Phi}(t_{k+1}, t_k)$, are updated through the use of a numerical integration routine between time steps, with

$$\begin{aligned} \dot{\mathbf{S}}^* &= \mathbf{f}(\mathbf{S}^*, t) \\ \mathbf{S}^*(t_{k-1}) &= \mathbf{S}^*_{k-1} \end{aligned} \quad \text{and} \quad \begin{aligned} \dot{\mathbf{\Phi}}(t, t_{k-1}) &= \mathbf{A}(t)\mathbf{\Phi}(t, t_{k-1}) \\ \mathbf{\Phi}(t_{k-1}, t_{k-1}) &= \mathbf{I}. \end{aligned} \tag{2.26}$$

This implementation is known as the linearized Kalman filter.

Frequently it is desirable to use the current estimated trajectory in place of the nominal reference trajectory since, under stable conditions, a better estimate will result. The process of updating the reference trajectory with the latest estimate of the true trajectory is known as rectification. Using rectification, the reference trajectory is updated as

$$\hat{\mathbf{S}}^*_k = \mathbf{S}^*_k + \hat{\mathbf{s}}_k \tag{2.27}$$

for each new measurement. This rectified linearized Kalman filter is more commonly known as the Extended Kalman Filter (EKF) and is the filter of choice for most astrodynamical tracking problems.

2.1.3 State Propagation

At time t_k, the state vector, $\mathbf{S}_k$, consists of the target's position, $\mathbf{r}_k$, and velocity, $\dot{\mathbf{r}}_k$ (or $\mathbf{v}_k$), in ECI coordinates

$$\mathbf{S}_k = (x_k, y_k, z_k, \dot{x}_k, \dot{y}_k, \dot{z}_k)^T. \tag{2.28}$$

The observation vector, $\mathbf{O}_k$, is

$$\mathbf{O}_k = (\rho_k, \dot{\rho}_k, \alpha_k, \varepsilon_k)^T, \tag{2.29}$$

where ρ_k is the range from the sensor to the target, $\dot{\rho}_k$ is the range rate, α_k is the target's azimuth, and ε_k is the target's elevation.

Given estimates for $\bar{\mathbf{S}}_m$ and $\bar{\mathbf{P}}_m$ at some time t_m prior to the time of the current observation, the method to be used to forecast the target state and covariance to the current observation time, t_o, can be developed using the specific dynamical model for this investigation.

As shown in Equation 2.26, the target state estimate, $\bar{\mathbf{S}}$, is updated through numerical integration of

$$\dot{\mathbf{S}} = \mathbf{f}(\mathbf{S}, t) \tag{2.30}$$

with initial conditions

$$\mathbf{S}(t_m) = \bar{\mathbf{S}}_m \tag{2.31}$$

to the current observation time, t_o. The specific function $\mathbf{f}(\mathbf{S}, t)$ depends on the system dynamics. In general, for a target in ballistic flight,

$$\ddot{\mathbf{S}} = \mathbf{G}(\mathbf{r}, t) + \mathbf{D}(\mathbf{r}, \mathbf{v}, t) + \frac{\mathbf{T}(t)}{M(t)} + \mathcal{R}(t). \tag{2.32}$$

Each term on the right-hand-side of Equation 2.32 represents a force per unit mass (i.e., acceleration) on the target. $\mathbf{G}(\mathbf{r}, t)$ is the combined gravitational acceleration on the target due to the earth, sun, and moon, $\mathbf{D}(\mathbf{r}, \mathbf{v}, t)$ is the effect of atmospheric drag, $\mathbf{T}(t)$ is the target's thrust, $M(t)$ is the target's mass, and $\mathcal{R}(t)$ is the acceleration due to all unmodeled forces. For this study, the target is subjected only to the gravitational acceleration of a spherical earth.

For such a target in a two-body orbit, the standard second-order differential equation is

$$\ddot{\mathbf{S}} = -\frac{\mu \mathbf{r}}{r^3}, \tag{2.33}$$

which, when expressed in the first-order form of Equation 2.12 gives

$$\dot{\mathbf{S}} = \mathbf{f}(\mathbf{S}, t) = \left(\dot{x}, \dot{y}, \dot{z}, -\frac{\mu x}{r^3}, -\frac{\mu y}{r^3}, -\frac{\mu z}{r^3}\right)^T \tag{2.34}$$

and can be integrated, using the initial condition $\mathbf{S}(t_m) = \bar{\mathbf{S}}_m$, from t_m to t_o. Equation 2.34 results from differentiating Equation 2.28 and applying Equation 2.33.

Propagating the state covariance, $\bar{\mathbf{P}}_m$, is a bit more difficult. From Equation 2.11,

$$\bar{\mathbf{P}}_o = \mathbf{\Phi}(t_o, t_m)\bar{\mathbf{P}}_m \mathbf{\Phi}(t_o, t_m)^T + \mathbf{Q}, \tag{2.35}$$

where the white noise process $\mathbf{u}$ is assumed to be homoscedastic and known. Otherwise adaptive filtering techniques are necessary to adaptively compute $\mathbf{Q}_m$. But, $\mathbf{\Phi}(t_o, t_m)$ itself must be numerically integrated, according to Equation 2.26 and the specific form of $\mathbf{A}(t)$ must be evaluated.

Letting $\mathbf{S} = (\mathbf{r}, \mathbf{v})^T$ and applying the definition of $\mathbf{A}(t)$ from Equation 2.23,

$$\mathbf{A}(t) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{S}}\right)^* = \left(\frac{\partial \dot{\mathbf{S}}}{\partial \mathbf{S}}\right)^* = \begin{pmatrix} \mathbf{f}_{rr} & \mathbf{f}_{rv} \\ \mathbf{f}_{vr} & \mathbf{f}_{vv} \end{pmatrix}, \tag{2.36}$$

where

$$\mathbf{f}_{rr} = \frac{\partial \dot{\mathbf{r}}}{\partial \mathbf{r}} = \mathbf{0}, \tag{2.37}$$

$$\mathbf{f}_{rv} = \frac{\partial \dot{\mathbf{r}}}{\partial \mathbf{v}} = \mathbf{I}, \tag{2.38}$$

$$\mathbf{f}_{vr} = \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{r}} = -\frac{\mu}{r^3} \begin{pmatrix} 1 - 3\left(\frac{x}{r}\right)^2 & -3\left(\frac{xy}{r^2}\right) & -3\left(\frac{xz}{r^2}\right) \\ -3\left(\frac{xy}{r^2}\right) & 1 - 3\left(\frac{y}{r}\right)^2 & -3\left(\frac{yz}{r^2}\right) \\ -3\left(\frac{xz}{r^2}\right) & -3\left(\frac{yz}{r^2}\right) & 1 - 3\left(\frac{z}{r}\right)^2 \end{pmatrix}, \tag{2.39}$$

$$\text{and} \quad \mathbf{f}_{vv} = \frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{v}} = \mathbf{0}. \tag{2.40}$$

Therefore, the system of equations to be integrated is

$$\begin{pmatrix} \dot{\mathbf{\Phi}}_{rr} & \dot{\mathbf{\Phi}}_{rv} \\ \dot{\mathbf{\Phi}}_{vr} & \dot{\mathbf{\Phi}}_{vv} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{f}_{vr} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{\Phi}_{rr} & \mathbf{\Phi}_{rv} \\ \mathbf{\Phi}_{vr} & \mathbf{\Phi}_{vv} \end{pmatrix} \tag{2.41}$$

or

$$\dot{\mathbf{\Phi}}_{rr} = \mathbf{\Phi}_{vr} \tag{2.42}$$

$$\dot{\mathbf{\Phi}}_{rv} = \mathbf{\Phi}_{vv} \tag{2.43}$$

$$\dot{\mathbf{\Phi}}_{vr} = \mathbf{f}_{vr}\mathbf{\Phi}_{rr} \tag{2.44}$$

$$\dot{\mathbf{\Phi}}_{vv} = \mathbf{f}_{vr}\mathbf{\Phi}_{rv} \tag{2.45}$$

subject to

$$\mathbf{\Phi}_{rr}(t_m, t_m) = \mathbf{\Phi}_{vv}(t_m, t_m) = \mathbf{I} \tag{2.46}$$

$$\text{and} \quad \mathbf{\Phi}_{rv}(t_m, t_m) = \mathbf{\Phi}_{vr}(t_m, t_m) = \mathbf{0} \tag{2.47}$$

So, to find $\bar{\mathbf{P}}_o$, $\mathbf{\Phi}(t_o, t_m)$ is integrated according to Equations 2.42 through 2.47 and Equation 2.35 is applied.

With $\bar{\mathbf{S}}_o$ and $\bar{\mathbf{P}}_o$ the state estimate and its covariance must now be mapped into the attribute space for use in completing the cluster assignment process.

2.1.4 Mapping Into the Attribute Space

Given the target state, $\bar{\mathbf{S}}_o = (\mathbf{r}_o, \dot{\mathbf{r}}_o)^T$, and the sensor state, $\bar{\mathbf{S}}_{os} = (\mathbf{r}_{os}, \dot{\mathbf{r}}_{os})^T$, the transformation from the state space into the attribute space is given by

$$\boldsymbol{\rho}_o = (\rho_{ox}, \rho_{oy}, \rho_{oz})^T = \mathbf{r}_o - \mathbf{r}_{os} \tag{2.48}$$

$$\dot{\boldsymbol{\rho}}_o = (\dot{\rho}_{ox}, \dot{\rho}_{oy}, \dot{\rho}_{oz})^T = \dot{\mathbf{r}}_o - \dot{\mathbf{r}}_{os} \tag{2.49}$$

$$\rho_o = \|\boldsymbol{\rho}_o\| \tag{2.50}$$

$$\dot{\rho}_o = \dot{\boldsymbol{\rho}}_o \cdot \hat{\boldsymbol{\rho}}_o \tag{2.51}$$

$$\alpha_o = \tan^{-1}\left(\frac{\rho_{oy}}{\rho_{ox}}\right) \tag{2.52}$$

$$\varepsilon_o = \tan^{-1}\left(\frac{\rho_{oz}}{\varrho_o}\right), \tag{2.53}$$

$$\text{where} \qquad \varrho_o = \sqrt{{\rho_{ox}}^2 + {\rho_{oy}}^2} \tag{2.54}$$

and $\hat{\boldsymbol{\rho}}_o$ is the unit vector along the range vector, $\boldsymbol{\rho}_o$. These transformations are based upon the sensor-target geometry depicted in Figure 2.2 and a standard transformation of rectangular coordinates of the state space into the spherical coordinate system of the attribute space.

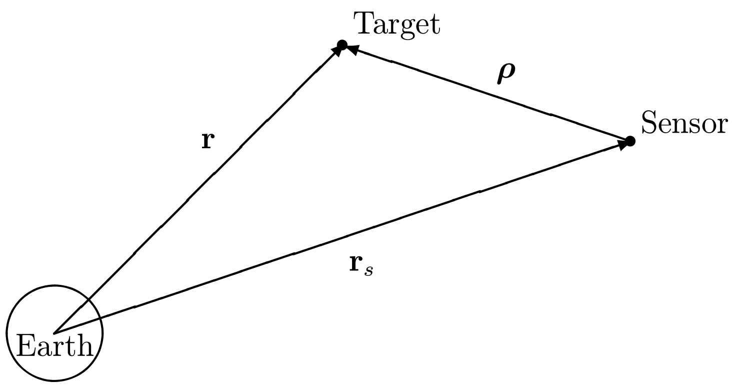

Figure 2.2: Sensor-Target Geometry

The vector of estimated observations at the current observation time, t_o, is designated by $\hat{\mathbf{O}}_o$, where

$$\hat{\mathbf{O}}_o = (\rho_o, \dot{\rho}_o, \alpha_o, \varepsilon_o)^T = \hat{\mathbf{O}}_o(\bar{\mathbf{S}}_o). \tag{2.55}$$

More properly, $\hat{\mathbf{O}}_o$ should be considered to be a function of the uncertain parameters $\mathbf{r}_o$ and $\dot{\mathbf{r}}_o$, with $\mathbf{r}_{os}$, $\dot{\mathbf{r}}_{os}$, and t_o treated as constants.

Using a Taylor series expansion and retaining only first-order terms, the state estimate can be shown to be an unbiased estimate. That is

$$E[\hat{\mathbf{O}}_o] = E[\hat{\mathbf{O}}_o(\bar{\mathbf{S}}_o)] \tag{2.56}$$

$$= E\left[\hat{\mathbf{O}}_o(\boldsymbol{\mu}_{\bar{\mathbf{S}}_o}) + \left(\frac{\partial\hat{\mathbf{O}}_o}{\partial\bar{\mathbf{S}}_o}\right)^* (\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})\right] \tag{2.57}$$

$$= E\left[\hat{\mathbf{O}}_o(\boldsymbol{\mu}_{\bar{\mathbf{S}}_o})\right] + E\left[\left(\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right)^* \bar{\mathbf{S}}_o\right] - E\left[\left(\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right)^* \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}\right] \tag{2.58}$$

$$= \hat{\mathbf{O}}_o(\boldsymbol{\mu}_{\bar{\mathbf{S}}_o}) + \left(\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right)^* E[\bar{\mathbf{S}}_o] - \left(\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right)^* \boldsymbol{\mu}_{\bar{\mathbf{S}}_o} \tag{2.59}$$

$$= \hat{\mathbf{O}}_o(\boldsymbol{\mu}_{\bar{\mathbf{S}}_o}), \tag{2.60}$$

where

$$\left(\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right)^* = \left.\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right|_{\bar{\mathbf{S}}_o = \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}} = \mathbf{H}_o. \tag{2.61}$$

Now, to find the state covariance matrix, a Taylor series expansion is again applied, so

$$\hat{\mathbf{O}}_o - E[\hat{\mathbf{O}}_o] = \hat{\mathbf{O}}_o(\boldsymbol{\mu}_{\bar{\mathbf{S}}_o}) + \mathbf{H}_o(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}) + \cdots - E[\hat{\mathbf{O}}_o] \tag{2.62}$$

$$= \mathbf{H}_o(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}) + \cdots \tag{2.63}$$

and, retaining only first-order terms,

$$\hat{\boldsymbol{\Xi}}_o = E\left[(\hat{\mathbf{O}}_o - E[\hat{\mathbf{O}}_o])(\hat{\mathbf{O}}_o - E[\hat{\mathbf{O}}_o])^T\right] \tag{2.64}$$

$$= E\left[(\mathbf{H}_o(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}))(\mathbf{H}_o(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o}))^T\right] \tag{2.65}$$

$$= E\left[(\mathbf{H}_o(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})^T \mathbf{H}_o^T\right] \tag{2.66}$$

$$= \mathbf{H}_o E\left[(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})^T\right] \mathbf{H}_o^T. \tag{2.67}$$

But

$$E\left[(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})(\bar{\mathbf{S}}_o - \boldsymbol{\mu}_{\bar{\mathbf{S}}_o})^T\right] \tag{2.68}$$

is merely the state covariance matrix, $\bar{\mathbf{P}}_o$. So, the estimated attribute covariance matrix, $\hat{\boldsymbol{\Xi}}_o$, can be written

$$\hat{\boldsymbol{\Xi}}_o = \mathbf{H}_o \bar{\mathbf{P}}_o \mathbf{H}_o^T. \tag{2.69}$$

Actually, $\hat{\boldsymbol{\Xi}}_o = \max(\mathbf{H}_o \bar{\mathbf{P}}_o \mathbf{H}_o^T, \mathbf{R})$ is used, where $\mathbf{R}$ is the observation covariance matrix. This prevents the estimated error associated with the observation from becoming smaller than the known error of the sensor.

Also, since $\boldsymbol{\mu}_{\bar{\mathbf{S}}_o}$ is not known, the reference state, $\bar{\mathbf{S}}_o^*$, is used instead, so

$$\mathbf{H}_o = \left.\frac{\partial \hat{\mathbf{O}}_o}{\partial \bar{\mathbf{S}}_o}\right|_{\bar{\mathbf{S}}_o = \bar{\mathbf{S}}_o^*}. \tag{2.70}$$

In particular,

$$\mathbf{H}_o = \begin{pmatrix} \dfrac{\rho_{ox}}{\rho_o} & \dfrac{\rho_{oy}}{\rho_o} & \dfrac{\rho_{oz}}{\rho_o} & 0 & 0 & 0 \\ \dfrac{\dot{\rho}_{ox}\rho_o - \rho_{ox}\dot{\rho}_o}{{\rho_o}^2} & \dfrac{\dot{\rho}_{oy}\rho_o - \rho_{oy}\dot{\rho}_o}{{\rho_o}^2} & \dfrac{\dot{\rho}_{z}\rho_o - \rho_{oz}\dot{\rho}_o}{{\rho_o}^2} & \dfrac{\rho_{ox}}{\rho_o} & \dfrac{\rho_{oy}}{\rho_o} & \dfrac{\rho_{oz}}{\rho_o} \\ -\dfrac{\rho_{oy}}{{\varrho_o}^2} & \dfrac{\rho_{ox}}{{\varrho_o}^2} & 0 & 0 & 0 & 0 \\ -\dfrac{\rho_{ox}\rho_{oz}}{{\rho_o}^2\varrho_o} & -\dfrac{\rho_{oy}\rho_{oz}}{{\rho_o}^2\varrho_o} & \dfrac{\varrho_o}{{\rho_o}^2} & 0 & 0 & 0 \end{pmatrix}. \tag{2.71}$$

Once reasonable estimates of $\hat{\mathbf{O}}_o$ and $\hat{\Xi}_o$ have been determined, they can be used to gate the new observations.

2.2 Cluster Assignment

Now, each actual observation is compared to each predicted observation and the "cost" of association (i.e., the cost of assigning an actual observation to an existing cluster) is computed. This cost is based upon the Euclidean metric in the attribute space. Because of the disparate scales of the various attributes, each attribute of the observations is standardized by subtracting the attribute minimum and dividing by the attribute range.

To minimize the likelihood of infeasible associations, actual observations which fall outside the predicted observations' gates are assigned an arbitrarily large cost. These gates are derived from the diagonal elements of the attribute covariance matrix, $\hat{\Xi}_o$. While the use of simple rectangular gates based on the diagonal covariance elements may be conservative (depending upon the relative magnitude of the off-diagonal terms), it is shown empirically to work quite well. An association is considered to be infeasible if any single attribute of an observation is outside the predicted attribute's 6-σ confidence interval or if any two attributes are outside the corresponding predicted attributes' 3-σ confidence intervals.

The rationale for using a two-step voting process for determining whether an association is infeasible or not is based upon the probability of Type I and Type II errors. In all the simulation scenarios run to date, each incorrect association resulted in at least one attribute being far outside the predicted attribute's 3-σ confidence interval, therefore making the probability of a false association (Type II error), even at the 6-σ confidence level, quite small.

And while the probability of denying a correct association (Type I error) at the 3-σ confidence level is small, the likelihood of such an occurrence over the life of the scenario must be considered. The probability of a Type I error is equal to the probability that one or more observation attributes fall outside their gates, or one minus the probability that none fall outside their gates. If p is the probability that a single observation attribute is within its 3-σ gate and $q = 1-p$, and n is the number of attributes, then the probability of disallowing a correct assignment is

$$P(\text{gating error}) = 1 - \binom{n}{0} p^n q^0 = 1 - p^n. \tag{2.72}$$

Applying Equation 2.72 to a typical 100-target scenario with four attribute types, 1.08 Type I errors would be expected in *each* observation frame, resulting in a lost observation. By requiring that two attributes (out of four) be outside the 3-σ confidence interval the probability of a Type I error is reduced considerably (by a factor of 250).

Once the costs of association have been computed, an assignment can be made of actual observations to existing clusters so that the total cost of these associations is a minimum. First, obvious assignments are made where only one assignment is possible. Then, since the remaining numbers of clusters and observations will likely be unequal, dummy clusters or observations are formed with association costs set to an arbitrarily large value. The remaining assignment is then solved (in this case using the Hungarian algorithm [40]), and all clusters receiving legitimate assignments are updated via the EKF as shown in Section 2.4.

All clusters for which no observation can feasibly be assigned are considered to have missed an observation and are annotated to indicate this. Obviously, no updating of this cluster's state or state covariance is possible. Finally, all remaining unassigned observations are then passed to the track initiation algorithm discussed in Chapter 3.

2.3 Track Termination

Conditions for terminating a cluster are now considered. Target tracks may require termination because they have reached their impact point, been destroyed, become obscured by the earth's surface or atmosphere, or simply because they exit the sensor's field-of-view.

In the scenarios investigated in this study, all observation frames are recorded at equally spaced intervals and all observations in the same frame have the same time tag, it makes sense to terminate a cluster after some pre-specified number of missing observations. There are several practical reasons for so doing. First, there is no point (computationally) in continuing to propagate a cluster which has either disappeared or been terminated due to a Type I error. After some period of time it must be accepted that the target is no longer being tracked. With fixed time interval observation frames a maximum number of consecutive missing observations can be used as a limit.

Another reason for not propagating clusters indefinitely relates to the increased probability of Type II errors. As a cluster is propagated without updating its state and state covariance, the state covariances will grow, making it more and more likely that a false association will result. This limitation could lead to another criterion for terminating a cluster. That is, the cluster could be terminated when the state covariance elements exceed certain bounds. The main drawback to this criterion, however, is that the magnitude of the covariance elements are highly dependent upon the sensor-target geometry, so determination of the bounds would be subjective.

In using the consecutive missing observations limit as a criterion for terminating a cluster, the likelihood of incorrectly terminating an active cluster due to a chance occurrence of the limit being exceeded must be considered. To do this, an upper bound on the likelihood of an observation being missing must be estimated. If the maximum number of missing observations allowed is n and the probability of an observation being missing is q (assuming equally likely and independent events), and $p = 1-q$, then the probability of incorrectly terminating a cluster is

$$P(\text{false termination}) = \binom{n+1}{0} p^0 q^{n+1} = q^{n+1}. \tag{2.73}$$

For a 100-target scenario with 5-second data intervals covering the first six minutes of an ICBM launch, the following number of false terminations based upon chance are expected:

Maximum Missing	Probability of Missing		
	0.05	0.10	0.20
1	365.00	730.00	1460.00
2	18.00	72.00	288.00
3	0.89	7.10	11.20
4	0.04	0.70	2.21
5	0.00	0.07	0.44
6	0.00	0.01	0.09
7	0.00	0.00	0.02

Table 2.1: Expected Number of False Terminations

Other criteria for terminating clusters could consider whether the target was predicted to be beyond the field-of-view of the sensor, below the surface of the earth, or beyond the earth's horizon. However, these considerations are not presently implemented in the simulation.

2.4 Track Update

Once an assignment has been made between the existing clusters and the observations at the current observation time, t_o, the state for each cluster is ready to be updated. Using the linearized versions of Equations 2.7, 2.8, and 2.9,

$$\hat{\mathbf{s}}_o = \bar{\mathbf{s}}_o + \mathbf{K}_o(\mathbf{o}_o - \mathbf{H}_o\bar{\mathbf{s}}_o) \tag{2.74}$$

$$\text{and} \quad \hat{\mathbf{P}}_o = (\mathbf{I} - \mathbf{K}_o\mathbf{H}_o)\bar{\mathbf{P}}_o, \tag{2.75}$$

where

$$\mathbf{K}_o = \bar{\mathbf{P}}_o\mathbf{H}_o^T(\mathbf{H}_o\bar{\mathbf{P}}_o\mathbf{H}_o^T + \mathbf{R})^{-1}. \tag{2.76}$$

Again, $\mathbf{R}$ is used rather than $\mathbf{R}_o$ in Equation 2.76 under an assumption that the white noise process $\mathbf{w}(t)$ is homoscedastic. And, because the Extended Kalman Filter is being used, $\bar{\mathbf{s}}_o = \mathbf{0}$, thus, Equation 2.74 becomes (assuming rectification occurs at each state update)

$$\hat{\mathbf{s}}_o = \mathbf{K}_o \mathbf{o}_o, \tag{2.77}$$

where

$$\mathbf{o}_o = \mathbf{O}_o - \mathbf{O}_o^*. \tag{2.78}$$

The final item needed is the measurement transform matrix, $\mathbf{H}_o$ from Equation 2.71. Evaluating $\mathbf{H}_o$ at $\bar{\mathbf{S}}_o$ and forming $\mathbf{K}_o$ according to Equation 2.76,

$$\hat{\mathbf{s}}_o = \mathbf{K}_o \mathbf{o}_o, \tag{2.79}$$

$$\hat{\mathbf{S}}_o = \bar{\mathbf{S}}_o + \hat{\mathbf{s}}_o, \tag{2.80}$$

$$\text{and} \quad \hat{\mathbf{P}}_o = (\mathbf{I} - \mathbf{K}_o \mathbf{H}_o)\bar{\mathbf{P}}_o. \tag{2.81}$$

While the implementation discussed above is theoretically correct, modification of the EKF is often necessary to prevent filter divergence. One of the primary causes of filter divergence is associated with errors which occur in the computation of the state error covariance matrix [55,25]. In particular, round-off errors in the calculation of this matrix can cause it to become non-positive semi-definite—a theoretical impossibility.

In this investigation, the standard EKF formulation was found to suffer from just this type of filter divergence. The state error covariance matrix immediately became non-symmetric and non-positive semi-definite during the first filter update. This result was due to the extremely poor conditioning of the matrix

$$\bar{\mathbf{M}} = \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + \mathbf{R}, \tag{2.82}$$

which must be inverted in Equation 2.76 as part of the EKF procedure. As a result, the EKF is re-formulated to take advantage of an approach which maintains the natural symmetry and positive semi-definiteness of the state covariance matrix.

2.4.1 Matrix Square Root

Since the *a priori* state error covariance matrix $\bar{\mathbf{P}}$ is a symmetric positive semi-definite matrix, it can be written as

$$\bar{\mathbf{P}} = \bar{\mathbf{W}}\bar{\mathbf{W}}^T, \tag{2.83}$$

where $\bar{\mathbf{W}}$ is the matrix square root of $\bar{\mathbf{P}}$. Such a square root can easily be found using Cholesky's Decomposition Algorithm [55]. Given the $n \times n$ elements of $\bar{\mathbf{P}}$, the elements of $\bar{\mathbf{W}}$ may be found using the following procedure

For $i = 1, 2, \ldots, n$

$$\bar{W}_{ii} = \sqrt{\bar{P}_{ii} - \sum_{k=1}^{i-1} \bar{W}_{ik}^2} \tag{2.84}$$

For $j = i + 1, \ldots, n$

$$\bar{W}_{ji} = \frac{\bar{P}_{ij} - \sum_{k=1}^{i-1} \bar{W}_{ik}\bar{W}_{jk}}{\bar{W}_{ii}}. \tag{2.85}$$

The resulting matrix is a lower-triangular square root matrix. The state error covariance update can now be reformulated using $\bar{\mathbf{W}}$ instead of $\bar{\mathbf{P}}$ to ensure that $\bar{\mathbf{P}}$ remains symmetric and positive semi-definite.

2.4.2 Covariance Update Reformulation

From Equations 2.75 and 2.76, the state error covariance update equation is

$$\hat{\mathbf{P}} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}, \tag{2.86}$$

where

$$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^T(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + \mathbf{R})^{-1}. \tag{2.87}$$

Equivalently,

$$\hat{\mathbf{P}} = \bar{\mathbf{P}} - \bar{\mathbf{P}}\mathbf{H}^T(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\bar{\mathbf{P}}. \tag{2.88}$$

Substituting $\bar{\mathbf{W}}\bar{\mathbf{W}}^T$ for $\bar{\mathbf{P}}$ and $\hat{\mathbf{W}}\hat{\mathbf{W}}^T$ for $\hat{\mathbf{P}}$ yields

$$\hat{\mathbf{W}}\hat{\mathbf{W}}^T = \bar{\mathbf{W}}\bar{\mathbf{W}}^T - \bar{\mathbf{W}}\bar{\mathbf{W}}^T\mathbf{H}^T(\mathbf{H}\bar{\mathbf{W}}\bar{\mathbf{W}}^T\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\bar{\mathbf{W}}\bar{\mathbf{W}}^T. \tag{2.89}$$

Letting $\bar{\mathbf{F}} = \bar{\mathbf{W}}^T\mathbf{H}^T$ and $\bar{\mathbf{M}} = \bar{\mathbf{F}}^T\bar{\mathbf{F}} + \mathbf{R}$, then Equation 2.89 becomes

$$\hat{\mathbf{W}}\hat{\mathbf{W}}^T = \bar{\mathbf{W}}(\mathbf{I} - \bar{\mathbf{F}}\bar{\mathbf{M}}^{-1}\bar{\mathbf{F}}^T)\bar{\mathbf{W}}^T. \tag{2.90}$$

Expressing $(\mathbf{I} - \bar{\mathbf{F}}\bar{\mathbf{M}}^{-1}\bar{\mathbf{F}}^T)$ as $\bar{\mathbf{\Lambda}}\bar{\mathbf{\Lambda}}^T$, then

$$\hat{\mathbf{W}}\hat{\mathbf{W}}^T = \bar{\mathbf{W}}\bar{\mathbf{\Lambda}}\bar{\mathbf{\Lambda}}^T\bar{\mathbf{W}}^T \tag{2.91}$$

or

$$\hat{\mathbf{W}} = \bar{\mathbf{W}}\bar{\mathbf{\Lambda}}. \tag{2.92}$$

The state error covariance update is found by first computing

$$\bar{\mathbf{W}} = \bar{\mathbf{P}}^{\frac{1}{2}} \tag{2.93}$$

using Cholesky's Decomposition Algorithm, then forming

$$\bar{\mathbf{F}} = \bar{\mathbf{W}}^T\mathbf{H}^T \tag{2.94}$$

$$\bar{\mathbf{M}} = \bar{\mathbf{F}}^T\bar{\mathbf{F}} + \mathbf{R} \tag{2.95}$$

$$\bar{\mathbf{\Lambda}} = (\mathbf{I} - \bar{\mathbf{F}}\bar{\mathbf{M}}^{-1}\bar{\mathbf{F}}^T)^{\frac{1}{2}} \tag{2.96}$$

$$\text{and} \quad \hat{\mathbf{W}} = \bar{\mathbf{W}}\bar{\mathbf{\Lambda}}. \tag{2.97}$$

Since $\bar{\mathbf{F}}^T\bar{\mathbf{F}} + \mathbf{R}$ is symmetric, symmetric inverse routines can be used to calculate $\bar{\mathbf{M}}^{-1}$.

