DISSERTATION

NETWORK MULTIPLE FRAME ASSIGNMENT ARCHITECTURES

Submitted by

Suihua Lu

Department of Mathematics

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2001

COLORADO STATE UNIVERSITY

July 31, 2001

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY SUIHUA LU ENTITLED "NETWORK MULTIPLE FRAME ASSIGNMENT ARCHITECTURES" BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Adviser

Department Head

ABSTRACT OF DISSERTATION

NETWORK MULTIPLE FRAME ASSIGNMENT ARCHITECTURES

Multiple target tracking methods divide into two broad classes, namely single frame and multiple frame methods. The most successful of the multiple frame methods are multiple hypothesis tracking (MHT) and multiple frame assignments (MFA). In dense tracking environments the performance improvements of multiple frame methods over single frame methods is very significant, making it the preferred solution for many tracking problems. Thus, in addition to the availability single frame processing, multiple frame data association methods are an essential class of methods for almost all tracking needs.

The application of multiple frame tracking methods must consider an architecture in which the sensors are distributed across multiple platforms. Such geometric and sensor diversity has the potential to significantly enhance tracking and discrimination accuracy. A centralized architecture in which all measurements are sent to one location and processed with tracks being transmitted back to the different platforms is a simple one that is probably optimal in that it is capable of producing the best track quality (e.g., purity and accuracy) and a consistent air picture. The centralized tracker is, however, unacceptable for several reasons, notably the communication overloads and single-point-failure. Thus, one must turn to a distributed architecture for both estimation/fusion and data association.

One of the simplest network-centric architectures is that of placing a centralized tracker on each platform. The architecture is called Network MFA Centralized,

which removes the problem of single-point-failure. However, due to communication delays in the network, the order the measurements arrive at different platforms varies. Each composite tracker is making its own tracking decisions based on the data it receives, regardless of decisions of other platforms. Therefore, a consistent air picture may not be achieved across the network.

Thus, the objective of this thesis is the development of two near-optimal Network-Centric MFA architectures, namely Network MFA on Local Data and Network Tracks and Network MFA on All data and Network Tracks, that preserve the quality of a centralized tracker across a network of platforms while managing communication loading and achieving a consistent air picture.

One technique that has proved useful for achieving SIAP is to require that each platform be in charge of assigning its own measurements to the network tracks. In the architecture of Network MFA on Local Data and Network Tracks, only local data are used in the sliding windows and track initiations are based on local data only. In the architecture of Network MFA on All Data and Network Tracks, all data (remote and local) are used in the sliding window.

Communication loading is only addressed by the architectures in that track states and their error covariances are not required to be transmitted back to the various platforms. The results of extensive computations are presented to validate the differences in four tracking architectures.

Suihua Lu
Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523
Fall 2001

ACKNOWLEDGEMENTS

Part I of this thesis was supported in its entirety by Numerica, Inc. through Internal Research and Development funds and from contracts from the Boeing Corporation and the Office of Naval Research Grant Number N00014-00-C-0264 to Numerica, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

Part I

Network Multiple Frame Assignment Architectures

Chapter 1

INTRODUCTION

Multiple target tracking methods divide into two broad classes, namely single frame and multiple frame methods. The single frame methods include nearest neighbor, global nearest neighbor and JPDA (joint probabilistic data association). The most successful of the multiple frame methods are multiple hypothesis tracking (MHT) [8] and multiple frame assignment (MFA) [60, 61]. The performance advantage of the multiple frame methods over the single frame methods follows from the ability to hold difficult decisions in abeyance until more information is available and the opportunity to change past decisions to improve current decisions. In dense tracking environments the performance improvements of multiple frame methods over single frame methods is very significant, making it the preferred solution for many tracking problems. Thus, in addition to the availability single frame processing, multiple frame data association methods are an essential class of methods needed for almost all tracking needs.

The application of multiple frame tracking methods must consider an architecture in which the sensors are distributed across multiple platforms. Such geometric and sensor diversity has the potential to significantly enhance tracking and discrimination accuracy. A centralized architecture in which all measurements are sent to one location and processed with tracks being transmitted back to the different platforms is a simple one that is probably optimal in that it is capable of

producing the best track quality (e.g., purity and accuracy) and a consistent air picture. The centralized tracker is, however, unacceptable for several reasons, notably the communication overloads and single-point-failure. Thus, one must turn to a distributed architecture for both estimation/fusion and data association.

Perhaps one of the most challenging tracking problems is the development of an advanced MFA/MHT that preserves the multiple frame tracking performance across a network of platforms comparable to that achieved for centralized tracking. The network centric algorithm architecture described by Moore and Blair [55] provides a consistent air picture across multiple platforms and limits the communications loads to within a practical limit. This architecture is, however, designed with single frame data association in mind. The multiple frame data association approaches of MFA/MHT offer substantially improved tracking performance compared to the current single frame process while maintaining consistent threat pictures across platforms and limited communications load.

While network-centric tracking is capable of achieving enhanced estimation of tracks using geometric diversity and sensor variety not available in single platform tracking, one must deal with a host of new problems including (1) distributed data association and estimation; (2) consistent air picture; (3) management of communication loading (4) sensor biases as well as location and registration errors (sometimes called gridlock); (5) pedigree problems in the case of a sparse communication network; and, (6) out-of-order, latent, and missing data due to both sensor and communication problems. These topics as well as others are discussed in the article by Moore and Blair [55] and the book by Blackman and Popoli [8].

Thus, objective of the current work is to develop two near-optimal Network-Centric MFA/MHT architectures that preserve the quality of a centralized tracker across a network of platforms while managing communication loading and achieving a consistent air picture. One technique that has proved useful for achieving

SIAP is to require that each platform be in charge of assigning its own measurements to the network tracks, but this technique has been implemented only for single frame processing. This technique is extended to multiple frame processing for two of the architectures explained in subsequent chapters. Communication is addressed in part by the architectures in that track states and their error covariances are not required to be transmitted back to the various platforms.

In Chapters 2 - 7, four architectures, namely, MFA Centralized, Network MFA Centralized, Network MFA on Local Data and Network Tracks, and Network MFA on All Data and Network Tracks are explained or developed. Computational experience with these architectures are summarized extensively in Chapters 8 and 9 with Conclusions presented in Sections 8.6 and 9.6.

Chapter 2

INTRODUCTION TO A CENTRALIZED TRACKING ARCHITECTURE

2.1 Introduction to Small Target Tracking

In the multi-target/multi-sensor tracking problem, there are cases where the sensors are located at a single site or on a single platform (e.g., a ship or an aircraft). However, sensors are generally located on multiple platforms which may have different geographic locations. Platforms can be ships or airborne surveillances that are moving around in the entire surveillance region, or they can be ground platforms that are fixed at certain locations. There are multiple sensors on board each platform. Some of the sensors are rotators that have a scan rate around 10 seconds, and provide $2D$ measurements with only range and azimuth (sometimes referred to as bearing) as well as $3D$ position measurements. The rotators may provide Doppler information (range rate) as well. Some of the sensors are electronically scanning radars (ESRs) that have a faster update rate and provide very accurate $3D$ measurements. Some ESRs have a sensor tracker that provides a track ID and a state in addition to the measurement.

Many benefits can be derived from the use of multiple sensors in multiple target surveillance systems. These benefits are derived from the manner in which the data from each sensor can be used to complement the data of the other sensors in order to obtain broader coverage and more accurate target state estimate and ID

decisions, and to reduce false tracks. The central problem in multi-target/multi-sensor surveillance system is the partitioning of observations from multiple sensors into tracks and false alarms.

The multi-target/multi-sensor tracking architectures that are widely used can be divided into two categories.

- Centralized Tracking Architecture (chapter 2).

  Observations from different platforms are all sent to a single centralized processing unit, which has a composite tracker that does all the tracking and sends tracking results back to each platform. The centralized processing unit may or may not reside on one of the platforms. Theoretically, the centralized architecture should produce the best performance.

- Network (or Distributed) Tracking Architecture (chapter 3, 4, 5 and 7)

  Practical considerations, such as communication limitations, registration error and resolution differences, have led to a wide variety of architectures. In a network distributed tracking architecture, each platform has its own composite tracker that does the tracking.

2.1.1 Overview of Centralized Tracking Architecture

A centralized architecture (Figure 2.1) is one in which all the sensors on different platforms send measurements to a central processing unit that contains a composite tracker. The composite tracker processes all the measurements, puts them into proper frames and processes them in the time order as they become ready. This architecture is conceptually the simplest and will ideally produce the most accurate data associations and tracking. Furthermore, the tracks (global tracks) are broadcast back to all the platforms, and they share a consistent air picture.

However, this architecture suffers from the serious problem of single point failure. If the central processing unit fails, or the communication is cut down, none of the platforms will have any tracking results. Furthermore, broadcasting track states (state vector and covariance matrix) at each update is a heavy communication load to the network.

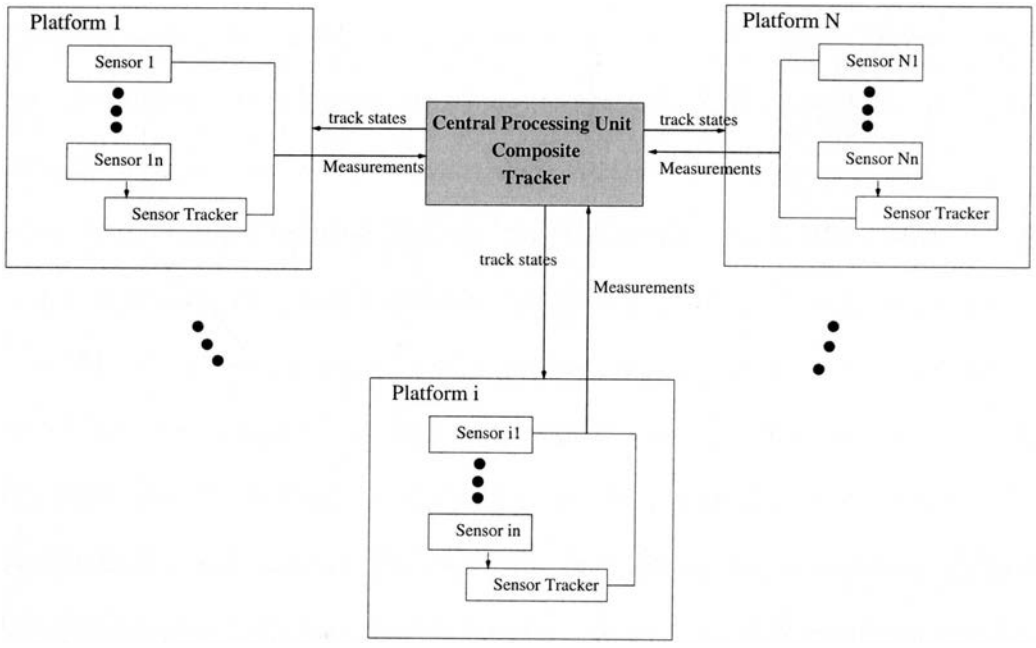


Figure 2.1: Centralized Architecture

2.1.2 Overview of Centralized Tracking Algorithms

Current algorithms for multi-target tracking generally fall into two categories: sequential and deferred logic. Sequential algorithms consider one frame of observations at a time. The term **frame** or **proper frame** is used to represent a grouping of observations where each target in the surveillance region can be seen at most once. Deferred logic considers several frames of observations all at once in making data association decisions. A popular deferred logic method used is called multiple hypothesis tracking (MHT) [8], in which one uses a sliding window of size N, builds a tree of possibilities, assigns a likelihood score to each track, develops an intricate pruning logic, and then solves the data association problem.

A newly proposed deferred logic method, which is called multiple frame assignment (MFA), is used in our centralized tracking architecture. It has been regarded as superior to the other methods. MFA is superior to single frame processing, because difficult data association decisions are held in abeyance until more information is available. For deferred logic, the central problem of tracking is formulated as a multi-dimensional assignment problem, which is NP-hard. Near optimal solutions, as opposed to optimal solutions, are used. MFA considers hundreds, or thousands, or even millions of hypothesis, while MHT only considers fifty or a hundred, thus leading to near optimal solutions that are much closer to optimal solutions, especially in dense scenarios. In the MFA method, a double pane sliding window (M/N) is used, where the window spans exactly M frames of data. All M frames within the window participate in track initiation calculations, but only the most recent $N < M$ frames participate in track continuation. A TrackTree that represents all feasible combinations of existing tracks and observations, each with a likelihood ratio score assigned, is designed to be the most efficient to navigate. The data association problem is formulated as an N dimensional assignment problem. Based on the solutions to the data association problem, the TrackTree is pruned and shifted.

2.2 Composite Tracker for the Centralized Architecture

A composite tracker for the centralized architecture has been well developed and tested. It is an Object Oriented software package written in C++ and owned by Numerica. It is based on the idea of an event driven code, which means the tracker processes events as they arrive, instead of waiting for the entire set of input data before processing.

2.2.1 Introduction to Event Driven

In traditional procedural programming, there exists a "main" program that handles input, processing and output sequentially. Therefore, such a traditional tracking program will read in the observations all at once, process them, and output the tracking result at the end. In surveillance systems, it won't meet real-time needs. Thus the modern tracking system is an event driven system.

Such a system has a main loop which just waits for events to occur. Whenever an event occurs, specific procedures are executed to handle the events. In the mean time, it might generate new events as well.

2.2.2 Event Driven Tracker

As discussed above, the composite tracker is an event driven system. It has a central-level event manager in which there is a global event queue that stores events that need to be processed. There are various event listeners, each of which listens to certain types of events (to perform specific operations required for those events). The tracker waits for an event to occur and then processes it. *Processing* means that the tracker goes through its entire list of event listeners to see if any of them can process it. New events may be generated while some events are being processed. They are pushed back onto the queue (*posted*). The top priority ones can be processed immediately (*fired*).

As is shown in Figure 2.2, it is the external events (observation event, metrics scoring event, and EOF event) that drive the tracker. Those external events can be read in from any specified input stream, such as stdin (keyboard), a particular file or a socket. Those external events are pushed onto the event queue of the tracker waiting to be processed.

There are basically three types of event listeners: the frameBuilder, the outputFormatter and the sliding window. Different event listeners process different

events. Furthermore, during the process of one event, new events (which are called internal events) can be generated and pushed onto the event queue.

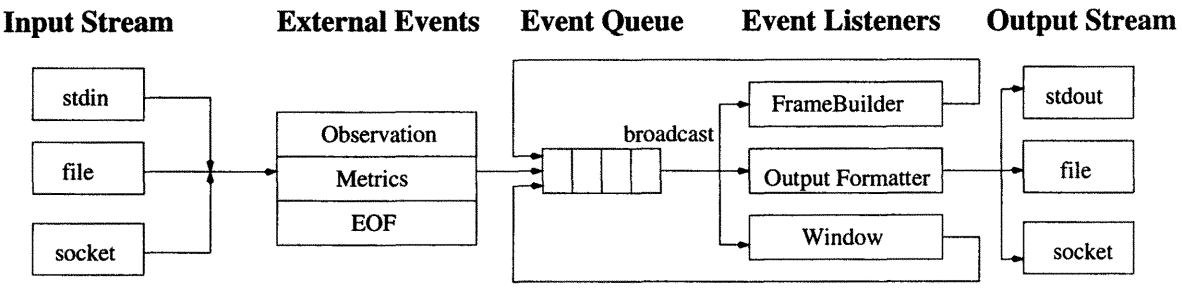


Figure 2.2: Event Driven Overview of the Centralized Tracker

External Events

- Observation Event

  The most common external event is an observation. An observation event contains a time tag, measurement type (such as $2D$ or $3D$, with or without Doppler), sensor ID (by which one can tell the sensor type), sensor navigation data (geodetic position and velocity of the sensor), necessary sensor characteristics (such as probability of detection, revisit time or Identity Friend or Foe (IFF) tag), the measurements (a subset of range, azimuth, elevation and Doppler) and the covariance associated with it.

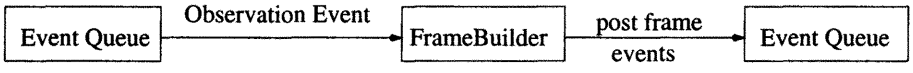


Figure 2.3: Observation Event

An incoming observation, from the input stream, is posted onto the event queue. The event listener, frameBuilder, processes observation events, and tries to put them into proper frames (which will be discussed later in the following sections). When a frame is ready, the frameBuilder generates a frame event. This is an internal event, which is posted on the event queue.

- Metrics Scoring Event

  A metrics scoring event is generated when we want to evaluate the performance of the tracker. A metrics event generally includes a time tag and platform navigational data (position and velocity). The composite tracker forks a child process to handle the metrics scoring event. One can choose what type of tracks they want:

  - "hard": If "hard" is chosen, then the child process searches the TrackTree for tracks that include the most recent firm decisions. Such a track set contains a consistent set of tracks.
  - "soft": If "soft" is chosen, then the child process searches the TrackTree for tracks that are the most recent soft decisions on the frames in the sliding window. However, as more information arrives, the associations may change for successive metrics scoring events.
  - "softPlus": If "softPlus" is chosen, then the child process processes all the observations that have already arrived, and chose the most probable tracks based all the data that are present. It collects all the unfinished frames that are still in the frameBuilder and appends them to the end of the sliding window. The data association problem is a $(M + p)$ dimensional assignment problem where p is the number of unfinished frames.

  Track Metrics Events are fired for the set of tracks, and they will be processed by the corresponding output formatter to be predicted to the required time, and transformed into the required coordinate system.

- EOF Event

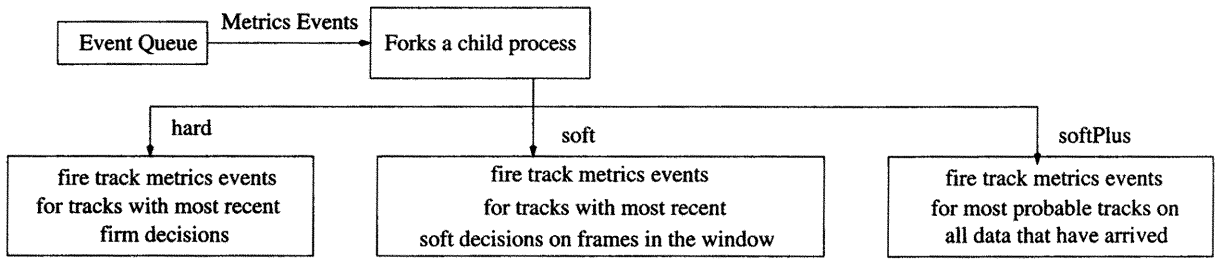


Figure 2.4: Metrics Scoring Event

The EOF event is encountered when the end of the input data stream is reached. At the end of each Monte Carlo run, the simulator generates an EOF event to indicate that no more measurements are left. The frameBuilder in the composite tracker posts all its unfinished frames as frame events. After processing all the frames left in the frameBuilder, it generates a close event and posts it onto the event queue.

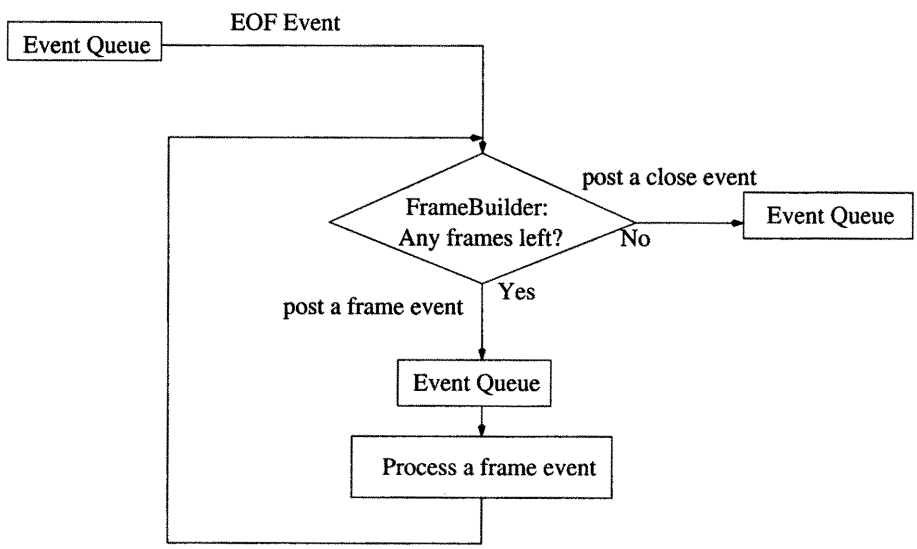


Figure 2.5: EOF Event

Internal Events

- Frame Event

It is the sliding window's responsibility to process a frame event. The frame is added to the sliding window's frame list, and the TrackTree is extended to

that frame. A data association problem is set up and solved. Solutions are parsed, the TrackTree is pruned and fixed DA decision events are fired. Then the window is shifted and the leftmost frame is moved out of the window and the TrackTree.

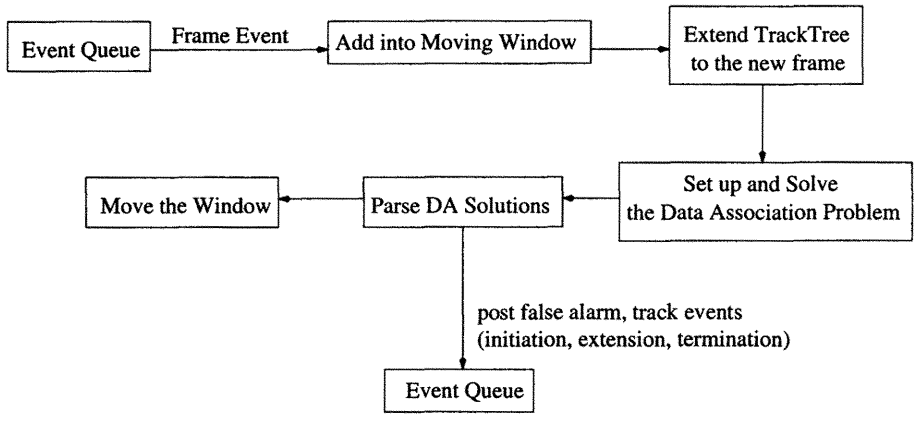


Figure 2.6: Frame Event

- Data Association Decisions Events

  DA decisions events can be divided into two categories: irrevocable decisions and soft decisions that may change. Those events include false alarms (FA) events, track initiation events, track extension events and track termination events. All those events are processed by the output formatter to generate the appropriate output and direct it to the specified output stream. All the processing details will be covered in section 2.3.2.

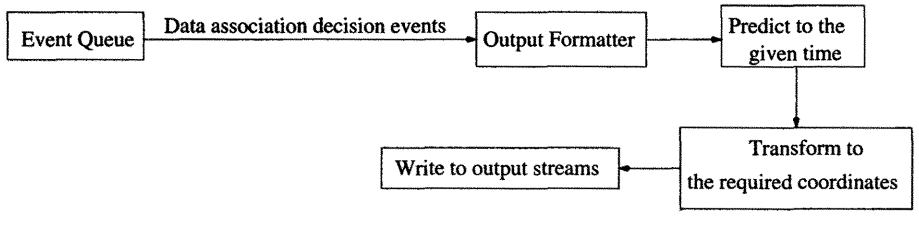


Figure 2.7: DA Decisions Event

- Close Event

The moving window starts to close by fixing the DA solutions and deleting the leftmost frames in the window. When the size of the window reaches zero, the tracker terminates normally.

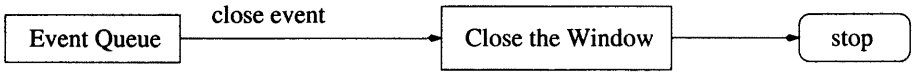


Figure 2.8: Close Event

2.3 Major Issues for the Centralized Architecture

2.3.1 FrameBuilder

The frameBuilder puts observations from the incoming stream into frames. A **proper frame** is a set of observations such that each target can be seen at most once. By definition, one could divide the observation stream into the smallest possible proper frames where each one contains a single observation. However, too many small frames will be built and it is impossible to relate them to the scoring formula introduced in Chapter 11, especially how to score missed detections. What we want to achieve in the frameBuilder here is to divide observation streams into **proper** frames and if the target is not seen in the frame, we can decide whether we need to score it as a missed detection or not.

The concept of a **sensorFrame** is introduced that contains a set of observations coming from the same sensor in which each target is seen at most once. In the multi-sensor scenario, each sensor has its own field of view (FOV). Some may overlap with those of others, some may not. If two sensors have non-overlapping FOV, then one can link two sensorFrames together to form a frame, where it is guaranteed that each target can be seen at most once in the frame. Look at the example of four sensors with FOVs shown in Figure 2.9. Sensor A and C have non-overlapping FOVs. Similarly, sensor B's and D's FOVs do not overlap. So, given

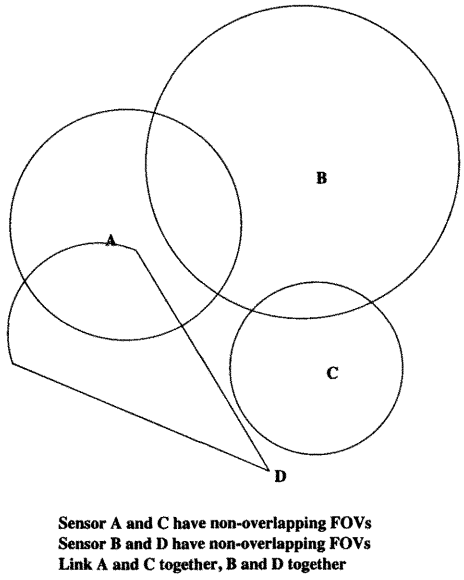


Figure 2.9: Sensors with different FOVs

one sensor frame from each sensor, it is obvious that one can link the sensorFrame of A and C together and B and D together to form two proper frames.

Thus, the algorithms for the frameBuilder in the centralized composite tracker can be summarized in the following steps:

1. Build sensorFrames for observations coming from the same sensor.

2. Link non-overlapping sensorFrames into frames in a timely fashion.

Building SensorFrames

The tracker has two different sensorFrame building schemes, due to the fact that it is now capable of handling observations from two different type of sensors.

- Rotators:

  A physical scan of the rotating radar is taken to be a frame. And it is assumed that the scanning rate/period of the rotators are known. Suppose the scanning period of a rotator is denoted to be T, then an observation at time t belongs to frame k, provided that $k * T \leq t \leq (k+1) * T$.

- Electronically Scanned Antenna (ESA) radars:

  An ESA radar has its own sensor tracker, so it can adaptively revisit targets, instead of following a fixed pattern. This means the target will not be revisited at a fixed rate. The method we use to divide the measurement streams into frames is highly heuristic.

  Each ESA measurement has a sensor track ID associated with it, so the criteria for frame building of ESA measurements are based on their own sensor track IDs.

  In order to avoid delays, the composite tracker wants to process the ESA measurements as soon as possible. Thus, the ESA sensor frameBuilder can have at most two sensorFrames in it.

  Let $\boldsymbol{F}^1, \boldsymbol{F}^2$ be the sensorFrames in the ESA sensor frameBuilder, and let ST^1_{ID}, ST^2_{ID} be the set of sensor track IDs that the observations in the sensorFrames have. Given a new observation $\boldsymbol{z}$, with its own sensor track ID st_z:

  1. If $st_z \in ST^1_{ID}$, then the sensorFrame $\boldsymbol{F}^1$ is ready to be processed. The ESA frameBuilder adds it into the second sensorFrame:

     $$\boldsymbol{F}^2 = \boldsymbol{z}, \qquad ST^2_{ID} = st_z.$$

  2. Otherwise, add the observation to the first sensorFrame:

     $$\boldsymbol{F}^1 = \boldsymbol{F}^1 \cup \boldsymbol{z}, \qquad ST^1_{ID} = ST^1_{ID} \cup st_z.$$

  If the sensorFrame $\boldsymbol{F}^1$ is ready, it will be taken to the frameBuilder to build frames after processing the observation $\boldsymbol{z}$. Then the ESA sensor frameBuilder sets $\boldsymbol{F}^1 = \boldsymbol{F}^2$ and $ST^1_{ID} = ST^2_{ID}$. Both $\boldsymbol{F}^2$ and ST^2_{ID} are reset to be empty sets.

Linking Non-Overlapping SensorFrames

Given two sensorFrames, how to tell if the two have overlapping FOVs is solved by the ideal of space partitioning as well. For efficiency considerations, a very coarse grid is applied to the entire surveillance region. It is assumed that observations in different grids can not emanate from the same target. Thus, if the FOVs have non-overlapping grids, it is concluded that the FOVs do not overlap. The total number of grids is generally of size 32 or 64, such that each grid can be represented by a bit in an integer or a long integer. For the sensorFrame of sensor i, there is an integer (SF_i^{FOV}) that denotes the grids the sensor's FOV lies in. During the course of building the sensorFrame, the integer is maintained whenever a new observation is added. If the sensorFrame has at least one observation lying in grid k, then the k^{th} bit of the SF_i^{FOV} is 1. The k^{th} bit of the SF_i^{FOV} is 0 otherwise. Thus, bit AND and OR operations can be used to link non-overlapping sensorFrames into frames. Figure 2.10 is an example of how to set the integer that represents which grids the sensor FOV lies in. For simplicity, the entire surveillance region is divided into 16 grids, and the sensor FOV is as shown in the figure. The actual sensor FOV lies in grids $4, 5, 6, 8, 9, 10, 12, 13$ and 14. However, based on the observations in the current sensorFrame, since there are no observations in grid 6 and 14, so there is no mark in grid 6, and 14.

For each sensor s_i, $i = 1, \cdots, N$, there is a queue of sensorFrames in time order that are finished and ready to be processed. Denote the queue to be $(SF_{s_i}(p_i), SF_{s_i}(p_i + 1), \cdots)$, then $t_{SF_{s_i}(p_i)} < t_{SF_{s_i}(p_i+1)}$, where $t_{SF_{s_i}(p_i)}$ can be the start time or the end time of the sensorFrame. The algorithm to build a frame F_k can be described as:

1. Order the sensorFrames queues such that

$$t_{SF_{s_{m_i}}(p_{m_i})} \leq t_{SF_{s_{m_j}}(p_{m_j})}, \quad i < j,\ m_i, m_j = 1, \cdots, N \tag{2.1}$$

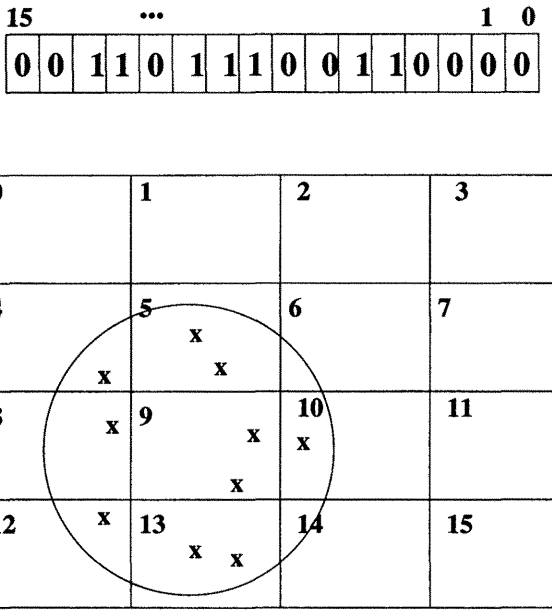


Figure 2.10: Example of Setting the SF^{FOV}

2. Set

$$F_k = SF_{s_{m_1}}(p_{m_1}), \tag{2.2}$$

$$F_k^{FOV} = SF_{s_{m_1}}^{FOV}(p_{m_1}), \tag{2.3}$$

where sensorFrame $SF_{s_{m_1}}(p_{m_1})$ has the earliest time tag of all the sensorFrames.

3. For $i = 2, \cdots, N$, check if

$$F_k^{FOV} \text{ AND } SF_{s_{m_i}}(p_{m_i}) = 0, \tag{2.4}$$

then

$$F_k = F_k \cup SF_{s_{m_i}}(p_{m_i}), \tag{2.5}$$

$$F_k^{FOV} = F_k^{FOV} \text{ OR } SF_{s_{m_i}}^{FOV}(p_{m_i}). \tag{2.6}$$

Otherwise, sensorFrame $SF_{s_{m_i}}(p_{m_i})$ has overlapping FOV with at least one sensorFrame in the frame F_k.

2.3.2 Double Pane Sliding Window

Introduction to the Sliding Window

The sliding window in our tracker is a double pane window, which is generally denoted as an M/N window. The window spans exactly M frames in the data. The most recent $N < M$ frames participate in track extension, and all M frames participate in track initiation. However, the first $(M - N)$ frames in the track initiation window contain only observations that have not been associated with any existing tracks yet.

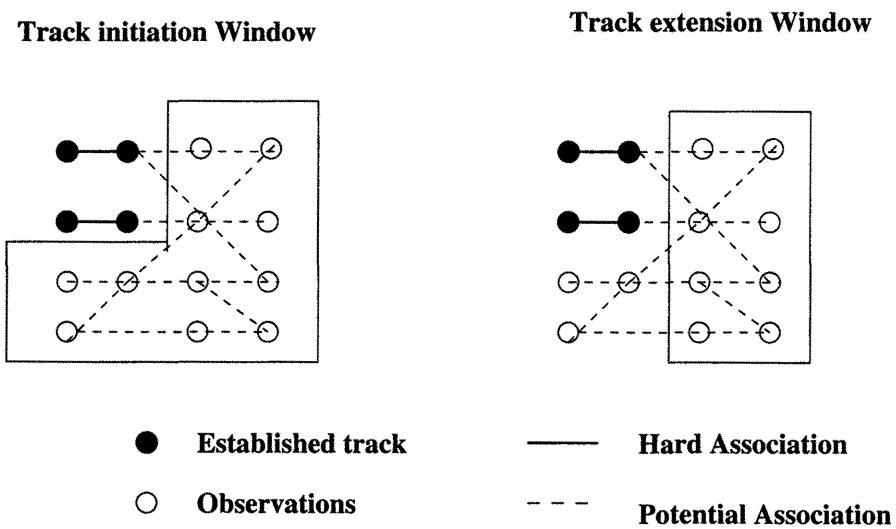


Figure 2.11: Tracking Initiation and Extension Window

A 4/2 window is shown in Figure 2.11, where 4 frames of observations can be used to initiate new tracks. The existing tracks extend into the $3rd$ and $4th$ frames (extension window).

Introduction to TrackTree

Let the frames in the window at step k be denoted by $f_{k_0}, \cdots, f_{k_{M-N-1}}, \cdots, f_{k_{M-1}}$, and the i^{th} observation in frame f_{k_j} is denoted by $z^i_{k_j}$.

TrackTree is the tree structure for maintaining the entire track database on each of the platforms. The TrackTree has only one root node, and the root node has no parents. All other nodes in the TrackTree have exactly one parent.

The TrackTree data structure represents all feasible combinations of existing tracks and observations, and each combination is represented as a distinct string of integers. For example, Figure 2.12 shows a TrackTree for a 4/2 window (initiation size of 4 and extension size is 2). The root node is always denoted by $(0 \cdots 0)$ and is depicted at the top of the tree. The number of zeros in the root node is the same as the number of frames in the sliding window. Nodes containing only non-negative integers $((i_{k_0} i_{k_1} \cdots i_{k_{M-1}})$, where $i_{k_j} \geq 0$ for all $j = 0, \cdots, M-1$) are collections of observations, and these nodes participate in track initiation calculations ($i_{k_j} = 0$ means there is no detection in frame f_{k_j}). The observations from a given string come from distinct frames, but a given observation may appear in the strings in more than one node. When $i_{k_j} = 0$, then the string does not contain any observation from frame k_j. Existing tracks are denoted by negative integers. For instance, a node represented by $(0 \cdots 0 - T_{k_l} i_{k_{M-N}} \cdots i_{k_{M-1}})$, extends the track with track ID T_{k_l}. The extension contains observations $i_{k_j} \geq 0$ for all $j = M-N, \cdots, M-1$. The number of zeros in the preamble to the string is always $M-N-1$, so that the integer strings in all the nodes have the same length on all nodes.

Each node represents a feasible combination of measurements in different frames or a combination of an existing track with measurements. For example, node 1001 means $z^1_{f_0}$ goes with $z^1_{f_3}$ and node $0-111$ means track number 1 is feasible with observation $z^1_{f_2}$ and $z^1_{f_3}$.

There is a *TrackString* data structure associated with each node, which can be denoted as $TS_{i_{k_0} i_{k_1} \cdots i_{k_{M-1}}}$ or $TS_{0 \cdots 0 - T_{k_l} i_{k_{M-N}} \cdots i_{k_{M-1}}}$. The *TrackString* records all the necessary filtering, gating and scoring information when a tracking filter has initiated.

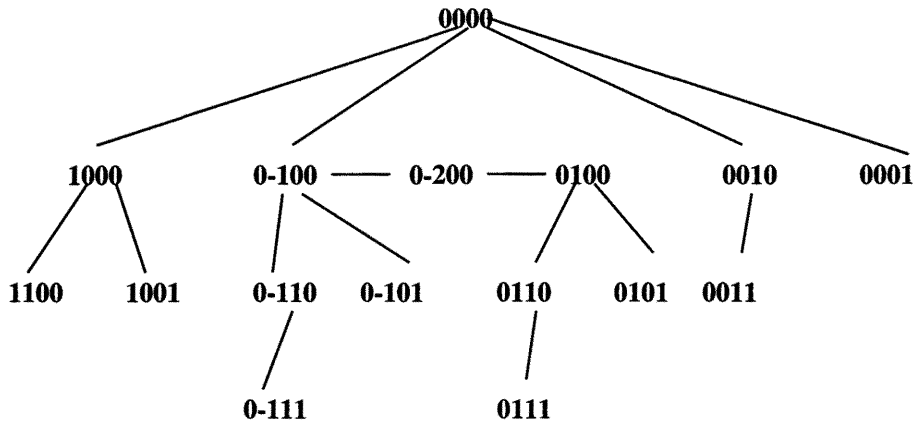


Figure 2.12: A simple example of the TrackTree

Introduction to the Data Association Problem

The data association problem can be posed as an M dimensional assignment problem. For instance, the four-dimensional assignment problem produced by the above TrackTree can be represented in a tableau as follows. Each line of the tableau corresponds to a different node in the TrackTree data structure, and each line is called an arc. The first four columns of the tableau contain the individual integers from the node string, and the last column is the cost, or the score, for each arc. All singleton arcs (arcs which have only one non-zero index, such as 1000 and $0-100$) are always added, since they are required by the assignment solver itself. All *valid* tracks are also added. A *valid* track has an associated tracking filter and produces a negative score. Thus, if a given track is required to have at least 3 observations in order to start a filter (according to a user definable parameter *initLengthMin*), then nodes that have only two observations are not added to the tableau.

```
1  0 0 0  0.0
0  1 0 0  0.0
0 -1 0 0  -40.0
0 -2 0 0  -50.0
0 -1 1 0  -52.0
```

```
0 -1 0 1  -50.0
0 -1 1 1  -60.0
0  1 1 1  -20.0
0  0 1 0  0.0
0  0 0 1  0.0
```

Introduction to Parsing DA Solutions

The DA problem associated with a M/N window is generally of dimension M as described in the previous section. All DA solutions are parsed into the following categories:

1. *False Alarms* are defined as solution arcs of the form $(0, \cdots, 0, i_{k_j}, 0, \cdots, 0)$. If $j > 0$, so that observation $z_{f_{k_j}}^{i_{k_j}}$, is not in the oldest frame of the window, then $z_{f_{k_j}}^{i_{k_j}}$ is temporarily treated as a false alarm. But if $j = 0$, then $z_{f_{k_0}}^{i_{k_0}}$ is declared as a false alarm permanently.

2. *Tentative Tracks* are solution arcs of the form $(0, \cdots, 0, i_{k_j}, \cdots, i_{k_{M-1}})$ where $j > M - N$ and which has at least two non-zero indices.

3. *Initiating Tracks* are defined as solution arcs of the form

   $(i_{k_0}, \cdots, i_{k_{M-N}}, i_{k_{M-N+1}}, \cdots, i_{k_{M-1}})$, which have at least two or more non-zero indices and such that the subString $(i_{k_0}, \cdots, i_{k_{M-N}})$ preceding the extension window has at least one non-zero index. It will be called as an initiating track if the trackString $TS_{i_{k_0},\cdots,i_{k_{M-N}},0,\cdots,0}$ meets the following criteria: (1) a tracking filter has been initiated, (2) it produces a negative score and (3) the track has at least *initLength* observations,. If the trackString $TS_{i_{k_0},\cdots,i_{k_{M-N}},0,\cdots,0}$ meets all three criteria, then association between all those observations will be fixed. Otherwise the association between i_{k_0} and i_{k_1} will be fixed.

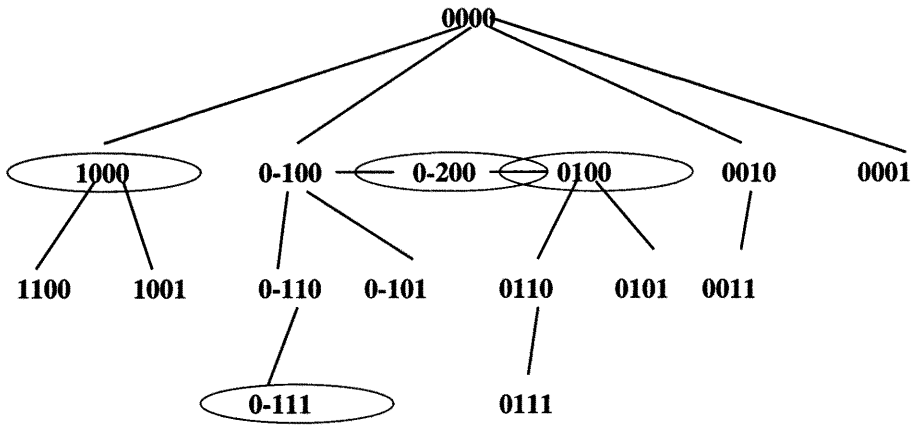


Figure 2.13: DA solutions expressed in the TrackTree

4. *Extending Tracks* are defined to be solution arcs of the form

   $(0, \cdots, 0, -T_{k_l}, i_{k_{M-N}}, \cdots, i_{M-1})$, where T_{k_l} is track ID. The association between track T_{k_l} and observation $z^{i_{k_{M-N}}}_{f_{k_{M-N}}}$ is fixed if $i_{k_{M-N}} \neq 0$.

5. *Terminating Tracks* are defined as solution arcs of the form

   $(0, \cdots, 0, -T_{k_l}, 0, \cdots, 0)$. If the corresponding track hasn't been seen for a while or the score degrades to a number greater than zero, then the track will be removed from the TrackTree permanently.

The DA solutions to the four-dimensional assignment problem above are circled in Figure 2.13. For solution arc 1000, observation $z^1_{f_{k_0}}$ is declared a false alarm permanently. For solution arc 0100, observation $z^1_{f_{k_1}}$ is declared a false alarm temporarily. Solution arc $0 - 111$ is declared as an extension of the existing track and the observation $z^1_{f_{k_2}}$ in first frame of the extension window is fixed to track 1. $0 - 200$ is declared a dropping track if it has not been seen for a certain period of time. Likewise, if the score for arc $0 - 200$ degrades to zero, then it is declared a dropping track. Otherwise, $0 - 200$ can be regarded as a track that has two missed detections.

Now the TrackTree is pruned based on firm decisions. For false alarm solutions, nothing else needs to be done. For track extensions and track initiations, the

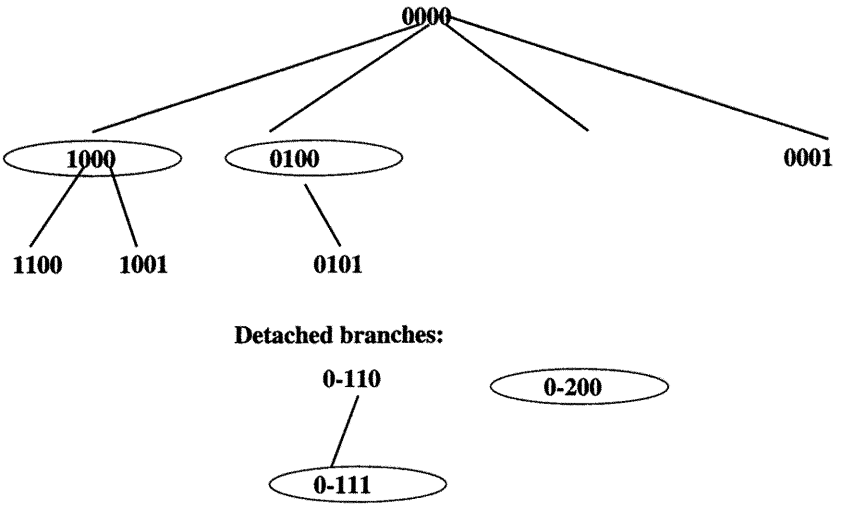


Figure 2.14: TrackTree after Pruning

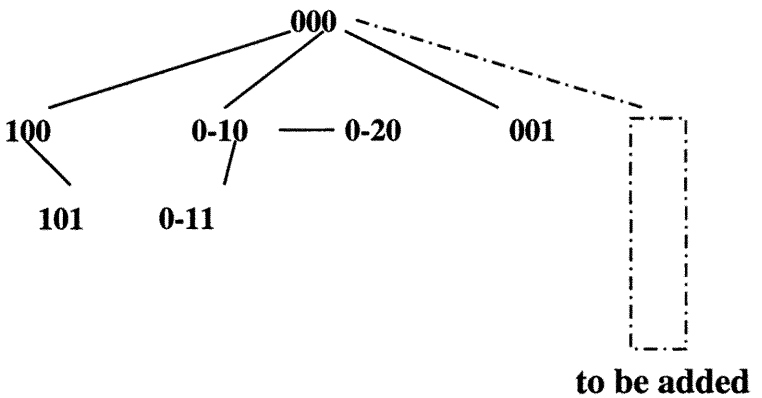


Figure 2.15: TrackTree after Shifting

corresponding sub-trees are detached, and any incompatible branches are pruned, as is shown in Figure 2.14. Then, the detached branches are grafted back into the TrackTree.

After the pruning process, the TrackTree is shifted to the left by one frame. Now the tracker is ready to process a new frame of data. The TrackTree, after pruning is shown in Figure 2.15.

A valid TrackTree satisfies the following conditions

- If node $(i_{k_0}, i_{k_1}, \cdots, i_{k_{j+1}}, 0, \cdots, 0)$ exists in the tree,

  then node $(i_{k_0}, i_{k_1}, \cdots, i_{k_j}, 0, \cdots, 0)$ must exist in the tree.

- If node $(i_{k_0}, i_{k_1}, \cdots, i_{k_{j+1}}, 0, \cdots, 0)$ exists in the tree,

  then node $(0, i_{k_1}, \cdots, i_{k_{j+1}}, 0, \cdots, 0)$ must exist.

2.3.3 Scoring Missed Detections

As explained in full detail in Chapter 11, the scoring formula for MFA has a nice recursive form. The likelihood ratios and the scores can be written recursively as:

$$L_{i_1 i_2 \cdots i_N} = \prod_{k=1}^{N} L_{i_k} = L_{i_1 i_2 \cdots i_{N-1}} L_{i_N} \tag{2.7}$$

$$c_{i_1 i_2 \cdots i_N} = \sum_{k=1}^{N} c_{i_k} = c_{i_1 i_2 \cdots i_{N-1}} + c_{i_N} \tag{2.8}$$

with

$$L_{i_k} = \left\{ P_\phi^k \right\}^{\Delta_{0 i_k}} \left\{ \left[\frac{(1-P_\chi^k) P_d^k p_t^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\delta_{i_k}^k} \left[\frac{\lambda_\nu^k p_\nu^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\nu_{i_k}^k} \right\}^{(1-\Delta_{0 i_k})}$$
$$\text{for } k = 1, 2, \cdots, N. \tag{2.9}$$

For a detection, when $\Delta_{0 i_k} = 0$, it is scored either as a track initiation or a track extension. However, in a multiple sensor scenario, the fact that $\Delta_{0 i_k} = 0$, or the track is associated with 0 in a certain frame does not necessarily mean it has a missed detection in that frame. A 0 shouldn't be scored as a missed detection if the target is out of the sensor's field of view (FOV). The criteria for scoring missed detections are heuristic.

Suppose frame f_k is made up of sensorFrames whose sensor IDs are denoted by $\{s_{k_1}, \cdots, s_{k_p}\}$. For each track, there is also a list of sensor IDs denoted by $\{s_{t_1}, \cdots, s_{t_q}\}$. The track is in the FOV of the sensor s_{t_i}, for $i = 1, \cdots, q$. For a 0 index in frame f_k:

1. It should not be scored as missed detection scored if

$$\{s_{k_1}, \cdots, s_{k_p}\} \cap \{s_{t_1}, \cdots, s_{t_q}\} = \emptyset \tag{2.10}$$

2. Suppose

$$\{s_{k_1}, \cdots, s_{k_p}\} \cap \{s_{t_1}, \cdots, s_{t_q}\} = \{s_1, \cdots, s_p\} \tag{2.11}$$

Then for each $s_j \in \{s_1, \cdots, s_p\}$, if the time difference between the frame end time and the time last seen by that particular sensor is greater than the revisit time for that sensor, then one should score a missed detection for s_j.

The list of sensor IDs that see the track is adaptively maintained. If a new sensor sees the target, then that new sensor should be added into the list. Otherwise, we should remove a sensor s from the list if it has not seen the track for Q successive frames, where Q can be decided based on

$$(1 - Pd_s)^Q < \alpha \tag{2.12}$$

where Pd_s is the probability of detection for sensor s, and α can be a user defined parameter (e.g. $\alpha = 0.05$).

2.3.4 Filtering and Estimation

For each collection of observations, a tracking filter is applied to do the filtering and estimation. The tracking filter can be a single model Kalman filter, or an IMM filter. Square root filtering algorithm can be used to achieve numerical stability.

Refiltering Window

As described above, a distinct string of integers represents a node in the TrackTree which corresponds to a *trackString*. Each trackString is a sequence of observations associated together. So a **refiltering** window whose length is totally independent of the double pane sliding window is used.

In the sliding window, observations are organized according to frames. In the refiltering window, the observations are organized according to their time tags. When the tracking filter has been initiated, the filter states and the scores corresponding to each observation are also stored. When a new observation arrives to

update the track, it first tries to insert the new observation in the correct position in the refiltering window (an old one might be moved out). It recovers the filter states and the scores of the observation immediately preceding the new one in time order and updates the track with the new observation and all the observations that have a later time tag.

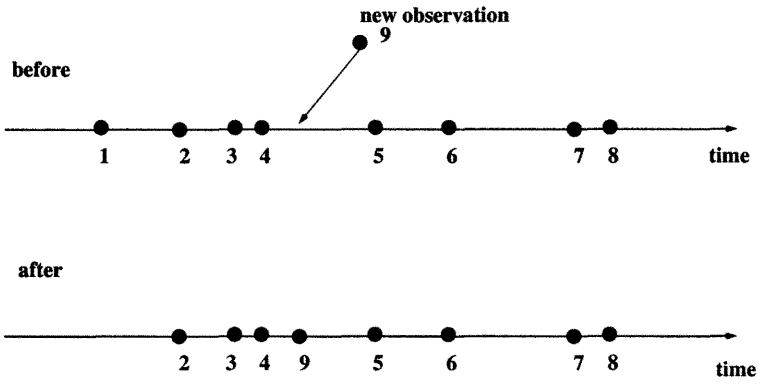


Figure 2.16: Illustration of the Refiltering Window of size 8

Notice here that after inserting the new observation into the correct position of the refiltering window, one needs to update the track with the new observation and the observations thereafter. As indicated in Figure 2.16, one needs to extract the filter state at observation 4 and update the track with observations $9, 5, 6, 7,$ and 8. After the update, observation 1 will be moved out.

This is the optimal solution to the latent data problem. The refiltering window size can be a user definable parameter, based on *a priori* knowledge. It has the disadvantage of possibly incurring huge computational loads. And it needs tremendously more memory space to store the observations and the track states associated with the observations at each step.

Unfortunately, it is still possible that the time tag of the new observation is even earlier than that of the first observation in the refiltering window, in which case, one can discard the observation (the association being regarded as not feasible) or go to the negative time update methods explained below.

Negative Time Update

Let's consider a linear model first:

$$\boldsymbol{x}_k = \boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \tag{2.13}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k, \tag{2.14}$$

where $\boldsymbol{Q}_k = E[\boldsymbol{w}_k\boldsymbol{w}_k^T]$ and $\boldsymbol{R}_k = E[\mathrm{v}_k\mathrm{v}_k^T]$.

The following methods are used to handle negative time updates.

- Retrodict

  A simple approach is to predict the state estimate back to the time of the measurement (retrodict), form the residual, and use the residual to update the state estimate in the same manner that a current observation would be used (process noise effects are ignored and the filter covariance matrix remains referenced to the current time). The gain and subsequent covariance update are also computed as if the observations were not delayed.

  For convenience, we only demonstrate how to incorporate a measurement arriving one time iteration late. Assume a measurement arrives late, with time tag, t_{k-1}, just after the filter update has been made for a measurement with time tag, t_k. We define $T = t_{k-1} - t_k$, such that $T < 0$. Two properties will be important in the following. First note that

$$\boldsymbol{F}(T) = \boldsymbol{F}^{-1}(|T|), \text{ or } \boldsymbol{F}^{-1}(T) = \boldsymbol{F}(|T|). \tag{2.15}$$

  For a nearly constant velocity model, we have the additional property that

$$\boldsymbol{F}^{-1}(T) = \boldsymbol{F}(-T). \tag{2.16}$$

  The principle of the negative time update can be described as follows: First we propagate the current estimates of state and covariance back to time

$k-1$ to produce the best estimate of these properties at this previous time on the basis of the previously estimated states. In a second step the current measurement $\mathbf{z}_{k-1}$ is included in the state and covariance update. Then, in a third step, we propagate the now best estimate at time $k-1$ forward to produce the new estimate at time k, which now includes the measurement $\mathbf{z}_{k-1}$.

1. Negative time-update:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k-1|k-} &= \boldsymbol{F}(T)\hat{\boldsymbol{x}}_{k|k-} \\ \boldsymbol{P}_{k-1|k-} &= \boldsymbol{F}(T)\boldsymbol{P}_{k|k-}\boldsymbol{F}^T(T). \end{aligned} \tag{2.17}$$

   $k-$ denotes the state at time k which does not contain yet the new measurement.

2. Measurement update:

$$\begin{aligned} \boldsymbol{K}_{k-1} &= \boldsymbol{P}_{k-1|k-}\boldsymbol{H}_{k-1}^T\left[\boldsymbol{H}_{k-1}\boldsymbol{P}_{k-1|k-}\boldsymbol{H}_{k-1}^T + \boldsymbol{R}_{k-1}\right]^{-1} \\ \hat{\boldsymbol{x}}_{k-1|k} &= \hat{\boldsymbol{x}}_{k-1|k-} + \boldsymbol{K}_{k-1}\left[\mathbf{z}_{k-1} - \boldsymbol{H}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-}\right] \\ \boldsymbol{P}_{k-1|k} &= \left[\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\right]\boldsymbol{P}_{k-1|k-}. \end{aligned} \tag{2.18}$$

3. Forward Projection:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k+} &= \boldsymbol{F}(|T|)\hat{\boldsymbol{x}}_{k-1|k} \\ \boldsymbol{P}_{k|k+} &= \boldsymbol{F}(|T|)\boldsymbol{P}_{k-1|k}\boldsymbol{F}^T(|T|) + \boldsymbol{Q}_{k-1}. \end{aligned} \tag{2.19}$$

   $k+$ denotes the state which includes the late measurement.

- Direct Update Methods

Delayed observations are expressed as a function of the current state estimates. Unlike the retrodict method explained above, it won't predict the state estimate back to the time of the latent measurement. Instead, it will make the measurement prediction based on the function between the latent

measurements and the current state estimates. Again, process noise effects are ignored. Then the state and covariance update (at the current time) can be carried out as in the procedures in a standard Kalman filter routine. The measurement matrix now takes the form of $\boldsymbol{H}_{k-1}\boldsymbol{F}(T)$, for $T \leq 0$.

The direct update method is a two-step method, as follows.

1. Measurement Prediction:

$$\begin{aligned} \hat{\boldsymbol{z}}_{k-1} &= \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\hat{\boldsymbol{x}}_{k|k-} \\ \boldsymbol{S}_{k-1} &= \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\boldsymbol{P}_{k|k-}\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T + \boldsymbol{R}_{k-1} \end{aligned} \tag{2.20}$$

2. Measurement update:

$$\begin{aligned} \boldsymbol{K}_{k-1} &= \boldsymbol{P}_{k|k-}\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T\boldsymbol{S}_{k-1}^{-1} \\ \hat{\boldsymbol{x}}_{k|k+} &= \hat{\boldsymbol{x}}_{k|k-} + \boldsymbol{K}_{k-1}\left[\boldsymbol{z}_{k-1} - \hat{\boldsymbol{z}}_{k-1}\right] \\ \boldsymbol{P}_{k|k+} &= \left[\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\right]\boldsymbol{P}_{k|k-}. \end{aligned} \tag{2.21}$$

- MMSE Approach

The MMSE approach accounts for the effects of the process noise that entered the system during the time interval $[t_{k-1}, t_k]$. The measurement $\boldsymbol{z}_{k-1}$ can be of no help in estimating the random target motion that entered the system after that time. The measurement equation can be expressed as:

$$\begin{aligned} \boldsymbol{z}_{k-1} &= \boldsymbol{H}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1} \\ &= \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\left[\boldsymbol{x}_k - \boldsymbol{w}_{k-1}\right] + \boldsymbol{v}_{k-1} \end{aligned} \tag{2.22}$$

Then, the update equation is of the form:

$$\hat{\boldsymbol{x}}_{k|k+} = \hat{\boldsymbol{x}}_{k|k-} + \boldsymbol{K}_{k-1}\{\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\left[\boldsymbol{x}_k - \boldsymbol{w}_{k-1}\right] + \boldsymbol{v}_{k-1} - \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\hat{\boldsymbol{x}}_{k|k-}\} \tag{2.23}$$

Define the estimation error residual:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{2.24}$$

Then,

$$\begin{aligned}\tilde{\boldsymbol{x}}_{k|k+} &= \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k+} \\ &= [\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)](\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-}) + \boldsymbol{K}_{k-1}[\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\boldsymbol{w}_{k-1} - \boldsymbol{v}_{k-1}] \\ &= [\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)]\tilde{\boldsymbol{x}}_{k|k-} + \boldsymbol{K}_{k-1}[\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\boldsymbol{w}_{k-1} - \boldsymbol{v}_{k-1}]\end{aligned} \tag{2.25}$$

The resulting covariance matrix equation is

$$\boldsymbol{P}_{k|k+} = E[\tilde{\boldsymbol{x}}_{k|k+}\tilde{\boldsymbol{x}}_{k|k+}^T] \tag{2.26}$$

The appropriate choice of the Kalman gain matrix $\boldsymbol{K}_{k-1}$ is found by minimizing the mean square error matrix (covariance matrix). Mathematically, $\boldsymbol{K}_{k-1}$ is found by setting the derivative of the trace to zero:

$$\frac{\partial}{\partial \boldsymbol{K}} \text{trace}(\boldsymbol{P}_{k|k+}) = 0 \tag{2.27}$$

which gives:

$$\begin{aligned}\boldsymbol{K}_{k-1} =& [\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & \{\boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & + \boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{Q}_{k-1} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}^T]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T + \boldsymbol{R}_{k-1}\}^{-1}\end{aligned} \tag{2.28}$$

where $\boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} = E[\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}^T]$. The fact that state estimation residual $\tilde{\boldsymbol{x}}_{k|k-}$ is cross-correlated with the plant noise $\boldsymbol{w}_{k-1}$ is due to the fact that when updating with observation $\boldsymbol{z}_k$, the plant noise in the time interval $[t_{k-2}, t_k]$ is taken into consideration already. Thus, the plant noise during the time interval $[t_{k-1}, t_k]$, which is now denoted as $\boldsymbol{w}_{k-1}$, is considered twice. And $\boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}$ has the form:

$$\boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} = [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{Q}_{k-1} \tag{2.29}$$

Notice here $\boldsymbol{K}_k$ is the Kalman gain matrix when updating with measurement $\boldsymbol{z}_k$, and $\boldsymbol{H}_k$ is the corresponding measurement matrix.

1. Measurement Prediction:

$$\hat{\boldsymbol{z}}_{k-1} = \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\hat{\boldsymbol{x}}_{k|k-} \tag{2.30}$$

2. Kalman Gain Calculation:

$$\begin{aligned}\boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} =& [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{Q}_{k-1} \\ \boldsymbol{K}_{k-1} =& [\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & \{\boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & + \boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{Q}_{k-1} - \boldsymbol{C}^T_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T + \boldsymbol{R}_{k-1}\}^{-1}\end{aligned} \tag{2.31}$$

3. Measurement Update

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k+} &= \hat{\boldsymbol{x}}_{k|k-} + \boldsymbol{K}_{k-1}\left[\boldsymbol{z}_{k-1} - \hat{\boldsymbol{z}}_{k-1}\right] \\ \boldsymbol{P}_{k|k+} &= \left[\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\right]\boldsymbol{P}_{k|k-} + \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\boldsymbol{C}^T_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}\end{aligned} \tag{2.32}$$

The MMSE approach is computational more intensive, and it has the disadvantage that it needs to store the Kalman gain $\boldsymbol{K}_k$ and the measurement matrix $\boldsymbol{H}_k$ at the previous step.

- Optimal Solution

Due to the assumption that all process noises and measurement noises are Gaussian, the optimal MAP estimator is actually the MV (minimum variance) estimator. Let $\hat{\boldsymbol{E}}[\cdot]$ denote the LMV estimator, then the optimal solution is

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k+} &= \hat{\boldsymbol{E}}[\boldsymbol{x}(k)|\boldsymbol{Z}^{k-} \cup \boldsymbol{z}_{k-1}] \\ &= \hat{\boldsymbol{E}}[\boldsymbol{x}(k)|\boldsymbol{Z}^{k-}] + \hat{\boldsymbol{E}}[\boldsymbol{x}(k)|\tilde{\boldsymbol{z}}_{k-1|k-}] \\ &= \hat{\boldsymbol{x}}_{k|k-} + \hat{\boldsymbol{E}}[\boldsymbol{x}(k)|\tilde{\boldsymbol{z}}_{k-1|k-}]\end{aligned} \tag{2.33}$$

The predicted measurement based on measurement set $\boldsymbol{Z}^{k-} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{k-2}, \boldsymbol{z}_k\}$ is denoted by:

$$\hat{\boldsymbol{z}}_{k-1|k-} = \hat{\boldsymbol{E}}[\boldsymbol{z}_{k-1}|\boldsymbol{Z}^{k-}] \tag{2.34}$$

Thus, the measurement update is

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k+} &= \hat{\boldsymbol{x}}_{k|k-} + \hat{\boldsymbol{E}}[\boldsymbol{x}(k)|\tilde{\boldsymbol{z}}_{k-1|k-}] \\ &= \hat{\boldsymbol{x}}_{k|k-} + \boldsymbol{E}[\boldsymbol{x}(k)\tilde{\boldsymbol{z}}^t_{k-1|k-}]\, \boldsymbol{E}[\tilde{\boldsymbol{z}}_{k-1|k-}\tilde{\boldsymbol{z}}^t_{k-1|k-}]\, \tilde{\boldsymbol{z}}_{k-1|k-}\end{aligned} \tag{2.35}$$

Similar to the MMSE approach, the optimal solution is computational more intensive, and it has the disadvantage that it needs to store the Kalman gain $\boldsymbol{K}_k$ and the measurement matrix $\boldsymbol{H}_k$ at the previous step.

1. Measurement Prediction:

$$\hat{\boldsymbol{z}}_{k-1} = \boldsymbol{H}_{k-1}\boldsymbol{F}(T)\left[\hat{\boldsymbol{x}}_{k|k-} - \boldsymbol{Q}_{k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1}\tilde{\boldsymbol{z}}_k\right] \tag{2.36}$$

   where $\boldsymbol{H}_k$ is the measurement matrix when updating observation $\boldsymbol{z}_k$, and $\tilde{\boldsymbol{z}}_k$ and $\boldsymbol{S}_k$ are the corresponding measurement predicting error and innovation matrix.