Obviously, $\hat{\mathbf{P}} = \hat{\mathbf{W}}\hat{\mathbf{W}}^T$ and

$$\hat{\mathbf{s}} = \bar{\mathbf{s}} + \mathbf{K}(\mathbf{o} - \mathbf{H}\bar{\mathbf{s}}), \tag{2.98}$$

where

$$\mathbf{K} = \bar{\mathbf{W}}\bar{\mathbf{F}}\bar{\mathbf{M}}^{-1}, \tag{2.99}$$

which for the EKF becomes

$$\hat{\mathbf{s}} = \mathbf{K}\mathbf{o}. \tag{2.100}$$

This method ensures that $\mathbf{P}$ remains symmetric and positive semi-definite as expected.

Chapter 3

Track Initiation

Once the track maintenance process has been completed, the remaining unassigned observations are passed to the track initiation algorithm. These unassigned observations are most likely the result of new targets which may appear in the sensor field-of-view because they were just launched, were just deployed as a multiple independently-targeted reentry vehicle (MIRV), entered the sensor's field-of-view, or emerged from being obscured by the earth's surface or atmosphere.

The goal of this process is to form an initial estimate of a cluster's state and state covariance using the minimum number of observations. Examination of various orbit determination techniques in [12,24,29,30] has shown Laplace's method, using three observations of a target's range, range rate, azimuth, and elevation, to be the most appropriate method for determining an initial state estimate in this study. How this method is applied will be shown in Section 3.3.1. However, since three consecutive observations cannot be guaranteed, the unassigned data must be stored in a buffer to permit forming the necessary combinations of observations.

As in the track termination process, the likelihood of missing an observation will determine the size of the buffer required. If a buffer covering the last m observation frames is used, the probability of failing to correctly initiate a cluster will be the probability that at most one observation of the target associated with that cluster occurs in the last $m-1$ observation frames given that an observation has been detected in the first observation frame. That is,

$$P(\text{initiation failure}) = \binom{m-1}{0} p^0 q^{m-1} + \binom{m-1}{1} p^1 q^{m-2}, \tag{3.1}$$

where p is the probability of detecting an observation and $q = 1 - p$.

As with the track termination process, the expected number of failures to initiate a cluster for various buffer sizes and probabilities of missing data is summarized below in Table 3.1. The table assumes a 100-target scenario.

Buffer	Probability of Missing		
Size	0.05	0.10	0.20
3	9.75	19.00	36.00
4	0.73	2.80	10.41
5	0.05	0.37	2.72
6	0.00	0.05	0.67
7	0.00	0.01	0.16

Table 3.1: Expected Number of Track Initiation Failures

Once the size of the track initiation buffer is determined, the track initiation process can be analyzed. Since the objective in this process is to form observation triples which can be assessed for suitability, it is desirable to form all feasible triples and select among these for the “best” overall assignment.

3.1 Problem Formulation

To find the “best” overall assignment for the track initiation problem requires that the problem to be solved be defined specifically as well as in what sense the solution is best. Briefly, the problem is to form triples of observations (tracks) such that no observation is included in more than one track and that the system dynamics are not violated. But there must be some means for assessing the “cost” of associating an observation with a track. Then, the problem becomes to choose a set of tracks which minimize this association cost subject to the restriction that no observation be used in more than one track and that no system dynamics be violated.

A measure of the quality of the tracks of targets in ballistic trajectories is a function of the total specific energy, $\mathcal{E}$. High quality is associated with low values of $\mathcal{E}$. The designers of an ICBM booster are faced with providing a system for delivering nuclear warheads to their targets for the minimum cost. Since the cost is energy, it is reasonable to assume that each warhead will follow a low energy trajectory to its designated impact point. Combining this assumption with the fact that a ballistic trajectory is a minimum energy path in the gravitational potential results in the conclusion that any misgrouping of observations into a track will require a higher energy orbit. Therefore, if one pair of observations defines an orbit and a second pair defines a similar orbit (with the mid-point observation common to each pair) there should be no change in energy between the two orbits if the three observations are properly grouped (in a conservative force field). Since imperfect measurements are involved, the assumption is that this change is smallest for properly grouped observations than otherwise.

The resulting problem can be formulated as a binary quadratic program. Consider m observation frames and n_p unassigned observations in each frame $p = 1, 2, \ldots, m$. Observation i in frame p is linked with observation j in frame q if and only if the variable $x_{ipjq} = 1$, otherwise, $x_{ipjq} = 0$. The cost associated with a given observation triple (p_i, q_j, r_k) (i.e., the ith observation from frame p, the jth observation from frame q, and the kth observation from frame r), c_{ipjqkr}, is equal to the specific energy of the orbit defined by the observation triple if the system dynamics are satisfied, and equal to infinity otherwise.

Problem BQP (Binary Quadratic Program)

$$\text{Minimize} \quad \sum_{p=1}^{m-2} \sum_{q>p}^{m-1} \sum_{r>q}^{m} \sum_{i=1}^{n_p} \sum_{j=1}^{n_q} \sum_{k=1}^{n_r} c_{ipjqkr} x_{ipjq} x_{jqkr} \tag{3.2}$$

$$\text{subject to} \quad \sum_{i=1}^{n_p} x_{ipjq} - \sum_{k=1}^{n_r} x_{jqkr} = 0, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ j = 1, 2, \ldots, n_q \end{array} \tag{3.3}$$

$$\sum_{i=1}^{n_p} x_{ipjq} \leq 1, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ j = 1, 2, \ldots, n_q \end{array} \tag{3.4}$$

$$\sum_{j=1}^{n_q} x_{ipjq} \leq 1, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ i = 1, 2, \ldots, n_p \end{array} \tag{3.5}$$

$$\sum_{j=1}^{n_q} x_{jqkr} \leq 1, \qquad \begin{array}{l} q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ k = 1, 2, \ldots, n_r \end{array} \tag{3.6}$$

$$x_{ipjq}, x_{jqkr} \text{ binary}, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ i = 1, 2, \ldots, n_p \\ j = 1, 2, \ldots, n_q \\ k = 1, 2, \ldots, n_r. \end{array} \tag{3.7}$$

Equation 3.3 (conservation of flow) ensures that if observation q_j is paired with some observation p_i, then it is also paired with some observation r_k, thus guaranteeing that a triple is always formed. And Equations 3.3 through 3.7 ensure that each observation is assigned to at most one track, and vice versa.

Since integer programs can be difficult to solve efficiently and quadratic integer programs even more so, the problem is reformulated into a binary linear program by setting

$$y_{ipjqkr} = x_{ipjq} x_{jqkr}. \tag{3.8}$$

This variable ensures that only observation triples are formed. That is, observation i in frame p, observation j in frame q, and observation k in frame r are linked if and only if the variable $y_{ipjqkr} = 1$, otherwise, $y_{ipjqkr} = 0$.

Problem BLP (Binary Linear Program)

$$\text{Minimize} \quad \sum_{p=1}^{m-2} \sum_{q>p}^{m-1} \sum_{r>q}^{m} \sum_{i=1}^{n_p} \sum_{j=1}^{n_q} \sum_{k=1}^{n_r} c_{ipjqkr} y_{ipjqkr} \tag{3.9}$$

$$\text{subject to} \quad \sum_{i=1}^{n_p} y_{ipjqkr} \leq 1, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ j = 1, 2, \ldots, n_q \\ k = 1, 2, \ldots, n_r \end{array} \tag{3.10}$$

$$\sum_{j=1}^{n_q} y_{ipjqkr} \leq 1, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ i = 1, 2, \ldots, n_p \\ k = 1, 2, \ldots, n_r \end{array} \tag{3.11}$$

$$\sum_{k=1}^{n_r} y_{ipjqkr} \leq 1, \qquad \begin{array}{l} p = 1,2,\ldots,m-2 \\ q = p+1,\ldots,m-1 \\ r = q+1,\ldots,m \\ i = 1,2,\ldots,n_p \\ j = 1,2,\ldots,n_q \end{array} \tag{3.12}$$

$$y_{ipjqkr} \text{ binary}, \qquad \begin{array}{l} p = 1,2,\ldots,m-2 \\ q = p+1,\ldots,m-1 \\ r = q+1,\ldots,m \\ i = 1,2,\ldots,n_p \\ j = 1,2,\ldots,n_q \\ k = 1,2,\ldots,n_r. \end{array} \tag{3.13}$$

Lemma 1 *Formulations BQP and BLP are equivalent.*

PROOF: The objective functions of both programs are equivalent by the definition of y_{ipjqkr}. To complete the proof, any solution to either formulation must be shown to satisfy the constraints of the other.

All feasible solutions to BQP satisfy BLP:

Multiplying both sides of Equation 3.4 by x_{jqkr} and applying Equation 3.8 proves that Equation 3.10 is satisfied. From Equation 3.3,

$$\sum_{i=1}^{n_p} x_{ipjq} = \sum_{k=1}^{n_r} x_{jqkr}, \qquad \begin{array}{l} p = 1,2,\ldots,m-2 \\ q = p+1,\ldots,m-1 \\ r = q+1,\ldots,m \\ j = 1,2,\ldots,n_q, \end{array} \tag{3.14}$$

which implies
$$\sum_{k=1}^{n_r} x_{jqkr} \leq 1, \qquad \begin{array}{l} q = p+1,\ldots,m-1 \\ r = q+1,\ldots,m \\ j = 1,2,\ldots,n_q. \end{array} \tag{3.15}$$

Again, multiplying both sides of Equation 3.15 by x_{ipjq} and applying Equation 3.8 proves that Equation 3.12 is satisfied. Finally, Equation 3.11 is shown to be satisfied by noting that

$$\sum_{j=1}^{n_q} x_{ipjq} \leq 1, \qquad \begin{array}{l} p = 1,2,\ldots,m-2 \\ q = p+1,\ldots,m-1 \\ i = 1,2,\ldots,n_p \end{array} \tag{3.16}$$

$$\sum_{j=1}^{n_q} {x_{ipjq}}^2 = \sum_{j=1}^{n_q} x_{ipjq}, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ i = 1, 2, \ldots, n_p \end{array} \tag{3.17}$$

$$\sum_{j=1}^{n_q} x_{ipjq} x_{jqkr} \leq \sum_{j=1}^{n_q} {x_{ipjq}}^2, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ i = 1, 2, \ldots, n_p \\ k = 1, 2, \ldots, n_r. \end{array} \tag{3.18}$$

All feasible solutions to BLP satisfy BQP:

Since all variables are binary, any feasible solution of Equation 3.8 can be adjusted to satisfy

$$x_{ipjq} = x_{jqkr}, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ i = 1, 2, \ldots, n_p \\ j = 1, 2, \ldots, n_q \\ k = 1, 2, \ldots, n_r \end{array} \tag{3.19}$$

such that there is no change in the objective function (Equation 3.2). Therefore, from Equation 3.10,

$$\sum_{i=1}^{n_p} x_{ipjq} x_{jqkr} \leq 1, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ j = 1, 2, \ldots, n_q \\ k = 1, 2, \ldots, n_r \end{array} \tag{3.20}$$

$$\sum_{i=1}^{n_p} x_{ipjq} x_{jqkr} = \sum_{i=1}^{n_p} {x_{ipjq}}^2 = \sum_{i=1}^{n_p} x_{ipjq}, \qquad \begin{array}{l} p = 1, 2, \ldots, m-2 \\ q = p+1, \ldots, m-1 \\ r = q+1, \ldots, m \\ j = 1, 2, \ldots, n_q \\ k = 1, 2, \ldots, n_r, \end{array} \tag{3.21}$$

showing Equation 3.4 to be satisfied. Applying this same technique to Equations 3.11 and 3.12 proves Equations 3.5, 3.6, and 3.15 to be satisfied. And, combining Equations 3.4 and 3.15 with Equation 3.19 shows Equation 3.3 to be satisfied, as well. Therefore, the two problem formulations are equivalent. □

3.2 Problem Reduction

One approach to solving the track initiation problem would be to form all possible triples. However, the computational load for large problems would be extensive, regardless of the method used to solve the binary linear program. To help visualize the size of such a problem, consider a track initiation buffer of m frames with the number of unassigned observations in frame p equal to n_p. Then the number of possible triples is

$$\sum_{p=1}^{m-2} \sum_{q>p}^{m-1} \sum_{r>q}^{m} n_p n_q n_r, \tag{3.22}$$

which has a worst-case complexity of $O(m^3 n^3)$, where n is the number of actual targets. For a 100-target scenario with a buffer of seven observation frames, such an explicit enumeration would involve 35,000,000 combinations. In practice, however, most of the unassigned observations will be assigned to clusters within the first three or four observation frames. This conclusion results from applying the probability of a successful track initiation,

$$P(\text{successful initiation}) = \sum_{i=3}^{m} \binom{m}{i} p^i q^{m-i}, \tag{3.23}$$

where p is the probability of a successful detection and $q = 1 - p$, to various buffer sizes and probabilities of missing data. Results for 100 targets are shown in Table 3.2. But even under these conditions, startup could require in excess of 1,000,000 combinations be examined. Therefore, the track initiation process

Buffer	Probability of Missing		
Size	0.05	0.10	0.20
3	85.74	72.90	51.20
4	98.60	94.77	81.92
5	99.88	99.14	94.21
6	99.99	99.87	98.30
7	100.00	99.98	99.53

Table 3.2: Expected Number of Clusters Initiated

is begun by first reducing the number of allowable arcs based upon the system dynamics. A three-step process is used which begins by forming gates around each observation to limit which observations can be linked (Section 3.2.1). For each allowable pair of linked observations, a preliminary estimate of the specific energy of the orbit defined by that pair is determined and evaluated for feasibility subject to the system dynamics and assumed booster characteristics (Section 3.2.2). Then, pairs sharing a common mid-point are linked to form observation triples and initial orbits are determined (Section 3.3.1). Each triple is further evaluated to ensure that it remains feasible in terms of the system dynamics and assumed booster characteristics (energy). Finally, from the remaining set of feasible triples, the binary linear program defined on page 34 is solved (Section 3.4).

3.2.1 Single Observation Gating

The goal of this section is to determine which pairs of observations may feasibly be linked subject to system dynamical constraints. To accomplish this goal, estimates of ρ_j, $\dot{\rho}_j$, α_j, and ε_j and a gate for each estimate are calculated given a single observation at time t_i consisting of ρ_i, $\dot{\rho}_i$, α_i, and ε_i. These estimates and gates are formed for every observation such that $t_i < t_o$ to assess links with observations in each observation frame where $t_i < t_j$. Actually, this process is much simpler than it might first appear since the targets being tracked are assumed to be acted upon only by the gravity of a spherical earth. Defining the ECI vector to the target as $\mathbf{r}$, the ECI vector to the sensor as $\mathbf{r}_s$, and the range vector from the sensor to the target as $\boldsymbol{\rho}$, then the three vectors are related by the equation

$$\mathbf{r} = \boldsymbol{\rho} + \mathbf{r}_s, \tag{3.24}$$

as seen in Figure 2.2.

The range vector, $\boldsymbol{\rho}_i$, can be estimated from a single observation as

$$\boldsymbol{\rho}_i = \rho_i \hat{\boldsymbol{\rho}}_i = \rho_i \begin{pmatrix} \cos\varepsilon_i \cos\alpha_i \\ \cos\varepsilon_i \sin\alpha_i \\ \sin\varepsilon_i \end{pmatrix}, \tag{3.25}$$

where $\hat{\boldsymbol{\rho}}_i$ is the unit range vector. Therefore, the sensor position, $\mathbf{r}_i$, can be estimated. And, since the only acceleration acting on either the target or the sensor is assumed to be due to the earth's gravity,

$$\ddot{\mathbf{r}}_i = -\frac{\mu}{{r_i}^3}\mathbf{r}_i \qquad \text{and} \qquad \ddot{\mathbf{r}}_{is} = -\frac{\mu}{{r_{is}}^3}\mathbf{r}_{is}. \tag{3.26}$$

Now, the target's position at time t_j can be estimated through the use of a Taylor series expansion as

$$\mathbf{r}_j = \mathbf{r}_i + \dot{\mathbf{r}}_i\tau_{ji} + \ddot{\mathbf{r}}_i\frac{{\tau_{ji}}^2}{2} + \cdots, \tag{3.27}$$

where $\tau_{ji} \equiv t_j - t_i$. While not enough information to estimate $\dot{\mathbf{r}}_i$ is available, it is possible to make approximations to provide reasonable estimates and bounds for those estimates.

Now, by definition

$$\rho_j = \|\boldsymbol{\rho}_j\| = \sqrt{\boldsymbol{\rho}_j \cdot \boldsymbol{\rho}_j}, \tag{3.28}$$

where

$$\boldsymbol{\rho}_j = \mathbf{r}_j - \mathbf{r}_{js}. \tag{3.29}$$

Since the sensor's state at t_j is known, $\boldsymbol{\rho}_j$ can be expressed (truncating the Taylor series expansion after the acceleration term) as

$$\boldsymbol{\rho}_j = \mathbf{r}_i + \dot{\mathbf{r}}_i\tau_{ji} + \ddot{\mathbf{r}}_i\frac{{\tau_{ji}}^2}{2} - \mathbf{r}_{js} \tag{3.30}$$

$$= A\mathbf{r}_i + \dot{\mathbf{r}}_i\tau_{ji} - \mathbf{r}_{js}, \tag{3.31}$$

where

$$A \equiv \left(1 - \frac{\mu{\tau_{ji}}^2}{2{r_i}^3}\right). \tag{3.32}$$

Therefore,

$${\rho_j}^2 = \boldsymbol{\rho}_j \cdot \boldsymbol{\rho}_j = (A\mathbf{r}_i + \dot{\mathbf{r}}_i\tau_{ji} - \mathbf{r}_{js}) \cdot (A\mathbf{r}_i + \dot{\mathbf{r}}_i\tau_{ji} - \mathbf{r}_{js}), \tag{3.33}$$

which, when expanded, yields

$${\rho_j}^2 = A^2(\mathbf{r}_i \cdot \mathbf{r}_i) + 2A\tau_{ji}(\mathbf{r}_i \cdot \dot{\mathbf{r}}_i) + {\tau_{ji}}^2(\dot{\mathbf{r}}_i \cdot \dot{\mathbf{r}}_i) - 2A(\mathbf{r}_i \cdot \mathbf{r}_{js}) \tag{3.34}$$

$$- 2\tau_{ji}(\dot{\mathbf{r}}_i \cdot \mathbf{r}_{js}) + \mathbf{r}_{js} \cdot \mathbf{r}_{js} \tag{3.35}$$

$$= (Ar_i)^2 + (v_i\tau_{ji})^2 - 2A(\mathbf{r}_i \cdot \mathbf{r}_{js}) + {r_{js}}^2 + 2\tau_{ji}(A\mathbf{r}_i - \mathbf{r}_{js}) \cdot \dot{\mathbf{r}}_i \tag{3.36}$$

$$= (Ar_i)^2 + (v_i\tau_{ji})^2 - 2A(\mathbf{r}_i \cdot \mathbf{r}_{js}) + {r_{js}}^2 + 2\tau_{ji}\|A\mathbf{r}_i - \mathbf{r}_{js}\|v_i\cos\beta. \tag{3.37}$$

The estimate of ρ_j is completed by making some approximations for the terms depending on $\dot{\mathbf{r}}_i$. Although the direction of the velocity vector is not known, its magnitude can be estimated in the problem under consideration. This result is due to the fact that the target is assumed to be operating in a conservative force field and, therefore, its energy is a constant. If the maximum velocity at burnout is assumed to be v_{bo} and it is also assumed that burnout occurs close to the earth's surface where r_{bo} is approximately the earth's radius, then the maximum specific energy is

$$\mathcal{E} = \frac{{v_{bo}}^2}{2} - \frac{\mu}{r_{bo}} \tag{3.38}$$

and the velocity at any subsequent time, such as t_i, is given by

$$v_i = \sqrt{2\left(\mathcal{E} + \frac{\mu}{r_i}\right)}. \tag{3.39}$$

The only term still not known is the value of β, the angle between $(A\mathbf{r}_i - \mathbf{r}_{js})$ and $\dot{\mathbf{r}}_i$. The angle β can be approximated, however, by noting that in extreme cases $\beta = \beta_1 \pm \beta_2$, where β_1 is the angle between $(A\mathbf{r}_i - \mathbf{r}_{js})$ and $\hat{\boldsymbol{\rho}}_i$ and β_2 is the angle between $\hat{\boldsymbol{\rho}}_i$ and $\dot{\mathbf{r}}_i$. Therefore,

$$\beta_1 = \cos^{-1} \frac{(A\mathbf{r}_i - \mathbf{r}_{js}) \cdot \hat{\boldsymbol{\rho}}_i}{\|A\mathbf{r}_i - \mathbf{r}_{js}\|} \tag{3.40}$$

and

$$\beta_2 = \cos^{-1} \frac{\dot{\mathbf{r}}_i \cdot \hat{\boldsymbol{\rho}}_i}{\|\dot{\mathbf{r}}_i\|}. \tag{3.41}$$

Although β_1 can be calculated directly, to calculate β_2 it should be noted that $\dot{\mathbf{r}}_i \cdot \hat{\boldsymbol{\rho}}_i$ is simply the target velocity along the line of sight, $v_{i_\|}$, and that

$$v_{i_\|} = (\dot{\boldsymbol{\rho}}_i + \dot{\mathbf{r}}_{is}) \cdot \hat{\boldsymbol{\rho}}_i = \dot{\rho}_i + \dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i. \tag{3.42}$$

Therefore,

$$\beta_2 = \cos^{-1} \frac{\dot{\rho}_i + \dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i}{v_i}. \tag{3.43}$$

From this development, the remaining uncertainty in the calculation of ρ_j results from not knowing the relative direction of the components of the sensor and target velocity normal to the line of sight. The extreme values of β correspond to an assumption that these components are coplanar and in either the same or opposite directions. Therefore, the larger of β_1 and β_2 is used as the nominal value and the smaller to determine the bounds for the value of ρ_j. That is

$$\beta = \max(\beta_1, \beta_2) \pm \min(\beta_1, \beta_2). \tag{3.44}$$

Now, to estimate $\dot{\rho}_j$,

$$\dot{\rho}_j = \dot{\boldsymbol{\rho}}_j \cdot \hat{\boldsymbol{\rho}}_j = \|\dot{\boldsymbol{\rho}}_j\| \cos\gamma_j, \tag{3.45}$$

where γ_j is the angle between $\dot{\boldsymbol{\rho}}_j$ and $\hat{\boldsymbol{\rho}}_j$. Since ρ_i and ρ_j are known, estimate can be completed if the angle between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ can be determined. Designating this angle as θ, the distance between $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_j$ is

$$\delta\rho = \sqrt{{\rho_i}^2 + {\rho_j}^2 - 2\rho_i\rho_j\cos\theta}. \tag{3.46}$$

However, the angle γ_j is approximately the angle between $\delta\boldsymbol{\rho}$ and $\boldsymbol{\rho}_j$ and since

$${\rho_i}^2 = \delta\rho^2 + {\rho_j}^2 - 2\delta\rho\rho_j\cos\gamma_j, \tag{3.47}$$

then

$$\begin{aligned}\cos\gamma_j &= \frac{\delta\rho^2 + {\rho_j}^2 - {\rho_i}^2}{2\delta\rho\rho_j} &(3.48)\\ &= \frac{{\rho_j}^2 - \rho_i\rho_j\cos\theta}{\delta\rho\rho_j}. &(3.49)\end{aligned}$$

Approximating $\dot{\rho}_j$ as $(\delta\rho/\tau_{ji})$,

$$\dot{\rho}_j = \frac{{\rho_j}^2 - \rho_i\rho_j\cos\theta}{\rho_j\tau_{ji}} \tag{3.50}$$

and θ can be found by noting that

$$\sin\theta = \frac{d_\perp}{\rho_j}, \tag{3.51}$$

where $d_\perp$ is the distance traveled perpendicular to the line of sight between t_i and t_j. Therefore,

$$\cos\theta = \frac{\sqrt{{\rho_j}^2 - {d_\perp}^2}}{\rho_j} \tag{3.52}$$

and

$$\dot{\rho}_j = \frac{{\rho_j}^2 - \rho_i\sqrt{{\rho_j}^2 - {d_\perp}^2}}{\rho_j \tau_{ji}}. \tag{3.53}$$

To calculate $d_\perp$, the velocities and accelerations of both the target and the sensor perpendicular to the line of sight are needed. Since v_i has already been estimated and v_{is} is known,

$$v_{i_\perp} = \sqrt{{v_i}^2 - {v_{i_\parallel}}^2} \tag{3.54}$$

$$= \sqrt{2\left(\mathcal{E} + \frac{\mu}{r_i}\right) - (\dot{\rho}_i + \dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i)^2} \tag{3.55}$$

and $$v_{is_\perp} = \sqrt{{v_{is}}^2 - (\dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i)^2}. \tag{3.56}$$

Therefore, the maximum relative velocity perpendicular to the line of sight is

$$\dot{\rho}_{i_\perp} = v_{i_\perp} + v_{is_\perp} \tag{3.57}$$

so

$$\dot{\rho}_{i_\perp} = \sqrt{2\left(\mathcal{E} + \frac{\mu}{r_i}\right) - (\dot{\rho}_i + \dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i)^2} + \sqrt{{v_{is}}^2 - (\dot{\mathbf{r}}_{is} \cdot \hat{\boldsymbol{\rho}}_i)^2}. \tag{3.58}$$

The approximate maximum distance moved perpendicular to the line of sight will then be

$$d_{max_\perp} = \dot{\rho}_{i_\perp}\tau_{ji} + \ddot{\rho}_{i_\perp}\frac{{\tau_{ji}}^2}{2}, \tag{3.59}$$

where $\ddot{\rho}_{i_\perp}$ is the relative acceleration perpendicular to the line of sight. And since

$$\ddot{\boldsymbol{\rho}}_i = \ddot{\mathbf{r}}_i - \ddot{\mathbf{r}}_{is} = -\frac{\mu}{{r_i}^3}\mathbf{r}_i + \frac{\mu}{{r_{is}}^3}\mathbf{r}_{is}, \tag{3.60}$$

then

$$\ddot{\rho}_{i_\perp} = \sqrt{\|\ddot{\boldsymbol{\rho}}_i\|^2 - (\ddot{\boldsymbol{\rho}}_i \cdot \hat{\boldsymbol{\rho}}_i)^2}. \tag{3.61}$$

The minimum distance moved perpendicular to the line of sight will result when

$$d_{min_\perp} = \left| |v_{i_\perp} - v_{is_\perp}| - \ddot{\rho}_{i_\perp}\frac{{\tau_{ji}}^2}{2} \right|. \tag{3.62}$$

The value of $\dot{\rho}_j$ will, therefore, be bounded by the variance in the estimates of ρ_j and $d_\perp$, with a nominal value based on the average of $d_{min_\perp}$ and $d_{max_\perp}$.

Unfortunately, the azimuth and elevation at time t_j cannot be estimated using a single observation since the target velocity vector cannot be fully estimated. Therefore, it is assumed that the new azimuth and elevation, α_j and ε_j, are in the neighborhood of the previous azimuth and elevation, α_i and ε_i (i.e., $\alpha_j = \alpha_i$ and $\varepsilon_j = \varepsilon_i$), and only the gates for these measurements will be determined.