   Unlike the previous solutions, the measurement prediction from time t_k to time t_{k-1} accounts in full for the process noise $\boldsymbol{w}_{k-1}$.

2. Kalman Gain Calculation:

$$\begin{aligned}\boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} =& [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{Q}_{k-1} \\ \boldsymbol{K}_{k-1} =& [\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & \{\boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{P}_{k|k-} - \boldsymbol{C}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} - \boldsymbol{C}^T_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T \\ & + \boldsymbol{H}_{k-1}\boldsymbol{F}(T)[\boldsymbol{Q}_{k-1} - \boldsymbol{Q}_{k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1}\boldsymbol{H}_k\boldsymbol{Q}_{k-1}]\boldsymbol{F}^T(T)\boldsymbol{H}_{k-1}^T + \boldsymbol{R}_{k-1}\}^{-1}\end{aligned} \tag{2.37}$$

3. Measurement Update

$$\hat{\boldsymbol{x}}_{k|k+} = \hat{\boldsymbol{x}}_{k|k-} + \boldsymbol{K}_{k-1}\left[\boldsymbol{z}_{k-1} - \hat{\boldsymbol{z}}_{k-1}\right]$$

$$\boldsymbol{P}_{k|k+} = \left[\boldsymbol{I} - \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\right]\boldsymbol{P}_{k|k-} + \boldsymbol{K}_{k-1}\boldsymbol{H}_{k-1}\boldsymbol{F}(T)\boldsymbol{C}^{T}_{\tilde{\boldsymbol{x}}_{k|k-}\boldsymbol{w}_{k-1}} \tag{2.38}$$

Chapter 3

INTRODUCTION TO NETWORK MFA

3.1 Problem Description

The multi-target/multi-sensor tracking systems can be divided into two categories, as discussed in Chapter 2. The Centralized Tracking Architecture (CTA) discussed in detail in Chapter 2 has optimal performance, and so the CTA provides a baseline for performance comparisons with Network (or distributed) Tracking Architectures (NTAs).

In the NTAs, each platform has its own composite tracker. Each composite tracker applies the MFA algorithm. The MFA algorithm itself belongs to a category of data association algorithms called deferred logic. All NTAs use deferred logic on all platforms.

3.1.1 Definition of Terms

The following terms that are used frequently in the tracking literature and these terms appear in later chapters:

1. **Local platform** is the platform on which one resides (also called as own-ship or own-platform).

2. **Local sensor** is a sensor on a local platform.

3. **Local data** are measurements from a local sensor.

4. **Remote platform** is a platform located a geographic distance from the local platform.

5. **Remote sensor** is a sensor on a remote platform.

6. **Remote data** are measurement from a remote sensor.

7. **Frame** is a set of measurements in which each target can be seen at most once.

8. **Local frame** is a frame of data from a local sensor.

9. **Remote frame** is a frame of data from a remote sensor.

10. **Sensor tracker** is a tracker that processes data from a designated sensor.

11. **Sensor tracks** are a set of tracks that are formed by the sensor tracker using data from a single sensor.

12. **Local tracker** is a tracker that processes local data only.

13. **Local tracks** are a set of tracks that are formed from local data.

14. **Composite tracker** is a tracker located on a particular platform (or else a central processing unit) that processes measurements from multiple platforms.

15. **Composite (Global, Network) tracks** are a set of tracks that combines information from multiple platforms.

16. **Measurement Report (MR)** is a message indicating a remote measurement.

17. **Associated Measurement Report (AMR)** is a message indicating a measurement and a composite track ID to which it has been assigned by the own ship platform.

18. **Associated Measurement Map (AMM)** is a message indicating which measurements in a frame of data are associated with which composite tracks.

3.1.2 Important Issues

Multiple Frame vs. Single Frame Processing

In Network-Distributed tracking architectures, multiple frame processing is still regarded as superior to single frame processing due to the fact that it can make data association decisions based on future data. So all the architectures we propose use Multiple Frame Assignment (MFA) as opposed to single frame processing. The basic idea of a Network MFA tracker is the same as a centralized MFA tracker which is explained in Chapter 2.

Consistent Air Picture

A consistent Air picture is one of the most important objectives in the design of a network-distributed tracking system. What consistent air picture means is that each platform should have exact the same set of tracks, (same association, same track states and covariance matrices).

Latent Data Problem

Because of the existence of multiple sensors on multiple platforms, various information such as measurements, track states and covariance, MRs, AMRs or AMMs are sent via the communication network. Random delays in communication are inevitable in real systems. So, a composite tracker must deal with the latent data problems and still achieve a consistent air picture.

Communication Load

Real communication networks are always band limited. There is a limit to the amount of information being passed between platforms at any time. Any design of a network architecture should reduce the communication load as much as possible. For example, passing track states and covariance matrices need 3-5 times more bandwidth than passing an MR. *Tracklets* [53, 31, 30] are also sometimes proposed to reduce communication loading.

Sensor Registration Bias

Sensor registration bias in a multiple sensor system needs be corrected so that multiple sensor measurements and/or tracks can be referenced to a common tracking coordinate system (frame). If uncorrected, registration errors can lead to large tracking errors and potentially to the formation of multiple tracks (ghosts) of the same target. There are three sources of registration biases: misalignments of the sensor measurement axes, electronic calibration bias errors, and sensor location errors. Techniques required to estimate and compensate these biases are highly dependent on conditions in the application.

Performance Evaluation and Metrics

Following are some of the important performance evaluation and metrics:

- Completeness History
  - Composite Completeness: proportion of real objects that should be tracked which are held as a declared composite track at each time in the scenario.
  - Relative Completeness: ratio of number of real objects that should be tracked which are held as declared composite tracks at each time in the

scenario to the maximum number of real objects that should be tracked which are held in any participating single sensor's track file.

- Composite track initiation time: time at which a particular object that should be tracked has a valid declared composite track.
- Track Continuity
  - Cumulative swaps of composite tracks: for real objects that should be tracked, the cumulative number of swaps (not counting breaks) of composite tracks for particular objects and averaged across all objects by time t into the scenario.
  - Cumulative broken composite tracks: for real objects that should be tracked, the cumulative number of breaks of composite tracks for particular objects and averaged across all objects by time t into the scenario.
- Ambiguity
  - Composite track redundant ratio: in a gated non-unique assignment, the number of declared composite tracks that are assignable to real objects that should be tracked, divided by the number of valid declared composite tracks.
  - Composite track spurious ratio: in a gated non-unique assignment, the number of declared composite tracks that are unassignable to real objects that should be tracked, divided by the number of valid declared composite tracks.
- Accuracy

- Composite track accuracy: as assessed over all Monte Carlo runs for a particular object that should be tracked, the root mean square error (RMSE) history in position, the RMSE history in velocity, the root sum squared average error (RSSAE) history in position, and the RSSAE history in velocity of the composite track assigned to that object compared to the truth states for that object.
- Composite track covariance consistency: as assessed over all Monte Carlo runs for a particular object that should be tracked, the mean normalized Chi-square statistic of the composite track assigned to that object.

- Cross-platform commonality history
  - Ratio of non-common composite track numbers: ratio of active composite track number that are different (additions or deletions) between pairs of composite tracking processors, divided by number of composite track numbers in the union of the two composite track files.
  - Composite track state estimate differences: the Euclidean differences between position and velocity state estimates of tracks held by pairs of composite tracking processors, for composite tracks with the same active composite track number.

- Communication data loading: the total amount of data that all the platforms send to the communication network.

3.2 Introduction to Some Tracking Fusion Architectures

3.2.1 Distributed Tracking with Central-Level Track Fusion

In distributed tracking with track fusion [5], platforms operate independently. Local data is processed by local trackers. The output is passed to some type of global processing entity. In this case, the data being passed to the global processor consists of a track state and the associated covariance matrix for each measurement update. The track fusion center performs a track-to-track association process to determine which state vectors from each of the platforms correspond to the same physical target. Having determined the likely association, the track fusion center collects the state vector and covariance matrices corresponding to common tracks from each platform and predicts each one to a common time. These predicted state vectors and predicted covariance matrices are then combined to produce a composite track state and covariance matrix for each track.

This approach has several disadvantages. The most significant disadvantage is the huge communication loading. Each local tracker on the platform passes track states and covariance matrices to the fusion center. In order to achieve a consistent air picture on all the platforms, the fusion center must broadcast track states and covariance matrices back to each platform. Another disadvantage is that the fusion center combines track states and covariance matrices that have a common process noise. The common process noise due to the common target dynamics observed by all the sensors makes combining all the states and state covariance matrices a sub-optimal approach, unless the cross-correlation between sensor observations is removed.

3.2.2 Distributed Track Fusion

In this approach, the state vector and associated covariance matrix for each tracker are passed from the tracker on each platform to a track fusion processor on

every other platform. The track fusion processor updates composite track states in an iterative fashion, as each update arrives.

This approach also suffers from the disadvantage of huge communication data loading and the sub-optimal fusion process to remove the cross correlation of the subsequent track states.

3.3 Proposed Network MFA Architectures

3.3.1 Communication Network Architecture

The communication network available to all trackers in all the proposed NTAs is fully connected. There is a direct path between any two platforms j and k. There is a random delay in the network, but there is no lost data. Figure 3.1 is an example of such a network with four platforms.

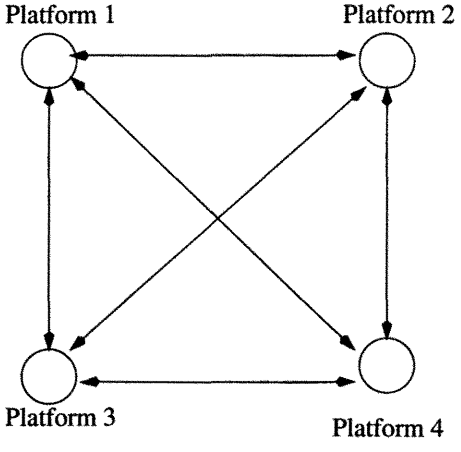


Figure 3.1: Communication Network

To simulate the communication network, and implement the different architectures we propose, we use the network architecture shown in Figure 3.2. There is a Communication/Control Unit (CCU) that simulates communications between the simulator and the network trackers. The CCU can read data from stdin, other file streams, or sockets. The data arrives in the form of messages, which contain a message header and a message body. The CCU extracts the message header,

which contains information identifying message types and platform IDs. Based on the message header, the CCU sends the message to the composite tracker on the corresponding platform (through sockets).

The CCU takes feedback from all the composite trackers. Feedback from the composite trackers can be used later to pass information that configures the network. For instance, a new tracker entering the network, or an existing tracker leaving the network requires a configuration change. Feedback can also be used for error control purposes. For example, if a communication link is down, then the CCU should stop sending observations to that particular tracker. Or, if the end of the input data stream is reached, then all the composite trackers in the network need to be shut down properly.

Each composite tracker handles its own communications with all the other platforms in the network. For different architectures we propose, the information sent is be very different, and differences will be discussed in detail in successive chapters.

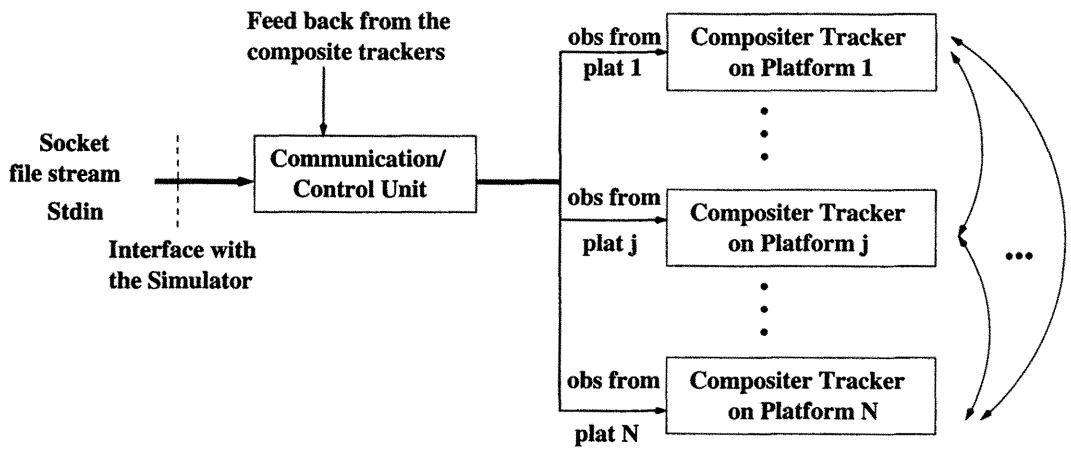


Figure 3.2: Overview of the Network MFA Architecture

3.3.2 Network MFA Centralized

In the network MFA centralized, each platform has a composite tracker which is similar to the one discussed in the centralized tracking architecture. The com-

posite tracker broadcasts all its local raw measurements to all the other platforms. Thus, every tracker has access to all the data. A centralized tracking algorithm (explained in Chapter 2) is used.

3.3.3 Network MFA on Local Data and Network Tracks

In this approach (Chapter 5), each platform has a composite tracker whose sliding window spans local frames only. Track initiation is based on local frames in the track initiation window only. Track extension decisions are based on network tracks and local frames in the track extension window. After the data association decisions are made, AMRs and newly initiated local tracks are broadcast to the network. The composite tracker uses the remote AMRs to update its network tracks and the remote, newly initiated tracks to update its track database. The messages being broadcast to the network are AMRs and initiating tracks.

3.3.4 Network MFA on All Data and Network Tracks

In this approach (Chapter 7), each platform has a composite tracker whose sliding window spans both local and remote frames. The composite tracker is allowed to make firm data association decisions on its local frames based on all the frames in the sliding window. It needs to wait for remote AMMs to fix the association between the network tracks and remote frames. The messages passed through the communication network include measurement reports, AMMs, and initiating tracks.

Chapter 4

INTRODUCTION TO NETWORK MFA CENTRALIZED

4.1 Overview of the Architecture

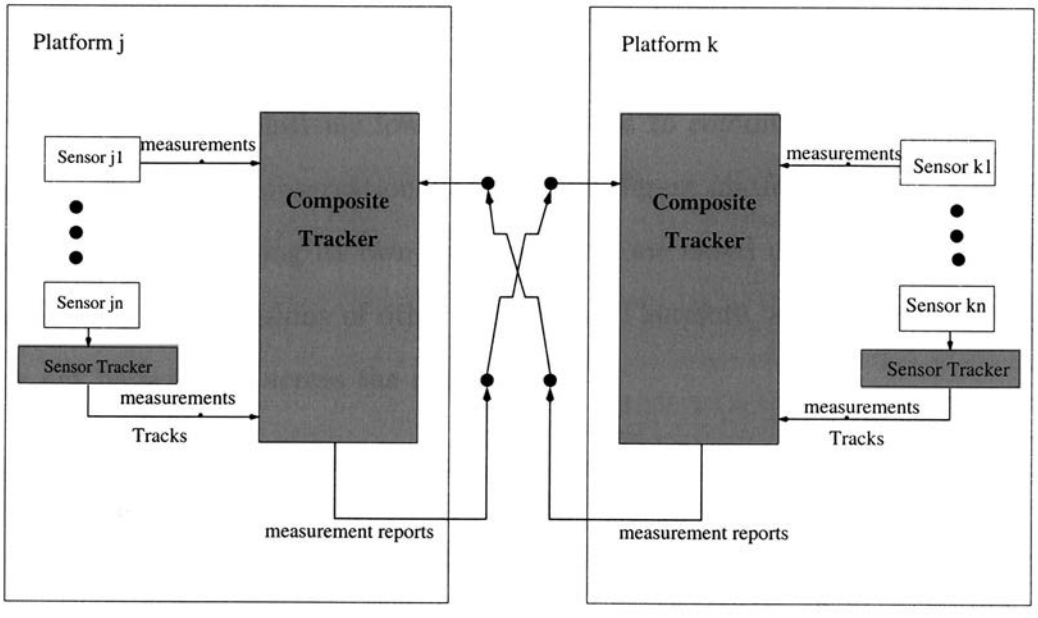


Figure 4.1: Network MFA Centralized

The network MFA centralized architecture places a centralized tracker on each platform. Raw observations generated by local sensors on the platform are fed into the local composite tracker. They are broadcast by the local composite tracker to all the other composite trackers in the network. As in real scenarios, all the trackers get local observations from the local sensors directly and remote observations

from remote platforms through the communication network. The communication network used is a fully connected network as is described in Chapter 3. Local observations are fed to the composite tracker through the CCU that interfaces with the simulator. Each composite tracker is responsible for broadcasting its local observations, as measurement reports, to all the other trackers in the network.

In this architecture, the single point failure problem is solved. If one or more composite trackers fail to perform correctly, or certain communication links are cut down, the composite trackers on other platforms are still capable of doing the tracking. However, the composite tracker may not have access to all remote measurement reports.

The communication network is used to broadcast measurement reports only. No composite track states are broadcast in this architecture. Thus the communication loading is relatively low. However, due to communication delays in the network, the order of observations arriving at different platforms varies. Each composite tracker is making its own tracking decisions based on the data it receives, regardless of the decisions of other platforms. Therefore, a consistent air picture may not be achieved across the network.

4.2 Implementation of A Network Composite Tracker

A network composite tracker (Figure 4.2) should be able to process observations (local and remote) as they arrive and perform the tracking tasks simultaneously. Since the arrival times of the messages (observations) are unpredictable, and they ought to be processed immediately, traditional sequential programming does not suffice. Multi-thread programming is used in implementing the network composite tracker. The interacting tasks are implemented as multiple threads within a single process (the tracker). They share a common environment such as the same static and dynamic storage. However, in order to protect multiple

threads from accessing the shared resource (the same piece of data) at the same time, a thread synchronization mechanism (*mutex* [79]) is used to provide mutually exclusive access to a shared resource.

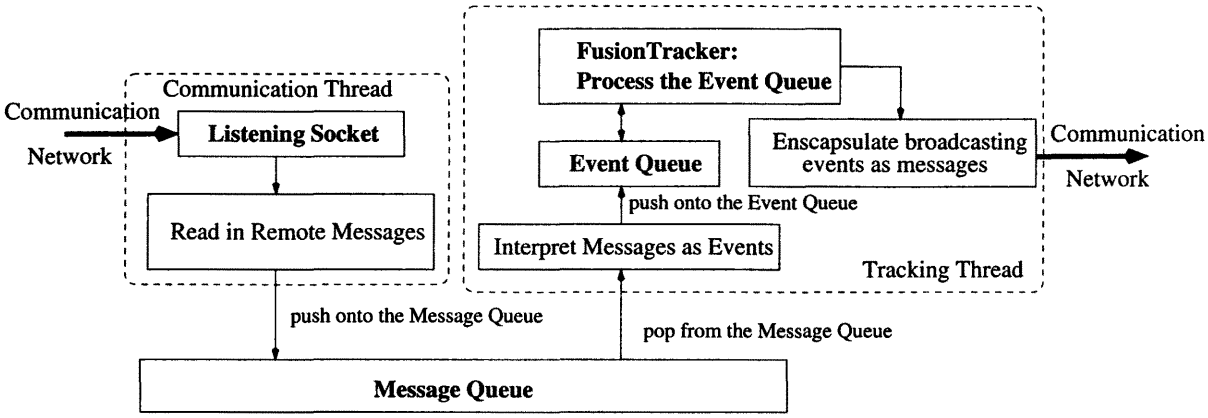


Figure 4.2: Network Composite Tracker

There are two threads on each composite tracker in the network. One of the threads handles input messages. It is called the Communication Thread. The other one is the major process thread and is called the Tracking Thread. Each tracker has a global message queue, which is shared by both threads. The message queue is protected by a *mutex* so that it can not be accessed by two threads at the same time.

4.2.1 Communication Thread

The Communication Thread handles all the incoming messages and pushes them onto the message queue as soon as they arrive.

Generally speaking, there are three types of incoming messages. The first type are local sensor messages, such as local observations and local sensor tracks generated by local sensors. The local sensor messages are now sent to the composite tracker by the communication/control unit that interfaces with the simulator. The second type are the metrics scoring requests, which also come from the CCU. The

third type are remote messages, which come from remote platforms. In the centralized network MFA architecture, the remote messages are remote measurement reports.

All types of messages are received through the listening socket [73] assigned to the composite tracker. The listening socket is opened (passive open) in the initialization step of the program and remains open during the entire simulation. The Communication Thread continually waits for remote connections through the listening socket. Whenever a connection occurs (either from the communication/control unit or from some remote platform), an active open of the socket is issued and a temporary connection is established. The thread then reads in the message and closes the connection. However, the socket can only buffer a finite number of connection requests, so if too many connection requests to the fusion node happen at the same time, some requests may be lost. (This may cause lost data in the communication network, but based on the basic assumptions, we will not consider this problem now.)

After the incoming messages are received from the listening socket, the communication thread pushes them onto the global message queue.

4.2.2 Tracking Thread

As discussed in the previous section, the incoming messages are pushed onto the global message queue by the communication thread, waiting to be processed by the Tracking Thread. The Tracking Thread takes messages out of the message queue and translates them into events. Specific operations are defined for all the possible events. The event queue stored inside the tracking thread drives the composite tracker. Therefore, in some sense, the tracking thread is equivalent to the event-driven tracker that applies the MFA algorithm.

The major difference is that there are broadcasting events where the tracker needs to broadcast messages to all the other network trackers through sockets.

In the centralized architecture, the messages broadcast are only local observation events.

4.3 Ideal Case

Consider the ideal case, where there is no delay in the communication network, and no processing delay in the composite tracker. As soon as a local observation is fed into the composite tracker, it generates a measurement report event which is then broadcast to the network immediately. The time difference between observations arriving on local and remote platforms is negligible. Thus, all composite trackers should receive the same observations in the same order. Conceptually, a consistent air picture should be achieved.

In order to simulate the ideal case, the CCU needs to be modified accordingly. Instead of sending the observations to the corresponding platform only, the CCU broadcasts observations to all platforms. Thus, there is no need for each tracker to broadcast its local observations to other platforms. In this way, all the composite trackers get the same data in exactly the same order.

4.4 Practical Case

A more realistic centralized type architecture takes into account communication delays and processing delays. There are delays inherent in socket communications. We use these delays to model communication delays between the platforms. As shown in Figure 4.2, messages are pushed onto the global message queue by the Communication Thread. The Tracking Thread keeps taking messages out from the message queue, interprets them as events and pushes them onto the event queue. According to the network architecture we proposed in Chapter 3, each composite tracker gets its local observations from the CCU which interfaces with the simulator. The Tracking Thread interprets them as observation events. While an observation event is being processed, it is broadcast to all other platforms.

The broadcasting step is done sequentially. To broadcast an observation event, it is transformed into a character string and sent to the listening socket of the tracker. This is done sequentially for each remote platform. In the future, the broadcasting step should be done in parallel. A new thread can be started to take over the task of sending the observation to one tracker.

On each composite tracker, the local observations arrive earlier than the remote measurement reports. Each tracker builds its own frames of data based on the observations received and processes them, regardless of the other composite trackers in the network. a consistent air picture may not be achieved.

In order to achieve a consistent air picture, different architectures and different rules are introduced to moderate the problem. In Chapter 5, the network architecture on local data and network tracks is explained. The network architecture on all data and network tracks is discussed in Chapter 7.

Chapter 5

PROPOSED NETWORK TRACKING ARCHITECTURE I: MFA ON LOCAL DATA AND NETWORK TRACKS

5.1 Overview of Architecture

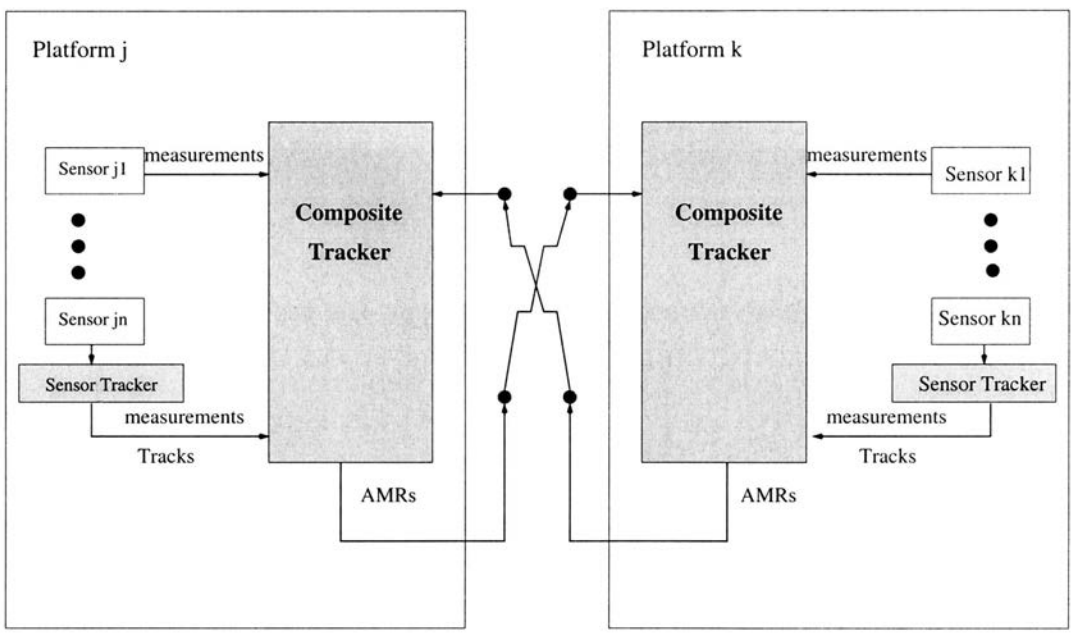


Figure 5.1: Network MFA on Local Data and Network Tracks

The network MFA architecture on local data and network tracks is explained in Figure 5.1. The composite tracker on each platform only has access to its local data. On each platform, there are multiple sensors, some of which may have their own sensor trackers. Raw measurements and sensor tracks are sent to the

composite tracker on the local platform to do the tracking. The double pane sliding window is made up of local frames only. The data association decisions are made based on scores of network tracks extending to the local frames in the window. Data association results are broadcast to all platforms in the network in the form of Associated Measurement Reports(AMRs) and local new tracks, which are used to update the track database on the remote platforms. The goal is that all composite trackers have the same set of network tracks.

5.1.1 Basic Assumptions

The architecture is based on the following assumptions:

- The communication network is fully connected. There are delays in the network, but there will be no lost data. (Lost data is discussed in Chapter 9.)
- Each composite tracker will broadcast to all the other platforms its firm (irrevocable) decisions on its local frames.
- Each composite tracker will update its network tracks according to the remote data association decisions.

5.1.2 What's Being Sent Out as Fixed DA Decisions

Data association decisions are assembled together as **AMRFrames**, which include:

- AMRs (Associated Measurement Reports): Each AMR has four parts, a measurement report, an associated track ID (network track ID), an AMRFrame ID that indicates which AMRFrame it is in, and a platform ID.
- Newly initiating tracks: Each new track should contain a Kalman state vector and covariance matrix with a time tag, a corresponding network track ID,

an AMRFrame ID, and a platform ID. However, it now contains the local observation history associated with it, which can be used by the remote side to reconstruct the track.

5.1.3 What's Being Done After Receiving Remote DA Decisions

Remote data association decisions are received as AMRs and new tracks, the remote composite tracker puts the pieces into the correct AMRFrames based on AMRFrame IDs and platform IDs. When a remote AMRFrame is ready to be processed, the composite tracker processes remote new tracks first, and then the AMRs.

Track-to-track correlation is performed between the remote new tracks and the existing tracks. If a remote new track correlates with one of the existing tracks, track fusion is performed to fuse the information of two tracks, and the TrackTree is updated. If the remote new track is actually a new one, then it is inserted as a new node in the TrackTree, and extended to the local frames in the extension window.

For AMRs, it finds the corresponding network track and updates it using the remote measurement. Then the network track is re-extended to all frames in the extension Window. The sub-tree structure might change due to the contribution of each incoming AMR.

5.2 Algorithm Overview

5.2.1 Ideal Case

The algorithm of the network MFA on local data and network tracks in the ideal case can be illustrated in Figure 5.2. There are three platforms i, j and k shown in the graph. A track extension window of size 2 is chosen for all three platforms.

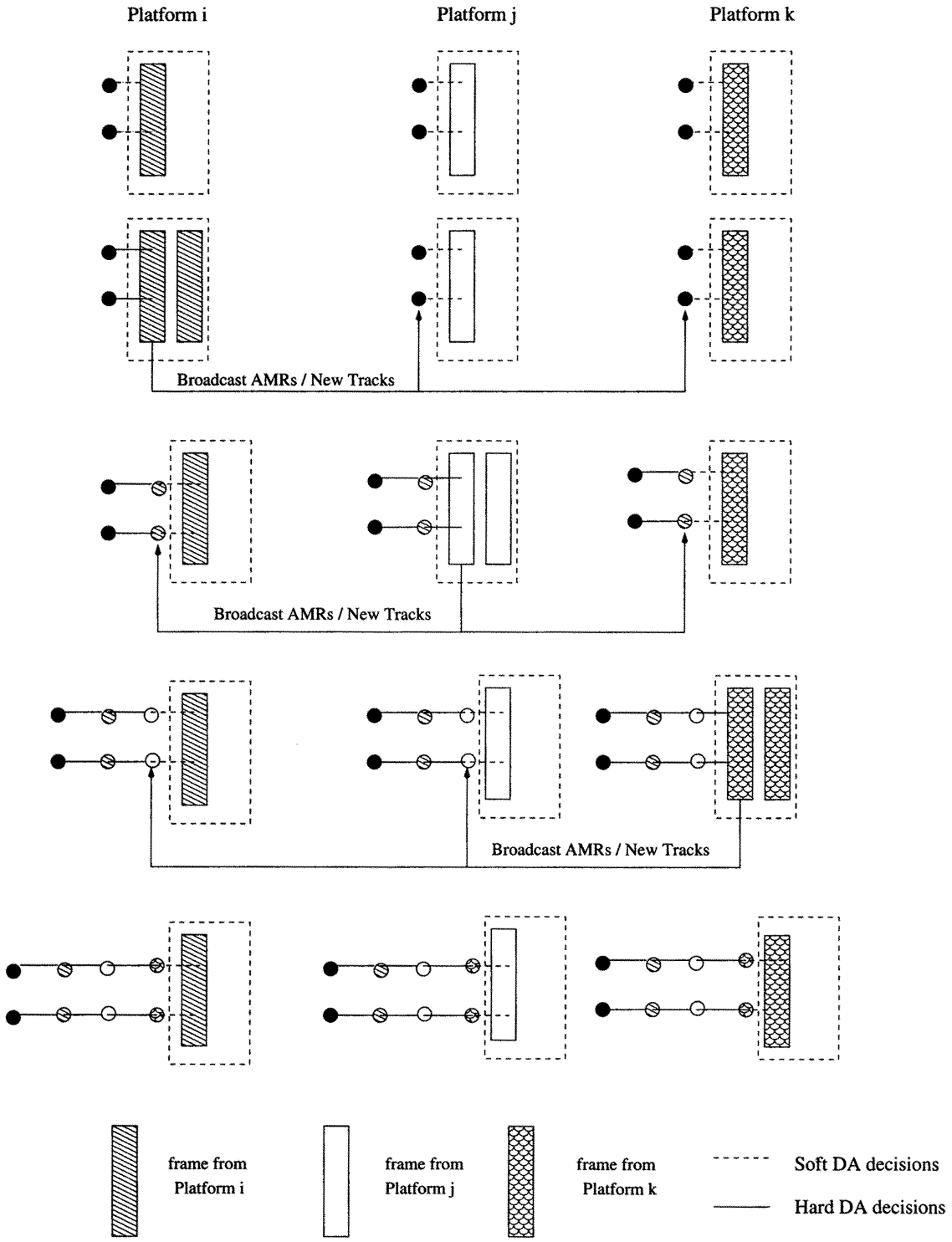


Figure 5.2: Ideal Case for Network MFA on Local Data and Network Tracks

The window on each platform contains only local frames, and there is only one local frame in each of the track extension window. We start with the same set of network tracks on platform i, j and k, which are represented by dark black dots in front of the track extension windows. Suppose another local frame on platform i is ready to be processed, then the frame is inserted into the extension window, and the TrackTree is extended to the frame. After setting up the data association problem, platform i makes irrevocable data association decisions on the first frame in its extension window based on network tracks and the two local frames. The data association decisions are broadcast as an AMRFrame to platform j and k. The composite tracker on platform i prunes its TrackTree and moves the sliding window forward. The composite trackers on both platform j and k update their network tracks based on the AMRFrame received and rebuild the TrackTrees in their extension window.

After all composite trackers finish processing the local frame from platform i, the network tracks are still the same in the sense that they all have the same observation history associated with each track. Then, another frame on platform j is ready to be processed. The composite tracker on platform j inserts it into its extension window, updates the TrackTree and makes its data association decisions and broadcasts them to platform i and k. The moving window on platform j is moved forward after that. Platform i and k update their network tracks and the TrackTrees according to the AMRFrame received.

Similarly, when a local frame from platform k is ready, the composite tracker on platform k makes its firm data association decisions and broadcasts them to platform i and j.

In the ideal case, the composite trackers across the network are well synchronized. By well synchronized, it means the following two conditions are satisfied. (1) The frames get ready to be processed alternatively on different platforms in

time order. For the example in Figure 5.2, let $t_j(p)$ denote the time at which p^{th} frame on platform j is ready, then

$$\cdots < t_i(p) < t_j(p) < t_k(p) < t_i(p+1) < t_j(p+1) < t_k(p+1) < \cdots$$

(2) The time interval between which two successive frames are ready is big enough for the local composite tracker to make irrevocable data association decisions and broadcast them to the remote platforms, for the AMRFrame be transmitted through the network, and for the remote composite tracker to process the corresponding AMRFrame.

It can be concluded that:

- Ideally, the network tracks are exactly the same across the platforms in the sense that they have the same observation history.
- The data association decisions on each platform are based on local frames in the extension window and network tracks which includes local and remote information.

5.2.2 More Realistic Case: Delays

In real life situations, there is no centralized control that synchronizes the process of all the composite trackers, and delays in the transmission of data are random. The order of the incoming AMRFrames is unpredictable. Thus, the composite tracker is designed to be event driven and is capable of handling out-of-order AMRFrames.

The composite tracker should try to process the AMRFrames according to the following rules:

- In the initialization phase, if the sliding window has less than $(M - N)$ frames, then the remote AMRFrame needs to be buffered.

- If the parameter *bufferAMRFrames* is set to be *true* by the user, then the composite tracker processes the local frames in the extension window and remote AMRFrames in a timely order. If all observations in the first frame in the extension window have time tags earlier than the AMRs in the remote AMRFrame, then the composite tracker buffers the remote AMRFrame and processes the local frame first. However, if the AMRs have time tags earlier than the observations in the first frame in the extension window, then the composite tracker updates its network tracks with the remote AMRFrame first.

- If the parameter *bufferAMRFrames* is set to be *false* by the user, the composite tracker processes the remote AMRFrames as soon as they are ready.

If the AMRFrames the composite tracker processes are out of order, then for each track, the observations used to update the network tracks may be out-of-order, which needs a lot of refiltering and rescoring. In the worst case, the observation may even fall out of the refiltering window to cause a negative time update. Thus, though the observation histories for the same network track are the same, the track states might be different due to negative time updates.

Sometimes, the composite tracker might receive an AMR for a track which does not exist in the local TrackTree yet. This can be caused by out of order AMRFrames from the same platform, or it can be caused by abnormalities in the communication network. Suppose there are three platforms in the network, denoted by A, B and C, and that the AMRFrames from the same platform are in the correct time order. However, the communication links between A-B and A-C are very efficient, while the communication link between B-C is heavily loaded. Thus, it takes significantly longer to communicate between B and C. Suppose the composite tracker on platform B broadcasts an AMRFrame (denoted by AMR_B)

that contains a new track T_B, the tracker on platform A processes the AMR_B after it arrives, and generates its own AMRFrame (AMR_A), which contains an AMR for track T_B. Suppose for the tracker on platform C, the AMRFrame AMR_A arrives sooner than AMR_B, then the tracker needs to handle the AMR for the non-existing track T_B.

There are two solutions to the above problem. One is to discard the AMR if the corresponding network track does not exist. This is easy to implement, but the consistent air picture may not be achieved any more. The other solution is to buffer the AMRs for non-existing tracks, and process them when the corresponding tracks arrive.

5.3 Even-Driven Overview

5.3.1 Modified Event Manager

The composite tracker is an event-driven software package. However, due to the fact that it needs to buffer AMRFrames (which are interpreted as AMRFrame events), the event manager needs to be modified.

There are now two event queues in the event manager, which are called as the primary event queue and the secondary event queue respectively. All events are posted onto the primary event queue waiting to be processed. If the event needs to be buffered, then it is moved from the primary queue to the secondary event queue. When the composite tracker processes all the primary events, it keeps processing the primary events until the queue is empty. However, when processing all the secondary events, the composite tracker goes through the queue again and again to process all the events that can be processed. It stops when the queue is empty or when no more events can be processed.

5.3.2 Introduction to Events

Adaptive Events

1. Observation Event

   An observation event contains a local observation, it is processed by the FrameBuilder to be put into proper frames. The frameBuilder builds sensorFrames according to sensor types and links the non-overlapping sensorFrames into frames as discussed in the centralized architecture.

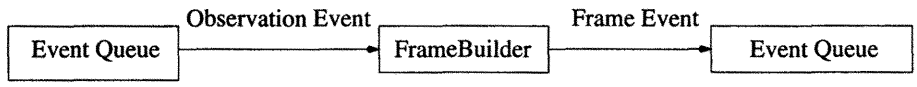


Figure 5.3: Observation Event

2. EOF Event

   At the end of each Monte Carlo run, an EOF event is generated. When the frameBuilder encounters an EOF event, it goes through its list of unfinished frames, and post them as frame events.

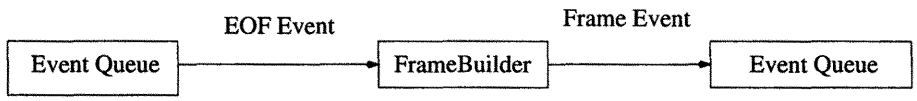


Figure 5.4: EOF Event

Data Association Decisions Events

1. Broadcasting Events

   The composite tracker on each platform builds the sliding window using only local frames. As discussed in the centralized architecture, whenever a frame event comes, the tracker extends its TrackTree to the frame and sets up the data association problem. The data association decisions on the first frame in the extension window are irrevocable and are broadcast to all other platforms. There are three types of data association decisions events that need to be broadcast.

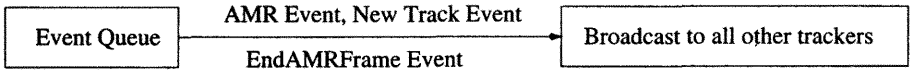


Figure 5.5: Broadcasting Events

- AMR Event

  An AMR event consists of four parts: an AMRFrame ID, a platform ID, a network track ID and the corresponding measurement report.

- New Track Event

  A new track event has an AMRFrame ID, a platform ID, a network track ID and an observation list. The length of the observation list is no less than *initLength* which is a user definable parameter that specifies the minimum number of observations required to initiate a new track. Thus, all estimation, filtering and scoring information for the new track can be reconstructed based on the observation list.

- End of AMR Frame Event

  An end of AMR Frame event consists of an AMRFrame ID, a platform ID and the number of data association decisions included in the AMRFrame which is the sum of AMR events and new track events in the AMRFrame. The End of AMR Frame event is used to help decide if AMRFrames are ready or not on the remote side.

2. Remote Events

Remote data association decisions events are events sent to the composite tracker from remote platforms. They are irrevocable data association decisions made by the remote composite trackers. Based on the AMRFrame ID and the platform ID, the AMRFrameBuilder (an event listener that listens to remote DA decision events) puts them into the correct AMRFrames.

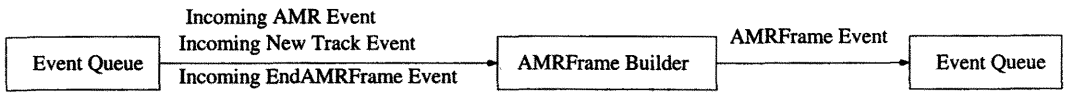


Figure 5.6: Remote Events

- Incoming AMR Event

  As discussed in the previous section, an incoming AMR event includes an AMRFrame ID, a platform ID, a network track ID and a corresponding measurement report.

- Incoming New Track Event

  The incoming new track event includes an AMRFrame ID, a platform ID, a network track ID and an observation list which consists of observations from the local platform only. The observation list is used to rebuild the track.

- Incoming End AMR Frame Event

  The incoming end AMR Frame Event consists of an AMRFrame ID, a platform ID and the number of DA decisions in that AMRFrame. The incoming end AMR frame event can be regarded as notifying the AMRFrameBuilder the number of data association decisions in the AMRFrame, so the AMRFrameBuilder can check if the AMRFrame is full and ready to be processed.

3. Output Events

Output events are events processed by outputFormatters, to be written in the form that can be recognized by some visualization tool. There are false alarm events, track initiation events, track update events and track termination events.

4. Track Metrics Events

   Track metrics events are processed by the Metrics outputFormatter. Track states are predicted to the required time and transformed to the required coordinate systems.

Metrics Scoring Event

Metrics scoring events are used to evaluate the performance of the composite trackers. Similar to the centralized architecture, the composite tracker forks a child process to handle the metrics scoring requests. The user can set the parameter *predictFrom* to be (1) "hard", in which case the track states are reported based on the most recent firm decisions; (2) "soft", in which case the track states are reported based on the most recent soft decisions; and (3) "softPlus", in which case the track states reported contain all the available information (local and remote).

All the available information for the composite tracker includes the network tracks, local frames that have already been added into the window, the unfinished local frames that are in the frameBuilder, and the unfinished AMRFrames that are in the AMRFrameBuilder. When the parameter *predictFrom* is set to be "softPlus", the metrics event collects all the unfinished frames in the frameBuilder and all the unfinished AMRFrames in the AMRFrameBuilder. The sliding window processes the metrics event by (1) updating its network tracks and the TrackTree with the unfinished AMRFrames, (2) appending all the unfinished local frames to the right end of the window, (3) extending the TrackTree to those frames, (4) setting up and solving a $(M + p)$ dimensional assignment problem, where p is the number of unfinished frames, and (5) firing track metrics events on the solution tracks.

Kernel Events

Kernel events are events that are processed by the sliding window.

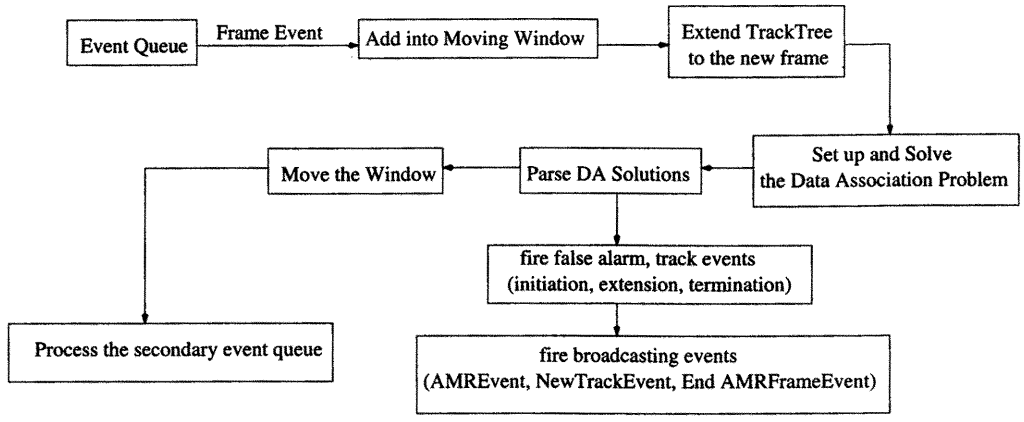


Figure 5.7: Frame Event

- Frame Event

  The way the frame event is processed is exactly the same as that of the centralized architecture. The difference is that when parsing the data association solutions, the window generates broadcasting events (AMRs, new tracks, and end AMRFrame events) based on the irrevocable decisions. After moving the window, it needs to process secondary event queue (which are buffered AMRFrame events) and process all the AMRFrames that can be processed.

- AMRFrame Event

  As discussed above, an AMRFrame is made up of remote data association decisions (AMRs and new tracks). The window uses the AMR frame to update its network tracks to include information from remote platforms. However, as is discussed in the previous section, if the AMRFrame needs to be buffered instead of being processed immediately, the AMRFrame event is moved to the secondary event queue.

- Close Event

  A close event is generated at the end of each Monte Carlo run. The size of the sliding window shrinks to zero by shifting the leftmost frame out.

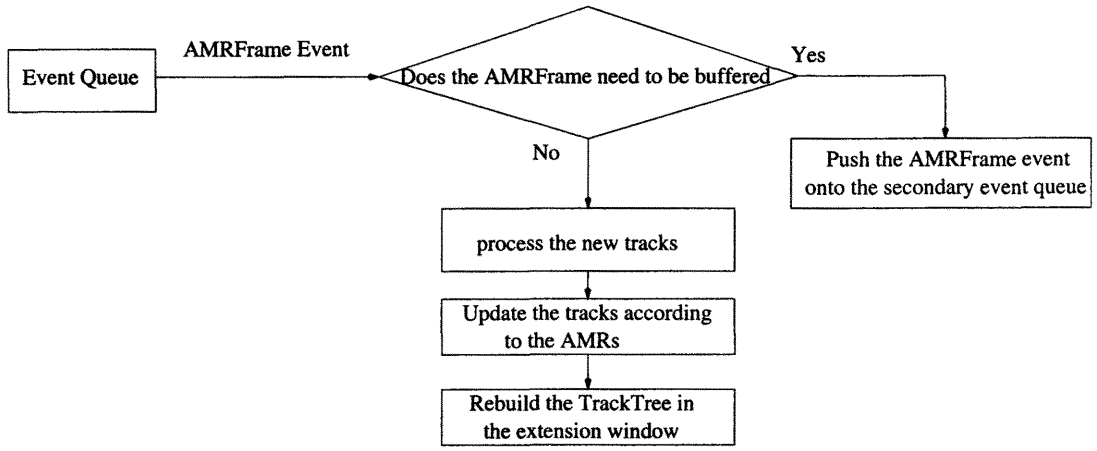


Figure 5.8: AMRFrame Event

5.4 Major Issues

5.4.1 Broadcasting AMRFrames

An AMRFrame is a group of irrevocable data association decisions on a local frame of observations. It includes new tracks and AMRs. When a local frame is ready and is inserted into the window, the composite tracker sets up and solves the data association problem (based on network tracks and local frames in the extension window). Then it parses the data association solutions and prunes the TrackTree. The decisions on the first frame in the extension window are irrevocable, and are broadcast to all other platforms as new tracks and AMRs, which are discussed in detail in the following subsections.

New Tracks

For a M/N window (track initiation window size of M and extension window size of N), the data association solutions can be divided into the following categories:

- False alarm arcs of the form $(0, \cdots, 0, i_{k_j}, 0, \cdots, 0)$, where $i_{k_j} \neq 0$, and $j = 0, \cdots, M-1$.

- Updating arcs of the form $(0, \cdots, 0, -T_{k_l}, i_{k_{M-N}}, \cdots, i_{M-1})$, where there should be at least one non-zero i_{k_j}, for $j = (M-N), \cdots, (M-1)$.

- Terminating arcs of the form $(0, \cdots, 0, -T_{k_l}, 0, \cdots, 0)$.

- Initiating arcs of the form $(i_{k_0}, \cdots, i_{k_{M-N}}, i_{k_{M-N+1}}, \cdots, i_{k_{M-1}})$, where there are at least two non-zero indices.

For initiating arcs of the form $(i_{k_0}, \cdots, i_{k_{M-N}}, i_{k_{M-N+1}}, \cdots, i_{k_{M-1}})$, the composite tracker assigns a local track ID and promotes it to the local track database, provided that the track string $TS_{i_{k_0},\cdots,i_{k_{M-N}}}$ satisfies the following requirements.

1. A tracking filter has been started and it produces a negative score.

2. It has at least *initLength* number of observations associated with it, where *initLength* is a user definable parameter that specifies the minimum number of observations required to initiate a new track.

3. The track state does not correlate with any of the existing remote tracks, who have no corresponding local tracks.

When a local track initiates, the associations in track string $TS_{i_{k_0},\cdots,i_{k_{M-N}}}$ are fixed and the new track event is generated to broadcast the new track to the network.

AMRs

Associated measurement reports are generated from the solution arcs of of the form $(0, \cdots, 0, -T_{k_l}, i_{k_{M-N}}, \cdots, i_{M-1})$, if $i_{k_{M-N}} \neq 0$. A broadcasting AMR event is generated, which consists of an AMR Frame ID (frame ID of $f_{k_{M-N}}$), a platform ID, a network track ID associated with track T_{k_l}, and the observation $z^{i_{k_{M-N}}}_{f_{k_{M-N}}}$.

End AMRFrame

The End AMRFrame message consists of an AMRFrame ID, a platform ID and the total number of data association decisions in that AMRFrame, which is just the summation of the number of new tracks reported and the number of AMRs broadcast.

5.4.2 Building AMR Frames

AMRFrames are built based on remote input events (incoming new track event, incoming AMR event and incoming end AMRFrame event). For each composite tracker, there is an event listener called AMRFrameBuilder which listens to remote input events and builds AMRFrames.

As discussed in Section 5.3.2, each remote input event has an AMRFrame ID and a platform ID associated with it. The *AMRFrameBuilder* puts pieces of events into the correct AMRFrames based on both the frame ID and the platform ID. The incoming end AMRFrame event tells the AMRFrameBuilder how many pieces of information the corresponding AMRFrame has. Thus the AMRFrameBuilder is able to tell if the AMR Frame is full (ready to be processed) by comparing the number of incoming events received and the number expected.

5.4.3 Processing AMRFrames

The sliding window listens to AMRFrame events. As explained before, an AMRFrame includes a list of new tracks and a list of AMRs.

Based on the discussions of the sliding window in the centralized architecture, the size of the sliding window between events is $(M-1)$. It is waiting for another frame of data to be added. Therefore, when an AMRFrame event is being processed, the window is not full.

Due to the fact that AMRFrames are irrevocable data association decisions, all corresponding operations are on the $(M-N)^{-th}$ branch in the TrackTree.

Thus, in the composite tracker initialization step, if the window size is too small (smaller than $(M - N)$), the AMRFrame events have to be buffered (moved onto the secondary event queue) instead of being processed immediately.

New Tracks

For incoming remote new tracks, track-to-track correlation is carried out between the remote new tracks and the existing tracks in the local track database (see Chapter 6 for detailed discussions). If the remote track does not correlate with any of the existing tracks, it is inserted as a new node into the TrackTree. Otherwise, it is fused with the existing track.

The TrackTree for a 5/3 sliding window is used for demonstration of the algorithm. The window is of size 4 when the AMRFrame is being processed. The dashed line and box represents that the window is waiting for another frame of data.

- Example of inserting a new track

  Figure 5.9 illustrates the steps when an incoming new track is regarded as emanating from a new target after the track-to-track correlation. In order to insert the track in the correct branch of the TrackTree, an integer sequence of size $(M - N)$ is formed $((0, \cdots, 0, -T_{new}) = (0 - 2))$, which serves as an address in the TrackTree. $T_{new} = 2$ is the network track ID for the incoming new track.

  Then, the remote new track needs to be extended to the frames that are in the extension window in a the order of $f_{k_{M-N+1}}, \cdots, f_{k_{M-2}}$. As is shown in Figure 5.9, it is first extended to frame f_{k_2} in the window by building the child $(0 - 210)$. It is later extended to frame f_{k_3} in the window, building child $(0 - 201)$ and $(0 - 211)$.

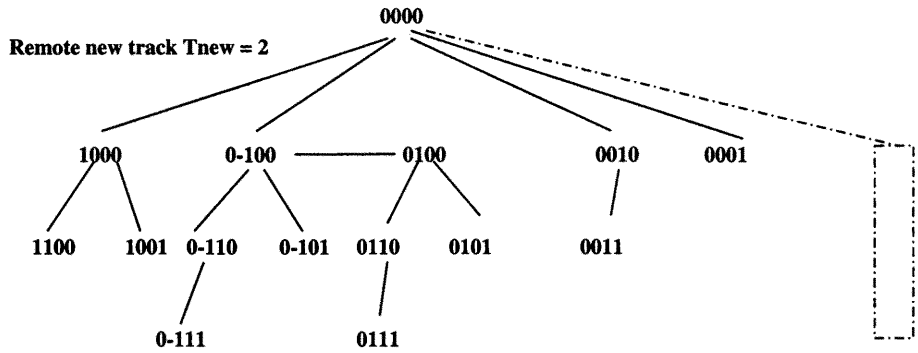
- Example of fusing two tracks emanating from the same target

  Figure 5.10 illustrates the steps when an incoming new track is regarded as emanating from the same target as one of the existing tracks. Suppose the remote new track $T_{new} = 3$ correlates with the local track $T_{local} = 2$.

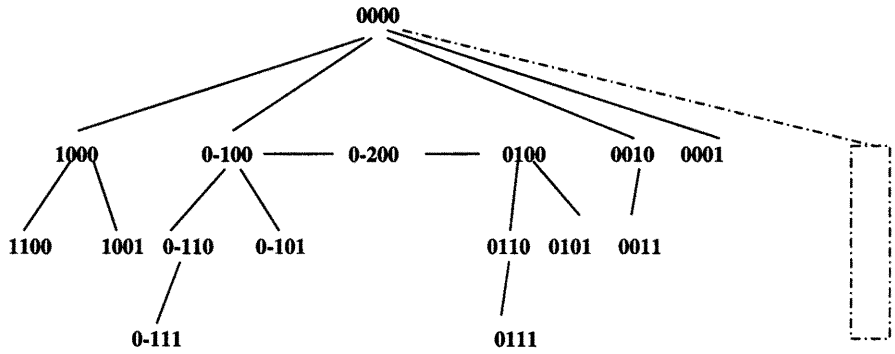
  After the fusing step, the track contains additional information. Thus, the track needs to be re-extended to the frames that are in the extension window $(f_{k_{M-N}}, \cdots, f_{k_{M-2}})$ in order. The sub-tree structure maybe changed due to the additional information of the incoming track, so the sub-tree is deleted first and then rebuilt. As is shown in the Figure 5.10, it is first extended to frame f_{k_2} in the window by building the child $(0 - 210)$. It is then extended to the frame f_{k_3} in the window, building child $(0 - 201)$ and $(0 - 211)$.

AMRs

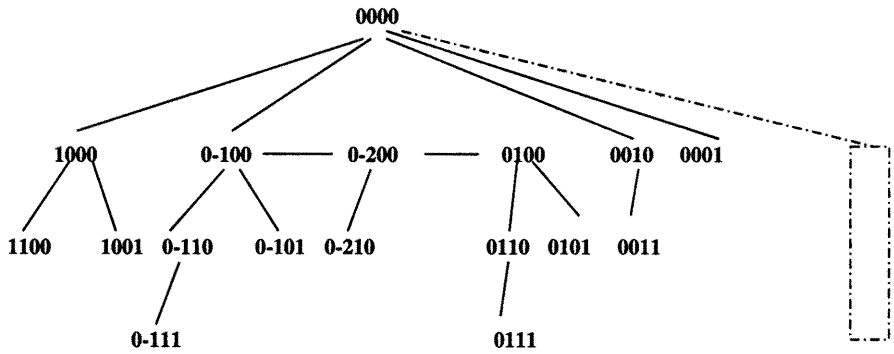
For remote AMRs, Figure 5.11 is an example of how to process them. For an AMR pair of (T_n, z), where T_n is the network track ID, an integer sequence $(0, \cdots, 0, -T_l)$ is formed to search the TrackTree for the track, where T_l is the corresponding local track ID. Then track T_l is updated with the remote observation z. The entire sub-tree is deleted first and then rebuilt by extending to the frames in the extension window.



Step I: Insert new Node (0-200) in the TrackTree



Step II: Update the Node (0-200) with the 3rd frame in the window



Step III: Update the Node (0-200) with the 4th frame in the window

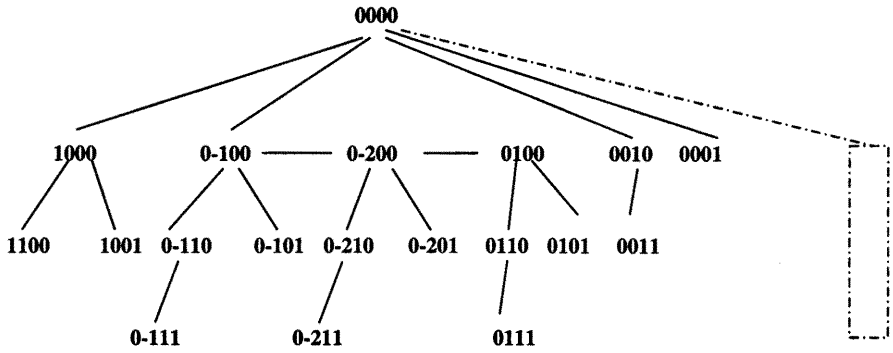


Figure 5.9: Example: Insert A Remote New Track into the TrackTree

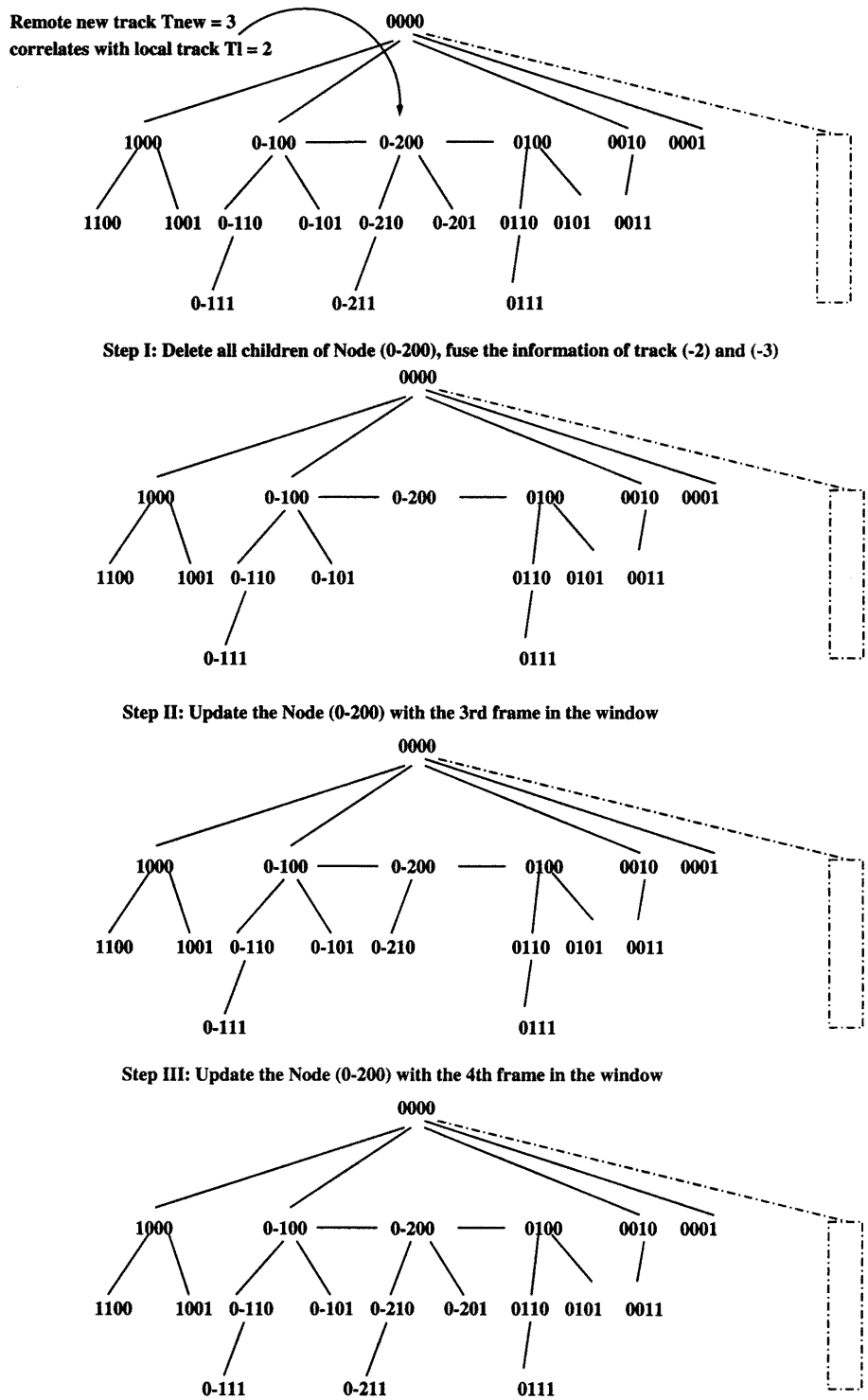
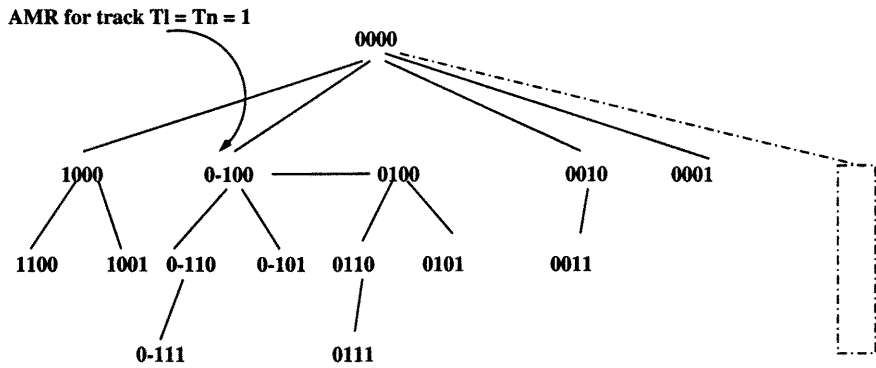
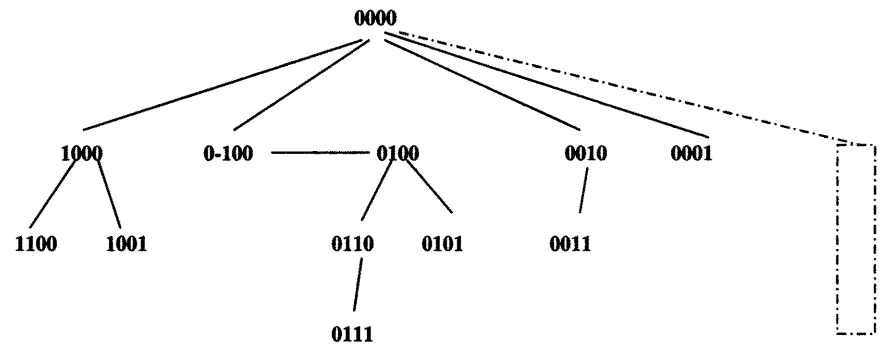


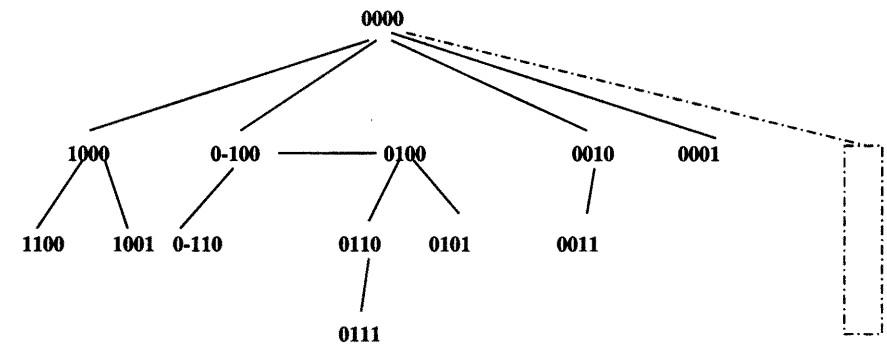
Figure 5.10: Example: Fuse A Remote New Track with An Existing Track



Step I: Delete all children of Node (0-100), update the Node (0-100) with the AMR



Step II: Update the Node (0-100) with the 3rd frame in the window



Step III: Update the Node (0-100) with the 4th frame in the window

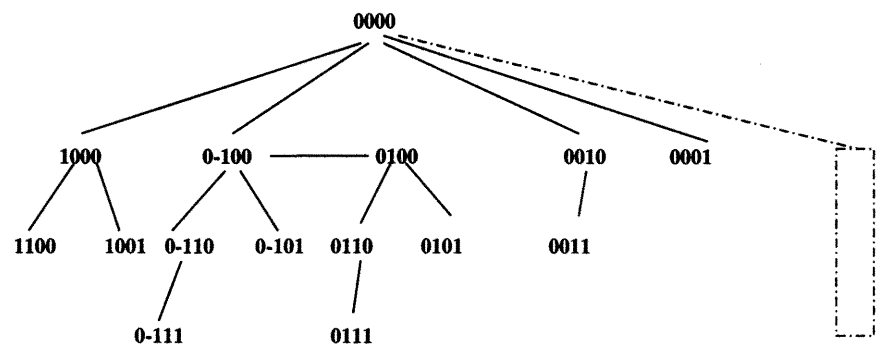


Figure 5.11: Example: Update with An AMR

Chapter 6

NETWORK MFA ON LOCAL DATA AND NETWORK TRACKS: TRACK INITIATION

6.1 Problem Description

The architecture discussed in Chapter 5 is network MFA on local data and network tracks, in which the sliding window is made up of local frames only, and the network tracks include information from all the remote platforms as well. Each composite tracker broadcasts to all the other platforms its AMRFrames (irrevocable data association decisions on local frames).