The maximum angular change, θ_{max}, will have components in both azimuth and elevation. However, since there is no way of knowing exactly which direction the target is moving θ_{max} is used as the gate for both the azimuth and the elevation, resulting in a somewhat conservative estimate. Caution must also be used in calculating $\delta\alpha$ when $\varepsilon \approx \pm\frac{\pi}{2}$ since a small change of θ_{max} will yield corresponding large changes in $\delta\alpha$. In fact, the relationship for the maximum change in α for a given θ_{max} is

$$\delta\alpha = \frac{\theta_{max}}{\cos\varepsilon_i} \tag{3.63}$$

and

$$\delta\varepsilon = \theta_{max}. \tag{3.64}$$

Each observation in an observation frame is assessed in this manner to determine an estimate and gate for each observation in the succeeding observation frames. Once this process is completed, each observation in the succeeding observation frames is compared to the estimate and gate for that frame. This process is $O(m^2n^2)$). If the observation satisfies the gating criteria, the pairing is considered for addition to a list of allowable pairs.

3.2.2 Dual Observation Gating

Before being admitted to the list of allowable pairs, a dual observation estimate is formed to determine the energy of the orbit defined by that pair. Since there is some assumed maximum energy which can be achieved based upon the known (or inferred) booster characteristics, any orbit satisfying this energy restriction is considered allowable and the pair is added to the list. In the empirical results to be presented later, only links between two observations of the same target are likely, and the number of pairs formed appears to be $O(n)$.

Because only the energy estimate is required and it is constant for a given orbit, the estimates for $\mathbf{r}_m$ and $\dot{\mathbf{r}}_m$ can be determined for any time t_m. In this section, therefore, two complete observations of range, range rate, azimuth, and elevation at times t_1 and t_2 will be used to determine $\mathbf{r}_m$ and $\dot{\mathbf{r}}_m$ so the energy and its error may be estimated.

Recalling

$$\boldsymbol{\rho}_i = \rho_i \hat{\boldsymbol{\rho}}_i = \rho_i \begin{pmatrix} \cos\varepsilon_i \cos\alpha_i \\ \cos\varepsilon_i \sin\alpha_i \\ \sin\varepsilon_i \end{pmatrix} \qquad i = 1, 2 \tag{3.65}$$

and

$$\mathbf{r}_i = \boldsymbol{\rho}_i + \mathbf{r}_{is}, \tag{3.66}$$

The range vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ can be expressed using Taylor series expansions at some common epoch t_m as

$$\mathbf{r}_i = \mathbf{r}_m + \dot{\mathbf{r}}_m \tau_{im} + \frac{1}{2}\ddot{\mathbf{r}}_m {\tau_{im}}^2 + \cdots \qquad i = 1, 2. \tag{3.67}$$

Since

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r}, \tag{3.68}$$

and the acceleration vector at time t_m can be expressed using Lagrange interpolation as

$$\ddot{\mathbf{r}}_m = \frac{\tau_{m2}}{\tau_{12}}\ddot{\mathbf{r}}_1 + \frac{\tau_{m1}}{\tau_{21}}\ddot{\mathbf{r}}_2 \tag{3.69}$$

$$= \frac{\tau_{2m}}{\tau_{21}}\ddot{\mathbf{r}}_1 - \frac{\tau_{1m}}{\tau_{21}}\ddot{\mathbf{r}}_2 \tag{3.70}$$

a system of two equations with two unknowns results

$$\begin{pmatrix} 1 & \tau_{1m} \\ 1 & \tau_{2m} \end{pmatrix} \begin{pmatrix} \mathbf{r}_m \\ \dot{\mathbf{r}}_m \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 - \dfrac{\ddot{\mathbf{r}}_m {\tau_{1m}}^2}{2} \\ \mathbf{r}_2 - \dfrac{\ddot{\mathbf{r}}_m {\tau_{2m}}^2}{2} \end{pmatrix}, \tag{3.71}$$

which, when solved for $\mathbf{r}_m$ and $\dot{\mathbf{r}}_m$ yields

$$\begin{pmatrix} \mathbf{r}_m \\ \dot{\mathbf{r}}_m \end{pmatrix} = \begin{pmatrix} \dfrac{\tau_{2m}}{\tau_{21}} & -\dfrac{\tau_{1m}}{\tau_{21}} \\ -\dfrac{1}{\tau_{21}} & \dfrac{1}{\tau_{21}} \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 - \dfrac{\ddot{\mathbf{r}}_m {\tau_{1m}}^2}{2} \\ \mathbf{r}_2 - \dfrac{\ddot{\mathbf{r}}_m {\tau_{2m}}^2}{2} \end{pmatrix}. \tag{3.72}$$

From this estimate of the state, the specific energy of the trajectory between these two observations can be estimated according to

$$\mathcal{E}_m = \frac{{v_m}^2}{2} - \frac{\mu}{r_m}. \tag{3.73}$$

To determine the error bounds on this estimate of $\mathcal{E}_m$, the same approach as used to determine $\hat{\boldsymbol{\Xi}}_o$ in Section 2.1.4 is applied. In this case,

$$\mathcal{E}_m = \mathcal{E}_m(\bar{\mathbf{S}}_m) \tag{3.74}$$

and

$$\bar{\mathbf{S}}_m = \bar{\mathbf{S}}_m(\boldsymbol{\Omega}), \tag{3.75}$$

where $\boldsymbol{\Omega}$ is the observation set required for the state estimate and may be written

$$\boldsymbol{\Omega} = \left(\rho_i, \dot{\rho}_i, \alpha_i, \varepsilon_i, \mathbf{S}_{is}, t_i\right)^T. \tag{3.76}$$

For the sake of analysis, the sensor states, $\mathbf{S}_{is}$, and the times of the observations, t_i, are again assumed to be known exactly. The only uncertainty is associated with the measurement of the observation attributes ρ_i, $\dot{\rho}_i$, α_i, and ε_i. It is further assumed that each attribute type is independently normally distributed and homoscedastic. That is

$$\rho_i \sim N(\mu_{\rho_i}, \sigma_\rho), \tag{3.77}$$

$$\dot{\rho}_i \sim N(\mu_{\dot{\rho}_i}, \sigma_{\dot{\rho}}), \tag{3.78}$$

$$\alpha_i \sim N(\mu_{\alpha_i}, \sigma_\alpha), \tag{3.79}$$

$$\text{and} \quad \varepsilon_i \sim N(\mu_{\varepsilon_i}, \sigma_\varepsilon). \tag{3.80}$$

Then, applying a Taylor series expansion and retaining only first-order terms, it can be shown that

$$\hat{\mathbf{P}}_m = \mathbf{J}_m \tilde{\mathbf{R}} {\mathbf{J}_m}^T, \tag{3.81}$$

where

$$\mathbf{J} = \left.\frac{\partial \hat{\mathbf{S}}_2}{\partial \boldsymbol{\Omega}}\right|_{\boldsymbol{\Omega}=\boldsymbol{\mu}_\Omega} \approx \left.\frac{\partial \hat{\mathbf{S}}_2}{\partial \boldsymbol{\Omega}}\right|_{\boldsymbol{\Omega}} \tag{3.82}$$

and

$$\tilde{\mathbf{R}} = \begin{pmatrix} \sigma_\rho & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\dot\rho} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\varepsilon & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\rho & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot\rho} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\varepsilon \end{pmatrix}. \tag{3.83}$$

Knowing $\hat{\mathbf{P}}_m$, this same approach can be applied to $\mathcal{E}_m(\bar{\mathbf{S}}_m)$ to show that

$$\sigma_{\mathcal{E}}{}^2 = \mathbf{E}_m \hat{\mathbf{P}}_m \mathbf{E}_m{}^T, \tag{3.84}$$

where

$$\mathbf{E}_m = \left.\frac{\partial \mathcal{E}_m}{\partial \bar{\mathbf{S}}_m}\right|_{\bar{\mathbf{S}}_m = \bar{\mathbf{S}}_m^*}. \tag{3.85}$$

For this specific estimate, where

$$\mathbf{r}_m = \frac{\tau_{2m}}{\tau_{21}}\left(1 - \frac{\mu}{r_1{}^3}\tau_{1m}\tau_{2m}\right)\mathbf{r}_1 - \frac{\tau_{1m}}{\tau_{21}}\left(1 - \frac{\mu}{r_2{}^3}\tau_{1m}\tau_{2m}\right)\mathbf{r}_2, \tag{3.86}$$

$$\dot{\mathbf{r}}_m = \frac{1}{\tau_{21}}\left(1 - \frac{\mu}{r_2{}^3}\tau_{1m}\tau_{2m}\right)\mathbf{r}_2 - \frac{1}{\tau_{21}}\left(1 - \frac{\mu}{r_1{}^3}\tau_{1m}\tau_{2m}\right)\mathbf{r}_1, \tag{3.87}$$

then

$$\mathbf{J}_m = \begin{pmatrix} \frac{\partial \mathbf{r}_m}{\partial \rho_1} & \frac{\partial \mathbf{r}_m}{\partial \dot\rho_1} & \frac{\partial \mathbf{r}_m}{\partial \alpha_1} & \frac{\partial \mathbf{r}_m}{\partial \varepsilon_1} & \frac{\partial \mathbf{r}_m}{\partial \rho_2} & \frac{\partial \mathbf{r}_m}{\partial \dot\rho_2} & \frac{\partial \mathbf{r}_m}{\partial \alpha_2} & \frac{\partial \mathbf{r}_m}{\partial \varepsilon_2} \\ \frac{\partial \dot{\mathbf{r}}_m}{\partial \rho_1} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \dot\rho_1} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \alpha_1} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \varepsilon_1} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \rho_2} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \dot\rho_2} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \alpha_2} & \frac{\partial \dot{\mathbf{r}}_m}{\partial \varepsilon_2} \end{pmatrix}, \tag{3.88}$$

where

$$\frac{\partial \mathbf{r}_m}{\partial \rho_1} = \frac{\tau_{2m}}{\tau_{21}}\left[\left(1 - \frac{\mu}{r_1{}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \rho_1} + \frac{3\mu}{r_1{}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1 \cdot \frac{\partial \mathbf{r}_1}{\partial \rho_1}\right)\mathbf{r}_1\right] \tag{3.89}$$

$$= \frac{\tau_{2m}}{\tau_{21}}\left[\left(1 - \frac{\mu}{r_1{}^3}\tau_{1m}\tau_{2m}\right)\hat{\boldsymbol{\rho}}_1 + \frac{3\mu}{r_1{}^5}\tau_{1m}\tau_{2m}(\mathbf{r}_1 \cdot \hat{\boldsymbol{\rho}}_1)\mathbf{r}_1\right] \tag{3.90}$$

$$\frac{\partial \mathbf{r}_m}{\partial \dot\rho_1} = \mathbf{0} \tag{3.91}$$

$$\frac{\partial \mathbf{r}_m}{\partial \alpha_1} = \frac{\tau_{2m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \alpha_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \mathbf{r}_1}{\partial \alpha_1}\right)\mathbf{r}_1\right] \tag{3.92}$$

$$= \rho_1\frac{\tau_{2m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \alpha_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \alpha_1}\right)\mathbf{r}_1\right] \tag{3.93}$$

$$\frac{\partial \mathbf{r}_m}{\partial \varepsilon_1} = \frac{\tau_{2m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \varepsilon_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \mathbf{r}_1}{\partial \varepsilon_1}\right)\mathbf{r}_1\right] \tag{3.94}$$

$$= \rho_1\frac{\tau_{2m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \varepsilon_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \varepsilon_1}\right)\mathbf{r}_1\right] \tag{3.95}$$

$$\frac{\partial \mathbf{r}_m}{\partial \rho_2} = -\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \rho_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \rho_2}\right)\mathbf{r}_2\right] \tag{3.96}$$

$$= -\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\hat{\boldsymbol{\rho}}_2+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}(\mathbf{r}_2\cdot\hat{\boldsymbol{\rho}}_2)\mathbf{r}_2\right] \tag{3.97}$$

$$\frac{\partial \mathbf{r}_m}{\partial \dot{\rho}_2} = \mathbf{0} \tag{3.98}$$

$$\frac{\partial \mathbf{r}_m}{\partial \alpha_2} = -\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \alpha_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \alpha_2}\right)\mathbf{r}_2\right] \tag{3.99}$$

$$= -\rho_2\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2}\right)\mathbf{r}_2\right] \tag{3.100}$$

$$\frac{\partial \mathbf{r}_m}{\partial \varepsilon_2} = -\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \varepsilon_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \varepsilon_2}\right)\mathbf{r}_2\right] \tag{3.101}$$

$$= -\rho_2\frac{\tau_{1m}}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2}\right)\mathbf{r}_2\right] \tag{3.102}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \rho_1} = -\frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \rho_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \mathbf{r}_1}{\partial \rho_1}\right)\mathbf{r}_1\right] \tag{3.103}$$

$$= -\frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\hat{\boldsymbol{\rho}}_1+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}(\mathbf{r}_1\cdot\hat{\boldsymbol{\rho}}_1)\mathbf{r}_1\right] \tag{3.104}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \dot{\rho}_1} = \mathbf{0} \tag{3.105}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \alpha_1} = -\frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \alpha_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \mathbf{r}_1}{\partial \alpha_1}\right)\mathbf{r}_1\right] \tag{3.106}$$

$$= -\frac{\rho_1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \alpha_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \alpha_1}\right)\mathbf{r}_1\right] \tag{3.107}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \varepsilon_1} = -\frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_1}{\partial \varepsilon_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \mathbf{r}_1}{\partial \varepsilon_1}\right)\mathbf{r}_1\right] \tag{3.108}$$

$$= -\frac{\rho_1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_1}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \varepsilon_1}+\frac{3\mu}{{r_1}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_1\cdot\frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \varepsilon_1}\right)\mathbf{r}_1\right] \tag{3.109}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \rho_2} = \frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \rho_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \rho_2}\right)\mathbf{r}_2\right] \tag{3.110}$$

$$= \frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\hat{\boldsymbol{\rho}}_2+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}(\mathbf{r}_2\cdot\hat{\boldsymbol{\rho}}_2)\mathbf{r}_2\right] \tag{3.111}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \dot{\rho}_2} = \mathbf{0} \tag{3.112}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \alpha_2} = \frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \alpha_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \alpha_2}\right)\mathbf{r}_2\right] \tag{3.113}$$

$$= \frac{\rho_2}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2}\right)\mathbf{r}_2\right] \tag{3.114}$$

$$\frac{\partial \dot{\mathbf{r}}_m}{\partial \varepsilon_2} = \frac{1}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \mathbf{r}_2}{\partial \varepsilon_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \mathbf{r}_2}{\partial \varepsilon_2}\right)\mathbf{r}_2\right] \tag{3.115}$$

$$= \frac{\rho_2}{\tau_{21}}\left[\left(1-\frac{\mu}{{r_2}^3}\tau_{1m}\tau_{2m}\right)\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2}+\frac{3\mu}{{r_2}^5}\tau_{1m}\tau_{2m}\left(\mathbf{r}_2\cdot\frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2}\right)\mathbf{r}_2\right] \tag{3.116}$$

and

$$\mathbf{E}_m = \left(-\frac{\mu x_m}{{r_m}^3}, -\frac{\mu y_m}{{r_m}^3}, -\frac{\mu z_m}{{r_m}^3}, \dot{x}_m, \dot{y}_m, \dot{z}_m\right) \tag{3.117}$$

The resulting estimates of $\mathcal{E}_m$ and $\sigma_{\mathcal{E}}$ are now used to assess the feasibility of the observation pair. If

$$\mathcal{E}_m < \mathcal{E}_{max} + 3\sigma_{\mathcal{E}}, \tag{3.118}$$

then the observation pair is added to the list of allowable pairs.

3.3 Track Selection

Once the list of allowable pairs is formed, the list of allowable triples is developed. To link two pairs of observations to form a triple, each pair must share a common observation. Therefore, to form a link, the second observation of the first pair must be the same as the first observation of the second pair. The list of pairs not ending in the current observation frame is evaluated for linkage with pairs in succeeding observation frames sharing a common mid-point and for each of these triples an initial estimate is computed. Once again, the energy of the orbit is evaluated as an additional check to determine if the triple is feasible. If it is, the calculated state and state covariance are stored and the triple is added to the list of allowable triples.

If only $O(n)$ triples are considered, then it is possible to limit the complexity of this phase to $O(n)$ by judicious application of indexing. This is due to the fact that the list of allowable pairs need be traversed only once and since it is known that the starting observation of the second pair is the same as the ending observation of the first pair. Knowing the index of the second pair in the list of allowable pairs and recalling the assumption that only one pair is likely for a given observation permits each triple to be formed directly.

3.3.1 Initial Estimate

Given that an observation triple is formed, estimates of the target position and velocity, $\mathbf{r}_m$ and $\dot{\mathbf{r}}_m$, at some time $t_m < t_o$ are now calculated using three observations, along with $\mathcal{E}_m$ and $\sigma_{\mathcal{E}}$ as was done in Section 3.2.2. If the calculated specific energy satisfies the energy restriction of Equation 3.118, the observation triple will be added to a list of allowable triples for subsequent consideration as a new track when the binary linear program of Section 3.1 is solved in Section 3.4.

Proceeding with Laplace's method, the position vector, $\mathbf{r}_m$, may be written

$$\mathbf{r}_m = \rho_m \hat{\boldsymbol{\rho}}_m + \mathbf{r}_{ms}, \tag{3.119}$$

and the velocity vector, $\dot{\mathbf{r}}_m$, is found by taking the derivative of Equation 3.119, giving

$$\dot{\mathbf{r}}_m = \dot{\rho}_m \hat{\boldsymbol{\rho}}_m + \rho_m \dot{\hat{\boldsymbol{\rho}}}_m + \dot{\mathbf{r}}_{ms}. \tag{3.120}$$

The only term not directly available from the sensor measurement set is $\dot{\hat{\boldsymbol{\rho}}}_m$, the time rate of change of the unit range vector. This term can be estimated, however, by using Lagrange interpolation to find an expression for the unit range vector

$$\hat{\boldsymbol{\rho}}_m = \frac{\tau_{m2}\tau_{m3}}{\tau_{12}\tau_{13}}\hat{\boldsymbol{\rho}}_1 + \frac{\tau_{m1}\tau_{m3}}{\tau_{21}\tau_{23}}\hat{\boldsymbol{\rho}}_2 + \frac{\tau_{m1}\tau_{m2}}{\tau_{32}\tau_{31}}\hat{\boldsymbol{\rho}}_3, \tag{3.121}$$

which, when differentiated with respect to t_m yields

$$\dot{\hat{\boldsymbol{\rho}}}_m = \frac{\tau_{m2}+\tau_{m3}}{\tau_{12}\tau_{13}}\hat{\boldsymbol{\rho}}_1 + \frac{\tau_{m1}+\tau_{m3}}{\tau_{21}\tau_{23}}\hat{\boldsymbol{\rho}}_2 + \frac{\tau_{m1}+\tau_{m2}}{\tau_{32}\tau_{31}}\hat{\boldsymbol{\rho}}_3. \tag{3.122}$$

Moreover, if $t_m = t_2$,

$$\dot{\hat{\boldsymbol{\rho}}}_2 = -\frac{\tau_{ji}}{\tau_{21}\tau_{31}}\hat{\boldsymbol{\rho}}_1 + \frac{\tau_{ji}-\tau_{21}}{\tau_{21}\tau_{ji}}\hat{\boldsymbol{\rho}}_2 + \frac{\tau_{21}}{\tau_{ji}\tau_{31}}\hat{\boldsymbol{\rho}}_3, \tag{3.123}$$

everything needed to calculate the initial state estimate is available. It is not necessary for t_m to equal one of the discrete observation times, although, if it did not, it would also be necessary to apply Lagrange interpolation to find estimates of $\hat{\boldsymbol{\rho}}_m$, ρ_m, $\dot{\rho}_m$, $\mathbf{r}_{ms}$, and $\dot{\mathbf{r}}_{ms}$. Therefore, it is simpler computationally to select the central observation time as the reference epoch.

Proceeding as in Section 3.2.2, the final feasibility check is performed on the observation triple using the energy restriction of Equation 3.118. That is, $\mathcal{E}_2$ and $\sigma_{\mathcal{E}}$ are computed by forming

$$\hat{\mathbf{P}}_2 = \mathbf{J}_2\tilde{\mathbf{R}}{\mathbf{J}_2}^T, \tag{3.124}$$

where

$$\mathbf{J}_2 = \begin{pmatrix} \dfrac{\partial \mathbf{r}_2}{\partial \rho_2} & \dfrac{\partial \mathbf{r}_2}{\partial \dot{\rho}_2} & \dfrac{\partial \mathbf{r}_2}{\partial \alpha_i} & \dfrac{\partial \mathbf{r}_2}{\partial \varepsilon_i} \\ \\ \dfrac{\partial \dot{\mathbf{r}}_2}{\partial \rho_2} & \dfrac{\partial \dot{\mathbf{r}}_2}{\partial \dot{\rho}_2} & \dfrac{\partial \dot{\mathbf{r}}_2}{\partial \alpha_i} & \dfrac{\partial \dot{\mathbf{r}}_2}{\partial \varepsilon_i} \end{pmatrix} \tag{3.125}$$

and

$$\tilde{\mathbf{R}} = \begin{pmatrix} \sigma_\rho & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{\rho}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\varepsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\varepsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\varepsilon \end{pmatrix}. \tag{3.126}$$

Specifically,

$$\frac{\partial \mathbf{r}_2}{\partial \rho_2} = \hat{\boldsymbol{\rho}}_2 \tag{3.127}$$

$$\frac{\partial \mathbf{r}_2}{\partial \dot{\rho}_2} = \mathbf{0} \tag{3.128}$$

$$\frac{\partial \mathbf{r}_2}{\partial \alpha_i} = \begin{pmatrix} \mathbf{0} & \rho_2 \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2} & \mathbf{0} \end{pmatrix} \tag{3.129}$$

$$\frac{\partial \mathbf{r}_2}{\partial \varepsilon_i} = \begin{pmatrix} \mathbf{0} & \rho_2 \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2} & \mathbf{0} \end{pmatrix} \tag{3.130}$$

$$\frac{\partial \dot{\mathbf{r}}_2}{\partial \rho_2} = \dot{\hat{\boldsymbol{\rho}}}_2 \tag{3.131}$$

$$\frac{\partial \dot{\mathbf{r}}_2}{\partial \dot{\rho}_2} = \hat{\boldsymbol{\rho}}_2 \tag{3.132}$$

$$\frac{\partial \dot{\mathbf{r}}_2}{\partial \alpha_i} = \begin{pmatrix} \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_1} & \dot{\rho}_2 \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2} + \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_2} & \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_3} \end{pmatrix} \tag{3.133}$$

$$\frac{\partial \dot{\mathbf{r}}_2}{\partial \varepsilon_i} = \begin{pmatrix} \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_1} & \dot{\rho}_2 \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2} + \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_2} & \rho_2 \frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_3} \end{pmatrix}, \tag{3.134}$$

where

$$\frac{\partial \hat{\boldsymbol{\rho}}_i}{\partial \alpha_i} = \begin{pmatrix} -\cos\varepsilon_i \sin\alpha_i \\ \cos\varepsilon_i \cos\alpha_i \\ 0 \end{pmatrix} \tag{3.135}$$

$$\frac{\partial \hat{\boldsymbol{\rho}}_i}{\partial \varepsilon_i} = \begin{pmatrix} -\sin\varepsilon_i \cos\alpha_i \\ -\sin\varepsilon_i \sin\alpha_i \\ \cos\varepsilon_i \end{pmatrix} \tag{3.136}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_1} = \frac{\tau_{23}}{\tau_{21}\tau_{31}} \frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \alpha_1} \tag{3.137}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_1} = \frac{\tau_{23}}{\tau_{21}\tau_{31}} \frac{\partial \hat{\boldsymbol{\rho}}_1}{\partial \varepsilon_1} \tag{3.138}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_2} = \frac{\tau_{ji} - \tau_{21}}{\tau_{21}\tau_{ji}} \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \alpha_2} \tag{3.139}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_2} = \frac{\tau_{ji} - \tau_{21}}{\tau_{21}\tau_{ji}} \frac{\partial \hat{\boldsymbol{\rho}}_2}{\partial \varepsilon_2} \tag{3.140}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \alpha_3} = \frac{\tau_{21}}{\tau_{ji}\tau_{31}}\frac{\partial \hat{\boldsymbol{\rho}}_3}{\partial \alpha_3} \tag{3.141}$$

$$\frac{\partial \dot{\hat{\boldsymbol{\rho}}}_2}{\partial \varepsilon_3} = \frac{\tau_{21}}{\tau_{ji}\tau_{31}}\frac{\partial \hat{\boldsymbol{\rho}}_3}{\partial \varepsilon_3}. \tag{3.142}$$

Then, the specific energy of the observation triple is

$$\mathcal{E}_m = \frac{{v_m}^2}{2} - \frac{\mu}{r_m} \tag{3.143}$$

and its variance is

$${\sigma_{\mathcal{E}}}^2 = \mathbf{E}_m \hat{\mathbf{P}}_m {\mathbf{E}_m}^T. \tag{3.144}$$

If the calculated energy, $\mathcal{E}_m$, satisfies Equation 3.118, the observation triple is added to the list of allowable triples and the final stage of the track initiation process is performed.

3.4 Cluster Initiation

With the set of allowable groups of observations to initiate clusters determined, the groups which provide the "best" overall assignment must now be chosen. To begin with, since a rather thorough examination of each triple has been performed, it can be concluded that each triple satisfies the physical constraints of the system dynamics and assumed booster characteristics, and therefore, the final solution should maximize the number of triples selected. And maximization of the total number of tracks initiated is ensured, because the total specific energy is being minimized and the specific energy of a ballistic orbit is negative. Selection is still subject to the overall assignment being feasible, however. That is, no observation can belong to more than one triple.

Therefore, if the initial list of triples is feasible, the assignment is complete and a new cluster is initiated for each triple. If the overall assignment is infeasible, however, one or more triples must be removed until the assignment is feasible. Since this process may result in more than one feasible solution, a means of discriminating among these to select the "best" one is also required.

Using the specific energy of the estimated orbits as the discriminant as developed in the binary linear program of Section 3.1, the set of clusters with the minimum total specific energy is defined to be the "best" solution. Such a discriminant is possible because targets on ballistic trajectories follow minimum energy paths through the geopotential. Any misgrouping of observations is assumed to result in a higher energy orbit.

To actually determine the optimal cluster assignments, the initial set of observation triples is evaluated for feasibility. If it is not feasible, a branch-and-bound procedure [28] is initiated which performs a depth-first-search for the optimal solution. One cluster is removed from consideration at each level and a branch is fathomed when it becomes feasible. The feasible solution with the lowest total specific energy is designated as the optimal solution and a cluster is initiated for each active triple. The worst-case complexity for this last phase is $O(2^n)$ if the entire branch-and-bound tree must be searched.

Through the multi-step elimination process the original binary linear program, of size $O(m^3n^3)$, is seen empirically to be reduced to $O(m^2n^2)$ by determining which triples are dynamically feasible. The final process of selecting the optimal solution is expected to require $O(2^n)$ time in the worst case for either approach.

Once the track initiation process is complete, all new clusters are added to the current list of clusters to be updated through the track maintenance process and their observations are removed from the pool of unassigned observations.

Chapter 4

Analysis of Results

4.1 Simulation Scenarios

To demonstrate the effectiveness of the method described in the last two chapters, five scenarios are developed. These scenarios are based upon the following assumptions:

- Launch points are randomly generated in the area 60°–70° East longitude, 50°–60° North latitude, an area covering known Soviet ICBM fields,
- 100 targets,
- Impact points are randomly generated in the area 80°–120° West longitude, 30°–50° North latitude, an area covering most of the continental United States,
- Each missile has identical characteristics, quantified by an instantaneous Q_{bo} value of 0.90 at launch,
- Launch occurs randomly over the first 30 seconds of the scenario, and
- A spherical rotating earth with no atmosphere,
- Simulation begins at 04:25:07 UTC 16 July 1986.

These scenarios assume that an ICBM attack is being observed from a single ICBM field which normally contains 50–100 ICBMs geographically dispersed to prevent multiple ICBMs from being destroyed in a single strike. The value of Q_{bo} was based upon data for existing Soviet ICBMs for which a ballistic missile defense is being contemplated[1]. From [12], knowing the maximum free-flight range angle, Ψ,

$$Q_{bo} = \frac{2\sin(\Psi/2)}{1+\sin(\Psi/2)}. \tag{4.1}$$

Each scenario uses the same launch points and launch times and pairs the launch points with impact points from a fixed set of 100 impact points. The scenarios differ in which impact points are assigned to a given launch point. The trajectories typical of these scenarios are illustrated in Figure 4.1 which shows a polar view of the complete trajectories of all 100 targets used in Scenario 1.

The sensors viewing the launch scenarios are placed in modified Molniya orbits which allow all targets to remain in view of the sensors for the duration of each of the five scenarios. The specific orbital elements used are given in Table 4.1.