The goal is that each composite tracker has the same track database. Each platform should have the consistent air picture. The tracks on different platforms that emanate from the same target may have different local track IDs, but should have the same network track ID. And the state estimate difference between the platforms should be as small as possible.

In order to achieve a consistent air picture, each composite tracker is capable of initiating new tracks based on local data only. For the remote incoming new tracks, the composite tracker performs track-to-track correlation to see if it emanates from the same target as one of the existing tracks. If this is the case, then two tracks are fused. Otherwise, the new track is inserted into the local track database. Similarly, the composite tracker broadcasts its track update as AMRs. When a remote AMR arrives, the tracker updates the corresponding network track with the remote observation.

6.2 Track Numbering Schemes

6.2.1 Local Track ID Bank

As discussed above, each composite tracker can initiate new tracks based on the local frames in the initiation window. For each newly initiated tracks, a unique local track ID is assigned to the track. The local track ID is used by the composite tracker to represent the track locally. It is used to form an integer sequence which serves as an address in the TrackTree. To locate a network track, the composite tracker needs to know its local track ID to search the TrackTree.

To achieve this goal, composite trackers have disjoint local track ID banks that they use to assign to the local tracks. Let TID_i denote the local track ID bank for platform i, then

$$TID_i \cap TID_j = \emptyset, \quad i \neq j \tag{6.1}$$

For each newly initiated local track, the composite tracker assigns it its next available track ID from the local track ID bank. It is then guaranteed that the same track ID will not be assigned to two different tracks.

Suppose there are at most M composite tracks in the network, then an easy example for the track ID bank is:

$$TID_i = \{j \times M + i, \; for \; j = 1, 2, \cdots\} \tag{6.2}$$

6.2.2 Network Track ID

There might be multiple local track IDs assigned to tracks emanating from the same target across the network. However, they should have the same network track ID. So that each target has one unique network track ID assigned to it.

Suppose the collection of local track IDs that represent the same target can be denoted as

$$\{ t_j^{(i)} \in TID_i, \; for \; i \in I, \; I \subset \{1, 2, \cdots, M\} \},$$

where I is some subset of platforms that track the target. Then the network track ID can be chosen as

$$nt_j = \min \{ t_j^{(i)} \in TID_i, \; for \; i \in I, \; I \subset \{1, 2, \cdots, M\} \}. \tag{6.3}$$

Whenever a new track is initiated locally, the network track ID is set to be the same as the local track ID and is later updated according to rule explained in equation (6.3).

6.2.3 Track ID Map

When broadcasting data association decisions, the network track ID is used. For remote platforms, it ought to know which local track ID the network track is corresponding to. A track ID map which maps network track ID to local track ID is used to implement the correspondence.

For platform i, the track ID Map is denoted as Map_{tid}^i. At the initialization step for the composite tracker, $Map_{tid}^i = \emptyset$. In the following cases, a new element is inserted into the track ID map if

1. A local new track $t_j^i \in TID_i$ initiates,

$$Map_{tid}^i = Map_{tid}^i \cup \{t_j^i \longrightarrow t_j^i\}. \tag{6.4}$$

2. A remote new track nt_j needs to be inserted into the TrackTree.

   Suppose after performing track-to-track correlation, the conclusion is made that the remote new track nt_j is actually a new one, then it is inserted into the TrackTree, and

$$Map_{tid}^i = Map_{tid}^i \cup \{nt_j \longrightarrow nt_j\}. \tag{6.5}$$

3. A remote new track nt_k needs to be fused with an existing one.

   Suppose after performing track-to-track correlation, the conclusion is made that the remote new track nt_k actually emanates from the same target as a local existing track t, then it needs to fuse the information from track nt_k into track t, and

   $$Map^i_{tid} = Map^i_{tid} \cup \{nt_k \longrightarrow t\}. \tag{6.6}$$

   The local track ID t can be from the local track ID bank ($t \in TID_i$), if it is initiated using local frames on platform i. Or it can belong to some remote track ID bank (*e.g.* $t \in TID_j$). In that case, the track has been inserted as a new track, based on the data association decision of platform j. Thus, one can conclude that $\{t \longrightarrow t\} \in Map^i_{tid}$.

   The network track ID for track t is updated if it is greater than nt_k.

When a track t_o terminates on platform i, the composite tracker goes through the track ID map Map^i_{tid}, and removes all terms such that $\{t_x \longrightarrow t_o\}$.

Based on the network to local track ID map, for all remote AMR events, the composite tracker checks the track ID Map to map the network track ID to local Track ID, and then tries to locate and update the corresponding track. If the remote track ID is not in the current track ID Map, the following reasons are possible:

- The corresponding network track has been declared to be terminated on the local platform.
- Due to communication delays in the network, the remote data association decision of the new track has not arrived yet.

The composite tracker cannot use the AMR at that time. If the AMR is for a terminated track, it will never be used again. However, if the AMR corresponds

to a remote new track that has not arrived yet, it can still be used after the arrival of the new remote track. The composite tracker now discards the AMR if the corresponding track does not exist in the TrackTree.

6.3 Track to Track Correlation

6.3.1 Mathematical Formulation as 2D Assignment Problem

Denote the remote incoming new tracks as $T_1^{new}, \cdots, T_{M_1}^{new}$ and the existing tracks as $T_1, \cdots, T_{M_2}$. It can be concluded that for each new network track T_i^{new}, it can emanate from the same target as at most one of the existing tracks T_j, and for each existing track, it can represent the same target as at most one of the remote ones. Thus, the problem can be formulated as a $2-D$ assignment problem. Let $x_{ij} \in \{0,1\}$ stand for whether track T_i^{new} emanates from the same target as the existing track T_j. If they do emanate from the same target, then $x_{ij} = 1$. Otherwise, $x_{ij} = 0$.

$$\begin{aligned} \min \quad & \sum_{i=0}^{M_1} \sum_{j=0}^{M_2} c_{ij} x_{ij} \\ s.t. \quad & \sum_{i=0}^{M_1} x_{ij} = 1, \quad j = 1, \cdots, M_2 \\ & \sum_{j=0}^{M_2} x_{ij} = 1, \quad i = 1, \cdots, M_1 \\ & x_{ij} \in \{0,1\} \end{aligned} \tag{6.7}$$

where c_{ij} is the cost function or distance function that measures the distance between track T_i^{new} and track T_j. A user definable D_{max} is used as a threshold, if $c_{ij} > D_{max}$, then the arc (i,j) is not even added, which means track T_i^{new} can not go with track T_j because they are too far apart. However, the assignment problem requires

$$c_{ij} - c_{i0} - c_{0j} < 0,$$

so one can set

$$c_{i0} = c_{0j} = \frac{D_{max}}{2}. \tag{6.8}$$

6.3.2 Distance Function

In order to compute the scores c_{ij}, each track needs to predict its state vector and covariance matrix to a common time and transform them to a common coordinate system to compute the distance.

- Euclidean Distance

  The most commonly used distance function is the Euclidean distance. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be the $3-D$ position vector of track T_i^{new} and T_j respectively. Then the distance d_{ij} is

$$d_{ij} = \sqrt{< \mathbf{x}_i - \mathbf{x}_j,\ \mathbf{x}_i - \mathbf{x}_j >} \tag{6.9}$$

- Statistical Distance

  Let $\mathbf{X}_i$ and $\mathbf{X}_j$ be state vector and $\mathbf{P}_i$ and $\mathbf{P}_j$ be the corresponding covariance matrix of track T_i^{new} and T_j. Then, the state difference vector

$$\tilde{\mathbf{X}}_{ij} = \mathbf{X}_i - \mathbf{X}_j$$

  Generally, $\mathbf{X}_i$ and $\mathbf{X}_j$ contain position and velocity information. The statistical distance is defined to be

$$d_{ij}^2 = \tilde{\mathbf{X}}_{ij}^T [\mathbf{P}_i + \mathbf{P}_j - \mathbf{P}_{ij} - \mathbf{P}_{ij}^T]^{-1} \tilde{\mathbf{X}}_{ij}, \tag{6.10}$$

  where $\mathbf{P}_{ij}$ is the cross-covariance matrix between track T_i^{new} and T_j's estimation errors. [8]

6.3.3 Parse Solutions to the $2D$ Assignment Problem

Solution Arcs of the Form $x_{i0} = 1$

The incoming new track T_i^{new} does not correlate with any of the existing ones, it needs to be added to the local track database.

1. It needs to be inserted into the current TrackTree. The corresponding integer sequence representing the node is $(0 \cdots 0 - t_i^{new})$, where there are $(M-N-1)$ zeros, and t_i^{new} is the network track ID of the new track. Furthermore, the track is extended to local frames in the extension window, *i.e.*, build all the children of node $(0 \cdots 0 - t_i^{new})$.

2. The network to local track ID map needs to be updated.

Solution Arcs of the Form $x_{ij} = 1$

The incoming new track T_i^{new} emanates from the same target as the existing track T_j. Suppose T_i^{new} is broadcast to the composite tracker by platform k, then it contains a list of observations associated with it and they are observations generated by sensors on platform k only. The existing track T_j couldn't have had those observations yet, so the two tracks need to be fused by adding the observations from T_i^{new} to T_j.

The network track ID of the track T_j will be changed to t_i^{new}, if the current network track ID is greater than t_i^{new}.

The track ID Map needs to be updated accordingly.

Solution Arcs of the Form $x_{0j} = 1$

This means that there is no incoming new tracks correlate with the existing track T_j. No operation is necessary.

Chapter 7

PROPOSED NETWORK TRACKING ARCHITECTURE II: MFA ON ALL DATA AND NETWORK TRACKS

7.1 Overview of Architecture

The network MFA architecture on all data and network tracks we propose is explained in Figure 7.1. There is one composite tracker on every platform. The composite tracker has access to all the data from the sensors on its local platform as well as data from sensors on remote platforms. The double pane sliding window is made up of both remote and local frames. The data association decisions are made based on the network tracks and the frames (remote and local) in the window.

7.1.1 Basic Assumptions

- The communication network is fully connected. There is delay in the network, but there is no lost data.
- Local observations are put into frames by the frameBuilder on the local composite tracker. Then, the observations are broadcast to all remote platforms with a local frame ID and the position in that frame. Thus, all composite trackers have the exact same frames.
- The sliding window on each tracker consists of both remote and local frames.

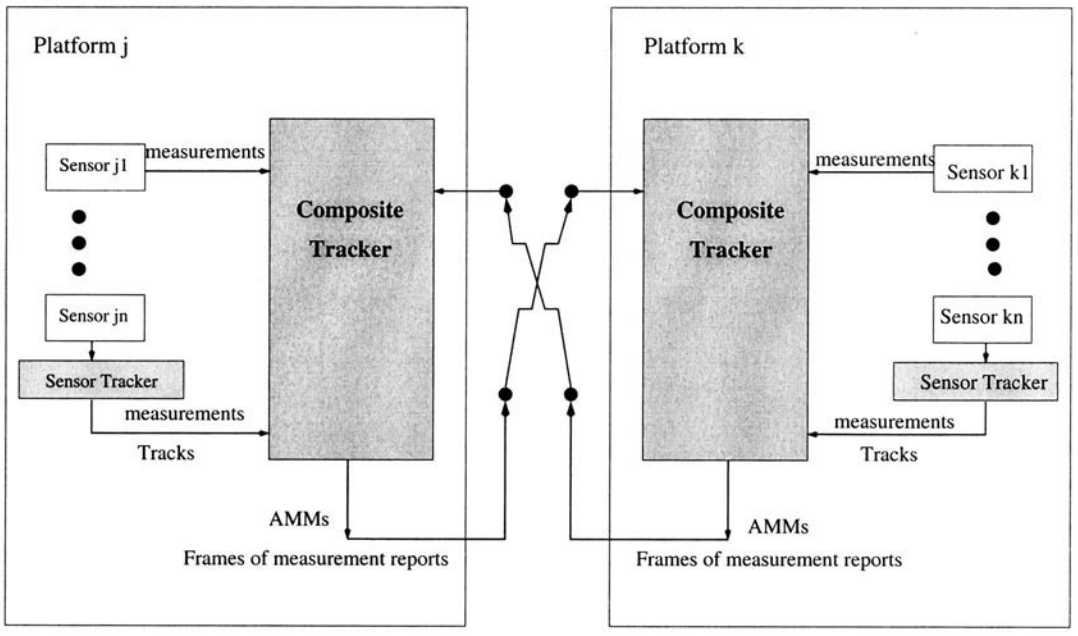


Figure 7.1: Network MFA on All Data and Network Tracks

- Each composite tracker can only make firm (irrevocable) data association decisions on its own frames (local frames). For remote frames, the tracker needs to update its network tracks based on the remote data association decisions.

7.1.2 What's Being Sent Out as Fixed DA Decisions

If the $(M-N+1)^{-th}$ frame ($f_{k_{M-N}}$) in the sliding window is a local frame, then the composite tracker sets up an M dimensional assignment problem and firm data association decisions on that frame are assembled together to form an AMMFrame, which includes:

- Associate Measurement Maps (AMMs): Each AMM has for parts: an AMM-Frame ID, a platform ID, a network track ID, and an observation ID in the corresponding frame.

- Newly initiating tracks: Each new track contains an AMMFrame ID, a platform ID, a network track ID, and an observation list that is associated with the new track.

7.1.3 What's Being Done After Receiving Remote DA Decisions

When a new frame of data is ready to be added into the sliding window, the composite tracker checks to see if the first frame in the extension window ($f_{k_{M-N}}$) is a local frame or a remote one. If $f_{k_{M-N}}$ is a local frame, then the composite tracker sets up the assignment problem and solves it. The AMMFrame is built as discussed in Section 7.1.2. Then the first frame (f_{k_0}) in the moving window is removed and the new frame is inserted.

However, if frame $f_{k_{M-N}}$ belongs to a remote platform, the composite tracker is not allowed to make data association decisions on that frame, instead, it needs to wait for the remote AMMFrame. If the corresponding AMMFrame is in the AMMFrame buffer, the tracker fixes the data association decisions between the network tracks and observations in frame $f_{k_{M-N}}$ based on the AMMFrame received. The TrackTree is properly updated and the remote frame ($f_{k_{M-N}}$) is deleted from the sliding window. Then the new frame is inserted into the sliding window.

However, if the corresponding AMMFrame has not arrived yet, then the composite tracker waits for the AMMFrame for a certain amount of time. If during the waiting period, the AMMFrame arrives, it is then used to update the TrackTree as discussed before. If the AMMFrame does not arrive, then frame $f_{k_{M-N}}$ is removed from the extension window and stored in the tardy queue waiting to be used when the corresponding AMMFrame arrives. Thus, whenever a remote AMMFrame arrives, the composite tracker checks to see if the corresponding frame is in the tardy queue. If the frame is not in the tardy queue, which means the frame is still in the window or has not been added into the window yet, the AMMFrame is buffered in the AMMFrame buffer. If the corresponding frame is in the tardy queue, it means that the frame has been removed from the window because the AMMFrame arrives too late. The composite tracker then uses the AMMFrame and the frame in the tardy queue to update the TrackTree immediately instead of buffering it.

7.2 Algorithm Overview in the Ideal Case

If there are no communication or processing delays, then all the frames on different platforms are in exactly the same order. It is further assumed that all the trackers are well synchronized. Figure 7.2 shows three platforms (i, j, and k) in the network, each has a track extension window of size 2.

Suppose at certain time t, the network tracks (represented as black dots) are the same, the first frame in the extension window belongs to platform i, and another frame (from platform j) is ready to be processed. The composite tracker on platform i sets up a data association problem based on the network tracks and frames in the window (one local frame and one remote frame from platform j). After solving the DA problem, it makes firm (irrevocable) decisions on its local frame, broadcasts the AMMs and new tracks as an AMMFrame to platform j and k, and moves the window forward. For the composite trackers on platform j and k, since the first frame in their extension windows is a remote frame (from platform i), they wait for the AMMFrame from platform i. After the AMMFrame is received, the trackers use them to update the network tracks. The remote frame is then removed from the sliding window.

Then another frame (from platform k) is ready. Now the first frame in the extension window belongs to platform j, so the composite tracker on platform j sets up the DA problem, solves it, fixes the decisions in that frame, broadcasts the AMMFrame to platform i and k, and moves the window forward. The composite trackers on platform i and k wait for the AMMFrame to update their network tracks.

Figure 7.2 illustrates three steps of such a process. After those three steps, the network tracks on all platforms are still the same because they have the same observation history.

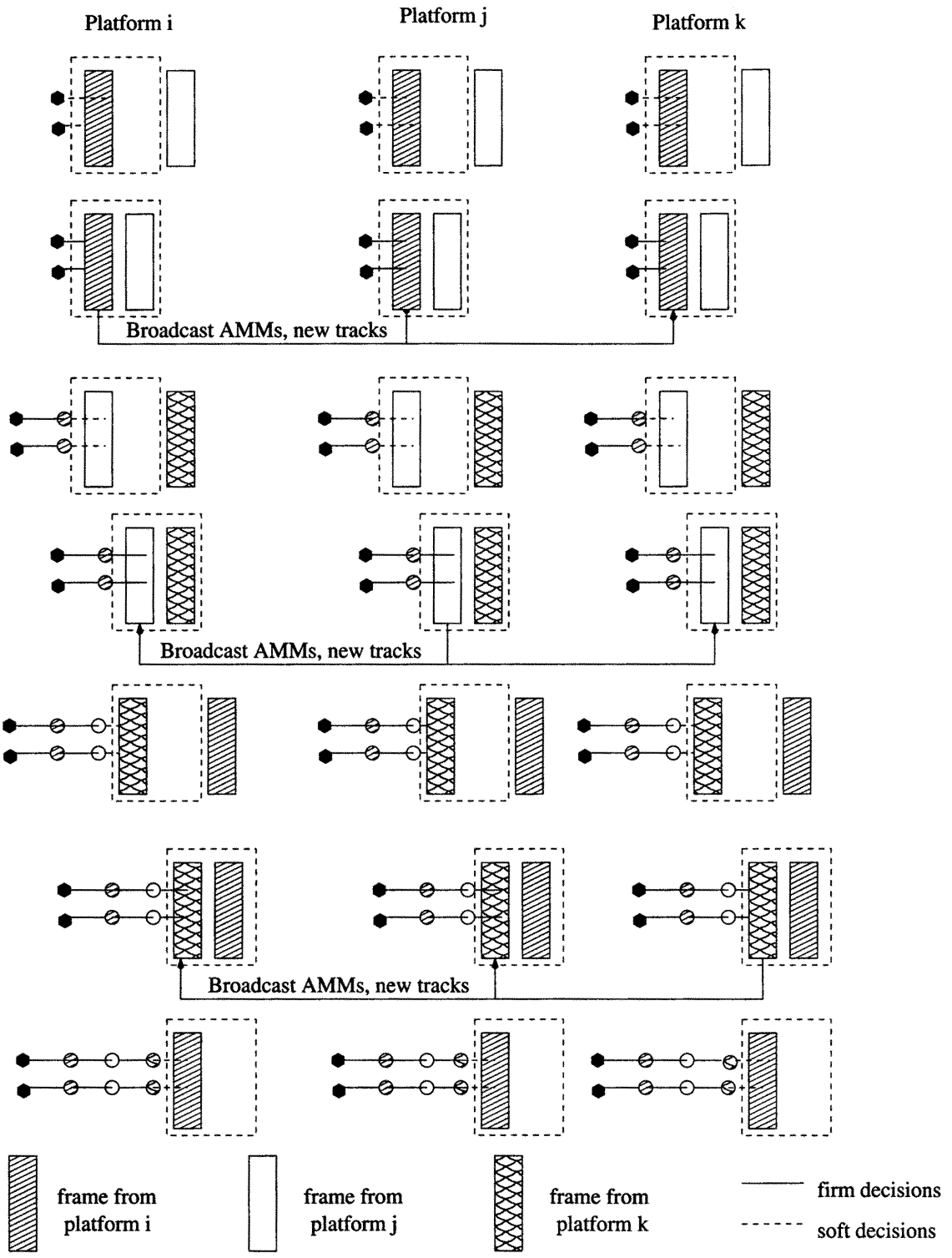


Figure 7.2: Ideal Case for Network MFA on All Data and Network Tracks

7.3 Algorithm Overview in More Realistic Case: Delays

In real life situations, random delay in the transmission network is inevitable. Furthermore, processing time needed to put a remote frame together is different for each tracker. Thus, the order of the frames being processed on each composite tracker is different. Similarly, for remote data association decisions, they arrive at different platforms at different times and in different order. The AMMFrame can be ready earlier than the time that the frame is shifted to be the first frame in the extension window, in which case the composite tracker buffers the AMMFrame before processing it. However, the AMMFrame can be ready much later than the time the frame is being processed.

These out-of-order data cause the problems which we call them conflicts in association and deadlock in processing.

7.3.1 Conflicts in Association

Problem Description

The problem is illustrated in Figure 7.3 using two platforms i and j. For the composite tracker on platform i, the first frame in the extension window is a local frame, so it sets up and solves the data association problem, fixes the decision, broadcasts the AMMs, and moves the window forward. In the mean time, for the composite tracker on platform j, the first frame in the extension frame is a local frame, so it sets up and solves its own DA problem, fixes the decision, broadcasts the AMMs, and moves the window forward.

When another frame is ready to be processed on platform i, the first frame in the extension window belongs to platform j. The composite tracker on platform i uses the AMMFrame from platform j to fix the data association decisions between network tracks and observations in the first frame of the extension window. However, the association may not exist in the TrackTree on platform i.

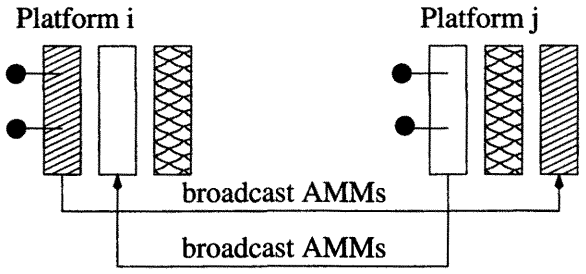


Figure 7.3: Conflicts in Association

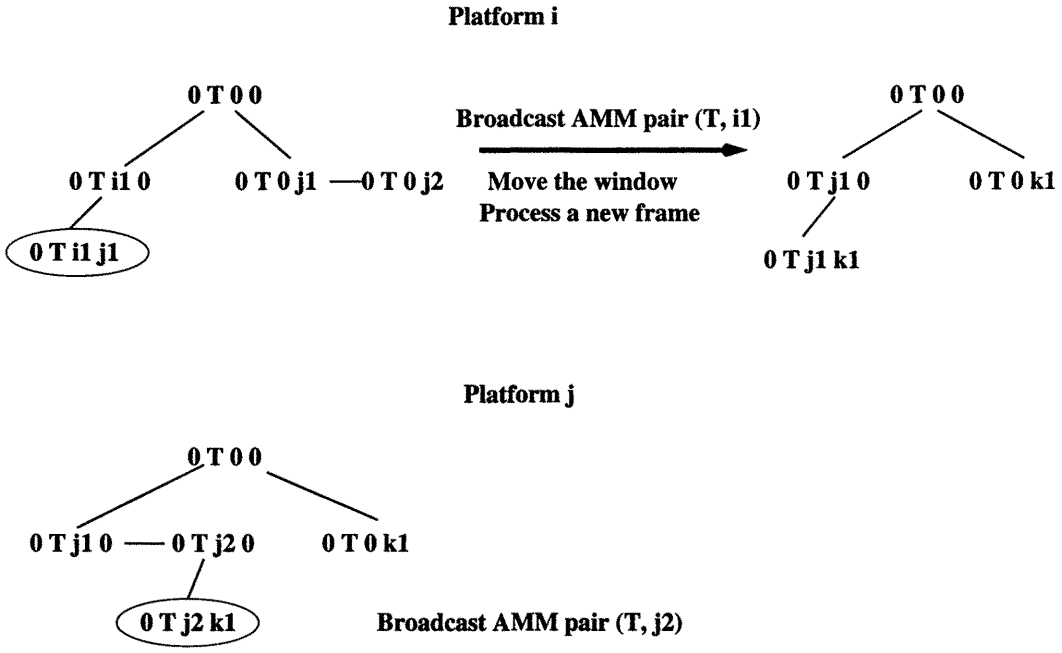


Figure 7.4: Example of Conflicts in Association

A specific example is given in Figure 7.4. Suppose the frame from platform i contains one measurement $\mathbf{z}_{i_1}$, the frame from platform j contains two measurement $\mathbf{z}_{j_1}$ and $\mathbf{z}_{j_2}$, and the frame from platform k has one measurement $\mathbf{z}_{k_1}$. In a 4/2 window, the sub-tree structure of network track T on platform i is shown in Figure 7.4 . Suppose the data association solution is $(0Ti_1j_1)$, which is represented by the node circled in the tree. The composite tracker broadcasts the AMM pair (T, i_1), prunes the TrackTree, and moves the window forward. Then a new frame from platform k is added and the TrackTree is extended to the new frame. Meanwhile, the composite tracker on platform j finishes processing its local frame and

broadcasts the AMM pair as (T, j_2). When the tracker on platform i has to fix the association between network track T and the observation $\mathbf{z}_{j_2}$ in the first frame in the extension window based on the AMMs received, the TrackTree indicates that track T is not feasible with observation $\mathbf{z}_{j_2}$, or the node $(0Tj_20)$ does not exist in the tree.

Proposed Solutions

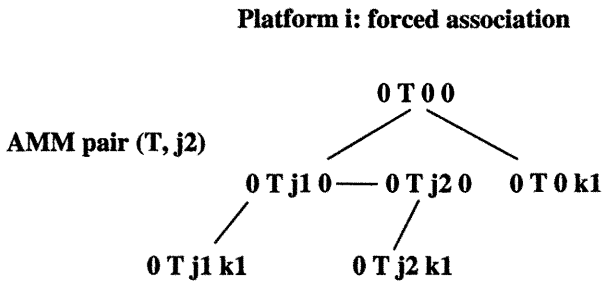


Figure 7.5: Proposed Solutions to Conflicts in Association: Forced Association

If a conflict in data association decisions ever occurs, in order to achieve a consistent air picture, the network tracks on different platforms need to be updated with the same sequence of observations. Thus, the composite tracker forces association between network tracks and observations based on the remote AMMs. As is shown in Figure 7.5, on platform i, a new node $(0Tj_20)$ is added and extended to the remaining frames in the extension window (building child node $(0Tj_2k_1)$).

7.3.2 Deadlock in Processing

Problem Description

As is shown in Figure 7.6, for the composite tracker one platform i, the first frame in its extension window is a remote frame from platform j, so it waits for the AMMFrame from platform j to update the TrackTree and move the window forward. However, platform j's first frame in the extension window belongs to

platform i, it can not process until the AMMFrame from platform i is received. Thus, both platforms are waiting for each other and the deadlock problem occurs.

Figure 7.6: Deadlock in Processing

Proposed Solutions

The key to solving the deadlock problem and maintaining a consistent air picture is to have the composite tracker wait for the corresponding AMMFrames for each remote frame for a certain number of steps which is specified by the user in the parameter file.

During the waiting period, if the remote AMMFrame is ready to be processed, then the composite tracker updates its network tracks with the remote AMMFrame. If after the specified number of steps, the AMMFrame still has not arrived yet, the remote frame will be removed from the sliding window, and put into the tardy queue waiting to be processed later. When the corresponding AMMFrame arrives, the frame is taken out from the tardy queue and used to update the network tracks.

In this way, the sliding window on the composite tracker can always move forward regardless of the status of the remote platforms in the network.

7.3.3 One Way to Moderate the Problems: Buffering Frames

On each composite tracker, there is a frame buffer to buffer both local and remote frames before they are processed by the sliding window. Thus, the frames are not put into the sliding window in the order in which they become ready. Instead, they are put into the frame buffer to be buffered for a certain amount of time T_{buffer} (Fixed Lag Buffer) or to be buffered for a certain length (Fixed

Length Buffer). The proposed method to order the frames in the frame buffer is to order them according to the time tag of the last observation in the frame.

The reason for choosing the last observation in the frame is that in most scenarios, one needs to deal with sensors with vastly different revisit rates. For instance, a rotator might only see the target once in 10 seconds, while an ESR may see the target every 0.1 second. If the time tag of the first observation in the frame is used to order the frames, then the composite tracker might need to process the rotator frame first before it can come to process those ESR frames. So if the tracker needs to report track states, there is a lot of valuable information not being used. By choosing the time tag of the last observation in a frame, short frames can be processed sooner.

In the optimal case for a fixed lag buffer, where $T_{buffer} \rightarrow \infty$, all the composite trackers on different platforms should order the frames exactly the same (according to the time tag of the last observation in the frame). However, since T_{buffer} is a finite number that is set according to some *a priori* knowledge of the communication network, the order of the frames in the frame buffer may not the same on different composite trackers. So the order of the frames being added into the window is different. In Figure 7.7, the horizontal axis stands for the frame duration time, and the vertical axis stands for the frame arrival time. The frame arrival times are different on platform i and platform k. If T_{buffer} is big enough, then the frames on both platforms are in the same order. However, if T_{buffer} is not big enough, the orders of frames on platform i and k are still different.

Similarly, for a fixed length buffer, if the buffer length is infinite, the order of the frames in the buffer on all platforms are the same. However, for a finite length buffer, the frame order may still be different.

From the above discussion, it can be concluded that buffering the frames moderates the problems caused by out-of-order frames. However, when the composite

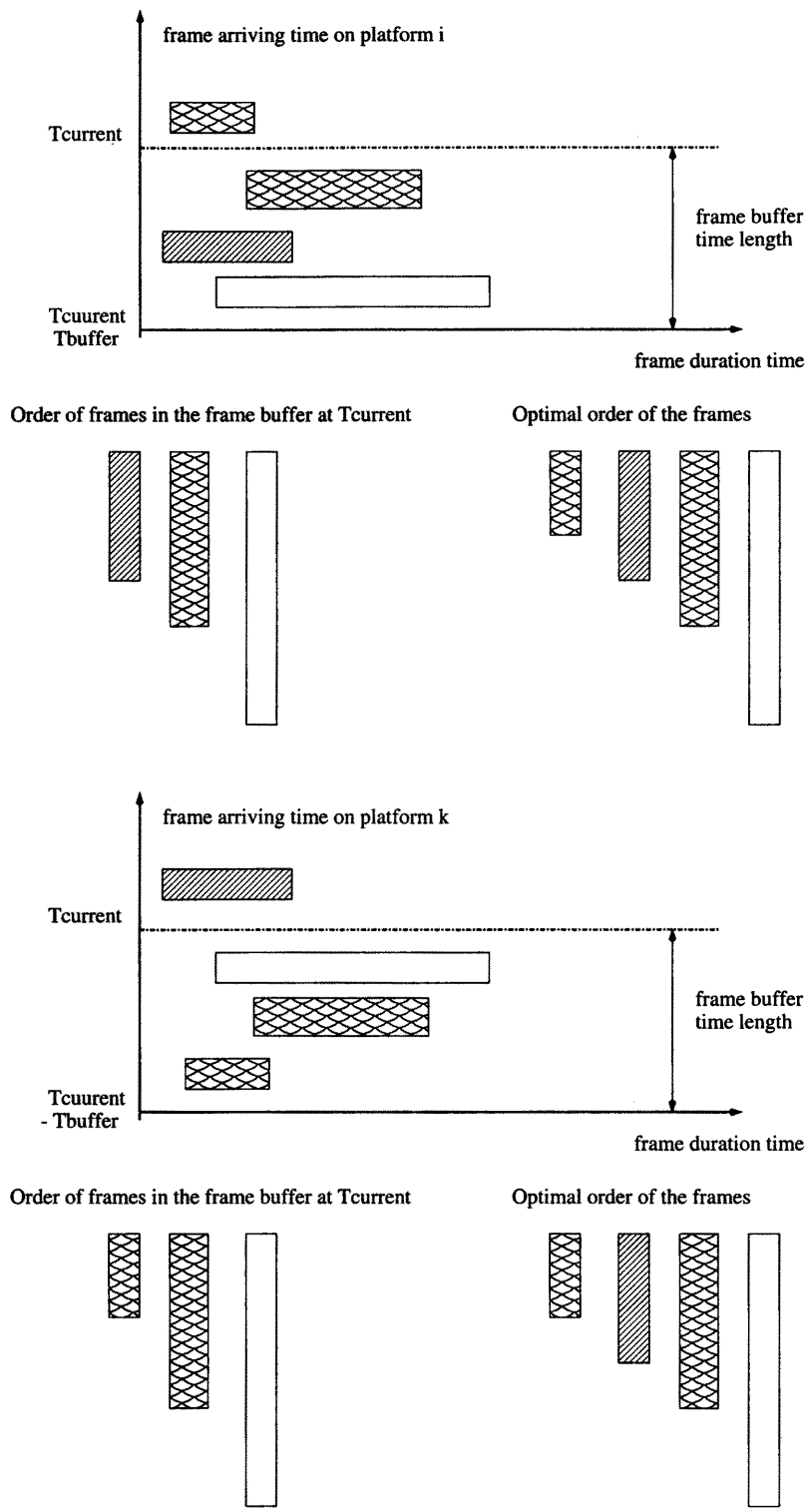


Figure 7.7: Illustration of Frame Buffer

tracker handles a metrics scoring request, the more frames in the frame buffer, the worse the state estimate accuracy is, due to the fact that more frames haven't been processed yet.

7.4 Event-Driven Overview

7.4.1 Modified Event Manager

The event manager in the proposed architecture is different from the event manager in the centralized tracker as explained in Chapter 2. The difference is caused by the solution to the deadlock problem as described in Section 7.3.2. For a remote frame, the composite tracker should wait for the corresponding AMMFrame for a finite number of steps.

In order to achieve the goal, an *isWaiting* flag is added in the event manager. If the flag is *true*, it means the tracker is waiting for the corresponding AMMFrame when processing a new frame of observations. During the waiting period, the first event in the event queue, which should be a kernel frame event, remains unprocessed. However, the event manager shouldn't block the incoming remote data association decision events. Thus, all the incoming remote data association decision events that are used to build AMMFrames are fired directly, instead of being posted onto the back of the event queue. If the AMMFrame is not ready after a certain number of steps, the first frame in the extension window is removed and put into the tardy queue, and the new frame is inserted into the window. If the *isWaiting* flag is *false*, the eventManager proceeds as discussed in the centralized case. The *isWaiting* flag is set to be *true* each time the composite tracker starts a waiting process, and is reset to be *false* either the corresponding AMMFrame is received or the frame is moved to the tardy queue.

7.4.2 Introduction to Events

Adaptive Events

1. Observation Event

   An observation event contains an observation from a local sensor, it is then processed by the frameBuilder to be put into proper frames. The frameBuilder builds sensorFrames according to sensor types, and links the non-overlapping sensorFrames into proper frames as discussed in the centralized architecture. The frameBuilder generates a measurement report event for local observations. It generates a frame event when the frame is ready.

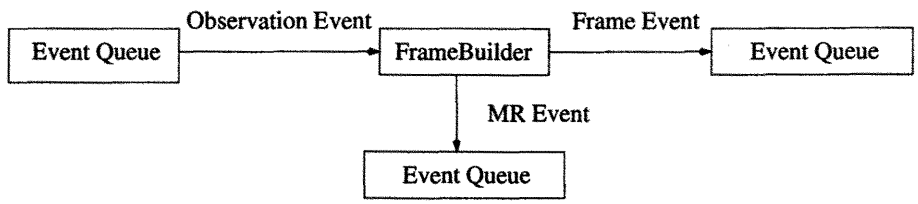


Figure 7.8: Observation Event

2. Measurement Report (MR) Event

   A measurement report event contains three parts, an observation, a platform ID, and a sensorFrame ID. It is broadcast to all the remote platforms by the composite tracker.

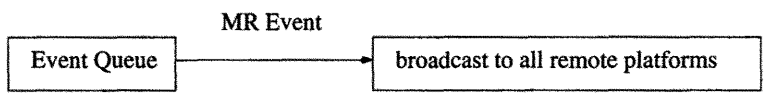


Figure 7.9: Measurement Report Event

3. Incoming Measurement Report (MR) Event

   An incoming measurement report contains a remote observation, a platform ID and a sensorFrame ID. The frameBuilder processes incoming MR events, putting observations into the corresponding sensor frames according to their platform ID and sensorFrame ID.

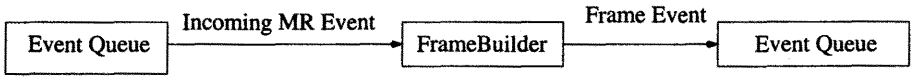


Figure 7.10: Incoming Measurement Report Event

4. Frame Event

   A frame event can be either a **local** frame or a **remote** frame. All frames are added into the frame buffer to be ordered based on the time tag of the last observation in the frame. And if there exists a frame that has been buffered long enough, the Frame Buffer generates a Kernel Frame event, which is then pushed onto the event queue. For a local frame, the composite tracker broadcasts the lists of sensorFrame IDs and the corresponding sensorFrame sizes to the remote platforms.

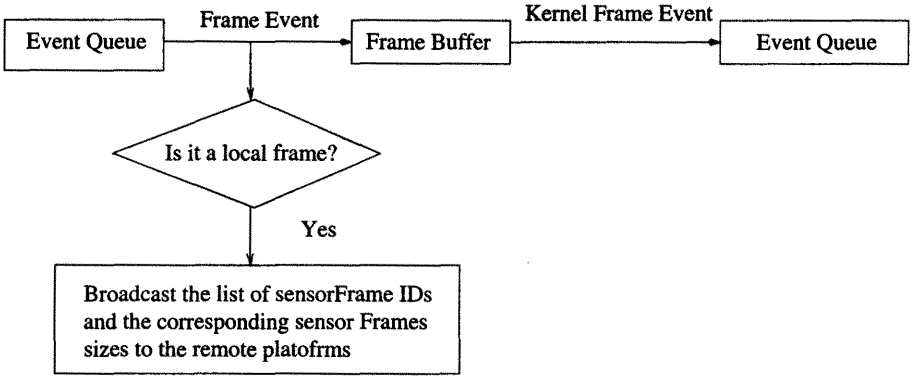


Figure 7.11: Frame Event

5. EndFrame Event

   An EndFrame Event contains four parts: a platform ID, a frame ID, a list of sensorFrame IDs and a list of corresponding sensorFrame sizes. The frameBuilder listens to the EndFrame event, and builds the remote frame by linking the corresponding sensorFrames together.

6. EOF Event

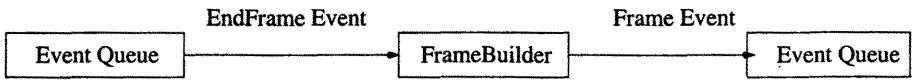


Figure 7.12: EndFrame Event

An EOF event is generated when it reaches the end of the input stream. When the frameBuilder encounters an EOF event, it goes through its list of unfinished local frames, and generates frame events for them.

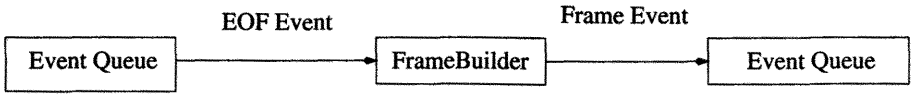


Figure 7.13: EOF Event

7. EOFrame Event

When an EOFrame event is generated by the composite tracker, it means there is no more frames (local or remote) ready. The frameBuffer generates kernel frames for all the frames it buffers.

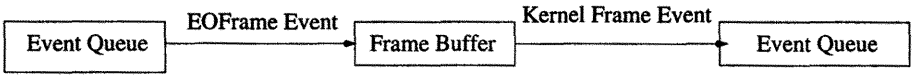


Figure 7.14: EOFrame Event

Data Association Decisions Events

Data association decision events are all fired instead of posted onto the back of the event queue to avoid processing delay.

1. Broadcasting Events

The composite tracker builds the window using both local and remote frames. As discussed in the centralized architecture (Chapter 2), whenever a frame is ready to be processed, the composite tracker checks to see whether the first frame in the track extension window is a local frame or not. If it is a local

frame, then the tracker sets up and solves the data association problem. The data association decisions on the first frame are irrevocable and are broadcasted to all other platforms. There are three types of broadcasting DA decisions events: AMM, new track and EndAMMFrame events.

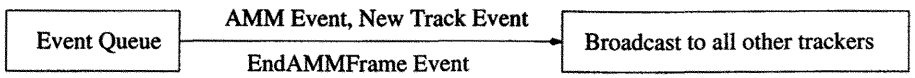


Figure 7.15: Broadcasting Events

- AMM Event

  An AMM event has four parts: a platform ID, an AMMFrame ID, a network track ID, and an observation ID. The track ID and observation ID pair indicates which network track goes with which observation in the frame.

- New Track Event

  A new track event contains a platform ID, an AMMFrame ID, a network track ID and an observation list associated with the track.

- End of AMMFrame Event

  An end of AMMFrame event contains a platform ID, an AMMFrame ID, and the number of data association decisions in that frame, which are used by the remote side to reconstruct the AMMFrame.

2. Remote Events

Correspondingly, there are three types of remote events sent from the remote tracker in the network: incoming AMM event, incoming new track event, and incoming EndAMMFrame Event. The AMMFrameBuilder collects those pieces and reconstructs the AMMFrames based on the platform IDs and AMMFrame IDs. All remote events are fired instead of posted not only to

avoid the processing delay, but also to avoid being blocked while the event manager is in its waiting stage for some AMMFrame.

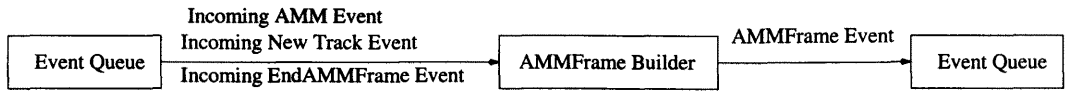


Figure 7.16: Remote Events

3. Output Events

   The output events are false alarm events, track initiate, update, and terminate events, which are all handled by the output formatter.

4. Track Metrics Events

   Track metrics events are generated by track metrics scoring events. They are processed by the metrics outputFormatter, to be predicted to the required time and transformed into the required coordinate systems.

Metrics Scoring Event

Metrics scoring events are used to evaluate the performance of the composite trackers. Similar to the centralized architecture, the composite tracker forks a child process to handle the metrics scoring request. The user can set the parameter *predictFrom* to be (1) " hard ", in which case the track states are reported based on the most recent firm decisions; (2) "soft", in which case the track states are reported based on the most recent soft decisions; and (3) "softPlus", in which case the track states reported contain all the available information (local and remote).

All the available information for the composite tracker includes the network tracks and all frames that have already been added into the window, the unfinished local and remote frames that are in the frameBuilder, and the finished local and remote frames in the frameBuffer. When the parameter *predictFrom* is set to be "softPlus", the metrics event collects all unfinished frames in the frameBuilder,

and finished frames in the frameBuffer. The sliding window processes the metrics event by appending all the unfinished/finished frames to the right end of the window, extending the TrackTree to those frames, setting up and solving a $(M+p)$ dimensional assignment problem, where p is the number of unfinished frames, and firing track metrics events on the solution tracks.

Kernel Events

Kernel events are processed by the sliding window.

1. Kernel Frame Event

   The kernel frame event is generated by the frame buffer and processed by the sliding window. When the window encounters a frame event, the operations are different based on single frame processing ($M/1$ window) or multi-frame processing (M/N window, for $N > 1$). Emphasis is laid on multi-frame processing, but the difference will be pointed out at the end.

   For an M/N window, the composite tracker checks the first frame ($f_{k_{M-N}}$) in the extension window to see if it is a local frame or not. If the frame is a local one, then the new kernel frame is added into the window, the TrackTree is extended to that frame, and the data association problem is set up and solved. Data association solutions are used to generate DA decision events as discussed above. The TrackTree is pruned, the leftmost frame is shifted out of the window, and the window is moved forward.

   If frame $f_{k_{M-N}}$ is a remote one, then the composite tracker can't make its own DA decisions on that frame. Instead, it needs to update the tracks based on the corresponding AMMFrame. Thus, it checks in the AMMFrame buffer to see if the AMMFrame has been received or not. If the AMMFrame has already been received, then the associations between network tracks and

observations in first frame in the extension window are fixed based on the AMMFrame. Then the first frame in the extension window ($f_{k_{M-N}}$) is removed and the TrackTree is pruned accordingly. Details will be explained in the Section 7.5.4. The sliding window is of size $(M-2)$ after removing the remote frame. The remote new tracks are then inserted or fused with the current network tracks and the TrackTree is updated with the remaining frames in the extension window. Finally the new kernel frame is added into the window and the TrackTree is extended to the new frame. If the corresponding AMMFrame is not available yet, the kernel frame will be pushed onto the front of the event queue and the *isWaiting* flag of the event manager is set to be *true*. Since the incoming AMM, new track, EndAMMFrame events, and the AMMFrame events are all fired, it is expected that during the waiting process, the corresponding AMMFrame can arrive and is ready to be processed. However, if the composite tracker has been waiting for a certain number of steps and the AMMFrame isn't ready, the first frame of the extension window is removed and put into the tardy queue. The TrackTree is pruned accordingly. Then the new kernel frame is added into the window and the TrackTree is extended to the newly added frame.

For single frame processing ($M/1$ window), the window size is $(M-1)$ when a kernel frame event is being processed. Thus, the first frame in the extension window is actually the frame being added. So if the frame is a local one, then the composite tracker processes it as discussed before. Otherwise, if the corresponding AMMFrame exists, the tracker adds the frame into the sliding window, extends the TrackTree, and then fixes the data associations based on the AMMFrame. The frame is then removed from the sliding window. If the corresponding AMMFrame does not exist, then the window waits for it

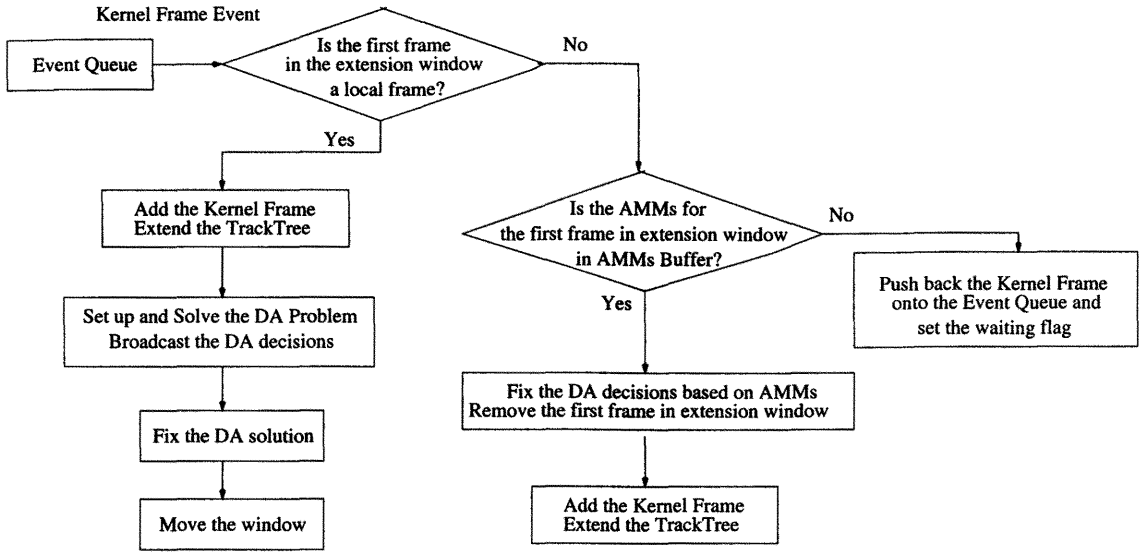


Figure 7.17: Kernel Frame Event for M/N Window ($N > 1$)

for a certain number of steps. And if the window has waited long enough, then the frame is put into the tardy queue.

2. AMMFrame Event

   The sliding window processes the AMMFrame event, it checks to see if the corresponding frame is in the tardy queue or not. If the corresponding frame is not in the tardy queue, then the AMMFrame is stored in the AMMFrame buffer waiting to be processed later.

   If the corresponding AMMFrame is in the tardy queue, then the TrackTree is updated using the AMMFrame and the frame taken out of the tardy queue. The procedures performed in the window and TrackTree are similar to the ones in Network MFA on local data and network tracks (Chapter 5). The network tracks are updated and the new tracks are inserted and fused with the existing ones. Then the TrackTree is re-extended to all the frames in the extension window.

7.5 Major Issues

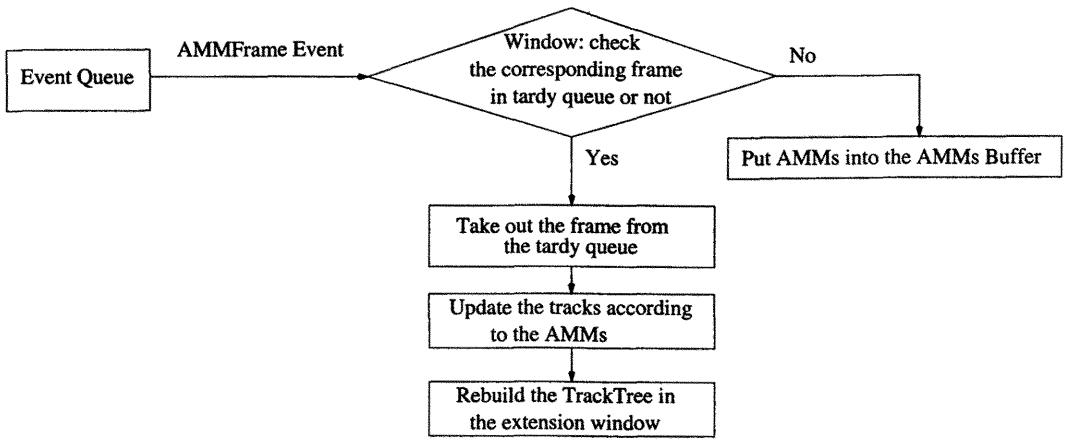


Figure 7.18: AMMFrame Event

7.5.1 Broadcasting AMMFrames

After irrevocable data association decisions are made on frame $f_{k_{M-N}}$ in the extension window, they are broadcast to all the remote platforms as AMMFrames.

New Tracks

For data association solution arcs of the form $(i_{k_0}, \cdots, i_{k_{M-N}}, i_{k_{M-N+1}}, \cdots, i_{k_{M-1}})$, the composite tracker assigns a local track ID and promotes it to the local track database, provided that the track string $TS_{i_{k_0},\cdots,i_{k_{M-N}}}$ satisfies the following requirements.

1. $i_{k_{M-N}} \neq 0$, which means the trackString has a detection in the frame $f_{k_{M-N}}$.

2. A tracking filter has been started and it produces a negative score.

3. It has at least *initLength* number of observations associated with it, where *initLength* is a user definable parameter that specifies the minimum number of observations required to initiate a new track.

4. The track state does not correlate with any of the existing remote tracks, who have no corresponding local tracks.

When a local track initiates, the associations in track string $TS_{i_{k_0},\cdots,i_{k_{M-N}}}$ are fixed and the new track event is generated to broadcast the new track to the network.

What is broadcast as a new track is a corresponding platform ID, an AMM-Frame ID, a track ID and a list of observations associated with the new track.

AMMs

For associated measurement maps, the composite tracker checks the list of updating tracks of the form $(0,\cdots,0,-T_{k_l},i_{k_{M-N}},\cdots,i_{M-1})$. If $i_{k_{M-N}} \neq 0$, then the composite tracker generates an AMM event, and broadcasts to remote platforms.

The AMM event consists of a Platform ID, an AMRFrame ID, a network track ID, which the local track ID T_{k_l} corresponds to, and the observation ID $i_{k_{M-N}}$.

End AMRFrame

The End AMMFrame message consists of a platform ID, an AMMFrame ID and the total number of data association decisions in the AMMFrame, which is the summation of the number of new tracks and the number of AMMs.

7.5.2 Building AMMFrames

The AMMFrameBuilder processes incoming AMMs, incoming new tracks and incoming end AMMFrame events, puts them into the correct AMMFrames according to the platform ID and the AMMFrame ID of each event. The incoming end AMMFrame event allows the AMMFrameBuilder to check if the corresponding AMMFrame is full by comparing the number of data association decisions received with the number expected. When an AMMFrame is ready to be processed, the event is fired instead of posted onto the back of the event queue.

7.5.3 Processing AMMFrame and the Corresponding Frame in Tardy Queue

When processing an AMMFrame event, if the corresponding frame is in the tardy queue, it means that the frame has been added into the window, and then been removed from the window as explained earlier in Section 7.4.2.

For a certain AMM pair (T_l, i_k) in the AMMFrame, with the help of the frame taken out from the tardy queue, the composite tracker is capable of locating the observation $\mathbf{z}_{i_k}$ to update the network track T_l. The algorithm used to update the TrackTree is exactly the same as the way the composite tracker updates its TrackTree in Network MFA on local data and network tracks architecture (Chapter 5).

For new tracks in the AMMFrame, the algorithm is the same as the one explained in Chapter 6. A two-dimensional assignment problem is set up to correlate the remote new tracks and the local existing tracks. Based on the solutions to the assignment problem, the new tracks are inserted in the TrackTree or fused with existing tracks.

7.5.4 Processing AMMFrame and the Corresponding Frame in Window

For an AMMFrame, if the corresponding frame is the first frame in the extension window (frame $f_{k_{M-N}}$), then the associations between network tracks and observations in frame $f_{k_{M-N}}$ are fixed based on the AMMs in the AMMFrame.

Update with AMMs

When processing an AMM pair $(T_l, i_{k_{M-N}})$, if the corresponding association exists in the TrackTree, as is shown in Figure 7.19, then followings steps are carried out:

1. Replace the track $T_{0\cdots 0-T_l}$ with $T_{0\cdots 0-T_l i_{k_{M-N}}}$.

2. Delete all children (or the sub-tree) of node $(0 \cdots 0 - T_l)$

3. Extend node $(0 \cdots 0 - T_l)$ to frames $f_{k_{M-N+1}}, \cdots, f_{k_{M-1}}$, while maintaining all the crossLinks.

4. Delete frame $f_{k_{M-N}}$ in the window, and prune the branches of the TrackTree that extends to that frame.

However, if the corresponding node does not exist in the TrackTree, then forced association is performed on the network track and the observation. Then the tracker continues with step 2.

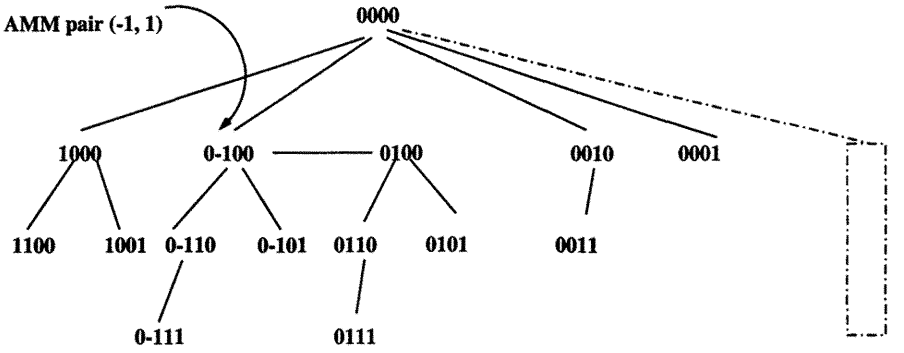
After processing all the AMMs, the sliding window is of size $(M-2)$.

Insert Remote New Track

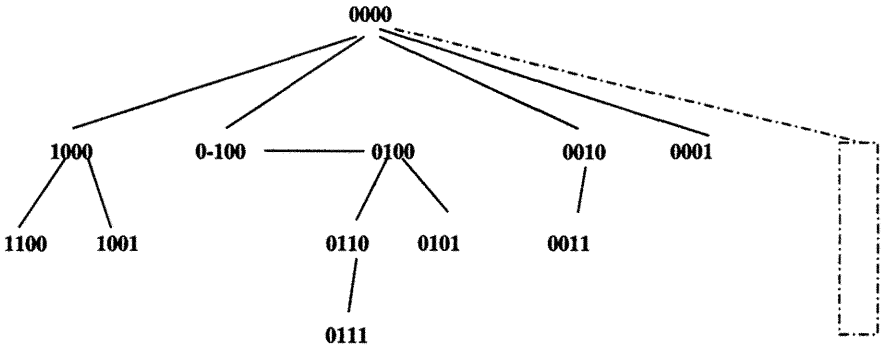
If a remote new track does not correlate with any of the existing tracks, it will be inserted in the TrackTree. After that, the new track is extended to all frames in the extending window. Figure 7.20 is an example of how the TrackTree changes.

Fuse Remote New Track with the Existing One

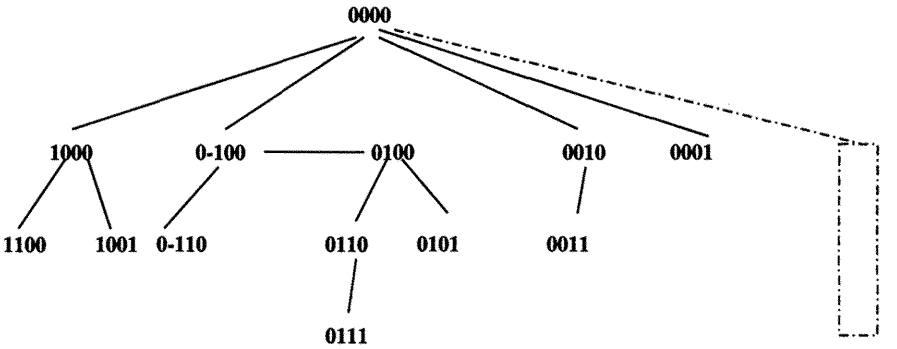
If the remote new track is regarded as emanating from the same target as one of the existing tracks, then the information of the two tracks are fused together, and then the corresponding sub-tree of the TrackTree is updated, as is shown in Figure 7.21.



Step I: Copy the Track of Node (0-110), delete all children of Node (0-100)



Step II: Update the Node (0-100) with the 4th frame in the window



Step III: Delete the 3rd frame in the window

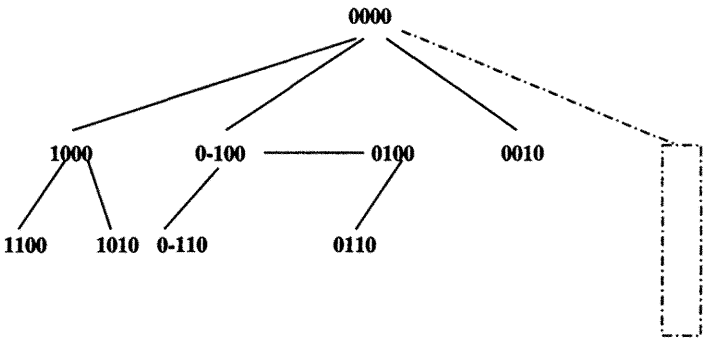
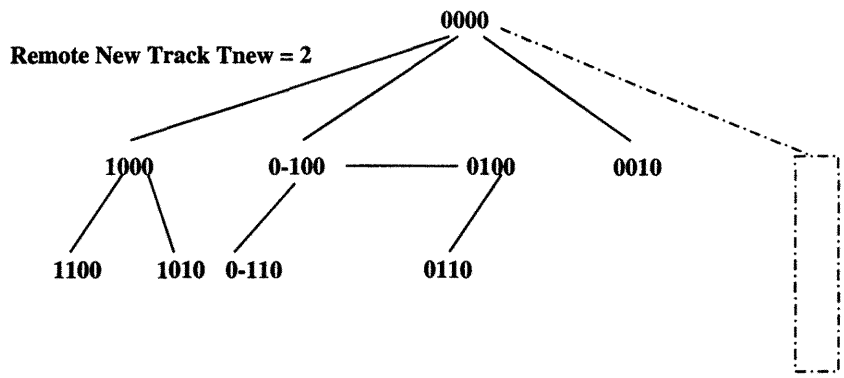
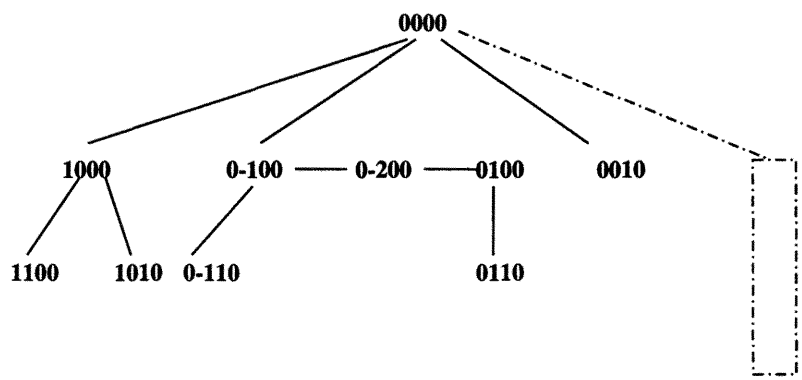


Figure 7.19: Example: Process An AMM



Step I: Insert New Node (0-200) in the TrackTree



Step II: Extend the Node (0-200) to the 3rd frame in the window

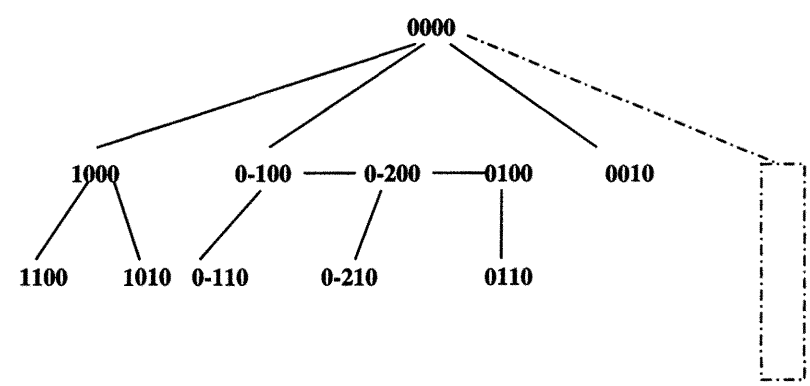
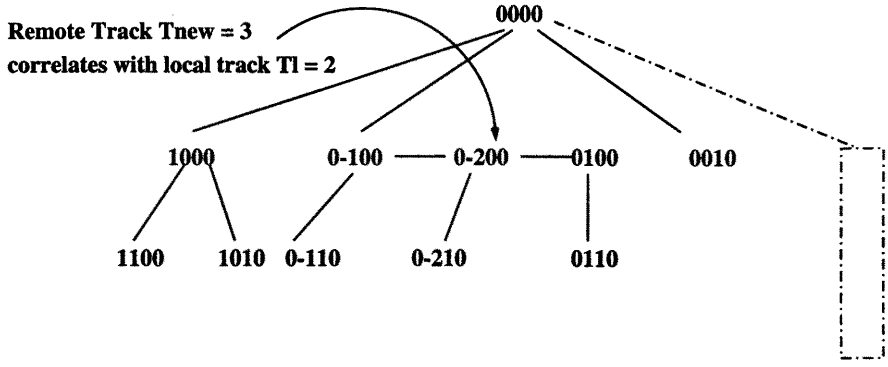
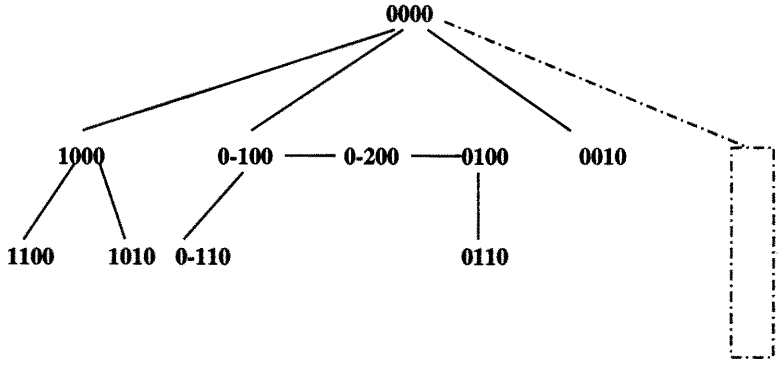


Figure 7.20: Example: Insert A Remote New Track into the TrackTree



Step I: Delete all children of Node (0-200), fuse the information of track (-2) and (-3)



Step II: Extend the Node (0-200) to the 3rd frame in the window

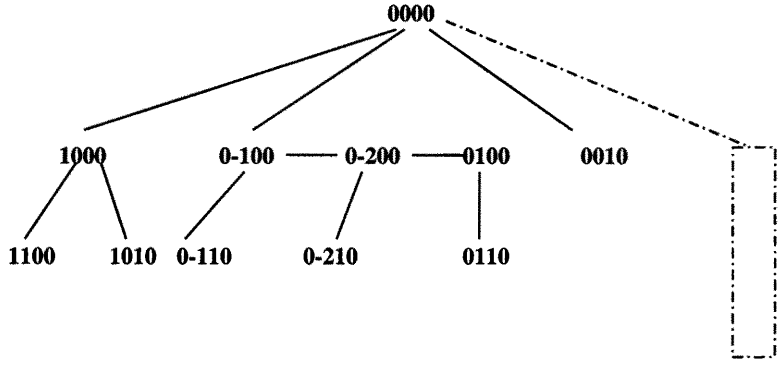


Figure 7.21: Example: Fuse A Remote New Track with An Existing Track

7.6 Track Initiation

7.6.1 Problem Description

For the Network MFA on all data and network tracks, the composite trackers initiate new tracks based on both remote and local data. Correlating the initiating tracks across the network is a little different from what is explained in Chapter 6.