Sensor	Inclination	Ascending Node	Argument of Perigee	Mean Anomaly	Eccentricity	Mean Motion
1	85.0000	70.0000	281.3211	132.4823	0.3689374	8.065369
2	85.0000	140.0000	281.3211	132.4823	0.3689374	8.065369
3	85.0000	210.0000	281.3211	132.4823	0.3689374	8.065369
4	85.0000	350.0000	281.3211	132.4823	0.3689374	8.065369

Table 4.1: Sensor Orbital Elements

Data from each sensor is provided at five-second intervals for the first six minutes of each attack scenario. This six-minute period covers the boost phase and post-boost phase prior to reentry vehicle deployment. This period is of particular interest in an SDI scenario because it is easier to destroy the targets before reentry vehicle deployment due to the smaller number and larger sizes of the targets.

Figures 4.2 through 4.5 illustrate the complexity of the problem. These figures give a good indication of the high densities of the targets being tracked.

[1] *Aviation Week & Space Technology*, March 14, 1988, page 153.

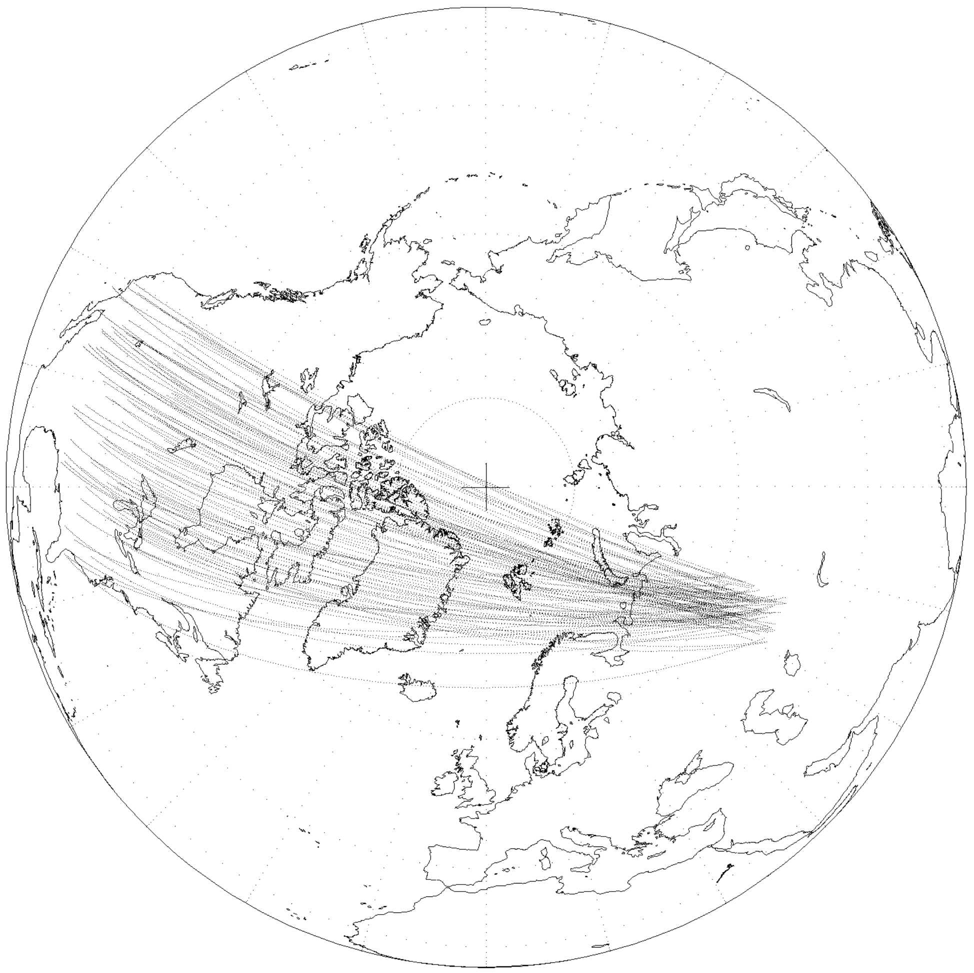

Figure 4.1: Polar View—Scenario 1

They also show large numbers of target crossings and similar tracks which typically cause the most difficulties for multi-target tracking algorithms. Each figure presents the view from one of the four sensors for the first six minutes of the ICBM attack in Scenario 1. All 100 targets are present in each view. At the bottom of each figure, the information pertaining to the sensor vantage point is displayed, showing the latitude and longitude of the satellite sub-point along with the altitude of the satellite above the earth's surface and the simulation time of that position.

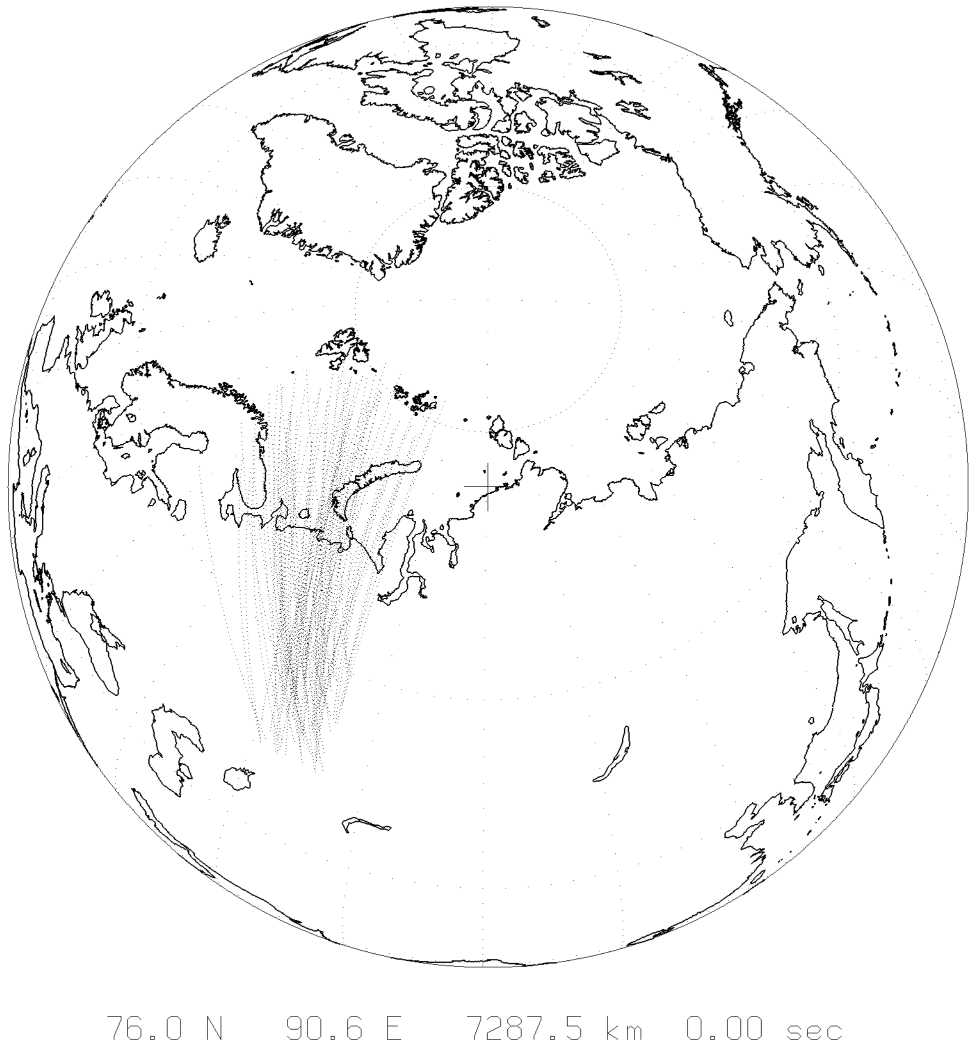

Figure 4.2: View from Sensor 1—Scenario 1

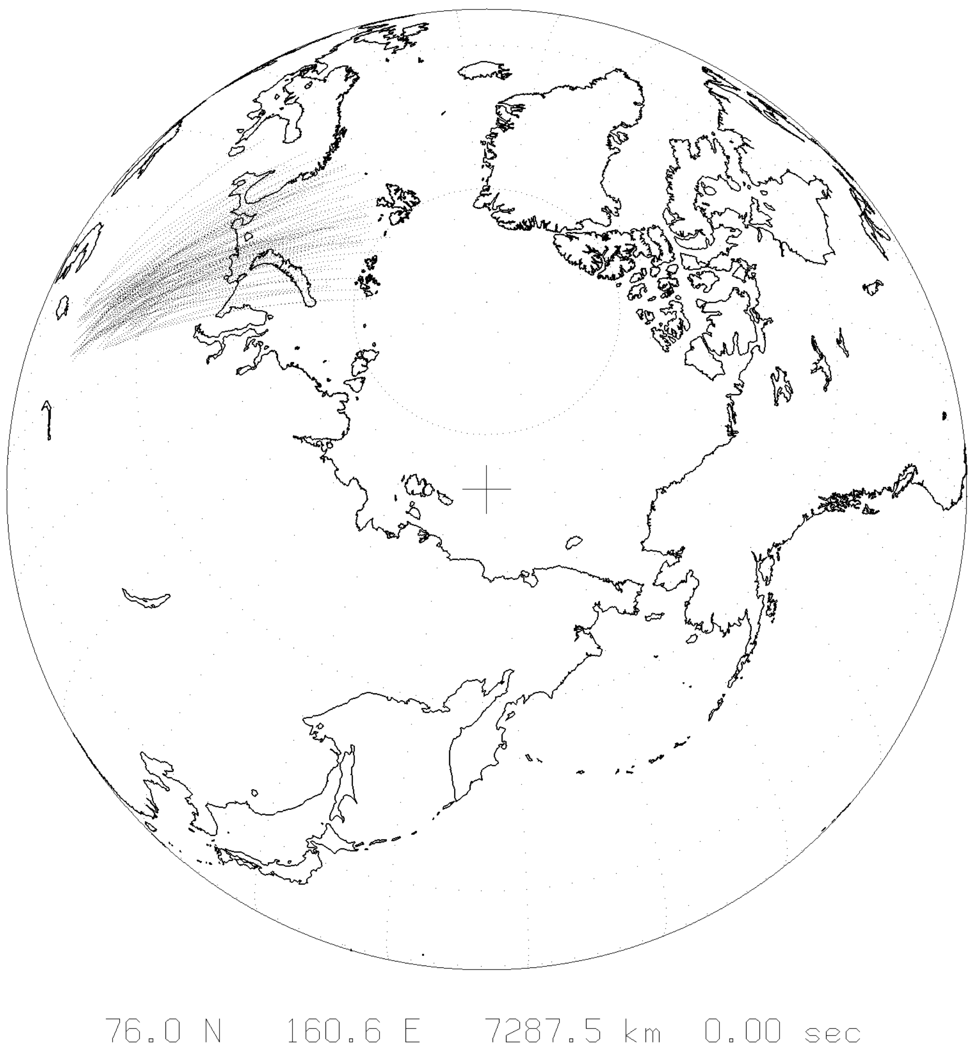

Figure 4.3: View from Sensor 2—Scenario 1

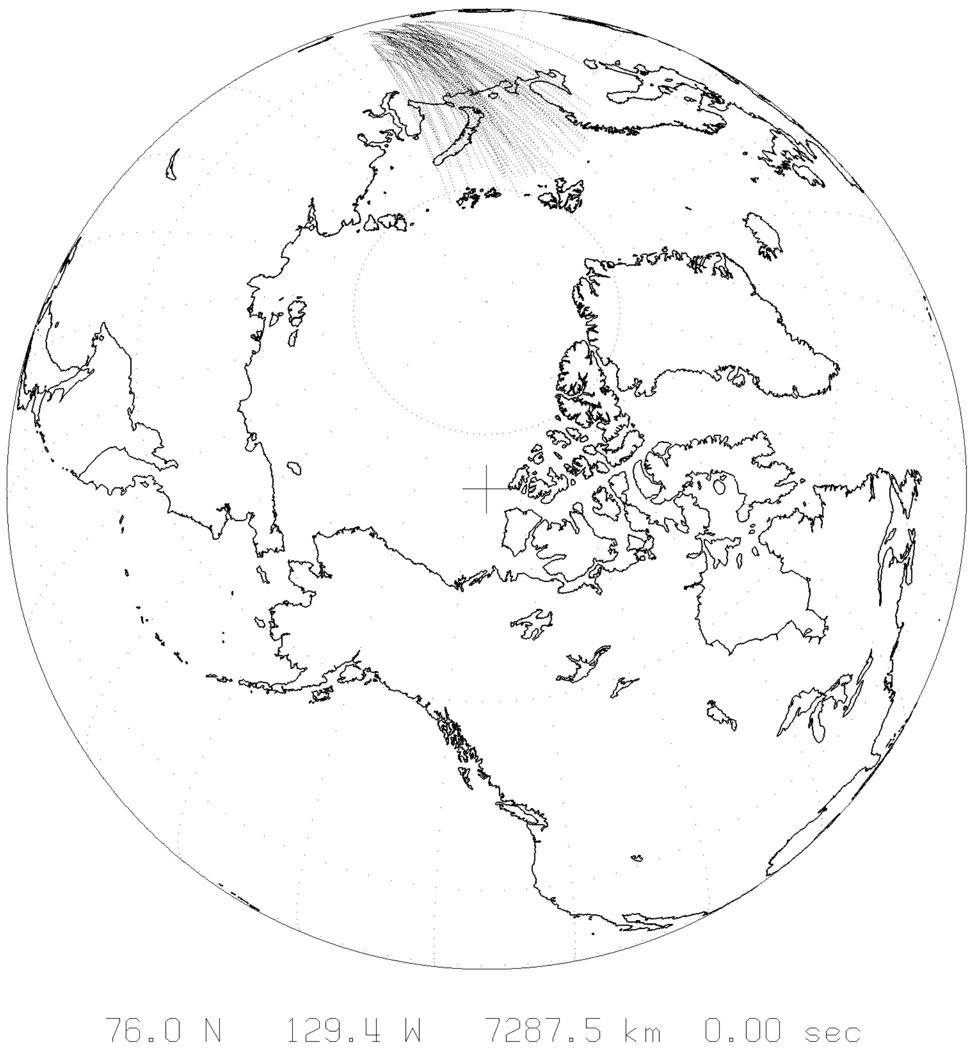

Figure 4.4: View from Sensor 3—Scenario 1

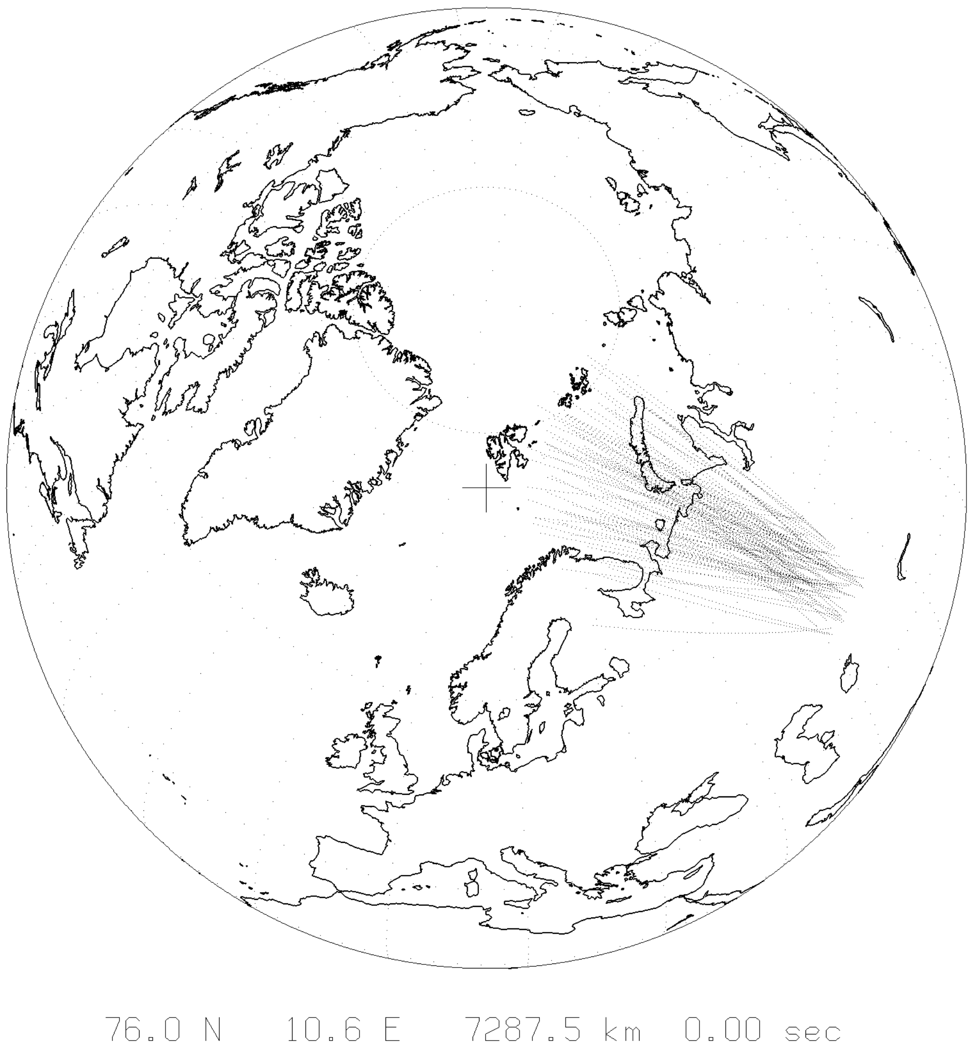

Figure 4.5: View from Sensor 4—Scenario 1

Four variance sets and four levels of missing data were considered during the course of this investigation to examine the effects of high and low quality data. The scenario variances are listed in Table 4.2. The four levels of missing data are 0, 5, 10, and 20 percent.

Variance Set	Variances			
	Range (meters)^2	Range Rate (meters/second)^2	Azimuth (radians)^2	Elevation (radians)^2
1	10.0	1.0	10^{-7}	10^{-7}
2	10.0	1.0	10^{-8}	10^{-8}
3	10.0	1.0	10^{-9}	10^{-9}
4	10.0	1.0	10^{-10}	10^{-10}

Table 4.2: Scenario Variances

To evaluate the performance of the temporal clustering algorithm, two sets of variances and levels of missing data are used. The first case, representing expected values for the measurement variances and levels of missing data, uses Variance Set 3 and 5 percent missing data, while the second case, representing worst-case values, uses Variance Set 1 and 20 percent missing data.

4.2 Discussion of Results

The temporal clustering procedure described in the previous two chapters is implemented in Pascal on the Cray X-MP/24 at The University of Texas at Austin's Center for High Performance Computing. A listing of the code used is included in Appendix B. Before the code can be run, however, determinations of the size of the track initiation buffer and the maximum number of missing observations for track termination are necessary. Based upon the results presented in Table 3.1 on page 32 and Table 2.1 on page 27, a track initiation buffer of seven observation frames and a maximum of five consecutive missing observations was selected as a termination criterion for the maximum value of 20 percent missing data, since the expected number of track initiation failures and false terminations over the life of each of the scenarios investigated is less than one.

Tables 4.3 and 4.4 are summaries of the results of the twenty simulation runs (five scenarios with four sensors each) used to investigate the performance of the temporal clustering algorithm for both the expected and worst-case values of measurement noise and missing data.

Scenario/	Perfect		Unassigned	Errors		
Sensor	Clusters	Targets	Observations	Termination	Gating	Observations
1/1	100	100	0	0	0	6582/6944
1/2	100	99	1	0	0	6588/6944
1/3	100	100	0	0	0	6591/6944
1/4	100	100	0	0	0	6647/6944
2/1	100	99	1	0	0	6582/6944
2/2	100	99	1	0	0	6588/6944
2/3	100	100	0	0	0	6591/6944
2/4	100	100	0	0	0	6647/6944
3/1	100	99	1	0	0	6582/6944
3/2	100	99	1	0	0	6588/6944
3/3	100	100	0	0	0	6591/6944
3/4	100	100	0	0	0	6647/6944
4/1	100	99	1	0	0	6582/6944
4/2	100	99	1	0	0	6588/6944
4/3	100	100	0	0	0	6591/6944
4/4	100	100	0	0	0	6647/6944
5/1	100	100	0	0	0	6582/6944
5/2	100	99	1	0	0	6588/6944
5/3	100	100	0	0	0	6591/6944
5/4	100	100	0	0	0	6647/6944
Total	2000	1992	8	0	0	132040/138880

Table 4.3: Summary of Results—Nominal Values

The first column of the table indicates the scenario number and sensor number of the observations. The next two columns show the number of perfect clusters and perfectly clustered targets for each run. A cluster is considered perfect if all of its observations are from a single target. No imperfect clusters were encountered in any of the runs examined. A target is considered perfectly clustered if all of its observations are contained in a single cluster.

On average, each run in the nominal case contained less than one imperfectly clustered target, resulting from a single observation being excluded from

Scenario/	Perfect		Unassigned	Errors		
Sensor	Clusters	Targets	Observations	Termination	Gating	Observations
1/1	102	97	1	1	1	5556/6944
1/2	101	99	0	0	1	5568/6944
1/3	101	99	0	1	0	5536/6944
1/4	101	98	2	0	1	5561/6944
2/1	102	97	1	1	1	5556/6944
2/2	102	98	0	0	2	5568/6944
2/3	101	99	0	1	0	5536/6944
2/4	101	98	2	0	1	5561/6944
3/1	102	97	1	1	1	5556/6944
3/2	101	99	0	0	1	5568/6944
3/3	101	99	0	1	0	5536/6944
3/4	101	98	2	0	1	5561/6944
4/1	102	97	1	1	1	5556/6944
4/2	102	98	0	0	2	5568/6944
4/3	101	99	0	1	0	5536/6944
4/4	101	98	2	0	1	5561/6944
5/1	102	96	2	1	1	5556/6944
5/2	101	99	0	0	1	5568/6944
5/3	101	99	0	1	0	5536/6944
5/4	101	98	2	0	1	5561/6944
Total	2027	1962	16	10	17	111105/138880

Table 4.4: Summary of Results—Extreme Values

a gate either during the track maintenance or track initiation phase. The frequency of occurrence is shown in column four of Table 4.3. However, *all* clusters developed were perfect. Considering the large number of target measurements observed (132,040) in the twenty cases evaluated, losing only eight observations is an exceptional performance.

Using the worst-case values, each run averages one split cluster and two improperly clustered targets, still quite good performance. These failures result from one of three causes and their frequencies are shown in columns 4–6 of Table 4.4. The first cause is the result of an observation not being assigned to any cluster. In five cases, two observations were lost during the track initiation process because of a failure to obtain three observations in the seven-frame track initiation buffer. Each of these failures could have been avoided by simply expanding the size of the buffer but the increased computational complexity and storage overhead may not merit such a change. This result is not unexpected based on the numbers in Table 3.1.

Additionally, one observation of a target was not included in any cluster because the problem reduction phase of the track initiation process incorrectly eliminated it from consideration in a feasible track based upon system dynamics. The remaining five unassigned observations were the result of correct assignments being discarded during the track maintenance process because they failed the gating criteria.

The other reasons for incorrectly clustered targets are due to track termination as the result of a termination or gating error. A termination error occurs when a track is terminated because more than the maximum number of missing observations occurred. Again, this type of failure could be avoided by increasing the maximum number of missing observations with a corresponding trade-off in computational cost. Increasing the buffer size to eight observation frames would, therefore, have eliminated ten tracking errors.

A gating error occurs when a track is terminated because the estimate of a target's state is sufficiently far from the true state to prevent the correct observations from falling within the observation gates. All seventeen gating errors occurred immediately after track initiation during the EKF stabilization phase. These gating errors could be reduced (or effectively eliminated) through either reduced measurement noise (improved sensor characteristics) or improved orbit determination procedures. This conclusion is supported by the total lack of gating errors in the cases using nominal values of measurement noise and missing data.

And, as expected, lower measurement noise results in better estimates of the target state. The effect of various levels of measurement noise on the estimates of target position and velocity are illustrated in Figures 4.6 and 4.7. Each figure shows the difference between the true and estimated position or velocity for the four levels of measurement noise considered. All data is for Scenario 1, Sensor 1, Cluster 1.

As a final test of the robustness of the temporal clustering algorithms, biases were introduced into the sensor attitude and, independently, in the sensor clock and applied to the worst-case runs. An attitude bias is possible due to

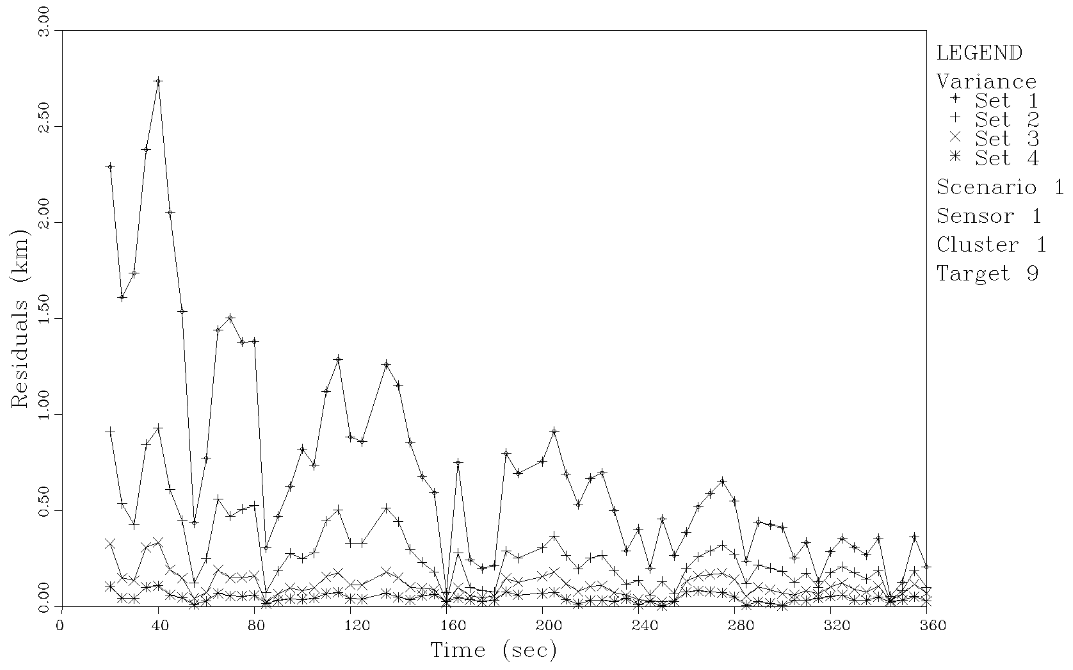

Figure 4.6: Position Error Due to Measurement Noise

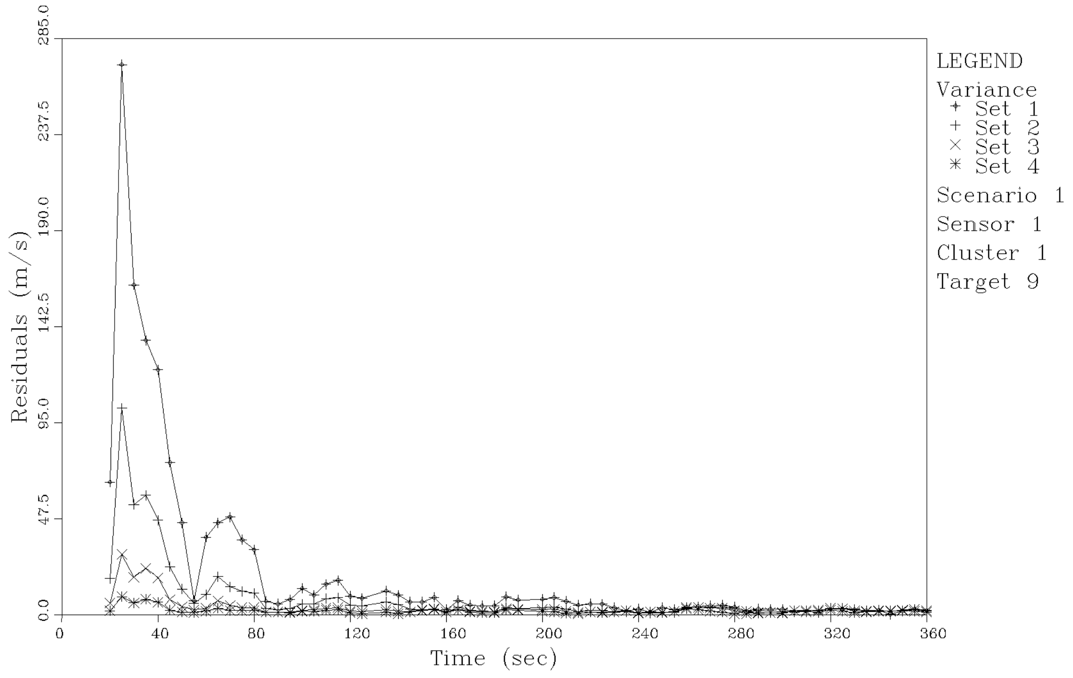

Figure 4.7: Velocity Error Due to Measurement Noise

precessional and nutational effects of the sensor platform during the course of its orbit. A constant bias of up to 60 arc seconds results in only two additional unassigned observations in the twenty runs considered, both failures in the track initiation process. These biases are much larger than one would expect in practice, since the slowly varying effects would be expected to be small and daily observation of the sensor platform attitude by ground personnel would allow the development of models to remove most of the remaining bias.

A clock bias would result in the sensor thinking it was in a position different from that indicated by its onboard ephemeris. Typically, with the atomic clocks available on most satellites these biases would be on the order of microseconds or, at worst, milliseconds. Yet, the temporal clustering algorithms performed exactly as shown in Table 4.4 with biases of up to 0.5 seconds, far above any bias which could reasonably be expected.

While the temporal clustering algorithm performs admirably even under these adverse conditions of attitude and clock biases, the effects on the state estimate and state covariance are considerably degraded, as can be seen in Figures 4.8 through 4.11. Each figure shows the difference between the true and estimated position or velocity for various levels of bias. All data is for Scenario 1, Sensor 1, Cluster 1. Obviously, while the algorithms perform well under these circumstances, it is highly desirable to reduce the sensor biases as much as possible to attain the high degree of accuracy necessary to direct a proper response.

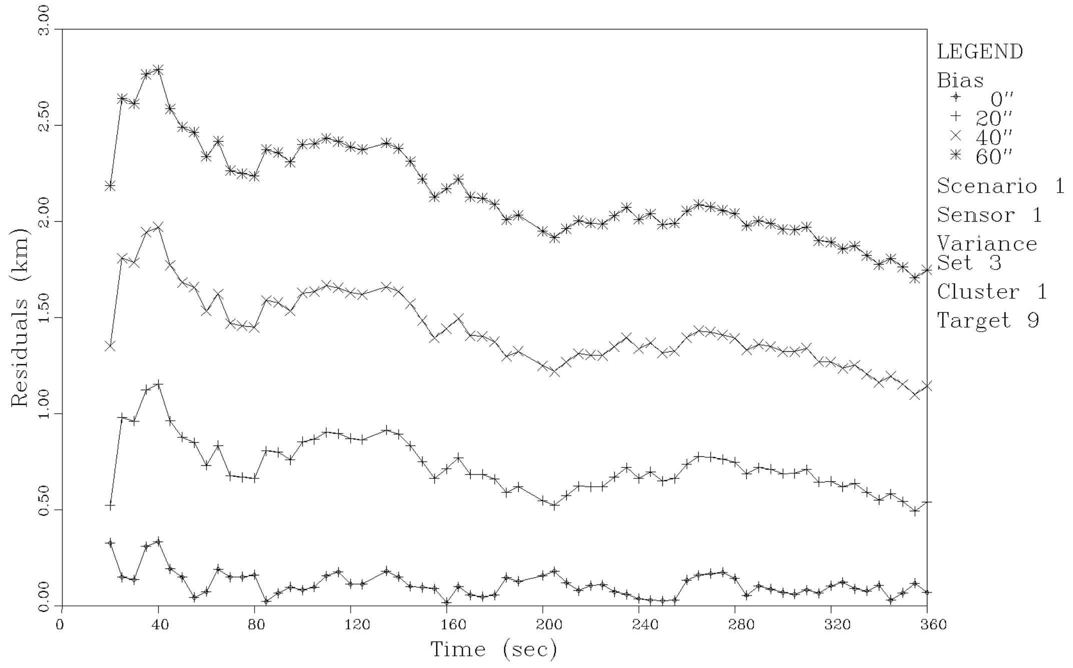

Figure 4.8: Position Error Due to Attitude Bias

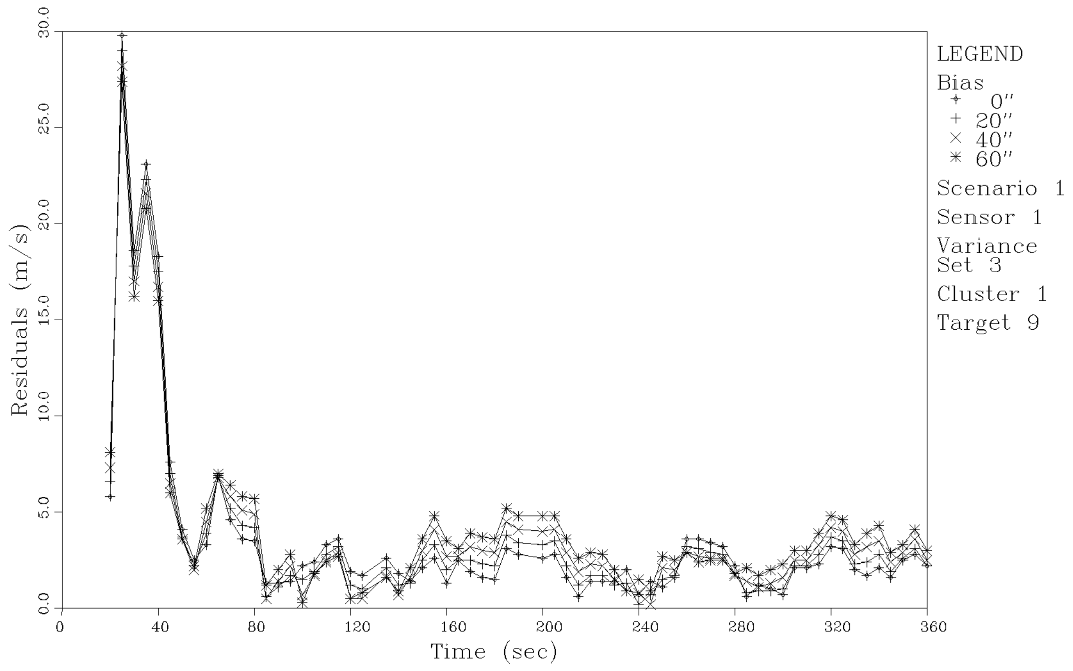

Figure 4.9: Velocity Error Due to Attitude Bias

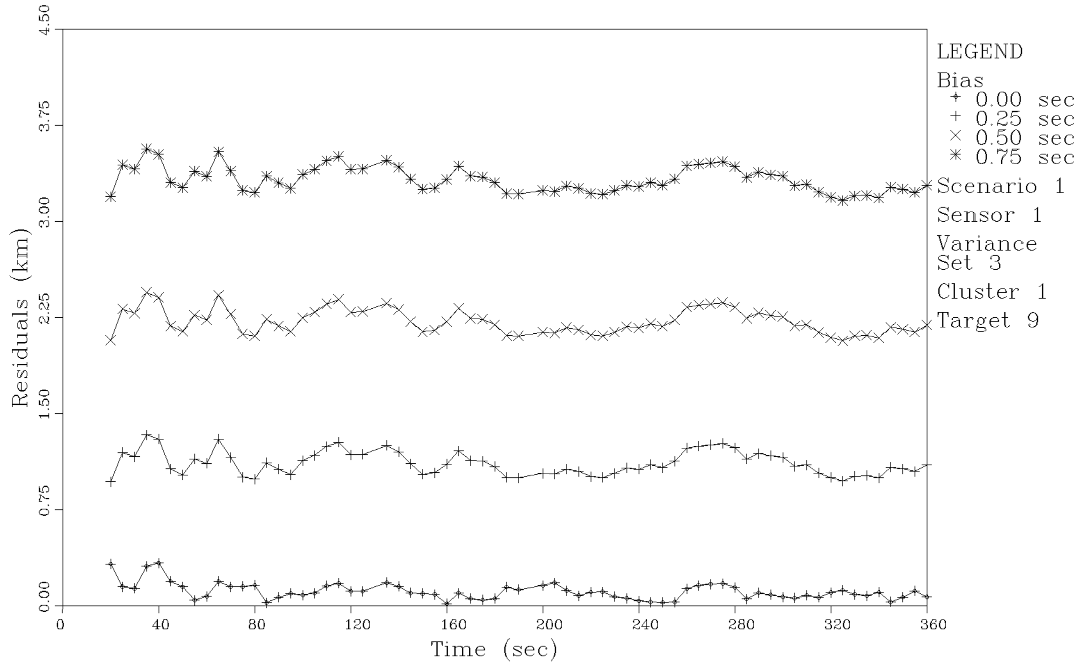

Figure 4.10: Position Error Due to Clock Bias

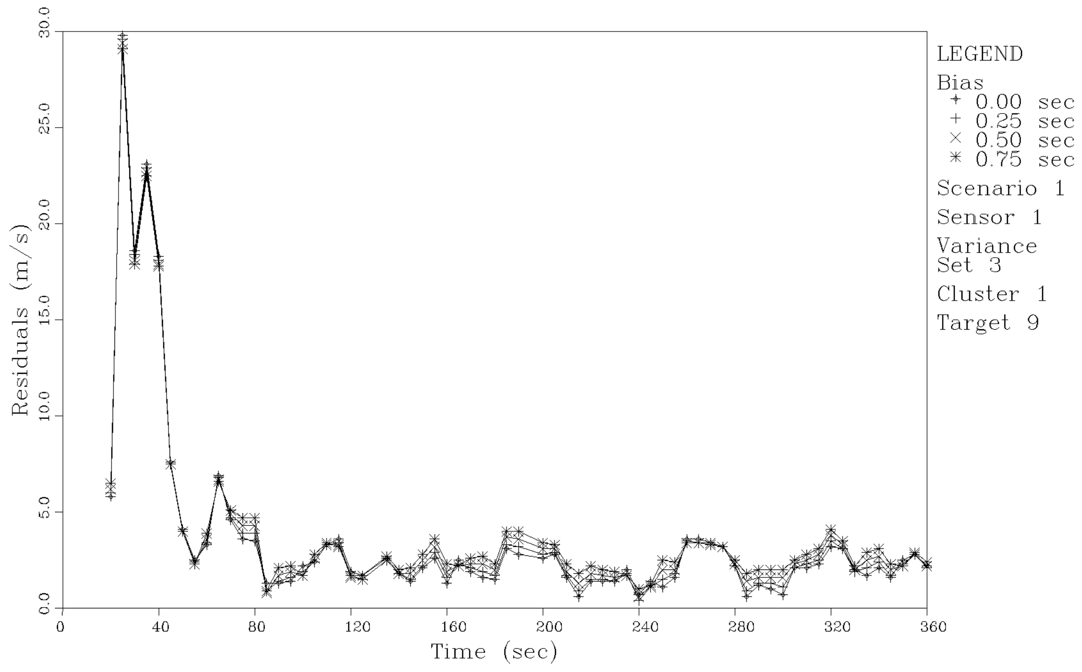

Figure 4.11: Velocity Error Due to Clock Bias

Chapter 5

Conclusions

The temporal clustering algorithm developed in Chapters 2 and 3 does indeed accomplish the initial objectives that it

- Perform both track initiation *and* track maintenance and
- Permit processing of data in real time while minimizing
  - Computational complexity and
  - Data storage requirements.

Through prudent application of existing solution techniques in astrodynamics, mathematical programming, numerical analysis, and statistical estimation, an integrated solution is developed which is not only capable of performing *both* track initiation and track maintenance, but also improves on previous work in the fields of multi-target tracking and clustering to effectively track large numbers of targets in real time.

In fact, the duty cycle for each run (the ratio of the total execution time to the elapsed scenario time) is only 4.4 percent. Execution time for all runs averages 15.9 CPU seconds per run. The portion of that time spent in each major procedure discussed in Figure 1.2 is provided in Table 5.1. Considering that only 17.5 percent of of the total execution time is taken up by algorithms of complexity worse than $O(n)$ and 76.7 percent is devoted to algorithms which are vectorizable and capable of being run in parallel, the timing results are quite impressive. In fact, empirical results from runs with 20, 50, and 100 targets showed the complexity of *Perform Cluster Assignments* and *Perform Track Initiation* to be only $O(n^2)$ and $O(n)$, respectively.

Procedure	Percent Total Execution Time	Complexity
1) Read Observation Frames	5.8	$O(k)$
2) Forecast Existing Clusters	47.5	$O(n)$
3) Calculate Assignment Costs	6.5	$O(n^2)$
4) Perform Cluster Assignments	8.5	$O(n^3)$
5) Update Clusters	29.2	$O(n)$
6) Perform Track Initiation	2.5	$O(2^n)$

Table 5.1: Timing Results

It should be cautioned, however, that these timing results apply only to the specific case investigated in this study. Any combination of measurement types and system dynamics which does not permit a significant reduction in the size of the binary linear program (as was done in Section 3.2) may not be able to satisfy the requirement that the problem be solved in real time. In such circumstances, more sophisticated methods of solving the binary linear program may be implemented to reduce the computational complexity, although there is no guarantee that these improved techniques will allow real time processing in all cases.

The limitations due to computational complexity and data storage requirements are reduced through the judicious application of existing algorithms for filtering (the Extended Kalman filter), assignment (the Hungarian method), and quadratic programming (branch-and-bound) while taking full advantage of the temporal component of the data and system dynamics. And, as seen in Section 2.3 and at the beginning of Chapter 3, additional reductions in complexity and data storage requirements are possible if the missing data rate is kept small.

A most remarkable feature of the temporal clustering algorithm is its ability to function well when faced with low data rates and high levels of both missing data and measurement noise. Even in the runs examined using the worst-case variances (Variance Set 1) and 20 percent missing data, the temporal clustering algorithm successfully clustered nearly 100 percent of the over 110,000 observations available. And for tracks with four or more observations all but 10 out of 2000 targets were tracked correctly throughout the scenarios, and those were the result of exceeding the size of the track termination buffer—something which can be avoided by simply increasing the size of the buffer by one frame. And even the addition of biases in the sensor attitude or clock to levels well in excess of those which can be reasonably expected failed to significantly affect the performance of the algorithm.

5.1 Future Research

There remain areas in which the temporal clustering algorithm can be improved or the scope of its application broadened. More attention can be applied to improving the overall efficiency of the tracking process through the application of state-of-the-art filters such as those discussed by Kaminski, Bryson, and Schmidt [35] and Verhaegen and Van Dooren [57] or simplification of the filters through the application of constant gain Kalman filters as discussed by Blackman [13].

In addition, significant advantage can be gained by exploiting the parallel structure of many existing computers and the application of pipelining as discussed by Allen, Kurien, and Washburn [1]. For example, considerable improvement in processing time can be achieved by developing a parallel structure capable of independently tracking each target, especially since massively parallel architectures with 65,536 processors exist today. This is particularly true considering that over 75 percent of the total execution time for the temporal clustering procedure is used by the forecasting and state update algorithms. These functions can easily be performed on separate parallel processors for each cluster. Use of vector processing is also helpful when integrating the large state and state covariance vectors in the forecasting algorithm.

And while the case presented assumes a spherical earth with no drag and no thrust, the method can be readily extended to cases using higher order gravitational potentials and atmospheric drag by simply reformulating the filter and track initiation gating process to specifically account for these effects. And, depending upon the specific system dynamics, it may also be necessary to choose another cost coefficient for the objective function of the binary linear program. Reformulation to account for other measurement combinations, such as the angles-only case, is possible, although preliminary investigations have shown this case to be somewhat more complex due to marginal observability. Investigations of vehicle thrust and maneuvering are certainly possible, as well, although the reformulation will likely be considerably more difficult and require the use of adaptive filtering techniques. Additional investigations of algorithm performance in the face of changing measurement covariance due to sensor degradation or failure are also possible through the application of adaptive filtering.

And, finally, the extension of this temporal clustering approach to the broader issue of multi-sensor correlation should be straightforward. Although issues of distributed processing [21,22,23] need to be examined in detail, it appears that once a cluster is established, its state estimate and state covariance matrix can be transmitted by each sensor to a central processor for correlation with data from other sensors. Transmission of this minimal amount of data significantly reduces the bandwidth required for data exchange and the correlation process can then apply gating and assignment procedures quite similar to those used in the single-sensor case.

While the temporal clustering process as developed here is specifically tailored for a single application, prudent modification of the application-specific portions should allow it to be applied to other ballistic tracking problems or even those in the areas and fluid dynamics or particle physics.

Appendix A

Simulation Design

To permit realistic assessment of the approach presented here, some means of providing a set of target measurements was necessary. To accomplish this objective, a data generating program (GENDAT) was developed by the author and Stuart H. Smith. A flowchart of GENDAT is presented in Figure A.1.

GENDAT was designed to generate both the target and sensor states using a Runge-Kutta 4(5) integrator and a user-provided force model. Input to GENDAT begins by determining the booster characteristic thrust, Q_{bo}, where

$$Q_{bo} = \frac{{v_{bo}}^2 r_{bo}}{\mu}, \tag{A.1}$$

and whether high or low trajectories are to be used for the targets. Then, the launch time and the launch and impact points for each target are input. Finally, the Keplerian orbital elements for each sensor are input.

An initial state estimate is then determined for each target at its specified launch time by computing the trajectory necessary to reach the assigned impact point from the designated launch point. Initial state estimates for the sensors are also determined by converting the orbital elements at the initial simulation time.

Once all the initial states are calculated, GENDAT begins forming observations of each target from each sensor while the target is in view. Currently, the target is in view if it has launched and not yet reached its impact point. No additional considerations such as sensor field-of-view, sensor range, or obscuration by the earth's limb are yet implemented. Observations are calculated using the transformation described in Section 2.1.4.

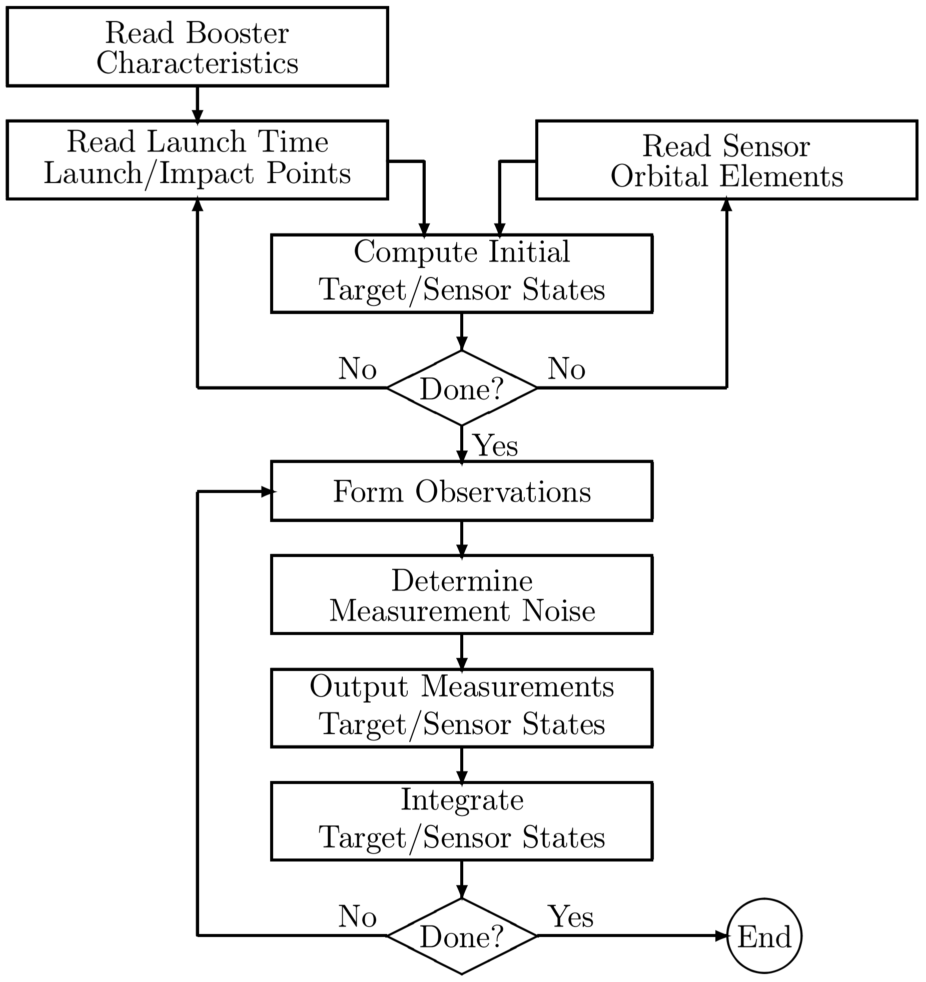
Read Booster Characteristics
Read Launch Time Launch/Impact Points
Read Sensor Orbital Elements
Compute Initial Target/Sensor States
No
Done?
No
Yes
Form Observations
Determine Measurement Noise
Output Measurements Target/Sensor States
Integrate Target/Sensor States
No
Done?
Yes
End

Figure A.1: GENDAT Flowchart

For each measurement attribute, a Gaussian random variable is computed to allow for expected measurement noise. These variables are $N(0, 1)$. In addition, a single uniform random variable is also computed for use in simulating the stochastic nature of detecting an observation.

The target and sensor states are output to separate files, the former to be available for comparison with estimated target states, the latter to act as the satellite ephemeris. The target measurements and associated random variables are output to a third file. The separation of the true measurements from the measurement noise allows the temporal clustering algorithm to incorporate differing measurement variances at runtime.

After all observations are formed and the resulting data is output, the target and sensor states are integrated to the next observation time and the process is repeated until the simulation end time or all targets have reached their destinations.