However, the idea of track numbering schemes discussed in Section 6.2 for Network MFA on local data and network tracks is adopted here. The composite tracker uses the same local track ID bank, and the same network to local track ID Map.

7.6.2 Track to Track Correlation

$2D$ Assignment Problem

Denote the remote incoming new tracks in the AMMFrame as $T_1^{new}, \cdots, T_{M_1}^{new}$, and the existing tracks as $T_1, \cdots, T_{M_2}$. As discussed in the Network MFA on local data and network tracks, the $2D$ assignment problem is as explained in Equation (6.7).

The cost, c_{ij}, is computed by some distance function between track T_i^{new} and T_j as discussed in Section 6.3.2. If $x_{i0} = 0$, then T_i^{new} is a new track. If $x_{ij} \neq 0$, then T_i^{new} is regarded as emanating from the same target as the existing track T_j.

Insert Remote Track T_i^{new}

If the remote incoming new track T_i^{new} is regarded as emanating from a new target, it is inserted in the TrackTree using the integer sequence $(0 \cdots 0 - T_i^{new})$ as the address. There are $(M - N - 1)$ zeros in front of the negative track ID T_i^{new}. Then the new node is extended into all frames in track extension window. (Figure 7.20)

Fuse Track T_i^{new} and T_j

If the remote incoming new track T_i^{new} correlates with the local track T_j, then the two tracks are merged.

Since both tracks contain remote and local observations, most likely, they have the same observation history associated with them. The local existing track has the opportunity to be initiated first and broadcast to the network, it is then assumed that the local track contains more information (more observations) than the remote new one. Thus, all information from the remote new one is neglected to avoid using the same information twice. However, if the remote track ID $T_i^{new} < T_{n_j}$, where T_{n_j} is the network track ID of track T_j, then the network track IDs of the existing track and all its child tracks are set to be T_i^{new}.

Chapter 8

SIMULATION RESULTS, COMPARISON AND DISCUSSIONS: PERFECT COMMUNICATION LINK

8.1 Scenario Description

This chapter contains a comparison of the four different tracking architectures centralized, network MFA centralized, network MFA on local data and network tracks, and network MFA on all data and network tracks presented in the previous chapters.

The scenario we use contains four fighter aircraft and is 600 seconds in length. There are four ship platforms defined. Each ship has an S-band Phased array radar and a UHF rotating radar. Figure 8.1 illustrates the target and platform trajectories and depicts the merging of the fighters.

The composite trackers in all architectures apply the MFA algorithm. A double pane sliding window of size 6/2 is used. A two-model IMM filter, which consists of two NCV models with noise level 10 and 1000, is used. At least four observations are required to start a tracking filter, and at least six observations to initiate a new track. A refiltering window of length 20 is used to avoid negative time update.

In this Chapter, all simulations are carried out using a perfect communication network, where there is no communication delay between the platforms and

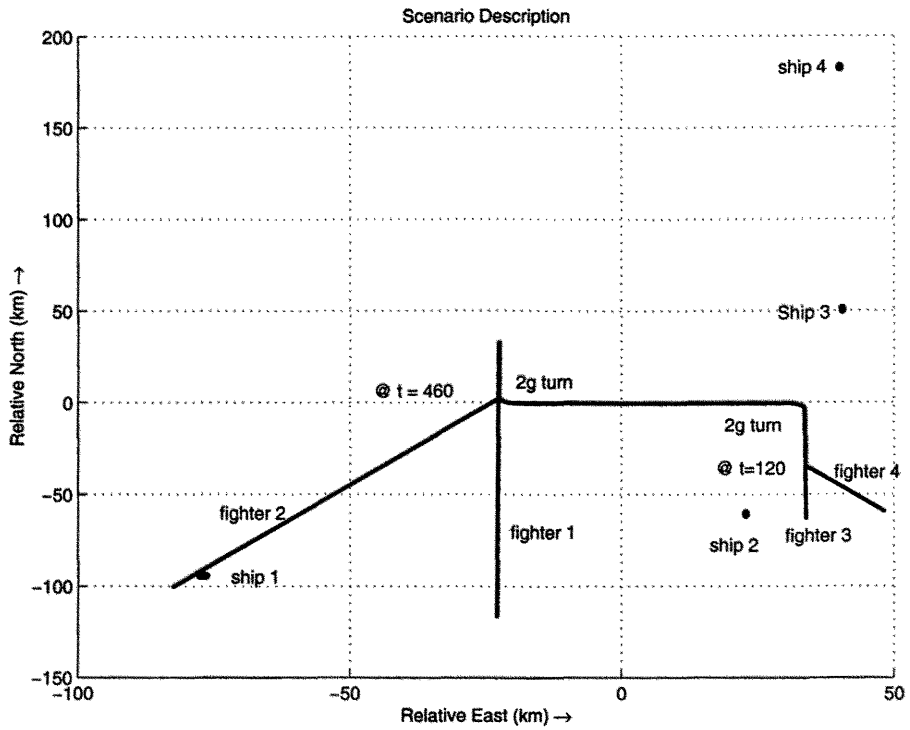


Figure 8.1: Scenario Description

there is no lost data. However, the order of the observations arriving on different platforms may still be different. For observations taken at the same time, the local observations always arrive on local platforms faster than the remote ones. For each architecture, five Monte Carlo runs are performed. For each metrics scoring request, the most recent "soft" decisions are used to give the state estimates.

8.2 Composite Track Ambiguity

The first metrics category is Ambiguity, which contains two metrics: Composite Redundant Track Mean Ratio and Composite Spurious Track Mean Ratio.

8.2.1 Spurious Track Mean Ratio

The composite spurious track mean ratio is equal to the number of un-assignable declared composite tracks divided by the number of valid, declared composite tracks. The metrics module attempts to assign every declared track on a platform

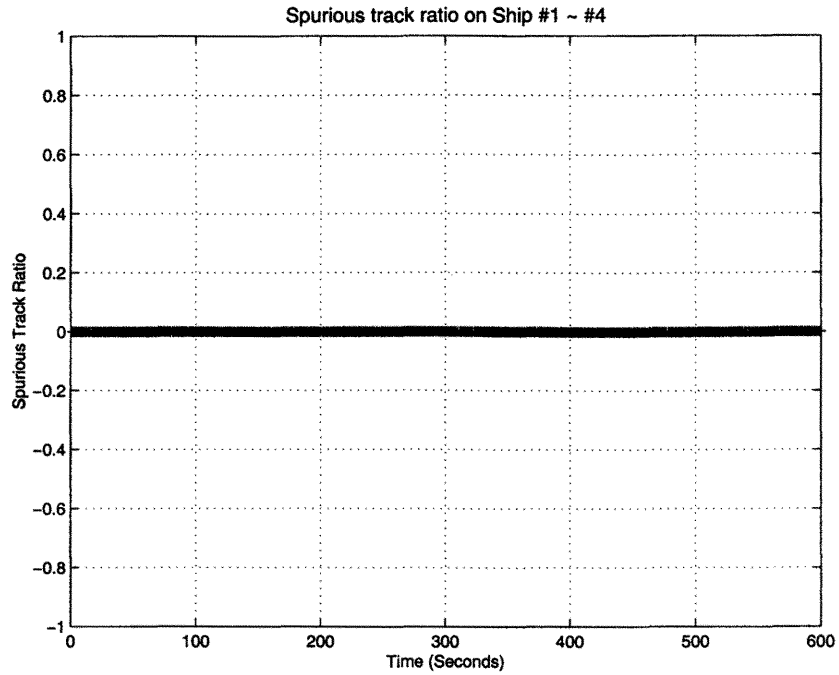


Figure 8.2: All four Architectures

to each truth object, if they pass the threshold for at least one truth object. A Euclidean threshold value defined to be $2e8$ is used. The number of un-assignable composite tracks is the number of composite tracks that do not pass the gating threshold with respect to any truth object.

For all four architectures, there are no spurious tracks on any of the platforms, and the spurious track mean ratio is always zero, as is shown in Figure 8.2.

8.2.2 Redundant Track Mean Ratio

The composite redundant track mean ratio is equal to the number of composite tracks assignable to a truth object divided by the number of valid, declared composite tracks. This metric is computed and plotted for each platform. A redundant track mean ratio of one is the ideal value, which means the composite tracker has exactly one track associated with each truth object.

For all the architectures, there are no redundant tracks. However, track initiation times differ on different platforms in different architectures. Figure 8.3

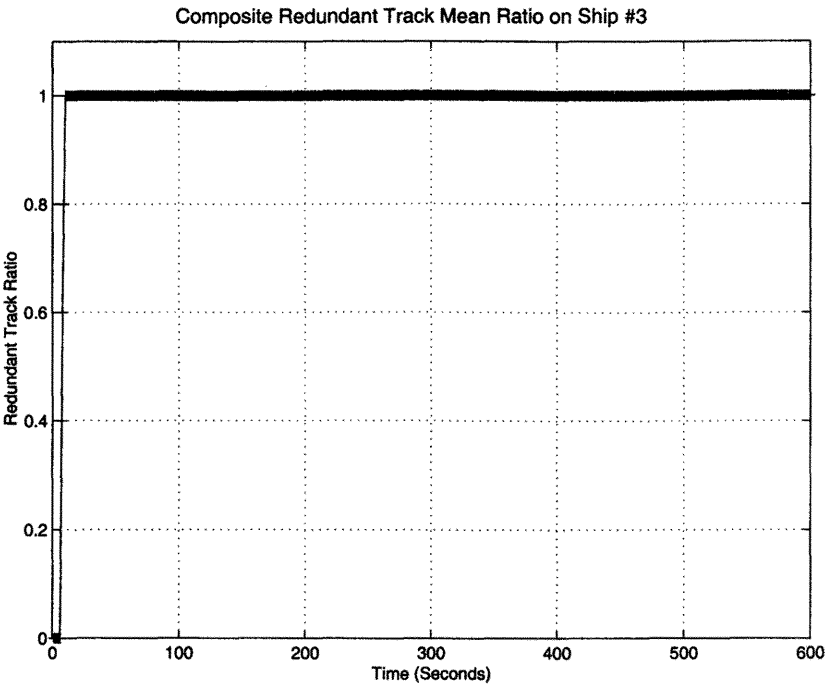


Figure 8.3: Network MFA on All Data and Network Tracks

shows the redundant track mean ratio on Ship 3 for network MFA on all data and network tracks.

8.3 Composite Track Accuracy

The second metrics category is Accuracy, which also contains two metrics: Composite Track Accuracy, and Composite Track Covariance Consistency.

8.3.1 Composite Track Position Accuracy

Composite track position accuracy is computed for each truth object on each platform as a function of metrics scoring times. At each metrics scoring time, the Euclidean distance between the truth object and the assigned composite track state estimate is calculated. The root mean square error (RMSE) statistic is computed and plotted for each truth object on each platform.

Composite Track Position Accuracy on Ship 4 tracking Fighter 4 is used as an example for comparison.

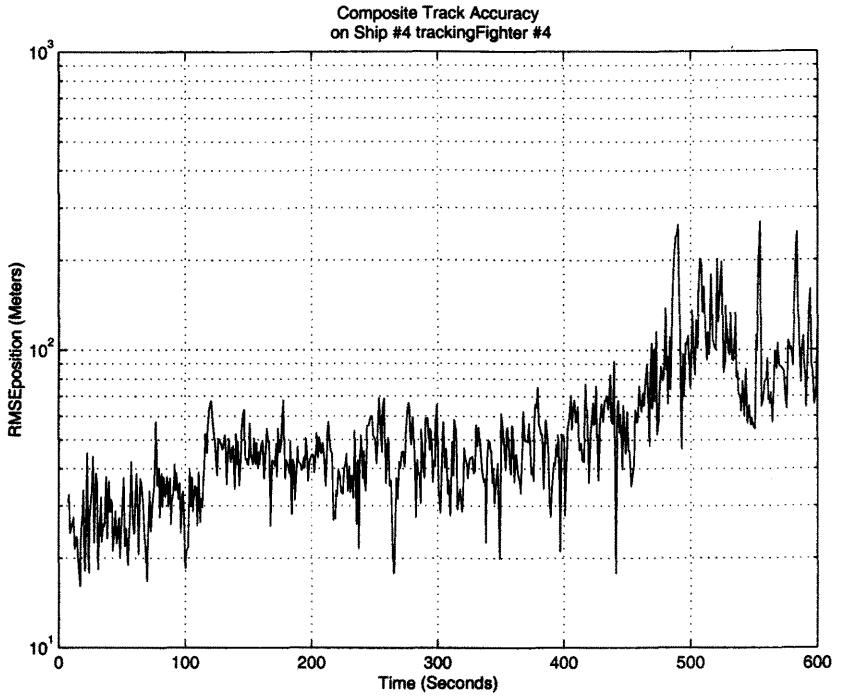


Figure 8.4: Centralized Architecture

- Centralized Architecture

  Figure 8.4 shows the position RMSE for the centralized architecture. As one can expect, the state estimates are very accurate for the first 120 seconds, before Fighter 1 merges with Fighter 3. After the merge, the accuracy worsens a little bit, but remains almost the same during the first $2g$ turn. The estimation error goes up after the four fighters merge together at 460 seconds.

- Network MFA Centralized

  The composite track position accuracy for the network MFA centralized architecture is close to that of the centralized architecture. The RMSE statistics from 400 to 600 seconds is shown in Figure 8.5.

- Network MFA on All Data and Network Tracks

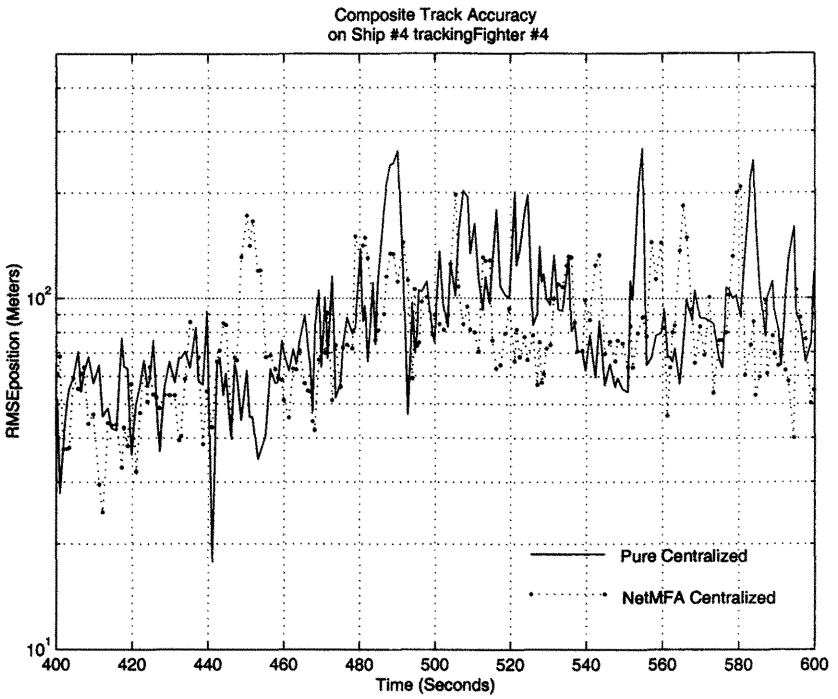


Figure 8.5: Comparison between Centralized and Network MFA Centralized

The composite track position accuracy differences between the centralized architecture and the network MFA on all data and network tracks architecture are negligible (Figure 8.6).

- Network MFA on Local Data and Network Tracks

  Figure 8.7 shows the RMSE statistics from 400 to 600 seconds for the centralized architecture and network MFA on local data. The composite trackers in network MFA on local data and network tracks only have access to their local data to make the soft data association decisions in the sliding window. Thus, especially during the $2g$ turn around 460 seconds, the estimation error is consistently bigger than the other architectures where the state estimates are made based on all data.

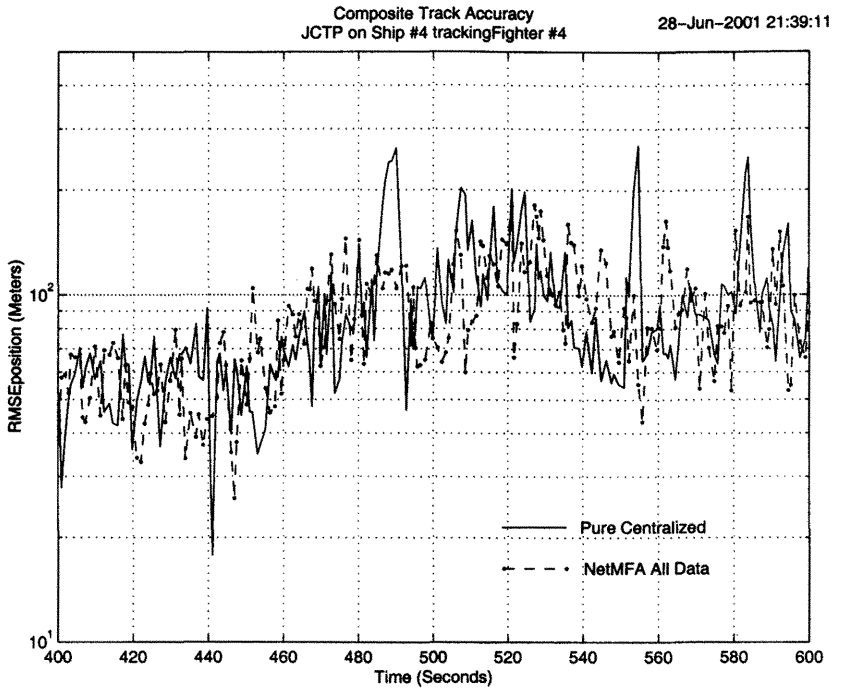


Figure 8.6: Comparison between Centralized and Network MFA on All Data

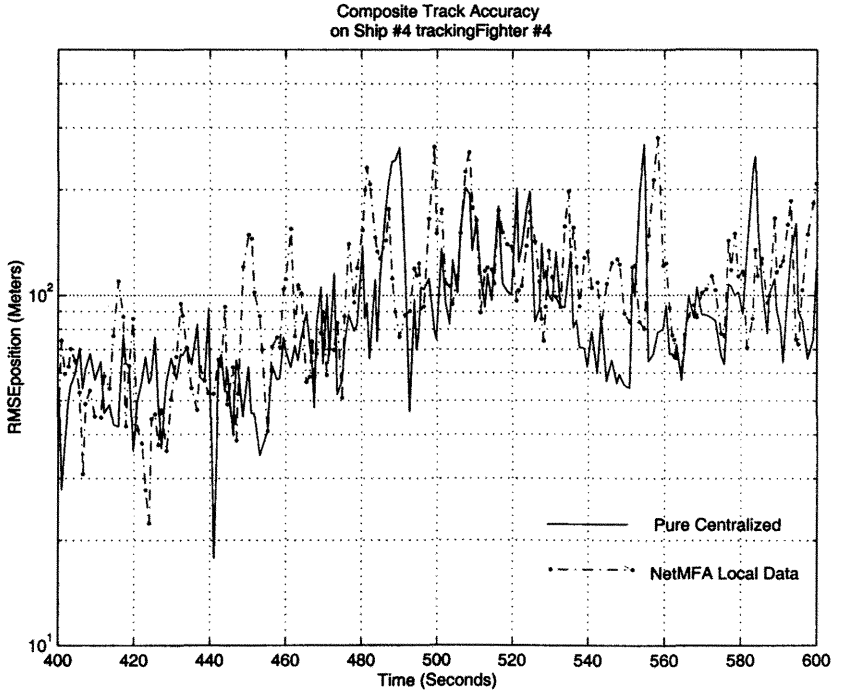


Figure 8.7: Comparison between Centralized and Network MFA on Local Data

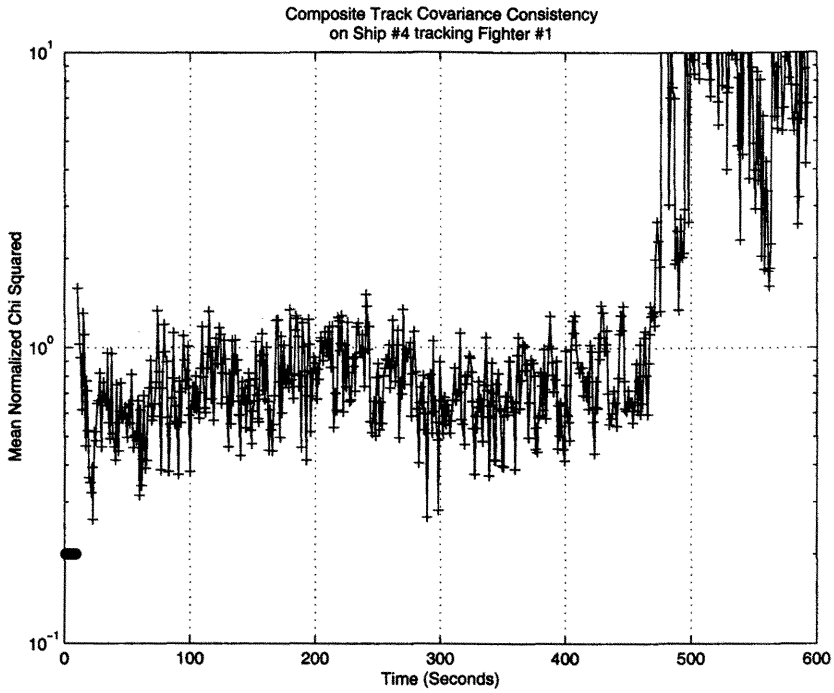


Figure 8.8: Network MFA on All Data and Network Tracks

8.3.2 Composite Track Covariance Consistency

Composite track covariance consistency is evaluated by computing the mean normalized Chi-square statistics of the composite tracker assigned to each truth object as assessed over all Monte Carlo runs.

For all four architectures, Composite Track Covariance Consistency is very similar. Figure 8.8 is an example of Ship 4 tracking Fighter 1 in the network MFA on all data and network tracks. Before Fighter 1 merges with other fighters, the observations associated to it are always correct, so the mean normalized Chi square value fluctuates around 1. However, after 460 seconds, when all fighters merge together, miss associations may occur, and the Chi square value goes up to around 10, which means the tracking filters are too optimal about the state estimations and the covariance matrices get too small.

8.4 Cross-platform Commonality History

The third category is Cross-platform Commonality History, which contains two metrics: Ratio of Non-common Composite Track Numbers and Composite Track State Estimate Difference.

8.4.1 Ratio of Non-common Composite Track Numbers

The rate of non-common composite track numbers is the number of active composite track numbers that are different between pairs of tracking platforms divided by the number of composite track numbers in the union of the two platforms' track databases. This metric is computed and plotted for each pair of platforms.

Ships 1 and 2 are chosen to show the metric here. For the centralized architecture, network MFA centralized architecture, and network MFA on all data and network tracks, where all composite trackers initiate tracks based on all data (local and remote), the ratio of non-common composite track numbers is zero. All platforms use the same network track IDs to identify the same truth object. (Figure 8.9)

For the architecture of network MFA on local data and network tracks, the ratio of non-common composite track numbers is different at the beginning couple of metrics scoring times, and then goes back to 0 afterwards, as is shown in Figure 8.10. The composite trackers in this architecture initiate new tracks based on local data only, and then broadcast them to remote platforms to be fused. Thus, different platforms initiate tracks (using different local track IDs) at different times due to different data rates of the sensors on board. However, after all platforms get the remote new tracks, the fusion process is correct, and all tracks associated with the same truth object have the same network track ID across the network.

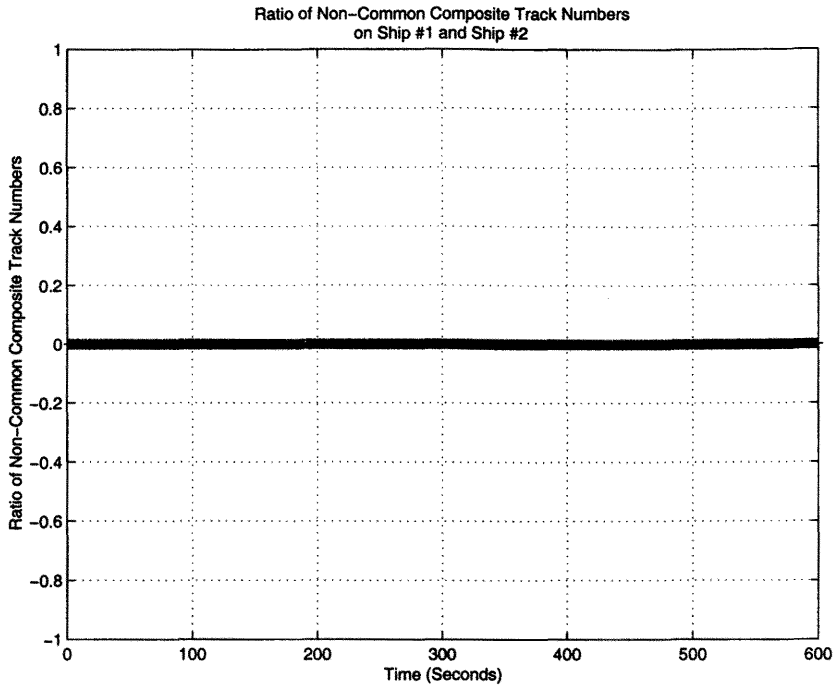


Figure 8.9: Centralized, Network MFA Centralized, Network MFA on All Data

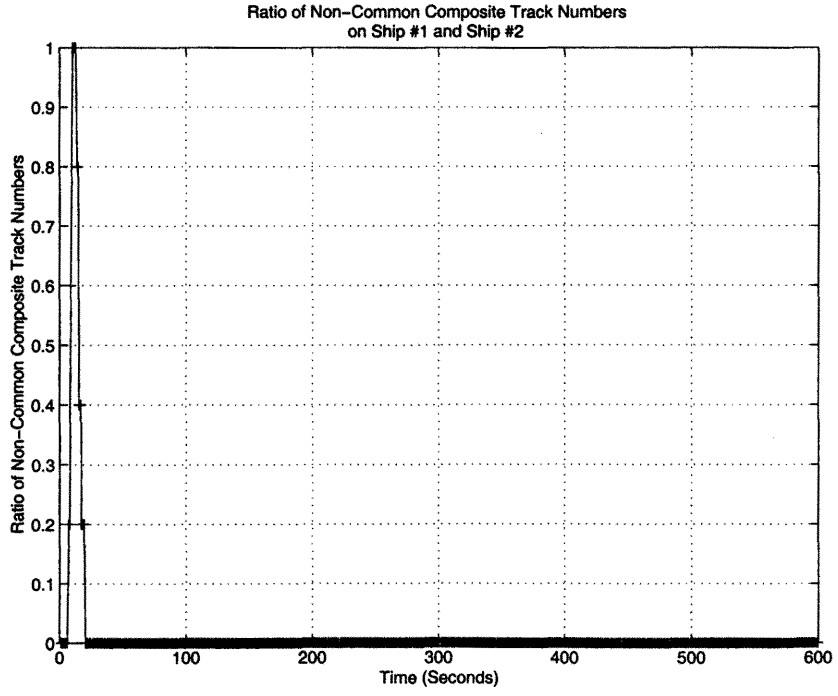


Figure 8.10: Network MFA on Local Data and Network Tracks

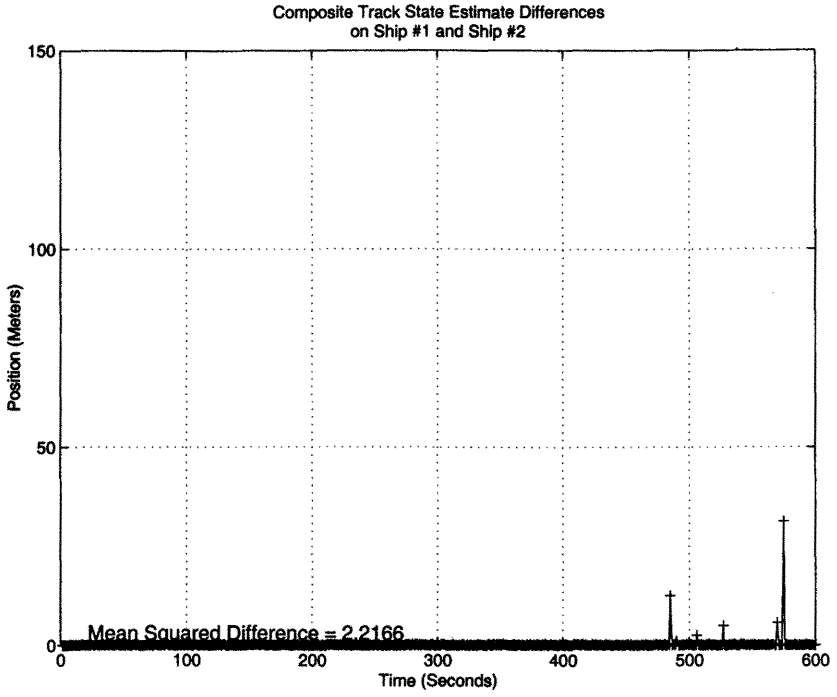


Figure 8.11: Centralized Architecture

8.4.2 Composite Track State Estimate Differences

The composite track state estimate differences are the Euclidean distances between the position estimates of tracks held by pairs of platforms for composite tracks with the same active composite track number.

- Centralized Architecture

  For the centralized architecture, the composite tracker broadcasts track states of its most recent soft (or hard) decisions to all other platforms in the network after processing each frame of data. As is shown in Figure 8.11, except for a couple metrics scoring times, the Composite Track State Estimate differences are negligible.