Appendix B

Program Listing

```
Program TCluster(tmdata,ssdata,tsdata,tcldata,trudata,csdata,input,output);
{              Author:  TS Kelso
{    Original Version:  26 January 1987 }
{    Current Revision:  26 May 1988 }
{ Program Description:  Program performs temporal clustering on time
                        successive data frames using an assignment from last
                        member of existing clusters to observations.  Track
                        initiation performed using quadratic program.  Assumes
                        range, range rate, azimuth, and elevation
                        measurements. }
(*#A+:R- *)

const
  clusters   = 201;                            {Maximum allowable clusters + 1}
  max_pairs  = 400;    {Maximum number of pairs in Perform_Cluster_Initiation}
  nest       =   6;                                  {Elements in state vector}
  block      =   3;                             {Axes in ECI coordinate system}
  nstack     =  42;         {Elements in stacked state vector = nest + nest^2}
  nterms     =   8;                      {Maximum number of terms in estimates}
  attributes =   4;        {Maximum number of possible measurement attributes}
  bad        =   5.0;       {Metric value for bad assignment = attributes + 1}
  frame1     =  -6;           {Data required for initial estimate = frame1..0}
  max_prop   =   5;                   {Maximum number of propagation intervals}
  max_missed =   1;                    {Maximum number of missed gates allowed}
  zero       =   1.0E-14;                   {Machine epsilon for real = double}
  big        =   1.0E+14;
  mu         =   3.9860064E+14;  {Geocentric gravitational parameter, m^3/s^2}
  small      =   1.0E-12;                      {RK78 integration control factor}
  pi         =   3.1415926535897932;
  max_energy =  -3.437E7;         {Maximum specific energy, meters^2/second^2}

type
  span           = frame1..0;
  atr_vector     = array [0..attributes] of real;
  obs_vector     = array [1..attributes] of real;
  state_vector   = array [1..nest] of real;
  stacked_vector = array [1..nstack] of real;
  limits         = array [1..2] of obs_vector;
  bvector        = array [1..clusters] of boolean;
  ivector        = array [0..clusters] of integer;
  frame          = array [1..clusters] of atr_vector;
  J_matrix       = array [1..nest,1..nterms] of real;
  ER_matrix      = array [1..nterms,1..nterms] of real;
  P_matrix       = array [1..nest,1..nest] of real;
  H_matrix       = array [1..attributes,1..nest] of real;
  K_matrix       = array [1..nest,1..attributes] of real;
  R_matrix       = array [1..attributes,1..attributes] of real;
  S_matrix       = array [span] of state_vector;
  O_matrix       = array [span] of obs_vector;
  M_matrix       = array [1..clusters,1..clusters] of real;
  states = record
    number : integer;
    time   : real;
    values : state_vector;
    end; {record}
  measures = record
    target,sensor : integer;
    time          : real;
    obs,error     : obs_vector;
    missing       : real;
    end; {record}

var
  EOI                               : boolean;                 {End of input}
  sensor,sen_nr,max_target,
  next_target,last_target,time      : integer;
  max_time,min_time,tbias,
  step,missing_flag,missing_limit   : real;
  nr_unassigned,nr_obs,
  nr_clusters,nr_active,nr_inactive : array [span] of integer;
  frame_time                        : array [span] of real;
  nr_missing,convert,
  col_basis,row_basis               : ivector;
  status,observation,targets        : array [span] of ivector;
  assigned                          : array [frame1..1] of bvector;
  R,bias                            : obs_vector;
  span_limits                       : limits;
  next                              : atr_vector;
  rkf                               : array [1..59] of real;
  Sxs                               : array [span] of state_vector;
  Identity,Q_k                      : P_matrix;
  R_k                               : R_matrix;
  metric,ametric                    : M_matrix;
  Rvar                              : array [1..3] of ER_matrix;
  attr                              : array [span] of frame;
  est,gate                          : array [1..clusters] of obs_vector;
  Sx                                 : array [1..clusters] of state_vector;
  P                                  : array [1..clusters] of P_matrix;
  stan                               : array [span,1..clusters] of obs_vector;
  atr_limits                         : array [frame1..1] of limits;
  current_obs                        : measures;
  current_sensor,current_target      : states;
  tmdata                             : file of measures;
  ssdata,tsdata                      : file of states;
  tcldata,trudata,csdata             : text;
  times                              : array [0..9] of integer;
  last,total                         : array [0..9] of real;

{*** Timing Functions ***************************************************}

Function Second : real; FORTRAN;

Procedure Init_Times;
  var
    i : integer;
  begin
  for i := 1 to 9 do
    begin
    last[i]  := 0.0;
    total[i] := 0.0;
    times[i] := 0;
    end; {for i}
  end; {Procedure Init_Times}

Procedure Start_Timer(arg : integer);
  begin
  last[arg] := Second;
  end; {Procedure Start_Timer}

Procedure Stop_Timer(arg : integer);
  var
    elapsed : real;
  begin
  elapsed := Second - last[arg];
  total[arg] := total[arg] + elapsed;
  times[arg] := times[arg] + 1;
  end; {Procedure Stop_Timer}

Procedure Report_Times(arg : integer);
  var
    i : integer;
  begin
  for i := 1 to arg do
    begin
    Write(i:2,') ');
    case i of
      1 : Writeln('Input Data');
      2 : Writeln('Forecast');
      3 : Writeln('Calculate Metrics');
      4 : Writeln('Perform Cluster Assignment');
      5 : Writeln('Perform Cluster Initiation');
      6 : Writeln('Update Estimates');
      end; {case}
    Writeln('    Elapsed time = ',total[i]:7:4,
              ', Average time = ',total[i]/times[i]:7:4,
              ', Percentage = ',100*total[i]/total[0]:4:1,'%');
    end; {for i}
  Writeln('    Total time   = ',total[0]:7:4);
  Writeln;
  end; {Procedure Report_Times}

{*** Global routines *******************************************************}

Function IMin(arg1,arg2 : integer) : integer;
  begin
  if arg1 < arg2 then
    IMin := arg1
  else
    IMin := arg2;
  end; {Function IMin}

Function IMax(arg1,arg2 : integer) : integer;
  begin
  if arg1 > arg2 then
    IMax := arg1
  else
    IMax := arg2;
  end; {Function IMax}

Function RMin(arg1,arg2 : real) : real;
  begin
  if arg1 < arg2 then
    RMin := arg1
  else
    RMin := arg2;
  end; {Function RMin}

Function RMax(arg1,arg2 : real) : real;
  begin
  if arg1 > arg2 then
    RMax := arg1
  else
    RMax := arg2;
  end; {Function RMax}
```

```
Procedure Increment(var arg : integer);
  begin
  arg := arg + 1;
  end; {Procedure Increment}

Procedure Decrement(var arg : integer);
  begin
  arg := arg - 1;
  end; {Procedure Decrement}

Function Missing(target,sen : integer;
                   time,val : real) : boolean;
  begin
  Missing := (val < missing_limit)
          or (target > max_target)
          or (sen <> sensor)
          or (time < min_time);
  end; {Function Missing}

Procedure Echo_True_Data(target,sen : integer;
                           time,val : real);
  begin
  if (sen = sensor)
  and (time >= min_time)
  and (target <= max_target) then
    begin
    while (target - last_target) > 1 do
      begin
      Write(trudata,'    ');
      Increment(last_target);
      end; {while}
    Increment(last_target);
    if val < missing_limit then
      begin
      Write(trudata,0:4);
      end {if}
    else
      begin
      Write(trudata,target:4);
      end; {else}
    end; {if}
  end; {Procedure Echo_True_Data}

{*** Initializations *******************************************************}

Procedure Init_Program;                                      {System Specific}
  var
    i,j,k,atr,var_set : integer;
    filename          : packed array[1..12] of char;
  begin
{Select data file to cluster}
  Readln(filename);
{Determine attribute and clock biases}
  for i := 1 to attributes do
    Readln(bias[i]);
  Readln(tbias);
{Determine observation sensor}
  sensor := Ord(filename[2]) - Ord('0');
{Set variances}
  var_set := Ord(filename[3]) - Ord('0');
  for i := 1 to attributes do
    for j := 1 to attributes do
      R_k[i,j] := 0.0;
  R_k[1,1] := 10.0;
  R_k[2,2] :=  1.0;
  R_k[3,3] := Exp(-(var_set+2)*Ln(10.0));
  R_k[4,4] := Exp(-(var_set+2)*Ln(10.0));
  for atr := 1 to attributes do
    R[atr] := Sqrt(R_k[atr,atr]);
  for i := 1 to 3 do
    for j := 1 to nterms do
      for k := 1 to nterms do
        Rvar[i,j,k] := 0.0;
  for atr := 1 to attributes do
    begin
    Rvar[1,atr,atr] := R_k[atr,atr];
    Rvar[2,atr,atr] := R_k[atr,atr];
    Rvar[2,atr+attributes,atr+attributes] := R_k[atr,atr];
    end; {for atr}
  Rvar[3,1,1] := R_k[1,1];
  Rvar[3,2,2] := R_k[2,2];
  for i := 3 to 5 do
    begin
    Rvar[3,i,i] := R_k[3,3];
    Rvar[3,i+3,i+3] := R_k[4,4];
    end; {for i}
  case filename[5] of  {Determine number of targets}
    'V','v' : max_target :=   5;
    'X','x' : max_target :=  10;
    'T','t' : max_target :=  20;
    'L','l' : max_target :=  50;
    'C','c' : max_target := 100;
    end; {case}
  case filename[6] of  {Determine time span}
    'A','a' : begin
              min_time :=   50.0;
              max_time :=  200.0;
              end; {Case A}
    'B','b' : begin
              min_time :=    0.0;
              max_time :=  200.0;
              end; {Case B}
    'C','c' : begin
              min_time :=    0.0;
              max_time :=  300.0;
              end; {Case C}
    'D','d' : begin
              min_time :=    0.0;
              max_time :=  100.0;
              end; {Case D}
    'E','e' : begin
              min_time :=    0.0;
              max_time := 2500.0;
              end; {Case E}
    'F','f' : begin
              min_time :=    0.0;
              max_time :=  360.0;
              end; {Case F}
    end; {case}
  case filename[7] of  {Determine missing data rate}
    'N','n' : missing_limit := 0.00;
    'Y','y' : missing_limit := 0.05;
    'X','x' : missing_limit := 0.10;
    'Z','z' : missing_limit := 0.20;
    end; {case}
{Initialize input data file}
  Connect(tmdata,'TMDATA  ');
  Reset(tmdata);
{Initialize sensor input data file}
  Connect(ssdata,'SSDATA  ');
  Reset(ssdata);
{Initialize target input data file}
  Connect(tsdata,'TSDATA  ');
  Reset(tsdata);
{Initialize cluster progress output file}
  Connect(tcldata,'TCLDATA ');
  Rewrite(tcldata);
  Writeln(tcldata,'Clusters for attributes:  ',
                  'Range, Range Rate, Azimuth, and Elevation');
  Writeln(tcldata);
{Initialize true observation output file}
  Connect(trudata,'TRUDATA ');
  Rewrite(trudata);
  Writeln(trudata,'Clusters for attributes:  ',
                  'Range, Range Rate, Azimuth, and Elevation (True)');
```

```
{Initialize cluster state output file}
  Connect(csdata,'CSDATA  ');
  Rewrite(csdata);
  end; {Procedure Init_Program}

{*** State Derivative Functions/Procedures **********************************}

Procedure Deriv(tm : real;
                FX : stacked_vector;
            var DX : stacked_vector);
  type
    matrix = array [1..block,1..block] of real;
  var
    i,j,split                                       : integer;
    r,ri,r2i,r3i,mf,xr,yr,zr,x3x,x3y,x3z,y3y,y3z,z3z : real;
    fvr                                             : matrix;
  begin
{General factors}
  r  := Sqrt(Sqr(FX[1])+Sqr(FX[2])+Sqr(FX[3]));
  ri := 1/r;  r2i := Sqr(ri);  mf := -mu*r2i/r;
{State transition factors}
  xr := FX[1]*ri;     yr := FX[2]*ri;     zr := FX[3]*ri;
  x3x := -3*Sqr(xr);  y3y := -3*Sqr(yr);  z3z := -3*Sqr(zr);
  x3y := -3*xr*yr;    x3z := -3*xr*zr;    y3z := -3*yr*zr;
  fvr[1,1] := mf*(1 + x3x);
  fvr[1,2] := mf*x3y;
  fvr[1,3] := mf*x3z;
  fvr[2,1] := mf*x3y;
  fvr[2,2] := mf*(1 + y3y);
  fvr[2,3] := mf*y3z;
  fvr[3,1] := mf*x3z;
  fvr[3,2] := mf*y3z;
  fvr[3,3] := mf*(1 + z3z);
{State derivatives}
  for i := 1 to 3 do
    begin
    DX[i] := FX[i+3];        {Position derivatives}
    DX[i+3] := mf*FX[i];     {Velocity derivatives}
    end; {for i}
{State transition derivatives}
  split := block*nest;
  for i := 1 to split do
    DX[nest+i] := FX[nest+split+i];
  for i := 1 to block do
    for j := 1 to nest do
      DX[split+i*nest+j] := fvr[i,1]*FX[  nest+j]
                          + fvr[i,2]*FX[2*nest+j]
                          + fvr[i,3]*FX[3*nest+j];
  end; {Procedure Deriv}
```

```
{*** Integration Functions/Procedures ***************************************}

Procedure Initialize_RK78;
  begin
  rkf[1]  :=    2/27;    rkf[2]  :=     1/9;     rkf[3]  :=    1/36;
  rkf[4]  :=    1/12;    rkf[5]  :=     1/6;     rkf[6]  :=    1/24;
  rkf[7]  :=    1/8;     rkf[8]  :=     5/12;    rkf[9]  :=  -25/16;
  rkf[10] :=    1/2;     rkf[11] :=     1/20;    rkf[12] :=    1/5;
  rkf[13] :=    1/4;     rkf[14] :=     5/6;     rkf[15] :=  -25/108;
  rkf[16] :=  125/108;   rkf[17] :=   125/54;    rkf[18] :=  -65/27;
  rkf[19] :=    1/6;     rkf[20] :=    13/900;   rkf[21] :=   31/300;
  rkf[22] :=   -2/9;     rkf[23] :=    61/225;   rkf[24] :=    2/3;
  rkf[25] :=   67/90;    rkf[26] :=   -53/6;     rkf[27] := -107/9;
  rkf[28] :=  704/45;    rkf[29] :=     1/3;     rkf[30] :=   -1/12;
  rkf[31] :=   23/108;   rkf[32] :=   -19/60;    rkf[33] :=  -91/108;
  rkf[34] :=   17/6;     rkf[35] :=   311/54;    rkf[36] := -976/135;
  rkf[37] :=   45/164;   rkf[38] :=    18/41;    rkf[39] := 2133/4100;
  rkf[40] :=   45/82;    rkf[41] :=  2383/4100;  rkf[42] := -341/164;
  rkf[43] := -301/82;    rkf[44] :=  4496/1025;  rkf[45] :=    3/205;
  rkf[46] :=   -3/41;    rkf[47] :=    -6/41;    rkf[48] :=   33/164;
  rkf[49] :=   12/41;    rkf[50] := -1777/4100;  rkf[51] := 2193/4100;
  rkf[52] :=   51/82;    rkf[53] :=  -341/164;   rkf[54] := -289/82;
  rkf[55] := 4496/1025;  rkf[56] :=     9/280;   rkf[57] :=   41/840;
  rkf[58] :=    9/35;    rkf[59] :=    34/105;
  end; {Procedure Initialize_RK78}

Procedure RK78(var x : stacked_vector;
            var t,dt : real;
                tout : real;
       relerr,abserr : real;
                neqn : integer;
           var iflag : integer);
  label 1,2,3;
  const
    bup = 2.821109907456E+12;
    blo = 1.6815125390625E-11;
  var
    dtfix,dtfail                                : boolean;
    i,nrej,nrejt,nstp                           : integer;
    dtold,delt,t0,rer,scale,ae,rte,te,xmag,pct   : real;
    f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,x0 : stacked_vector;
  begin
  nrejt := 0;
  nstp := 0;
  if (abserr = 0) and (relerr = 0) then {Set flag if fixed step mode desired}
    dtfix := true
  else
    dtfix := false;
  dtold := dt;
```

```
1:dtfail := false;
  nrej := 0;
{Reset step size if this will put t greater than tout}
  delt := tout - t;
  if Abs(dt) >= Abs(delt) then
    dt := delt
  else
    if Abs(2*dt) >= Abs(delt) then
      dt := delt/2;
  if Abs(dt) >= 2.84E-14*Abs(t) then
    begin
{First Evaluation}
    t0 := t;
    for i := 1 to neqn do
      x0[i] := x[i];
    Deriv(t,x,f0);
{Second Evaluation}
2:  t := t0 + rkf[1]*dt;
    for i := 1 to neqn do
      x[i] := rkf[1]*f0[i]*dt + x0[i];
    Deriv(t,x,f1);
{Third Evaluation}
    t := t0 + rkf[2]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[3]*f0[i] + rkf[4]*f1[i])*dt + x0[i];
    Deriv(t,x,f2);
{Fourth Evaluation}
    t := t0 + rkf[5]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[6]*f0[i] + rkf[7]*f2[i])*dt + x0[i];
    Deriv(t,x,f3);
{Fifth Evaluation}
    t := t0 + rkf[8]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[8]*f0[i] +rkf[9]*(f2[i] - f3[i]))*dt + x0[i];
    Deriv(t,x,f4);
{Sixth Evaluation}
    t := t0 + rkf[10]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[11]*f0[i] + rkf[12]*f4[i] + rkf[13]*f3[i])*dt + x0[i];
    Deriv(t,x,f5);
{Seventh Evaluation}
    t := t0 + rkf[14]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[15]*f0[i] + rkf[16]*f3[i] + rkf[17]*f5[i]
             + rkf[18]*f4[i])*dt + x0[i];
    Deriv(t,x,f6);
{Eighth Evaluation}
    t := t0 + rkf[19]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[20]*f6[i] + rkf[21]*f0[i] + rkf[22]*f5[i]
             + rkf[23]*f4[i])*dt + x0[i];
    Deriv(t,x,f7);
{Ninth Evaluation}
    t := t0 + rkf[24]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[25]*f6[i] + 2*f0[i] + 3*f7[i] + rkf[26]*f3[i]
             + rkf[27]*f5[i] + rkf[28]*f4[i])*dt + x0[i];
    Deriv(t,x,f8);
{Tenth Evaluation}
    t := t0 + rkf[29]*dt;
    for i := 1 to neqn do
      x[i] := (rkf[30]*f8[i] + rkf[31]*f3[i] + rkf[32]*f6[i] + rkf[33]*f0[i]
             + rkf[34]*f7[i] + rkf[35]*f5[i] + rkf[36]*f4[i])*dt + x0[i];
    Deriv(t,x,f9);
{Eleventh Evaluation}
    t := t0 + dt;
    for i := 1 to neqn do
      x[i] := (rkf[37]*f8[i] + rkf[38]*f9[i] + rkf[39]*f6[i] + rkf[40]*f7[i]
             + rkf[41]*f0[i] + rkf[42]*f3[i] + rkf[43]*f5[i]
             + rkf[44]*f4[i])*dt + x0[i];
    Deriv(t,x,f10);
{Twelfth Evaluation}
    t := t0;
    for i := 1 to neqn do
      x[i] := (rkf[45]*(f0[i] - f6[i]) + rkf[46]*(f7[i] - f8[i])
             + rkf[47]*(f5[i] - f9[i]))*dt + x0[i];
    Deriv(t,x,f11);
{Thirteenth Evaluation}
    t := t0 + dt;
    for i := 1 to neqn do
      x[i] := (rkf[48]*f8[i] + rkf[49]*f9[i] + rkf[50]*f0[i] + rkf[51]*f6[i]
             + rkf[52]*f7[i] +        f11[i] + rkf[53]*f3[i] + rkf[54]*f5[i]
             + rkf[55]*f4[i])*dt + x0[i];
    Deriv(t,x,f12);
{Compute state at t+dt}
    for i := 1 to neqn do
      x[i] := (rkf[56]*(f8[i] + f9[i]) + rkf[57]*(f11[i] + f12[i])
             + rkf[58]*(f6[i] + f7[i]) + rkf[59]*f5[i])*dt + x0[i];
    if not dtfix then
      begin {Compute max local truncation error}
      rer := RMax(relerr,2.572E-13);
      scale := 2/rer;
      ae := scale * abserr + 1.0E-14;
      rte := 0;
      for i := 1 to neqn do
        begin
        te := Abs(rkf[57]*(f0[i] + f10[i] - f11[i] - f12[i]));
        xmag := Abs(x[i]) + Abs(x0[i]) + ae;
        rte := RMax(rte,te/xmag);
        end; {for i}
      rte := rte * scale;
      if rte >= 1 then
        begin {Reject this step}
        dtfail := true;
        nrej := nrej + 1;
        nrejt := nrejt + 1;
        if nrej >= 10 then
          begin
          for i := 1 to neqn do
            x[i] := x0[i];
          t := t0;
          iflag := 7;
          Goto 3;
          end {if}
        else
          begin
          pct := 0.025;
          if rte < bup then
            pct := 0.9/Sqrt(Sqrt(Sqrt(rte)));
          dt := pct * dt;
          dtold := dt;
          Goto 2;
          end; {else}
        end; {if}
      end; {if not dtfix}
{This step is acceptable - eighth order evaluation}
    t := t0 + dt;
    nstp := nstp + 1;
    if Abs(tout-t) <= 1.0E-14 then
      begin
      dt := dtold;
      iflag := 2;
      Goto 3;
      end;
    if dtfix then
      Goto 1;
    pct := 20;
    if rte > blo then
      pct := 0.9/Sqrt(Sqrt(Sqrt(rte)));
    if dtfail then
      pct := RMin(pct,1.0);
    dt := dt * pct;
    dtold := dt;
    Goto 1;
    end; {if}
```

```
{Check for too small a step size}
    if Abs(delt) <= Abs(dt) then
      begin {Extrapolate}
      Deriv(t,x,f0);
      for i := 1 to neqn do
        x[i] := f0[i]*dt + x[i];
      t := t + dt;
      dt := dtold;
      iflag := 2;
      end
    else {Attempted to use too small a step size}
      iflag := 8;
3:end; {Procedure RK78}

{*** Integrator Interface Procedures ****************************************}

Procedure Stack_Vector(x_vector : state_vector;
                     Phi_matrix : P_matrix;
                    var x_stack : stacked_vector);
  var
    i,j : integer;
  begin
  for i := 1 to nest do
    x_stack[i] := x_vector[i];
  for i := 1 to nest do
    for j := 1 to nest do
      x_stack[nest*i+j] := Phi_matrix[i,j];
  end; {Procedure Stack_Vector}

Procedure Unstack_Vector(var x_vector : state_vector;
                       var Phi_matrix : P_matrix;
                              x_stack : stacked_vector);
  var
    i,j : integer;
  begin
  for i := 1 to nest do
    x_vector[i] := x_stack[i];
  for i := 1 to nest do
    for j := 1 to nest do
      Phi_matrix[i,j] := x_stack[nest*i+j];
  end; {Procedure Unstack_Vector}

Procedure Get_Sensor;
  var
    time,result : integer;
    step        : real;
    t           : array [1..2] of real;
    SSV         : stacked_vector;
    Phi         : P_matrix;
  begin
  for time := frame1 to -1 do
    Sxs[time] := Sxs[time+1];
  repeat
    Read(ssdata,current_sensor);
  until (current_sensor.time = frame_time[0])
  and (current_sensor.number = sensor);
  Sxs[0] := current_sensor.values;
  Phi := Identity;
  Stack_Vector(Sxs[0],Phi,SSV);
  t[1] := 0.0;  t[2] := tbias;  step := tbias;
  RK78(SSV,t[1],step,t[2],small,zero,nstack,result);
  Unstack_Vector(Sxs[0],Phi,SSV);
  end; {Procedure Get_Sensor}

Procedure Get_Observations;
  var
    time,ntime,obs,atr : integer;
    noise              : real;
  begin
{Shift previous data}
  for time := frame1 to 0 do
    begin
    ntime := time + 1;
    atr_limits[time] := atr_limits[ntime];
    assigned[time] := assigned[ntime];
    end; {for time}
  for time := frame1 to -1 do
    begin
    ntime := time + 1;
    nr_obs[time] := nr_obs[ntime];
    nr_unassigned[time] := nr_unassigned[ntime];
    nr_clusters[time] := nr_clusters[ntime];
    nr_active[time] := nr_active[ntime];
    nr_inactive[time] := nr_inactive[ntime];
    attr[time] := attr[ntime];
    status[time] := status[ntime];
    targets[time] := targets[ntime];
    observation[time] := observation[ntime];
    frame_time[time] := frame_time[ntime];
    end; {for time}
{Read new observations}
  obs := 1;
  attr[0,obs] := next;
  targets[0,obs] := next_target;
  frame_time[0] := attr[0,obs,0];
  repeat
    EOI := EOF(tmdata);
    Echo_True_Data(targets[0,obs],sen_nr,attr[0,obs,0],missing_flag);
    if not Missing(targets[0,obs],sen_nr,attr[0,obs,0],missing_flag) then
      begin
      for atr := 1 to attributes do
        begin
        atr_limits[0,1,atr] := RMin(atr_limits[0,1,atr],attr[0,obs,atr]);
        atr_limits[0,2,atr] := RMax(atr_limits[0,2,atr],attr[0,obs,atr]);
        end; {for atr}
      Increment(obs);
      end; {if not Missing}
    if not EOI then
      begin
      Read(tmdata,current_obs);
      targets[0,obs] := current_obs.target;
      sen_nr := current_obs.sensor;
      attr[0,obs,0] := current_obs.time;
      for atr := 1 to attributes do
        attr[0,obs,atr] := current_obs.obs[atr]
                     + (bias[atr] + current_obs.error[atr])*R[atr];
      missing_flag := current_obs.missing;
      end; {if not EOI}
  until (attr[0,obs,0] > frame_time[0]) or EOI;
  last_target := 0;
  EOI := EOI or (attr[0,obs,0] > max_time);
  Writeln(trudata);
  if not EOI then
    begin
    Write(trudata,attr[0,obs,0]:7:1);
    next := attr[0,obs];
    next_target := targets[0,obs];
    end; {if not EOI}
  nr_obs[0] := obs - 1;
  nr_unassigned[0] := nr_obs[0];
  nr_inactive[0] := nr_inactive[-1];
  nr_active[0] := nr_active[-1];
  nr_clusters[0] := nr_active[0] + nr_inactive[0];
  end; {Procedure Get_Observations}

Procedure Initialize_Clustering;
  var
    missed                     : boolean;
    time,ntime,cluster,obs,atr : integer;
    noise,ltime                : real;
  begin
{Initialize attribute minimums, maximums}
  for atr := 1 to attributes do
    begin
    atr_limits[1,1,atr] :=  big;
    atr_limits[1,2,atr] := -big;
    end; {for atr}
  for obs := 1 to clusters do
    assigned[1,obs] := false;
  for time := 0 downto frame1 do
    begin
    ntime := time + 1;
    atr_limits[time] := atr_limits[ntime];
    assigned[time] := assigned[ntime];
    end; {for time}
{Initialize cluster status}
  for time := frame1 to 0 do
    begin
    nr_clusters[time] := 0;
    nr_unassigned[time] := 0;
    nr_obs[time] := 0;
    nr_active[time] := 0;
    nr_inactive[time] := 0;
    targets[time,0] := 0;                       {* For missing observations *}
    frame_time[time] := -1.0;                       {* For output buffering *}
    for cluster := 1 to clusters do
      begin
      observation[time,cluster] := 0;
      status[time,cluster] := 0;
      end; {for cluster}
    end; {for time}
{Find first measurement}
  ltime := -1.0;
  last_target := 0;
  repeat
    Read(tmdata,current_obs);
    next_target := current_obs.target;
    sen_nr := current_obs.sensor;
    next[0] := current_obs.time;
    for atr := 1 to attributes do
      next[atr] := current_obs.obs[atr]
                 + (bias[atr] + current_obs.error[atr])*R[atr];
    missing_flag := current_obs.missing;
    if next[0] > ltime then                             {* Output true data *}
      begin
      ltime := next[0];
      Writeln(trudata);
      Write(trudata,ltime:7:1);
      end; {if}
    missed := Missing(next_target,sen_nr,next[0],missing_flag);
    if missed then
      Echo_True_Data(next_target,sen_nr,ltime,missing_flag);
    until not missed;
  for time := 1 to 2 do
    begin {Get minimum data frames minus 1 to start cluster}
    Get_Observations;  {Read remainder of observation frame}
    Get_Sensor;        {Read sensor state}
    end; {for time}
  repeat  {Read beginning of true target state data frame}
    Read(tsdata,current_target);
  until (current_target.time = frame_time[0]);
  end; {Procedure Initialize_Clustering}

{*** Major Procedures ****************************************************}

Procedure Input_Data;
  var
    time,obs,atr : integer;
  begin
  Start_Timer(1);                                                  {Timing}
  Get_Observations;
  Get_Sensor;
{Calculate span minimums and maximums and scale factors}
  span_limits := atr_limits[frame1];
  for atr := 1 to attributes do
    for time := frame1+1 to 0 do
      begin
      span_limits[1,atr] := RMin(span_limits[1,atr],atr_limits[time,1,atr]);
      span_limits[2,atr] := RMax(span_limits[2,atr],atr_limits[time,2,atr]);
      end; {for time}
  Stop_Timer(1);                                                   {Timing}
  end; {Procedure Input_Data}

{*** Estimation Initialization Procedures ***********************************}

Procedure Initialize_Estimation;
  var
    i,j : integer;
  begin
{Identity matrix}
  for i := 1 to nest do
    begin
    for j := 1 to nest do
      Identity[i,j] := 0.0;
    Identity[i,i] := 1.0;
    end; {for i}
{State covariance}
  Q_k := Identity;
  for i := 1 to nest do
    Q_k[i,i] := 1.0;
  step := 10.0;
  end; {Procedure Initialize_Estimation}
```

```
{*** Estimation Propagation Procedures ***************************************}

Function ArcTan2(num,den : real) : real;
  var
    answer : real;
  begin
  answer := ArcTan(num/den);
  if den < 0 then
    answer := answer + pi;
  if answer > pi then
    answer := answer - 2.0*pi;
  ArcTan2 := answer;
  end; {Function ArcTan2}

Procedure Map_State(Xr : state_vector;
                var Ov : obs_vector);
  begin
  Ov[1] := Sqrt(Sqr(Xr[1])+Sqr(Xr[2])+Sqr(Xr[3]));
  Ov[2] := (Xr[1]*Xr[4] + Xr[2]*Xr[5] + Xr[3]*Xr[6])/Ov[1];
  Ov[3] := ArcTan2(Xr[2],Xr[1]);
  Ov[4] := ArcTan(Xr[3]/Sqrt(Sqr(Xr[1])+Sqr(Xr[2])));
  end; {Procedure Map_State}

Procedure Calculate_H(var H : H_matrix;
                          Xr : state_vector;
                      var Ov : obs_vector);
  var
    i                                             : integer;
    rhoi,rhoi2,varrho,varrhoi,varrhoi2,rhoi2_vari : real;
  begin
{Calculate coefficients}
  Map_State(Xr,Ov);
  rhoi := 1/Ov[1];
  rhoi2 := Sqr(rhoi);
  varrho := Sqrt(Sqr(Xr[1]) + Sqr(Xr[2]));
  varrhoi := 1/varrho;
  varrhoi2 := Sqr(varrhoi);
  rhoi2_vari := rhoi2*varrhoi;
{Form H matrix}
  for i := 1 to block do
    begin
    H[1,i] := Xr[i]*rhoi;
    H[1,i+block] := 0.0;
    H[2,i] := (Xr[i+block]*Ov[1] - Xr[i]*Ov[2])*rhoi2;
    H[2,i+block] := H[1,i];
    H[3,i+block] := 0.0;
    H[4,i+block] := 0.0;
    end; {for i}
  H[3,1] := -Xr[2]*varrhoi2;
  H[3,2] :=  Xr[1]*varrhoi2;
  H[3,3] :=  0.0;
  H[4,1] := -Xr[1]*Xr[3]*rhoi2_vari;
  H[4,2] := -Xr[2]*Xr[3]*rhoi2_vari;
  H[4,3] := varrho*rhoi2;
  end; {Procedure Calculate_H}

Procedure Map_State_to_Attribute_Space(Xv : state_vector;
                                       Pm : P_matrix;
                                var Ov,OP : obs_vector);
  var
    i,j,k : integer;
    Xr,SP : state_vector;
    H,H_P : H_matrix;
    M     : R_matrix;
  begin
  for j := 1 to nest do
    Xr[j] := Xv[j] - Sxs[0,j];
  Calculate_H(H,Xr,Ov);
  for i := 1 to attributes do
    for j := 1 to nest do
      begin
      H_P[i,j] := 0.0;
      for k := 1 to nest do
        H_P[i,j] := H_P[i,j] + H[i,k]*Pm[k,j];
      end; {for j}
  M := R_k;
  for i := 1 to attributes do
    for j := 1 to attributes do
      for k := 1 to nest do
        M[i,j] := M[i,j] + H_P[i,k]*H[j,k];
  for i := 1 to attributes do
    OP[i] := 3.0*Sqrt(M[i,i]);
  end; {Procedure Map_State_to_Attribute}

Function ArcSin(arg : real) : real;
  begin
  ArcSin := ArcTan(arg/Sqrt(1.0-Sqr(arg)));
  end; {Function ArcSin}

Function ArcCos(arg : real) : real;
  begin
  ArcCos := pi/2.0 - ArcTan(arg/Sqrt(1.0-Sqr(arg)));
  end; {Function ArcCos}

Procedure Forecast;
  var
    cluster,i,j,k,result : integer;
    t                    : array [1..2] of real;
    SSV                   : stacked_vector;
    Phi,Phi_P             : P_matrix;
  begin
  Start_Timer(2);                                                  {Timing}
  for cluster := 1 to nr_clusters[-1] do
    if status[0,cluster] < 0 then
      begin {Integrate state and state transition matrix}
      Phi := Identity;
      Stack_Vector(Sx[cluster],Phi,SSV);
      t[1] := 0.0;
      t[2] := frame_time[0] - frame_time[status[0,cluster]];
      status[0,cluster] := -1;
      RK78(SSV,t[1],step,t[2],small,zero,nstack,result);
      Unstack_Vector(Sx[cluster],Phi,SSV);
{Propagate state covariance matrix}
      for i := 1 to nest do
        for j := 1 to nest do
          begin
          Phi_P[i,j] := 0.0;
          for k := 1 to nest do
            Phi_P[i,j] := Phi_P[i,j] + Phi[i,k]*P[cluster,k,j];
          end; {for j}
        P[cluster] := Q_k;
        for i := 1 to nest do
          for j := 1 to nest do
            for k := 1 to nest do
              P[cluster,i,j] := P[cluster,i,j] + Phi_P[i,k]*Phi[j,k];
      Map_State_to_Attribute_Space(Sx[cluster],P[cluster],
                                   est[cluster],gate[cluster]);
      end; {if}
  Stop_Timer(2);                                                   {Timing}
  end; {Procedure Forecast}

Procedure Calculate_Metrics;
  label 1;
  var
    cluster,obs,atr,mrow,missed_gates : integer;
    delta                             : real;
    factor                            : obs_vector;
    nr_feasible                       : array [1..2] of ivector;
  begin
  Start_Timer(3);                                                  {Timing}
  mrow := 0;
{Calculate standardization factors}
  for atr := 1 to attributes do
    begin
    factor[atr] := span_limits[2,atr] - span_limits[1,atr];
    if Abs(factor[atr]) < zero then
      factor[atr] := 1.0
    else
      factor[atr] := 1.0/factor[atr];
    end; {for atr}
{Initialize pre-assignment indices}
  for cluster := 1 to clusters do
    nr_feasible[1,cluster] := 0;
  nr_feasible[2] := nr_feasible[1];
  col_basis := nr_feasible[1];
  row_basis := nr_feasible[1];
{Calculate metrics}
  for cluster := 1 to nr_clusters[-1] do
    if status[0,cluster] = -1 then
      begin
      Increment(mrow);
      convert[mrow] := cluster;
      for obs := 1 to nr_obs[0] do
        begin
        metric[mrow,obs] := 0.0;
        missed_gates := 0;
        for atr := 1 to attributes do
          begin
          delta := Abs(est[cluster,atr] - attr[0,obs,atr]);
          if delta > gate[cluster,atr] then
            if (delta > 2.0*gate[cluster,atr])
            or (missed_gates >= max_missed) then
              begin
              metric[mrow,obs] := bad;
              goto 1;
              end {if missed}
            else
              Increment(missed_gates);
          metric[mrow,obs] := metric[mrow,obs] + Sqr(factor[atr]*delta);
          end; {for atr}
1:      if metric[mrow,obs] < bad then
          begin
          Increment(nr_feasible[1,mrow]);
          Increment(nr_feasible[2,obs]);
          col_basis[mrow] := obs;
          row_basis[obs] := mrow;
          end; {if}
        end; {for obs}
      end; {if status}
{Check pre-assignment}
  for cluster := 1 to mrow do
    if (nr_feasible[1,cluster] > 1) or
       (nr_feasible[2,col_basis[cluster]] > 1) then
      col_basis[cluster] := 0;
  for obs := 1 to nr_obs[0] do
    if (nr_feasible[2,obs] > 1) or
```

```
       (nr_feasible[2,row_basis[obs]] > 1) then
      row_basis[obs] := 0;
{Assign dummy values, as necessary}
  for cluster := nr_clusters[-1]+1 to nr_obs[0]+nr_inactive[-1] do
    begin
    Increment(mrow);
    convert[mrow] := cluster;
    status[0,cluster] := 0;                              {Dummy cluster}
    for obs := 1 to nr_obs[0] do
      metric[mrow,obs] := bad;
    end; {for cluster}
  for obs := nr_obs[0]+1 to nr_active[-1] do
    begin
    targets[0,obs] := -1;                            {Dummy-observation}
    for cluster := 1 to mrow do
      metric[cluster,obs] := bad;
    end; {for obs}
  Stop_Timer(3);                                              {Timing}
  end; {Procedure Calculate_Metrics}

{*** State Update Procedures ***********************************************}

Procedure Invert(M : R_matrix;
          var MInv : R_matrix);
  label 1;
  var
    i,j,k,l,irow,icol,l1           : integer;
    determ,pivot,hold,sum,t,ab,big : real;
    index                          : array[1..nest,1..3] of integer;
  Procedure Swap(var a,b : real);
    var
      hold : real;
    begin
      hold := a;
      a := b;
      b := hold;
    end; {Procedure Swap}
{Gauss-Jordan inversion}
  begin
  for i := 1 to attributes do
    index[i,3] := 0;
  determ := 1;
  for i := 1 to attributes do
    begin {Search for largest element}
    big := 0;
    for j := 1 to attributes do
      begin
      if index[j,3] <> 1 then
        begin
        for k := 1 to attributes do
          begin
          if index[k,3] > 1 then
            begin
            writeln('ERROR:  Matrix singular');
            goto 1;
            end;
          if index[k,3] < 1 then
            if Abs(M[j,k]) > big then
              begin
              irow := j;
              icol := k;
              big := Abs(M[j,k]);
              end; {if}
          end; {for k}
        end; {if}
      end; {for j}
    Increment(index[icol,3]);
    index[i,1] := irow;
    index[i,2] := icol;
{Interchange rows to put pivot on diagonal}
    if irow <> icol then
      begin
      determ := -determ;
      for l := 1 to attributes do
        Swap(M[irow,l],M[icol,l]);
      end; {if irow <> icol}
{Divide pivot row by pivot column}
    pivot := M[icol,icol];
    determ := determ * pivot;
    M[icol,icol] := 1;
    for l := 1 to attributes do
      M[icol,l] := M[icol,l] / pivot;
{Reduce nonpivot rows}
    for l1 := 1 to attributes do
      begin
      if l1 <> icol then
        begin
        t := M[l1,icol];
        M[l1,icol] := 0;
        for l := 1 to attributes do
          M[l1,l] := M[l1,l] - M[icol,l] * t;
        end; {if l1 <> icol}
      end; {for l1}
    end; {for i}
{Interchange columns}
  for i := 1 to attributes do
    begin
    l := attributes - i + 1;
    if index[l,1] <> index[l,2] then
      begin
      irow := index[l,1];
      icol := index[l,2];
      for k := 1 to attributes do
        swap(M[k,irow],M[k,icol]);
      end; {if index}
    end; {for i}
  for k := 1 to attributes do
    if index[k,3] <> 1 then
      begin
      writeln('ERROR:  Matrix singular');
      goto 1;
      end;
  for i := 1 to attributes do
    for j := 1 to attributes do
      MInv[i,j] := M[i,j];
1:end; {Procedure Invert}

Procedure Cholesky(M : P_matrix;
               var S : P_matrix);
  var
    i,j,k : integer;
    sum   : real;
  begin
  for i := 1 to nest do
    begin
    for j := 1 to i-1 do
      S[j,i] := 0.0;
    sum := 0.0;
    for k := 1 to i-1 do
      sum := sum + Sqr(S[i,k]);
    S[i,i] := Sqrt(M[i,i] - sum);
    for j := i+1 to nest do
      begin
      sum := 0.0;
      for k := 1 to i-1 do
        sum := sum + S[i,k]*S[j,k];
      S[j,i] := (M[i,j] - sum)/S[i,i];
      end; {for i}
    end; {for i}
  end; {Procedure Cholesky}

Procedure Update_Estimates;
  var
    cluster,i,j,k,obs            : integer;
    Oref,od                      : obs_vector;
    dS,dSx                       : state_vector;
    H                            : H_matrix;
    K_k,Fbar,F_M                 : K_matrix;
    M,MI                         : R_matrix;
    Wbar,What,I_FMF,Lambda,Phat : P_matrix;
  begin
  Start_Timer(6);                                                      {Timing}
  for cluster := 1 to nr_clusters[0] do
    if (status[0,cluster] = -1) and
       (nr_missing[cluster] = 0) then
      begin
{Calculate relative state vector}
      for i := 1 to nest do
        dS[i] := Sx[cluster,i] - Sxs[0,i];
{Calculate H}
      Calculate_H(H,dS,Oref);
{Calculate W bar}
      Cholesky(P[cluster],Wbar);
{Calculate F bar}
      for i := 1 to nest do
        for j := 1 to attributes do
          begin
          Fbar[i,j] := 0.0;
          for k := i to nest do {Lower triangular multiplication}
            Fbar[i,j] := Fbar[i,j] + Wbar[k,i]*H[j,k];
          end; {for j}
{Calculate M matrix}
      MI := R_k;
      for i := 1 to attributes do
        for j := 1 to attributes do
          for k := 1 to nest do
            MI[i,j] := MI[i,j] + Fbar[k,i]*Fbar[k,j];
      Invert(MI,M);
{Calculate Lambda matrix}
      for i := 1 to nest do
        for j := 1 to attributes do
          begin
          F_M[i,j] := 0.0;
          for k := 1 to attributes do
            F_M[i,j] := F_M[i,j] + Fbar[i,k]*M[k,j];
          end; {for j}
      I_FMF := Identity;
      for i := 1 to nest do
        for j := 1 to nest do
          for k := 1 to attributes do
            I_FMF[i,j] := I_FMF[i,j] - F_M[i,k]*Fbar[j,k];
      Cholesky(I_FMF,Lambda);
{Calculate W hat}
      for i := 1 to nest do
        begin
        for j := 1 to i do
```

```
          begin
          What[i,j] := 0.0;
          for k := 1 to i do
            What[i,j] := What[i,j] + Wbar[i,k]*Lambda[k,j];
          end; {for j}
        for j := i+1 to nest do
          What[i,j] := 0.0;
        end; {for i}
{Calculate P hat}
      for i := 1 to nest do
        for j := 1 to i do
          begin
          Phat[i,j] := 0.0;
          for k := 1 to i do
            Phat[i,j] := Phat[i,j] + What[i,k]*What[j,k];
          Phat[j,i] := Phat[i,j];
          end; {for j}
{Calculate K matrix}
      for i := 1 to nest do
        for j := 1 to attributes do
          begin
          K_k[i,j] := 0.0;
          for k := 1 to i do
            K_k[i,j] := K_k[i,j] + Wbar[i,k]*F_M[k,j];
          end; {for j}
      obs := observation[0,cluster];
      for i := 1 to attributes do
        od[i] := attr[0,obs,i] - Oref[i];
      for i := 1 to nest do
        begin
        dSx[i] := 0.0;
        for j := 1 to attributes do
          dSx[i] := dSx[i] + K_k[i,j]*od[j];
        Sx[cluster,i] := Sx[cluster,i] + dSx[i];     {Rectify reference state}
        end; {for i}
      P[cluster] := Phat;
      end; {if}
  Stop_Timer(6);                                                      {Timing}
  end; {Procedure Update_Estimates}

Procedure Assignment(n : integer;                           {Hungarian Method}
                  cost : M_matrix;
               var ans : ivector);
  label 1,2;
  type
    rvector = array [1..clusters] of real;
    bvector = array [1..clusters] of boolean;
    imatrix = array [1..clusters,1..clusters] of integer;
```

```
var
  i,j,k,m                : integer;
  delta                  : real;
  alpha,beta,slack       : rvector;
  labels,exposed,nhbor,Q : ivector;
  labeled                : bvector;
  mate                   : array [1..2] of ivector;
  A                      : imatrix;
Procedure Augment(v : integer);
  begin
  if labels[v] = 0 then
    begin
    mate[1,v] := exposed[v];
    mate[2,exposed[v]] := v;
    end {if}
  else
    begin
    exposed[labels[v]] := mate[1,v];
    mate[1,v] := exposed[v];
    mate[2,exposed[v]] := v;
    augment(labels[v]);
    end; {else}
  end; {Procedure Augment}
Function Modify : boolean;
  label 1;
  var
    i,j           : integer;
    theta1,theta2 : real;
  begin
  theta1 := big;
  for j := 1 to n do
    if slack[j] > 0 then
      theta1 := RMin(theta1,slack[j]);
  theta2 := theta1/2;
  for i := 1 to n do
    if labeled[i] then
      alpha[i] := alpha[i] + theta2
    else
      alpha[i] := alpha[i] - theta2;
  for j := 1 to n do
    if slack[j] = 0 then
      beta[j] := beta[j] - theta2
    else
      beta[j] := beta[j] + theta2;
  for j := 1 to n do
    if slack[j] > 0 then
      begin
      slack[j] := slack[j] - theta1;
      if slack[j] = 0 then
```

```
          if mate[2,j] = 0 then
            begin
            exposed[nhbor[j]] := j;
            augment(nhbor[j]);
            Modify := true;
            goto 1;
            end {if}
          else
            begin
            labels[mate[2,j]] := nhbor[j];
            labeled[mate[2,j]] := true;
            Q[mate[2,j]] := 1;
            A[nhbor[j],mate[2,j]] := 1;
            end; {else}
        end; {if}
    Modify := false;
1:  end; {Function Modify}
  Function Q_not_empty(var index : integer) : boolean;
    begin
    index := 0;
    repeat
     Increment(index);
    until (Q[index] = 1) or (index = n);
    if (index = n) and (Q[index] = 0) then
      Q_not_empty := false
    else
      Q_not_empty := true;
    end; {Function Q_not_empty}
  begin
  for i := 1 to n do
    begin
    mate[1,i] := 0;
    alpha[i] := 0;
    labels[i] := 0;
    end; {for i}
  for j := 1 to n do
    begin
    mate[2,j] := 0;
    beta[j] := big;
    for i := 1 to n do
      beta[j] := RMin(beta[j],cost[i,j]);
    end; {for j}
{Repeat for n stages}
  for m := 1 to n do
    begin
    for i := 1 to n do
      for j := 1 to n do
        A[i,j] := 0;
    for i := 1 to n do
```

```
      exposed[i] := 0;
    for j := 1 to n do
      slack[j] := big;
    for i := 1 to n do
      for j := 1 to n do
        if cost[i,j] - alpha[i] - beta[j] < zero then
          if mate[2,j] = 0 then
            begin
            exposed[i] := j;
            end {if}
          else
            begin
            A[i,mate[2,j]] := 1;
            end; {else}
{Construct auxiliary graph}
    for i := 1 to n do
      begin
      Q[i] := 0;
      labeled[i] := false;
      end; {for i}
    for i := 1 to n do
      if mate[1,i] = 0 then
        begin
        if exposed[i] <> 0 then
          begin
          augment(i);
          goto 2;
          end; {if}
        Q[i] := 1;
        labels[i] := 0;
        labeled[i] := true;
        for k := 1 to n do
          begin
          delta := cost[i,k] - alpha[i] - beta[k];
          if delta < zero then
            delta := 0.0;
          if (0 <= delta) and (delta < slack[k]) then
            begin
            slack[k] := delta;
            nhbor[k] := i;
            end; {if}
          end; {for k}
        end;{if}
1:  while Q_not_empty(i) do
      begin
      Q[i] := 0;
      if exposed[i] <> 0 then
        begin
        augment(i);
        goto 2;
        end;
        for k := 1 to n do
          begin
          delta := cost[i,k] - alpha[i] - beta[k];
          if delta < zero then
            delta := 0.0;
          if (0 <= delta) and (delta < slack[k]) then
            begin
            slack[k] := delta;
            nhbor[k] := i;
            end; {if}
          end; {for k}
      for j := 1 to n do
        if (A[i,j] = 1) and not labeled[j] then
          begin
          labels[j] := i;
          labeled[j] := true;
          Q[j] := 1;
          if exposed[j] <> 0 then
            begin
            augment(j);
            goto 2;
            end; {if}
          for k := 1 to n do
            begin
            delta := cost[j,k] - alpha[j] - beta[k];
            if delta < zero then
              delta := 0.0;
            if (0 <= delta) and (delta < slack[k]) then
              begin
              slack[k] := delta;
              nhbor[k] := j;
              end; {if}
            end; {for k}
          end; {if}
      end; {while}
    if not Modify then
      goto 1;
2:  end; {for m}
  ans := mate[2];
  end; {Procedure Assignment}

Procedure Perform_Cluster_Assignment;
  var
    cluster,obs,k,atr,size,
    asize,col_index,row_index : integer;
    brow,row_convert          : ivector;
    done                      : boolean;
  begin
  Start_Timer(4);                                                   {Timing}
  size := IMax(nr_active[-1],nr_obs[0]);
{Perform pre-assignment packing}
  row_index := 0;
  asize := 0;
  for cluster := 1 to size do
    if col_basis[cluster] = 0 then
      begin
      Increment(asize);
      Increment(row_index);
      row_convert[row_index] := cluster;
      col_index := 0;
      for obs := 1 to size do
        if row_basis[obs] = 0 then
          begin
          Increment(col_index);
          ametric[row_index,col_index] := metric[cluster,obs];
          end; {if}
      end; {if}
  if asize > 0 then
    begin
    Assignment(asize,ametric,brow);
    col_index := 0;
    for obs := 1 to size do  {Unpack solution}
      if row_basis[obs] = 0 then
        begin
        Increment(col_index);
        row_basis[obs] := row_convert[brow[col_index]];
        end; {if}
    end; {if}
  brow := row_basis;
{Check for invalid assignments and assign targets to clusters}
  for obs := 1 to size do
    begin
    cluster := convert[brow[obs]];
    if metric[brow[obs],obs] < bad then
      begin {Good assignment}
      observation[0,cluster] := obs;
      assigned[0,obs] := true;
      Decrement(nr_unassigned[0]);
      nr_missing[cluster] := 0;
      end {if}
    else
      if status[0,cluster] = -1 then
        begin {Missing assignment}
        {Cluster has been propagated maximum number of times}
        if nr_missing[cluster] >= max_prop then
          begin {Terminate old cluster}
          status[0,cluster] := 0;
          Increment(nr_inactive[0]);
          Decrement(nr_active[0]);
          end {if}
        else
          begin {Propagate old cluster}
          Increment(nr_missing[cluster]);
          observation[0,cluster] := 0;
          end; {else}
      end; {else}
    end; {for obs}
  Stop_Timer(4);                                                        {Timing}
  end; {Procedure Perform_Cluster_Assignment}

Procedure Calculate_SP_Estimates(time1,obs1 : integer;
                               var est,gate : O_matrix);
  type
    vector = array [1..block] of real;
  var
    i,j,time2                            : integer;
    A,acc,acc_n,acc_p,acc_t2,Arrs,
    Arrs_dot_urho,Arrsv,b1,b2,
    beta_1,beta_2,beta_min,beta_max,
    Cos_az,Cos_el,Cos_beta,Cos_beta_1,
    Cos_beta_2,d,d1,d2,d3,dmax,dmin,
    mu_factor1,mu_factor2,r1_dot_rs2,
    rng_min,rng_max,Sin_az,Sin_el,Sqr_t21,
    t21,theta,vmag2,vsmag2,vs_dot_urho,
    rmag2,rmag3 : real;
    rsmag2       : array [1..2] of real;
    urho,rvec                           : vector;
  begin
{Calculate unit range vector}
  Cos_az := Cos(attr[time1,obs1,3]);
  Cos_el := Cos(attr[time1,obs1,4]);
  Sin_az := Sin(attr[time1,obs1,3]);
  Sin_el := Sin(attr[time1,obs1,4]);
  urho[1] := Cos_el*Cos_az;
  urho[2] := Cos_el*Sin_az;
  urho[3] := Sin_el;
{Compute magnitudes; dot products}
  rmag2  := 0.0;  rsmag2[1]   := 0.0;
  vsmag2 := 0.0;  vs_dot_urho := 0.0;
  for i := 1 to block do
    begin
    j := i + block;
    rvec[i] := attr[time1,obs1,1]*urho[i] + Sxs[time1,i];
    rmag2 := rmag2 + Sqr(rvec[i]);
    rsmag2[1] := rsmag2[1] + Sqr(Sxs[time1,i]);
    vsmag2 := vsmag2 + Sqr(Sxs[time1,j]);
    vs_dot_urho := vs_dot_urho + Sxs[time1,j]*urho[i];
    end; {for i}
  vmag2 := 2.0*(Max_Energy + mu/Sqrt(rmag2));
  mu_factor1 := -mu/(Sqrt(rmag2)*rmag2);
  mu_factor2 := -mu/(Sqrt(rsmag2[1])*rsmag2[1]);
  acc_p := 0.0;  acc_t2 := 0.0;
  for i := 1 to block do
    begin
    acc    := mu_factor1*rvec[i] - mu_factor2*Sxs[time1,i];
    acc_t2 := acc_t2 + Sqr(acc);
    acc_p  := acc_p + acc*urho[i];
    end; {for i}
  acc_n := Sqrt(acc_t2 - Sqr(acc_p));
  Cos_beta_1 := (attr[time1,obs1,2] + vs_dot_urho)/Sqrt(vmag2);
  beta_1 := ArcCos(Cos_beta_1);
  d1 := Sqrt(vmag2 - Sqr(attr[time1,obs1,2] + vs_dot_urho));
  d2 := Sqrt(vsmag2 - Sqr(vs_dot_urho));
  for time2 := time1+1 to 0 do
    begin
{Calculate time interval}
    t21 := frame_time[time2] - frame_time[time1];
    Sqr_t21 := 0.5*Sqr(t21);
{Compute magnitudes; dot products}
    rsmag2[2]  := 0.0;
    r1_dot_rs2 := 0.0;
    for i := 1 to block do
      begin
      rsmag2[2] := rsmag2[2] + Sqr(Sxs[time2,i]);
      r1_dot_rs2  := r1_dot_rs2 + rvec[i]*Sxs[time2,i];
      end; {for i}
    A := 1.0 + mu_factor1*Sqr_t21;
{Calculate accelerations; more dot products}
    Arrs  := 0.0;  Arrs_dot_urho := 0.0;
    for i := 1 to block do
      begin
      Arrsv := A*rvec[i] - Sxs[time2,i];
      Arrs  := Arrs + Sqr(Arrsv);
      Arrs_dot_urho := Arrs_dot_urho + Arrsv*urho[i];
      end; {for i}
    Arrs := Sqrt(Arrs);
{Compute angle beta}
    Cos_beta_2 := Arrs_dot_urho/Arrs;
    Cos_beta   := RMin(Cos_beta_1,Cos_beta_2);
    beta_2 := ArcCos(Cos_beta_2);
    beta_max := beta_1 + beta_2;
    beta_min := Abs(beta_1 - beta_2);
{Calculate range estimate and gate}
    b1 := Sqr(A)*rmag2 + Sqr(t21)*vmag2 - 2.0*A*r1_dot_rs2 + rsmag2[2];
    b2 := 2.0*t21*Arrs*Sqrt(vmag2);
    est[time2,1] := Sqrt(b1 + b2*Cos_beta_1);
    gate[time2,1] := Abs(Sqrt(b1 + b2*Cos(beta_max)) - est[time2,1]);
    gate[time2,1] := RMax(Abs(Sqrt(b1 + b2*Cos(beta_min)) - est[time2,1]),
                          gate[time2,1]) + 3.0*R[1];
{Calculate range rate estimate and gate}
    d3 := Sqr_t21*acc_n;
    dmax := t21*(d1+d2) + d3;
    dmin := t21*Abs(d1-d2) - d3;
    d := 0.5*(dmax + dmin);
    est[time2,2] := (Sqr(est[time2,1]) - attr[time1,obs1,1]
                    *Sqrt(Sqr(est[time2,1]) - Sqr(d)))/(est[time2,1]*t21);
    rng_min := est[time2,1] - gate[time2,1];
    rng_max := est[time2,1] + gate[time2,1];
    gate[time2,2] := Abs((Sqr(rng_max) - attr[time1,obs1,1]*Sqrt(Sqr(rng_max)
                         - Sqr(dmax)))/(rng_max*t21) - est[time2,2]);
    gate[time2,2] := RMax(Abs((Sqr(rng_max) - attr[time1,obs1,1]
                         *Sqrt(Sqr(rng_max) - Sqr(dmin)))/(rng_max*t21)
                         - est[time2,2]),gate[time2,2]);
    gate[time2,2] := RMax(Abs((Sqr(rng_min) - attr[time1,obs1,1]
                         *Sqrt(Sqr(rng_min) - Sqr(dmax)))/(rng_min*t21)
                         - est[time2,2]),gate[time2,2]);
    gate[time2,2] := RMax(Abs((Sqr(rng_min) - attr[time1,obs1,1]
                         *Sqrt(Sqr(rng_min) - Sqr(dmin)))/(rng_min*t21)
                         - est[time2,2]),gate[time2,2]) + 3.0*R[2];
    gate[time2,2] := 1.3*gate[time2,2];
{Calculate azimuth and elevation gates}
    est[time2,3] := attr[time1,obs1,3];
    est[time2,4] := attr[time1,obs1,4];
    theta := ArcSin(dmax/est[time2,1]);
    gate[time2,3] := theta/Cos(attr[time1,obs1,4]) + 3.0*R[3];
    gate[time2,4] := theta + 3.0*R[4];
    end; {for time2}
  end; {Procedure Calculate_SP_Estimates}

Procedure Calculate_DP_Estimate(time1,obs1,
                                time2,obs2 : integer;
                    var energy,delta_energy : real);
  type
    vector = array [1..block] of real;
  var
    cluster,i,j,k,time,atr                  : integer;
    t,obs                                   : array [1..2] of integer;
    Cos_az,Cos_el,Sin_az,Sin_el,dti,r2,v2,rf : real;
    state,Emat,E_P                          : state_vector;
    rmag2,rmag3,r_dot_urho                  : array [1..2] of real;
    mu_factor                               : array [1..2,1..2] of real;
    dt,tf                                   : array [1..3] of real;
    urho,rvec                               : array [1..2] of vector;
    r_dot_part                             : array [1..2,3..4] of real;
    partial_urho                           : array [1..2,3..4] of vector;
    Jmat,J_R                               : J_matrix;
    Pmat                                   : P_matrix;
  begin
  t[1] := time1;
  t[2] := time2;
  obs[1] := obs1;
  obs[2] := obs2;
  dt[3] := frame_time[time2] - frame_time[time1];
  dti   := 1.0/dt[3];
  dt[2] := 0.5*dt[3];
  dt[1] := -dt[2];
  tf[1] := dt[1]*dti;
  tf[2] := dt[2]*dti;
  for time := 1 to 2 do
    begin
    Cos_az := Cos(attr[t[time],obs[time],3]);
    Cos_el := Cos(attr[t[time],obs[time],4]);
    Sin_az := Sin(attr[t[time],obs[time],3]);
    Sin_el := Sin(attr[t[time],obs[time],4]);
    urho[time,1] := Cos_el*Cos_az;
    urho[time,2] := Cos_el*Sin_az;
    urho[time,3] := Sin_el;
    partial_urho[time,3,1] := -Cos_el*Sin_az;
    partial_urho[time,3,2] :=  Cos_el*Cos_az;
    partial_urho[time,3,3] :=  0.0;
    partial_urho[time,4,1] := -Sin_el*Cos_az;
    partial_urho[time,4,2] := -Sin_el*Sin_az;
    partial_urho[time,4,3] :=  Cos_el;
    rmag2[time] := 0.0;
    r_dot_urho[time] := 0.0;
    r_dot_part[time,3] := 0.0;
    r_dot_part[time,4] := 0.0;
    for i := 1 to block do
      begin
      rvec[time,i] := attr[t[time],obs[time],1]*urho[time,i] + Sxs[t[time],i];
      rmag2[time] := rmag2[time] + Sqr(rvec[time,i]);
      r_dot_urho[time] := r_dot_urho[time] + rvec[time,i]*urho[time,i];
      for atr := 3 to 4 do
        r_dot_part[time,atr] := r_dot_part[time,atr]
                              + rvec[time,i]*partial_urho[time,atr,i];
      end; {for i}
    rmag3[time] := Sqrt(rmag2[time])*rmag2[time];
    mu_factor[time,1] := mu*dt[1]*dt[2]/(2.0*rmag3[time]);
    mu_factor[time,2] := 3.0*mu_factor[time,1]/rmag2[time];
    mu_factor[time,1] := 1.0 - mu_factor[time,1];
    end; {for time}
  r2 := 0.0;  v2 := 0.0;
```

```
for i := 1 to block do
  begin
  j := i + block;
  state[i] := tf[2]*mu_factor[1,1]*rvec[1,i]
            - tf[1]*mu_factor[2,1]*rvec[2,i];
  r2 := r2 + Sqr(state[i]);
  state[j] :=  dti*(mu_factor[2,1]*rvec[2,i]
                  - mu_factor[1,1]*rvec[1,i]);
  v2 := v2 + Sqr(state[j]);
  Jmat[i,1] := tf[2]*(mu_factor[1,1]*urho[1,i]
                    + mu_factor[1,2]*r_dot_urho[1]*rvec[1,i]);
  Jmat[i,2] := 0.0;
  Jmat[i,5] := -tf[1]*(mu_factor[2,1]*urho[2,i]
                     + mu_factor[2,2]*r_dot_urho[2]*rvec[2,i]);
  Jmat[i,6] := 0.0;
  Jmat[j,1] := -dti*(mu_factor[1,1]*urho[1,i]
                   + mu_factor[1,2]*r_dot_urho[1]*rvec[1,i]);
  Jmat[j,2] := 0.0;
  Jmat[j,5] := dti*(mu_factor[2,1]*urho[2,i]
                  + mu_factor[2,2]*r_dot_urho[2]*rvec[2,i]);
  Jmat[j,6] := 0.0;
  for atr := 3 to 4 do
    begin
    Jmat[i,atr] := attr[time1,obs1,1]*tf[2]
                 *(mu_factor[1,1]*partial_urho[1,atr,i]
                 + mu_factor[1,2]*r_dot_part[1,atr]*rvec[1,i]);
    Jmat[i,atr+4] := -attr[time2,obs2,1]*tf[1]
                   *(mu_factor[2,1]*partial_urho[2,atr,i]
                   + mu_factor[2,2]*r_dot_part[2,atr]*rvec[2,i]);
    Jmat[j,atr] := -attr[time1,obs1,1]*dti
                 *(mu_factor[1,1]*partial_urho[1,atr,i]
                 + mu_factor[1,2]*r_dot_part[1,atr]*rvec[1,i]);
    Jmat[j,atr+4] := attr[time2,obs2,1]*dti
                   *(mu_factor[2,1]*partial_urho[2,atr,i]
                   + mu_factor[2,2]*r_dot_part[2,atr]*rvec[2,i]);
    end; {for atr}
  end; {for i}
  energy := v2/2.0 - mu/Sqrt(r2);
  rf := mu/(r2*Sqrt(r2));
  for i := 1 to block do
    begin
    j := i + block;
    Emat[i] := state[i]*rf;
    Emat[j] := state[j];
    end; {for i}
  for i := 1 to nest do
    for j := 1 to nterms do
      begin
      J_R[i,j] := 0.0;
```

```
        for k := 1 to nterms do
          J_R[i,j] := J_R[i,j] + Jmat[i,k]*Rvar[2,k,j];
        end; {for j}
    for i := 1 to nest do
      for j := 1 to nest do
        begin
        Pmat[i,j] := 0.0;
        for k := 1 to nterms do
          Pmat[i,j] := Pmat[i,j] + J_R[i,k]*Jmat[j,k];
        end; {for j}
    for j := 1 to nest do
      begin
      E_P[j] := 0.0;
      for k := 1 to nest do
        E_P[j] := E_P[j] + Emat[k]*Pmat[k,j];
      end; {for j}
    delta_energy := 0.0;
    for j := 1 to nest do
      delta_energy := delta_energy + E_P[j]*Emat[j];
  end; {Procedure Calculate_DP_Estimate}

Procedure Calculate_TP_Estimate(time1,obs1,
                                time2,obs2,
                                time3,obs3 : integer;
                                 var state : state_vector;
                                  var Pmat : P_matrix;
                   var energy,delta_energy : real);
  type
    vector = array [1..block] of real;
  var
    cluster,i,j,k,time,atr               : integer;
    t,obs                                : array [1..3] of integer;
    Cos_az,Cos_el,Sin_az,Sin_el,r2,v2,rf : real;
    Emat,E_P                             : state_vector;
    rmag2,rmag3,r_dot_urho               : array [1..3] of real;
    r_dot_part                           : array [1..3,3..4] of real;
    mu_factor                            : array [1..3,1..2] of real;
    drho                                 : vector;
    dt,tf                                : array [1..3] of real;
    urho,rvec                            : array [1..3] of vector;
    partial_urho                         : array [1..3,3..4] of vector;
    Jmat,J_R                             : J_matrix;
  begin
  t[1] := time1;  t[2] := time2;  t[3] := time3;
  obs[1] := obs1;  obs[2] := obs2;  obs[3] := obs3;
  dt[1] := frame_time[time2] - frame_time[time1];
  dt[2] := frame_time[time3] - frame_time[time2];
  dt[3] := dt[1] + dt[2];
  tf[1] := -dt[2]/(dt[1]*dt[3]);
```

```
tf[2] := (dt[2]-dt[1])/(dt[1]*dt[2]);
tf[3] :=  dt[1]/(dt[2]*dt[3]);
for time := 1 to 3 do
  begin
  Cos_az := Cos(attr[t[time],obs[time],3]);
  Cos_el := Cos(attr[t[time],obs[time],4]);
  Sin_az := Sin(attr[t[time],obs[time],3]);
  Sin_el := Sin(attr[t[time],obs[time],4]);
  urho[time,1] := Cos_el*Cos_az;
  urho[time,2] := Cos_el*Sin_az;
  urho[time,3] := Sin_el;
  partial_urho[time,3,1] := -Cos_el*Sin_az;
  partial_urho[time,3,2] :=  Cos_el*Cos_az;
  partial_urho[time,3,3] :=  0.0;
  partial_urho[time,4,1] := -Sin_el*Cos_az;
  partial_urho[time,4,2] := -Sin_el*Sin_az;
  partial_urho[time,4,3] :=  Cos_el;
  end; {for time}
r2 := 0.0;
v2 := 0.0;
for i := 1 to block do
  begin
  j := i + block;
  state[i] := attr[time2,obs2,1]*urho[2,i] + Sxs[time2,i];
  r2 := r2 + Sqr(state[i]);
  drho[i] := tf[1]*urho[1,i] + tf[2]*urho[2,i] + tf[3]*urho[3,i];
  state[j] := attr[time2,obs2,2]*urho[2,i] + attr[time2,obs2,1]*drho[i]
            + Sxs[time2,j];
  v2 := v2 + Sqr(state[j]);
  Jmat[i,1] := urho[2,i];
  Jmat[i,2] := 0.0;
  Jmat[i,3] := 0.0;
  Jmat[i,4] := attr[time2,obs2,1]*partial_urho[2,3,i];
  Jmat[i,5] := 0.0;
  Jmat[i,6] := 0.0;
  Jmat[i,7] := attr[time2,obs2,1]*partial_urho[2,4,i];
  Jmat[i,8] := 0.0;
  Jmat[j,1] := drho[i];
  Jmat[j,2] := urho[2,i];
  for atr := 3 to 4 do
    begin
    k := 3*(atr - 2);
    Jmat[j,k]   :=  attr[time2,obs2,1]*tf[1]*partial_urho[1,atr,i];
    Jmat[j,k+1] := (attr[time2,obs2,2] + attr[time2,obs2,1]*tf[2])
                   *partial_urho[2,atr,i];
    Jmat[j,k+2] :=  attr[time2,obs2,1]*tf[3]*partial_urho[3,atr,i];
    end; {for atr}
  end; {for i}
  energy := v2/2.0 - mu/Sqrt(r2);
```

```
    rf := mu/(r2*Sqrt(r2));
    for i := 1 to block do
      begin
      j := i + block;
      Emat[i] := state[i]*rf;
      Emat[j] := state[j];
      end; {for i}
{Form state covariance matrix}
    for i := 1 to nest do
      for j := 1 to nterms do
        begin
        J_R[i,j] := 0.0;
        for k := 1 to nterms do
          J_R[i,j] := J_R[i,j] + Jmat[i,k]*Rvar[3,k,j];
       end; {for j}
    for i := 1 to nest do
      for j := 1 to nest do
        begin
        Pmat[i,j] := 0.0;
        for k := 1 to nterms do
          Pmat[i,j] := Pmat[i,j] + J_R[i,k]*Jmat[j,k];
        end; {for j}
    for j := 1 to nest do
      begin
      E_P[j] := 0.0;
      for k := 1 to nest do
        E_P[j] := E_P[j] + Emat[k]*Pmat[k,j];
      end; {for j}
    delta_energy := 0.0;
    for j := 1 to nest do
      delta_energy := delta_energy + E_P[j]*Emat[j];
  end; {Procedure Calculate_TP_Estimate}

Procedure Perform_Cluster_Initiation;
  label 1;
  type
    pairs = record
      time1,obs1,time2,obs2 : integer;
      end; {record}
    triples = record
      time1,obs1,time2,obs2,time3,obs3 : integer;
      metric                           : real;
      end; {record}
  var
    move                           : boolean;
    time,obs,atr,ntime,start,stop,
    count,ocount1,ocount2,cluster,
    missed_gates,point1,point2,arc  : integer;
    delta,specific_energy           : real;
    active                          : bvector;
    estimates,gates                 : O_matrix;
    index,back_index,incident       : array [span] of ivector;
    pointer                         : array [frame1..-1] of ivector;
    pair                            : array [frame1..-1,1..clusters] of pairs;
    triple                          : array [1..clusters] of triples;
  Procedure Solve_QP(arcs : integer;
                 var best : bvector);
    var
      i        : integer;
      mincost  : real;
      solution : bvector;
    Function Feasible : boolean;
      label 1;
      var
        result   : boolean;
        time,obs : integer;
      begin
      result := true;
      for time := frame1 to 0 do
        for obs := 1 to nr_unassigned[time] do
          if incident[time,obs] > 1 then
            begin
            result := false;
            goto 1;
            end; {if}
1:    Feasible := result;
      end; {Function Feasible}
    Procedure Search(j : integer;
                     c : real);
      var
        k : integer;
        d : real;
      begin
      for k := j+1 to arcs do
        begin
        solution[k] := false;
        with triple[k] do
          begin
          Decrement(incident[time1,back_index[time1,obs1]]);
          Decrement(incident[time2,back_index[time2,obs2]]);
          Decrement(incident[time3,back_index[time3,obs3]]);
          d := c - metric;
          end; {with}
        if not Feasible then
          Search(k,d)
        else
          if d < mincost then
            begin
            best := solution;
            mincost := d;
            end; {if}
        solution[k] := true;
        with triple[k] do
          begin
          Increment(incident[time1,back_index[time1,obs1]]);
          Increment(incident[time2,back_index[time2,obs2]]);
          Increment(incident[time3,back_index[time3,obs3]]);
          end; {with}
        end; {for}
      end; {Procedure Search}
    begin
    solution := best;
    mincost := big;
    if not Feasible then
      Search(0,0);
    end; {Procedure Solve_QP}
  begin
  Start_Timer(5);                                                     {Timing}
  for time := frame1 to 0 do  {Index unassigned observations}
    begin
    obs := 0;
    for count := 1 to nr_unassigned[time] do
      begin
      repeat
        Increment(obs);
        back_index[time,obs] := 0;
      until not assigned[time,obs];
      index[time,count] := obs;
      back_index[time,obs] := count;
      end; {for count}
    end; {for time}
  for start := frame1 to -1 do  {Determine possible pairs}
    begin
    obs := 0;
    for ocount1 := 1 to nr_unassigned[start] do
      begin
      pointer[start,ocount1] := obs + 1;
      Calculate_SP_Estimates(start,index[start,ocount1],
                             estimates,gates);
      for stop := start+1 to 0 do
        begin
        for ocount2 := 1 to nr_unassigned[stop] do
          begin
          missed_gates := 0;
          for atr := 1 to attributes do
            begin
```

```
            delta := Abs(estimates[stop,atr]
                       - attr[stop,index[stop,ocount2],atr]);
            if delta > gates[stop,atr] then
              if (delta > 2.0*gates[stop,atr])
              or (missed_gates >= max_missed) then
                goto 1
              else
                Increment(missed_gates);
            end; {for atr}
          Calculate_DP_Estimate(start,index[start,ocount1],
                                stop,index[stop,ocount2],
                                specific_energy,delta);
          if specific_energy <= max_energy + 3.0*Sqrt(delta) then
            begin
            Increment(obs);
            with pair[start,obs] do
              begin
              time1 := start;
              obs1  := index[start,ocount1];
              time2 := stop;
              obs2  := index[stop,ocount2];
              end; {with}
            end; {if}
1:        end; {for ocount2}
        end; {for stop}
      end; {for ocount1}
    pointer[start,nr_unassigned[start]+1] := obs + 1;
    pointer[start,0] := obs;
    end; {for start}
{Form all possible triples and calculate metric; eliminate infeasible triples}
  for time := frame1 to 0 do
    for obs := 1 to nr_unassigned[time] do
      incident[time,obs] := 0;
  count := 0;
  cluster := nr_clusters[0];
  for start := frame1 to -2 do
    begin
    for ocount1 := 1 to pointer[start,0] do
      begin
      stop := pair[start,ocount1].time2;
      if stop <> 0 then
        begin
        obs := back_index[stop,pair[start,ocount1].obs2];
        point1 := pointer[stop,obs];
        point2 := pointer[stop,obs+1];
        for ocount2 := point1 to point2-1 do
          begin
          Increment(cluster);
          Calculate_TP_Estimate(start,pair[start,ocount1].obs1,
                                stop,pair[stop,ocount2].obs1,
                                pair[stop,ocount2].time2,
                                pair[stop,ocount2].obs2,
                                Sx[cluster],P[cluster],
                                specific_energy,delta);
          if specific_energy <= max_energy + 3.0*Sqrt(delta) then
            begin
            Increment(count);
            nr_missing[cluster] := 0;
            with triple[count] do
              begin
              time1  := start;
              obs1   := pair[start,ocount1].obs1;
              time2  := stop;
              obs2   := pair[stop,ocount2].obs1;
              time3  := pair[stop,ocount2].time2;
              obs3   := pair[stop,ocount2].obs2;
              metric := specific_energy;
              Increment(incident[time1,back_index[time1,obs1]]);
              Increment(incident[time2,back_index[time2,obs2]]);
              Increment(incident[time3,back_index[time3,obs3]]);
              active[count] := true;
              end; {with}
            end; {if}
          end; {for ocount2}
        end; {if}
      end; {for ocount1}
    end; {for start}
{Solve remaining quadratic program using implicit enumeration}
  Solve_QP(count,active);
  cluster := nr_clusters[0];
  move := false;
  for arc := 1 to count do
    if active[arc] then
      begin
      Increment(cluster);
      nr_missing[cluster] := 0;
      if move then
        begin
        Sx[cluster] := Sx[nr_clusters[-1]+arc];
        P[cluster] := P[nr_clusters[-1]+arc];
        end; {if move}
      with triple[arc] do
        begin
        assigned[time1,obs1] := true;
        Decrement(nr_unassigned[time1]);
        observation[time1,cluster] := obs1;
        assigned[time2,obs2] := true;
        Decrement(nr_unassigned[time2]);
        observation[time2,cluster] := obs2;
        assigned[time3,obs3] := true;
        Decrement(nr_unassigned[time3]);
        observation[time3,cluster] := obs3;
        for time := time1 to 0 do
          begin
          status[time,cluster] := time2 - 1;
          Increment(nr_clusters[time]);
          Increment(nr_active[time]);
          if (time <> time1) and (time <> time2) and (time <> time3) then
            observation[time,cluster] := 0;
          end; {for time}
        end; {with}
      end {if}
    else
      move := true;
  Stop_Timer(5);                                                    {Timing}
  end; {Procedure Perform_Cluster_Initiation}

{*** Outputs ***************************************************************}

Procedure Echo_Cluster_Assignment(time : integer);
  var
    cluster,count : integer;
  begin
  if frame_time[time] >= 0.0 then
    begin
    write(tcldata,frame_time[time]:7:1);
    cluster := 0;
    for count := 1 to nr_active[time] do
      repeat
      Increment(cluster);
      if status[time,cluster] < 0 then
        Write(tcldata,targets[time,observation[time,cluster]]:4)
      else
        Write(tcldata,'    ');
      until status[time,cluster] < 0;
    Writeln(tcldata);
    end; {if}
  end; {Procedure Echo_Cluster_Assignment}

Procedure Output_Cluster_Residuals;
  var
    count,cluster,target,i,j : integer;
    position,velocity        : real;
    Sxt                      : array [1..clusters] of state_vector;
  begin
{Input target states for current observation frame}
  repeat
    Sxt[current_target.number] := current_target.values;
    Read(tsdata,current_target);
  until (current_target.time > frame_time[0]);
{Compare estimated to true target states}
  cluster := 0;
  for count := 1 to nr_active[0] do
    repeat
    Increment(cluster);
    if status[time,cluster] < 0 then
      begin
      target := targets[0,observation[0,cluster]];
      position := 0.0;
      velocity := 0.0;
      if target > 0 then
        begin
        for i := 1 to block do
          begin
          j := i + block;
          position := position + Sqr(Sx[cluster,i] - Sxt[target,i]);
          velocity := velocity + Sqr(Sx[cluster,j] - Sxt[target,j]);
          end; {for i}
        position := Sqrt(position);
        velocity := Sqrt(velocity);
        end; {if target > 0}
      Write(csdata,cluster:4,target:4,frame_time[0]:7:1);
      Writeln(csdata,position:11:1,velocity:7:1);
      end; {if status < 0}
    until status[0,cluster] < 0;
  end; {Procedure Output_Cluster_Residuals}

{*** Main Program ***********************************************************}

BEGIN

{*****************************}
{** Perform Initializations **}
{*****************************}
  Init_Times;                                                           {Timing}
  Start_Timer(0);                                                       {Timing}
  Init_Program;
  Initialize_RK78;
  Initialize_Estimation;
  Initialize_Clustering;
```

```
{**********************************}
{** Perform sequential clustering **}
{**********************************}
  repeat
    Input_Data;
    Forecast;
    Calculate_Metrics;
    Perform_Cluster_Assignment;
    Update_Estimates;
    Perform_Cluster_Initiation;
    Echo_Cluster_Assignment(frame1);
    Output_Cluster_Residuals;
  until EOI;
{******************************}
{** End sequential clustering **}
{******************************}
  for time := frame1+1 to 0 do
    Echo_Cluster_Assignment(time);
  Stop_Timer(0);                                                        {Timing}
  Report_Times(6);                                                      {Timing}

END.
```

BIBLIOGRAPHY

[1] Thomas G. Allen, Thomas Kurien, and Robert B. Washburn, "Parallel Computational Structures for Multiobject Tracking Algorithms on Associative Processors," *Proceedings of the 1986 American Control Conference*, Vol. 3, 18–20 June 1986, pp. 1869–1875.

[2] Michael R. Anderberg, *Cluster Analysis for Applications*, New York: Academic Press, 1973.

[3] M.M. Astrahan, *Speech Analysis by Clustering, or the Hyperphoneme Method*, Stanford Artificial Intelligence Project Memorandum AIM–124 (AD 709067), Stanford University, 1970.

[4] S.N. Balakrishnan, M.M. Crawford, T.S. Kelso, S. Lee, J.B. Lundberg, and B.D. Tapley, "Application of Clustering Techniques for the Detection, Classification, and Estimation of Multiple Targets in Space," presented at the *AIAA 26th Aerospace Sciences Meeting*, January 11–14, 1988.

[5] G.H. Ball and D.J. Hall, *PROMENADE—An On-Line Pattern Recognition System*, RADC–TR–67–310 (AD 822174), Stanford Research Institute, Menlo Park, CA, 1967.

[6] Y. Bar-Shalom and A.G. Jaffer, "Adaptive Nonlinear Filtering for Tracking with Measurements of Uncertain Origin," *Proceedings of the 1972 IEEE Conference on Decision and Control and 11th Symposium on Adaptive Processes*, 13–15 December 1972, pp. 243–247.

[7] Yaakov Bar-Shalom and Edison Tse, "Tracking in a Cluttered Environment with Probabilistic Data Association," *Proceedings of the Fourth Symposium on Nonlinear Estimation Theory and Its Applications*, 10–12 September 1973, pp. 13–22.

[8] Y. Bar-Shalom, "Extension of the Probabilistic Data Association Filter to Multitarget Tracking," *Proceedings of the Fifth Symposium on Nonlinear Estimation Theory and Its Applications*, 23–25 September 1974, pp. 16–21.

[9] Yaakov Bar-Shalom and Edison Tse, “Tracking in a Cluttered Environment With Probabilistic Data Association,” *Automatica*, Vol. 11, September 1975, pp. 451–460.

[10] Yaakov Bar-Shalom, “Tracking Methods in a Multitarget Environment,” *IEEE Transactions on Automatic Control*, Vol. AC–23, No. 4, August 1978, pp. 618–626.

[11] Y. Bar-Shalom and K. Birmiwal, “Consistency and Robustness of PDAF for Target Tracking in Cluttered Environments, *Automatica*, Vol. 19, No. 4, July 1983, pp. 431–437.

[12] Roger R. Bate, Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics*, New York: Dover Publications, Inc., 1971.

[13] S.S. Blackman, *Multiple-Target Tracking With Radar Applications*, Dedham, MA: Artech House, Inc., 1986.

[14] Christopher L. Bowman, “Maximum Likelihood Track Correlation for Multisensor Integration,” *Proceedings of the 18th IEEE Conference on Decision & Control Including the Symposium on Adaptive Processes*, Vol. 1, 12–14 December 1979, pp. 374–376.

[15] C.B. Chang, “Ballistic Trajectory Estimation with Angle-Only Measurements,” *IEEE Transactions on Automatic Control*, Vol. AC–25, No. 3, June 1980, pp. 474–480.

[16] C.B. Chang and L.C. Youens, *An Algorithm for Multiple Target Tracking and Data Correlation*, TR–643, MIT Lincoln Laboratory, 13 June 1983.

[17] Chaw-Bing Chang and John A. Tabaczynski, “Application of State Estimation to Target Tracking,” *IEEE Transactions on Automatic Control*, Vol. AC–29, No. 2, February 1984, pp. 98–109.

[18] Kuo-Chu Chang and Y. Bar-Shalom, “Joint Probabilistic Data Association for Multitarget Tracking with Possibly Unresolved Measurements,” *Proceedings of the 1983 American Control Conference*, Vol. 2, 22–24 June 1983, pp. 466–471.

[19] Kuo-Chu Chang and Yaakov Bar-Shalom, “Joint Probabilistic Data Association for Multitarget Tracking with Possibly Unresolved Measurements and Maneuvers,” *IEEE Transactions on Automatic Control*, Vol. AC–29, No. 7, July 1984, pp. 585–594.

[20] Kuo-Chu Chang, Chee-Yee Chong, and Y. Bar-Shalom, "Joint Probabilistic Data Association in Distributed Sensor Networks," *Proceedings of the 1985 American Control Conference*, Vol. 2, 19–21 June 1985, pp. 817–822.

[21] C.Y. Chong, S. Mori, E. Tse, and R.P. Wishner, "Distributed Estimation in Distributed Sensor Networks," *Proceedings of the 1982 American Control Conference*, Vol. 2, 14–16 June 1982, pp. 820–826.

[22] C.Y. Chong, E. Tse, and S. Mori, "Distributed Estimation in Networks," *Proceedings of the 1983 American Control Conference*, Vol. 1, 22–24 June 1983, pp. 294–300.

[23] C.Y. Chong, S. Mori, and E. Tse, "Distributed Estimation Systems," *Proceedings of the 6th MIT/ONR Workshop on C^3 Systems*, 25–29 July 1983, pp. 158–163.

[24] Pedro Ramon Escobal, *Methods of Orbit Determination*, New York: John Wiley & Sons, Inc., 1965.

[25] Robert J. Fitzgerald, "Divergence of the Kalman Filter," *IEEE Transactions on Automatic Control*, Vol. AC–16, No. 6, December 1971, pp. 736–747.

[26] E.W. Forgy, *Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications*, Biometric Society Meetings, Riverside, CA (Abstract in *Biometrics*, Vol. 21, No. 3, p. 78), 1965.

[27] Thomas E. Fortmann, Yaakov Bar-Shalom, and Molly Scheffé, "Multitarget Tracking Using Joint Probabilistic Data Association," *Proceedings of the 19th IEEE Conference on Decision and Control*, Vol. 2, 10–12 December 1980, pp. 807–812.

[28] Robert S. Garfinkel and George L. Nemhauser, *Integer Programming*, New York: John Wiley & Sons, 1972.

[29] Paul Herget, *The Computation of Orbits*, Paul Herget, 1948.

[30] Samuel Herrick, *Astrodynamics: Orbit Determination, Space Navigation, Celestial Mechanics*, Vol. 1, London: Van Nostrand Reinhold Company, 1971.

[31] A.G. Jaffer and Y. Bar-Shalom, "On Optimal Tracking in Multiple Target Environments," *Proceedings of the Third Symposium on Nonlinear Estimation Theory and Its Applications*, 11–13 September 1972, pp. 112–117.

[32] R.C. Jancey, "Multidimensional Group Analysis," *Australian Journal of Botany*, Vol. 14, No. 1, 1966, pp. 127–130.

[33] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME: Journal of Basic Engineering*, Vol. 82, Series D, No. 1, March 1960, pp. 35–45.

[34] R.E. Kalman and R.S. Bucy, "New Results in Filtering and Prediction Theory," *Transactions of the ASME: Journal of Basic Engineering*, Vol. 83, Series D, No. 1, March 1961, pp. 95–108.

[35] Paul G. Kaminski, Arthur E. Bryson, Jr., and Stanley F. Schmidt, "Discrete Square Root Filtering: A Survey of Current Techniques," *IEEE Transactions on Automatic Control*, Vol. AC–16, No. 6, December 1971, pp. 727–736.

[36] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the 5th Symposium of Mathematical Statistics and Probability*, AD 669871, Vol. 1, 1967, pp. 281–297.

[37] D.J. McRae, "MIKCA: A FORTRAN IV Iterative K-Means Cluster Analysis Program," *Behavioral Science*, Vol. 16, No. 4, 1971, pp. 423–424.

[38] C.L. Morefield, "Application of 0–1 Integer Programming to a Track Assembly Problem," *Proceedings of the 1975 IEEE Conference on Decision and Control Including the 14th Symposium on Adaptive Processes*, 10–12 December 1975, pp. 428–433.

[39] Charles L. Morefield, "Application of 0–1 Integer Programming to Multitarget Tracking Problems," *IEEE Transactions on Automatic Control*, Vol. AC–22, No. 3, June 1977, pp. 302–312.

[40] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

[41] H. Charles Romesburg, *Cluster Analysis for Researchers*, Belmont, CA: Lifetime Learning Publications, 1984.

[42] Ronald G. Sea, "An Efficient Suboptimal Decision Procedure for Associating Sensor Data With Stored Tracks in Real-Time Surveillance Systems," *Proceedings of the 1971 IEEE Conference on Decision and Control (Including the 10th Symposium on Adaptive Processes)*, 15–17 December 1971, pp. 33–37.

[43] R.A. Singer and A.J. Kanyuck, "Computer Control of Multiple Site Track Correlation," *Automatica*, Vol. 7, July 1971, pp. 455–463.

[44] R.A Singer and J.J. Stein, "An Optimal Tracking Filter for Processing Sensor Data of Imprecisely Determined Origin in Surveillance Systems," *Proceedings of the 1971 IEEE Conference on Decision and Control (Including the 10th Symposium on Adaptive Processes)*, 15–17 December 1971, pp. 171–175.

[45] Robert A. Singer and Ronald G. Sea, "New Results in Optimizing Surveillance System Tracking and Data Correlation Performance in Dense Multitarget Environments," *IEEE Transactions on Automatic Control*, Vol. AC–18, No. 6, December 1973, pp. 571–582.

[46] Robert A. Singer, Ronald G. Sea, and Kim B. Housewright, "Derivation and Evaluation of Improved Tracking Filters for use in Dense Multitarget Environments," *IEEE Transactions on Information Theory*, Vol. IT–20, No. 4, July 1974, pp. 423–432.

[47] R.W. Sittler, "An Optimal Data Association Problem in Surveillance Theory," *IEEE Transactions on Military Electronics*, Vol. MIL–8, April 1964, pp. 125–139.

[48] P. Smith and G. Buechler, "A Branching Algorithm for Discriminating and Tracking Multiple Objects," *IEEE Transactions on Automatic Control*, Vol. AC–20, No. 1, February 1975, pp. 101–104.

[49] P.H.A. Sneath and R.R. Sokal, *Numerical Taxonomy*, San Francisco: W.H. Freeman, 1973.

[50] R.R. Sokal and P.H.A. Sneath, *Principles of Numerical Taxonomy*, San Francisco: W.H. Freeman, 1963.

[51] Helmut Späth, *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*, New York: John Wiley & Sons, 1985.

[52] J.J. Stein and S.S. Blackman, "Generalized Correlation of Multi-Target Track Data," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES–11, No. 6, November 1975, pp. 1207–1217.

[53] B.D. Tapley, B.E. Schutz, C.S. Ho, and T. Wilson, *Data Classification for Multitarget Tracking Applications*, CSR–83–5, Center for Space Research, The University of Texas at Austin, Austin, TX, May 1983.

[54] B.D. Tapley, B.E. Schutz, P.A.M. Abusali, S.N. Balakrishnan, and C.S. Ho, "A New Method for Enhancement of Data Separability and Data Classification in Multisensor-Multitarget Tracking Problems," *Proceedings of the 1985 American Control Conference*, Vol. 2, 19–21 June 1985, pp. 1056–1058.

[55] Byron D. Tapley, Bob E. Schutz, and George H. Born, *Satellite Orbit Determination: Fundamentals and Applications*, August 1986.

[56] E. Tse, R.E. Larson, and Y. Bar-Shalom, "Applications of Optimum Discrete Filter to Target Tracking with Angle Only Measurements," *Proceedings of the Fourth Symposium on Nonlinear Estimation Theory and Its Applications*, 10–12 September 1973, pp. 328–333.

[57] Michel Verhaegen and Paul Van Dooren, "Numerical Aspects of Different Kalman Filter Implementations," *IEEE Transactions on Automatic Control*, Vol. AC–31, No. 10, October 1986, pp. 907–917.

VITA

Thomas Sean Kelso was born on 18 October 1954 in Denver, Colorado, the son of Mary Margaret Kelso and Thomas John Kelso. After graduating from high school in 1972 from Miami Carol City, Florida, he was appointed to the United States Air Force Academy, where he earned the degree of Bachelor of Science in Physics and Mathematics and a regular commission in the United States Air Force in June 1976. He then performed duty as a Minuteman ICBM Missile Combat Crew Member at Whiteman AFB, Missouri, where he also was awarded the degree of Master of Business Administration in Quantitative Methods from the University of Missouri-Columbia in August 1979. He was selected as a Minuteman ICBM Operations Training Instructor for the 4315 CCTS at Vandenberg AFB, California in 1979, where he performed instructor duties until 1981. From there he went to Wright-Patterson AFB, Ohio, where he graduated as a Distinguished Graduate with a Master of Science in Space Operations from the School of Engineering, Air Force Institute of Technology, in December 1982. He was then selected as Chief, Training Plans Branch for the Consolidated Space Operations Center Activation Division at Sunnyvale AFS, California in 1983, where his daughter, Shannon, was born in April. In 1984, he became the Chief of the Global Positioning System (GPS) Spacecraft Operations Team, where he was responsible for two GPS launches, planning for GPS deployments from the Space Shuttle, and daily operations for a constellation of eight satellites. In 1985, he was selected to enter The Graduate School at The University of Texas at Austin.

Permanent address: 7130 Tyler Street
Hollywood, FL 33024

This dissertation was typeset[1] with LaTeX by the author.

[1]LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for The University of Texas at Austin dissertation format was written by Khe-Sing The and modified by the author.