- Network MFA Centralized

  For the network MFA centralized architecture, each composite tracker processes frames of data, and makes its own fixed data association decisions,

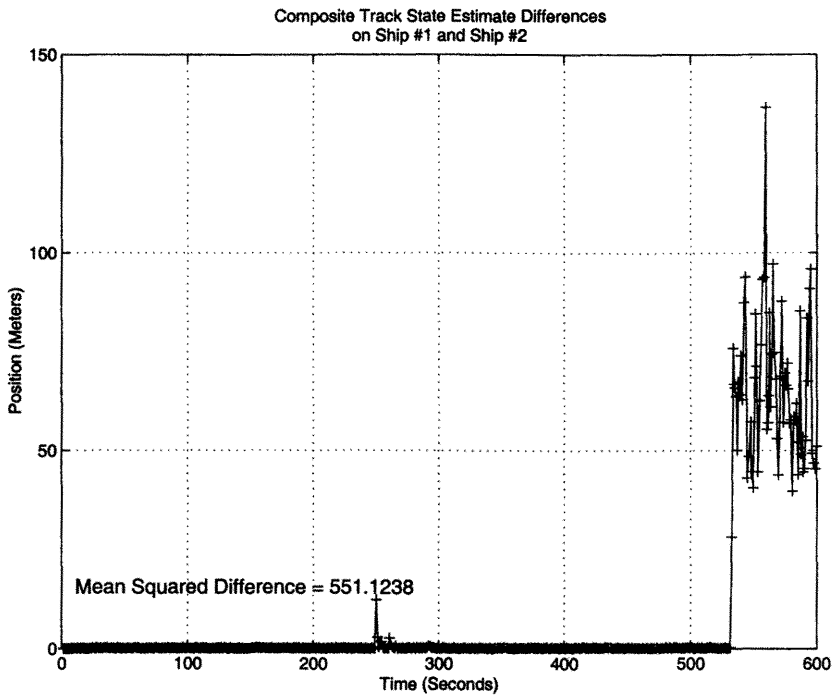


Figure 8.12: Network MFA Centralized

regardless of other trackers in the network. Thus, as Figure 8.12 illustrates, there is almost no differences before all fighters merge together. However, once Ship 1 and Ship 2 make one different association decision, that effects all successive decisions, because there is no mechanism built in to correct the association differences. Therefore, after four fighters merge together, the position estimate differences stays around 70 meters.

- Network MFA on Local Data and Network Tracks

  For the architecture of network MFA on local data and network tracks, the composite track position estimate differences are shown in Figure 8.13. The state estimates are computed using the most recent soft decisions based on the network tracks and the local frames in the window. Different platforms have different local frames in the sliding window, so the state estimates are different across the platforms.

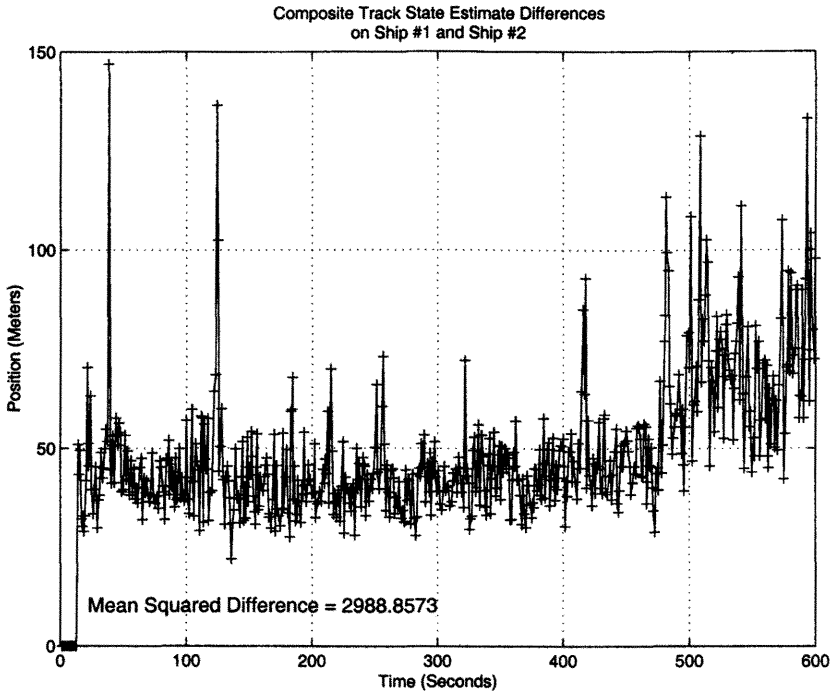


Figure 8.13: Network MFA on Local Data on Network Tracks

- Network MFA on All Data and Network Tracks

  For the architecture of network MFA on all data and network tracks, the composite tracker is allowed to make data association decisions on local frames. The associations between network tracks and remote frames are fixed based on remote AMMFrames. As is shown in Figure 8.14, the composite tracker position estimate differences are pretty small, except at a couple of metrics scoring times. The differences are mainly caused by processing delays. After the local composite tracker makes irrevocable decisions on its local frame, the time it takes for remote composite trackers to finish processing the remote AMMFrame may not be negligible at all times. However, unlike the network MFA centralized architecture, the composite trackers can correct their differences, so that the differences only occur at a discrete set of metrics scoring times.

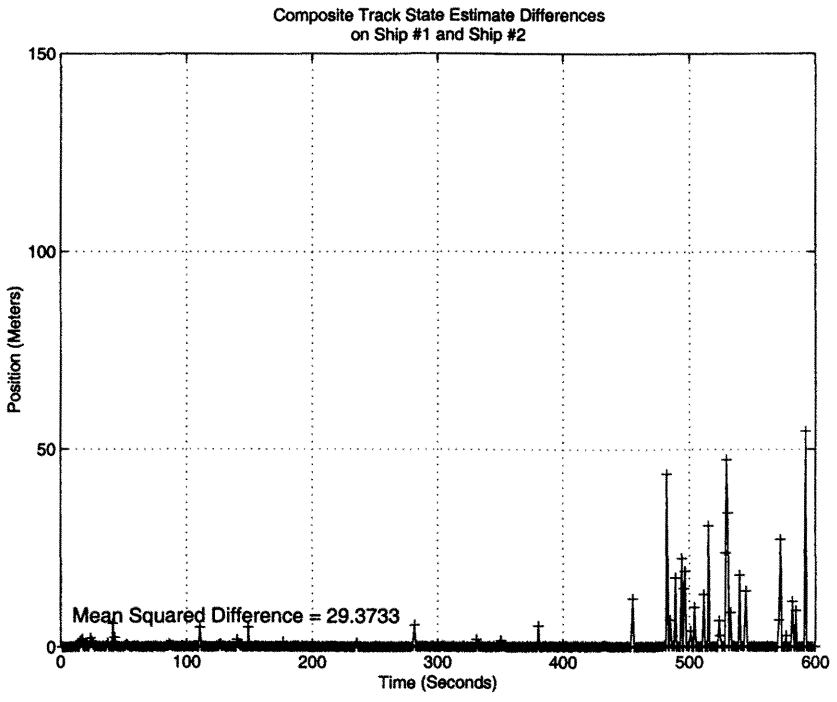


Figure 8.14: Network MFA on All Data

8.5 Communication Data Loading

Communication data loading is the total amount of data that all of the platforms send to the communication link for transmission to other platforms during a scoring interval.

- Centralized Architecture

  There are three types of messages transmitted across the network:

  1. Measurement report: all measurement reports are sent to the composite tracker by remote platforms. The composite tracker by default resides on platform 1.

  2. Track state: for each track update, a track state message, which includes a state vector (6×1), an upper half of the covariance matrix (6×6) and a track ID, is sent back to all the platforms by the composite tracker.

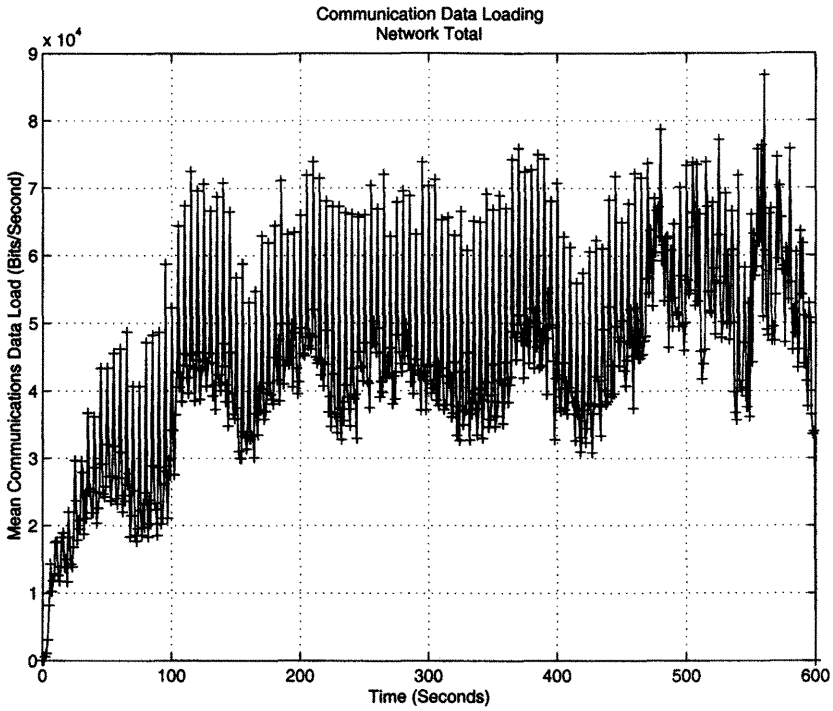


Figure 8.15: Centralized Architecture

3. Track drop: for each dropped track, the composite tracker broadcasts a track drop message that contains a track ID.

As discussed in Chapter 2, the communication loading is huge in this architecture (Figure 8.15), due to the fact that track states are broadcast back to all platforms at each update.

- Network MFA Centralized

  As discussed in Chapter 4, the messages passed around the network are only measurement reports. As is shown in Figure 8.16, the communication loading is now only a third compared to the centralized architecture.

- Network MFA on Local Data and Network Tracks

  In this architecture (Chapter 5), the messages broadcast by the composite trackers are AMRs, new tracks, and end AMRFrame messages.

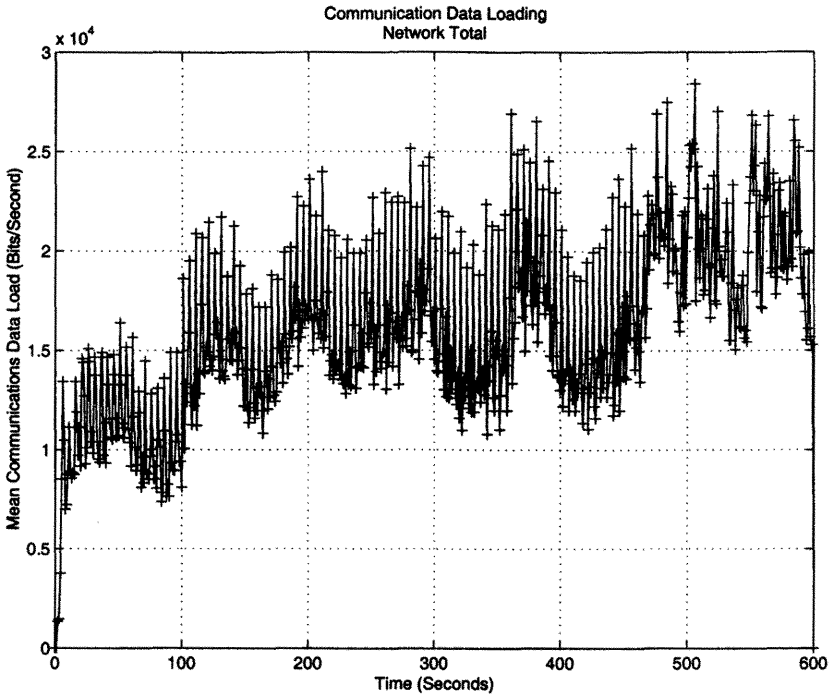


Figure 8.16: Network MFA Centralized

The communication loading is depicted in Figure 8.17. Compared with Figure 8.16, the communication loading is actually a little bit higher than the network MFA centralized case. The reason is three fold:

1. The scenario we are running does not have clusters. Thus, there are hardly any false alarms. almost all measurements are returns from truth objects, which are then categorized by the local composite tracker as AMRs.
2. AMRs add two additional fields compared with measurement reports: a tracker ID and an AMRFrame ID.
3. The new track messages are expensive to be transmitted across the network.

- Network MFA on All Data and Network Tracks

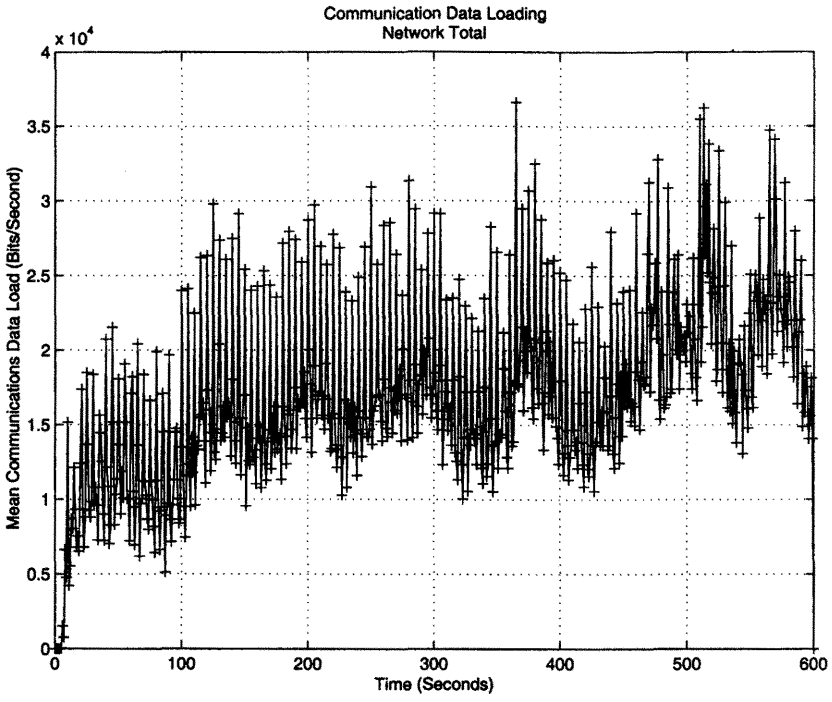


Figure 8.17: Network MFA on Local Data and Network Tracks

For the network MFA on all data and network tracks, the messages being broadcast are measurement reports, end frame messages, AMMs, new tracks and end AMMFrame messages. Figure 8.18 illustrates the communication loading for this architecture.

8.6 Conclusions and Comparisons

The computational results presented in this chapter demonstrate that all four architectures have a perfect or near perfect score on a large class of metrics, namely, spurious track mean ratio, redundant track mean ratio, track breakage, composite completeness. These metrics generally measure the continuity of the tracks and a one-to-one match of the computed tracks to truth objects. This is quite remarkable given that the local trackers and data arising from the local trackers are often very imperfect in these metrics. The reason for the success of the network tracker is

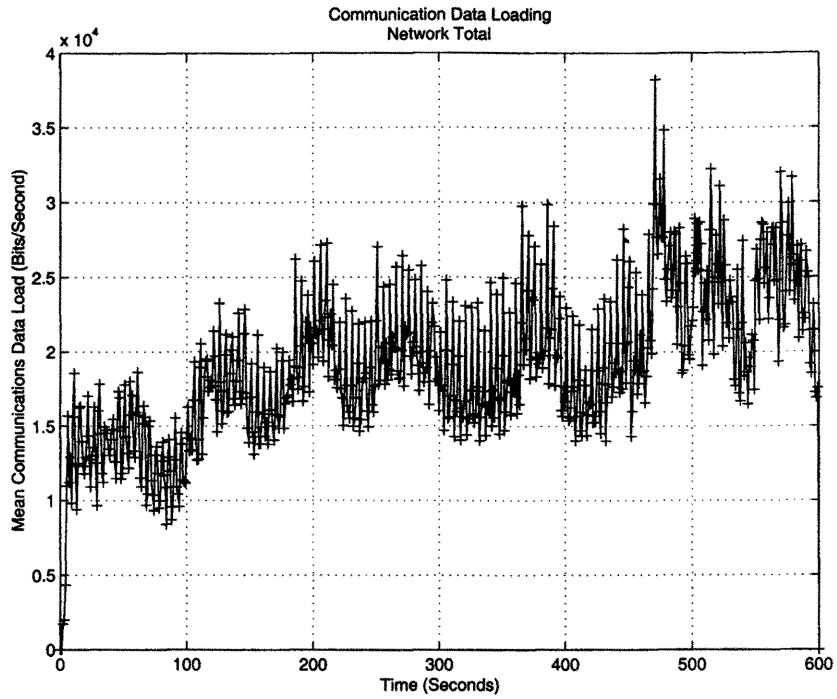


Figure 8.18: Network MFA on All Data and Network Tracks

that MFA at the network level can correct many if not all of the errors in the local tracker due to the more global information available at the network level.

The accuracy and consistency of all four architectures are reasonably close to each other with Network MFA on Local Data and Network Tracks being slightly worse than the other three architectures which use all data. Often it is the case that the local sensor tracker on a platform may produce erroneous tracks. Using local data only in the MFA may not be able to correct the problems as well as using all data in the MFA.

With respect to a consistent air picture, the centralized architecture naturally has an almost perfect cross-platform commonality history as measurement by non-common composite tracker numbers and composite state estimate differences. Imperfections are due primarily to processing delays. The network MFA centralized architecture puts a centralized tracker on each platform and no track matching mechanism is applied. Each composite tracker makes its own tracking

decisions regardless of the other trackers in the network. Thus, the performance in cross-platform commonality history degrades. The use of the rule that each platform is in charge of assigning its own measurements to the network tracks leads to improvements found in the architectures of network MFA on local data and network MFA on all data. Each composite tracker is only allowed to make tracking decisions on its local data only, and associations between the network tracks and remote data are fixed based on remote decisions.

We might note that the tracking accuracy is slightly better when using Network MFA on All Data and Network Tracks than with Network MFA on Local Data and Network Tracks. The network MFA on all data and network tracks has an almost perfect cross-platform commonality history except at a discrete set of metrics scoring times due to processing delays. Network MFA on Local Data and Network Tracks is worse, primarily due to the use of soft data association decisions, but improves in these metrics if hard decisions are used.

The centralized architecture has the biggest communication loading, because track states are broadcast to all platforms at each measurement update. All network MFA architectures proposed cut down the communication loadings to one third of that of the centralized architecture. The communication loading for network MFA on all data is always a little heavier than the network MFA centralized architecture, which is caused by the extra messages sent in the network (end frame messages, AMMs, new tracks and end AMMFrame messages). In the future, the composite trackers can only broadcast gated measurements instead of all raw measurements, which might help to cut down the communication loading even more in dense clusters. In the architecture of network MFA on local data and network tracks, the messages transmitted are fixed data association decisions instead of raw measurements.

Thus, in dense clusters, if the communication bandwidth is limited, the network MFA on local data and network tracks is the preferred architecture.

With respect to processing loads, the network architectures are computationally more expensive than the centralized one. In the centralized architecture, there is only one composite tracker in the network, whereas in all network architectures, each platform has its own composite tracker. However, network MFA on all data and network MFA on local data require ten to twenty percent more computing power than network MFA centralized.

Chapter 9

SIMULATION RESULTS, COMPARISON AND DISCUSSIONS: IMPERFECT COMMUNICATION LINK

9.1 Scenario Description

The same scenario shown in Figure 8.1 is used. However, an imperfect communication network is used. It models two general phenomena: the time delay incurred while transmitting a message from one platform to another, and the possibility that a message does not arrive at its intended destination.

The time delay consists of two parts. The first part is deterministic based on the relative positions of the platforms. These deterministic delays are unique for each pairing of originating platform to destination platform. The second component of the time delay results from queuing at the input of the originating communication node. It is modeled as a random quantity.

The probability that a message is not successfully transmitted also results from a combination of factors. The first factor is the small probability that a message is lost due to the imperfect reliability of the transmission. The second factor results from a process that is referred to as triage. If the the communication link determines that the total time delay associated with message transmission is greater than 1.35 seconds, it does not send the message.

On the tracker side, the max communication delay in the parameter file is set to be 1.8 for all the architectures, to account for the communication delay.

Due to the fact that there is lost data in the network, for the network MFA on Local data and network tracks, a remote AMRFrame is regarded as ready if *maxCommunicationDelay* seconds have passed, and no more information has arrived yet.

Similarly, for network MFA on all data and network tracks, a remote AMM-Frame is regarded as ready if *maxCommunicationDelay* seconds have passed, and no more information has arrived. For a remote frame of observations, if the end-Frame message has arrived, *maxCommunicationDelay* seconds have passed, and no more observations for that frame arrive, then the remote frame is regarded as ready. However, if the endFrame message is lost, then the entire frame won't be used, because in that case, the remote platforms don't know the exact rules to link the sensorFrames together.

9.2 Composite Track Ambiguity

9.2.1 Spurious Track Mean Ratio

For the centralized architecture, the network MFA centralized, and the network MFA on all data and network tracks, the spurious track mean ratio is zero on all the platforms.

However, for the architecture of network MFA on local data and network tracks, the Spurious Track Mean Ratio is up to 0.05 on Ship 1 and 3 for the first 200 seconds. As is shown in Figure 9.1, there are two un-assignable composite tracks that diverge on one of the Monte Carlo runs. One is short, and the other is around 60 seconds long. This may due to the fact that the composite trackers in this architecture initiate tracks based on local data only.

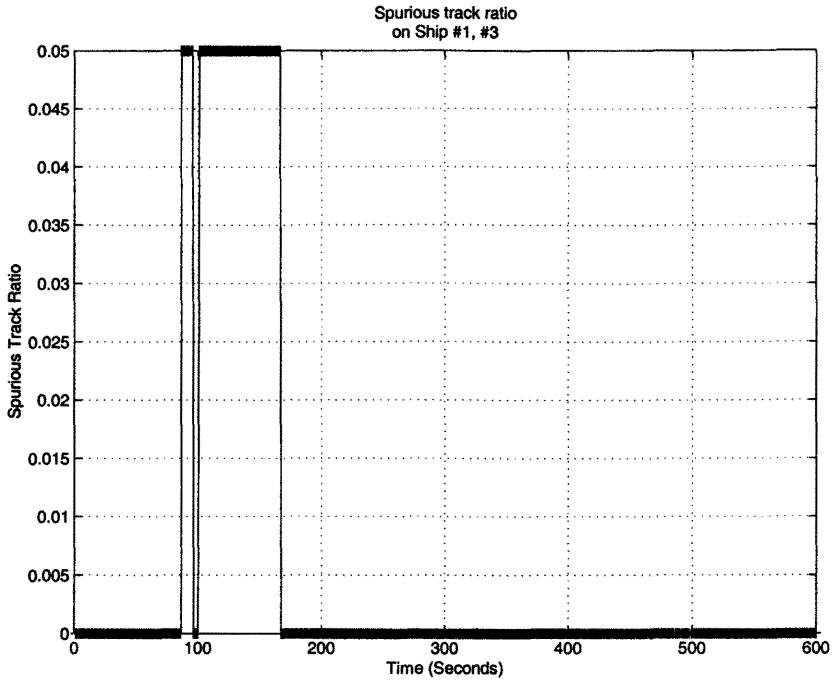


Figure 9.1: Network MFA on Local Data and Network Tracks

9.2.2 Redundant Track Mean Ratio

- Centralized Architecture

  The redundant track mean ratio is always 1 on all the platforms.

- Network MFA Centralized

  As is shown in Figure 9.2, there are no redundant tracks on Ship 3. Ship 2's redundant track ratio goes up to 1.2 at the beginning, and then drops back to 1.0. Ship 1's redundant ratio goes up to 1.05 after 250 seconds or so and remains there. The redundant ratio on Ship 4 is 1.05 at 120 seconds, when Fighters 3 and 4 merge together, and goes up to 1.1 after all four fighters merge together at 460.

- Network MFA on Local Data and Network Tracks

  Figure 9.3 shows the redundant track ratio in network MFA on local data and network tracks. Only on Ship 3, the redundant track ratio goes back to

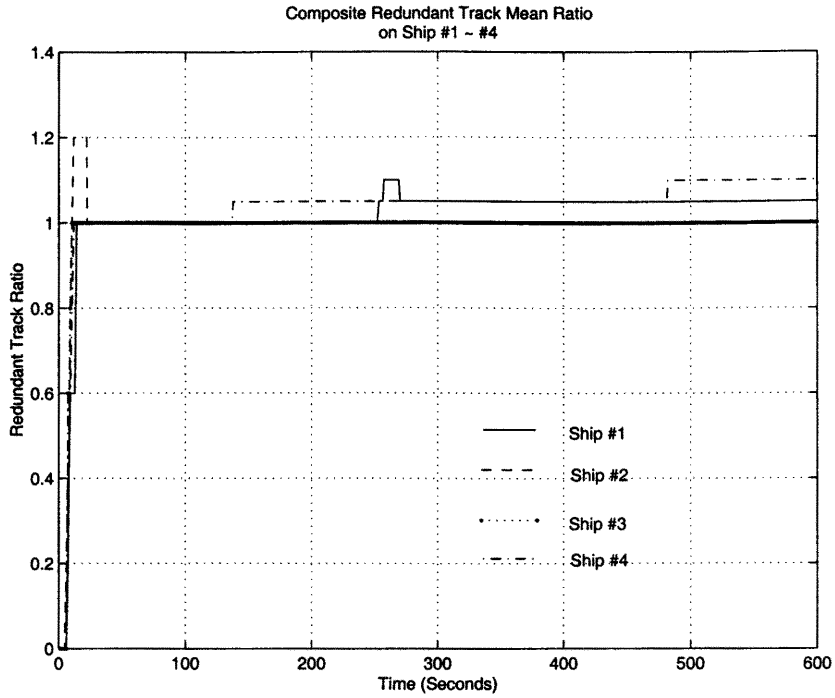


Figure 9.2: Network MFA Centralized

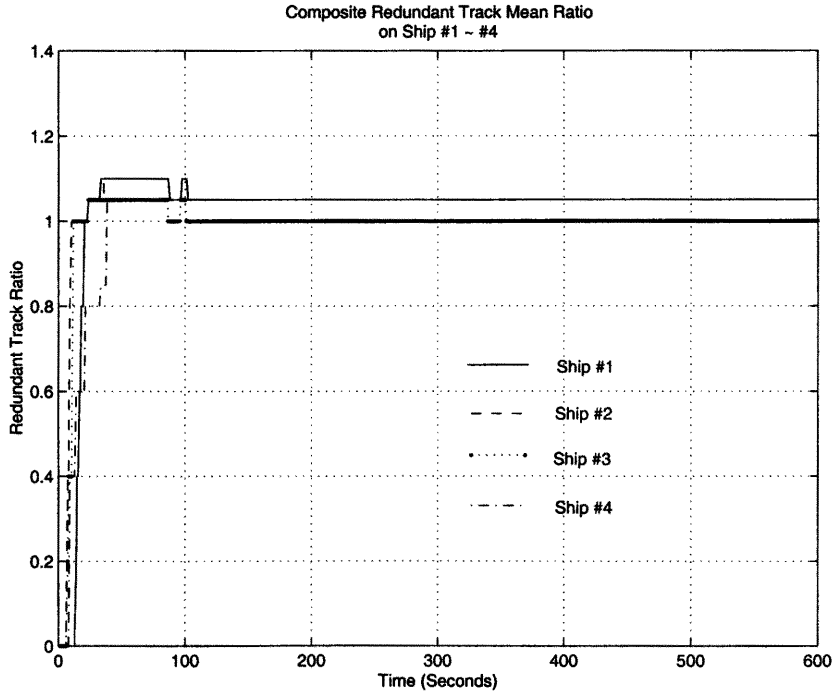


Figure 9.3: Network MFA on Local Data and Network Tracks

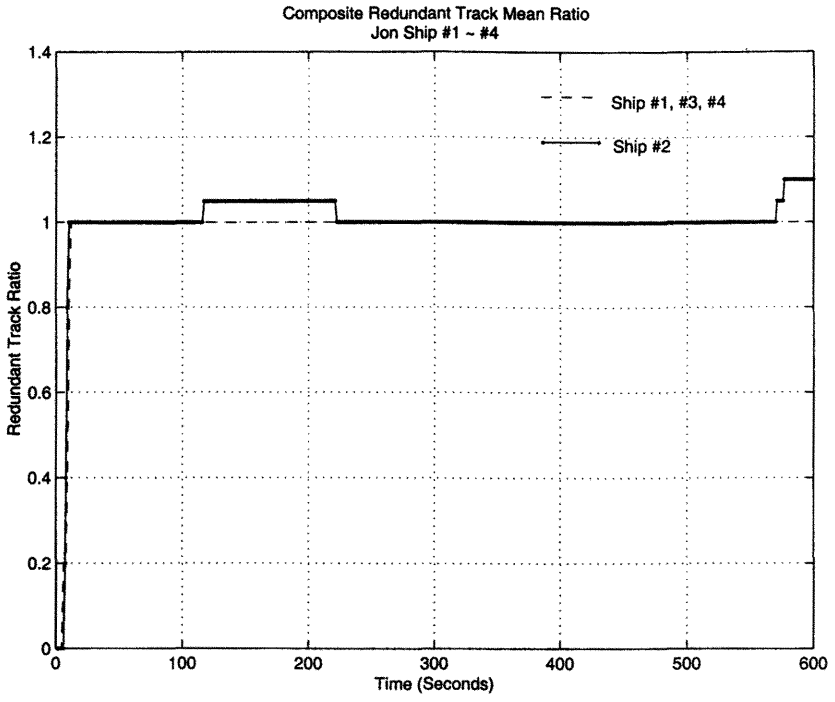


Figure 9.4: Network MFA on All Data and Network Tracks

1.0, on all others ships, the ratio remains at 1.05, which means there is one redundant track in one of the Monte Carlo runs.

- Network MFA on All Data and Network Tracks

  The redundant track ratio is shown in Figure 9.4. Only on Ship 2, there are some redundant tracks. All other ships have the correct number of tracks initiated as the truth objects.

  It can be concluded that the network MFA on all data and network tracks architecture performs better in the redundant track mean ratio metric compared with the other two network architectures.

9.2.3 One Way to Improve: Longer Tracking Filter Initiation Length

As explained in Chapter 8, four observations are required to initiate a tracking filter. If the filter initiation length is set to be five, the spurious track mean ratio

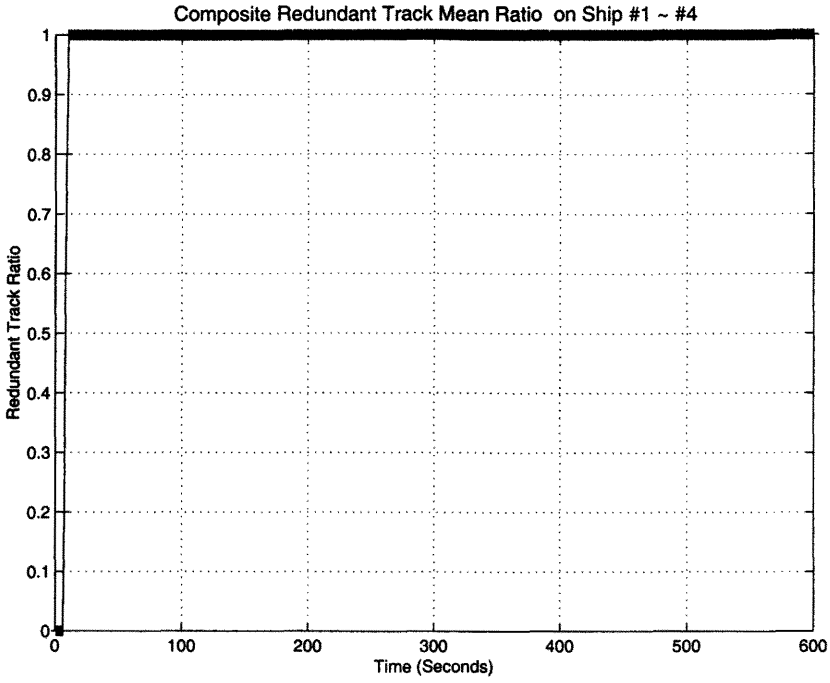


Figure 9.5: Network MFA on Local Data and Network Tracks: Improved

and redundant track mean ratio improve in all architectures. The reason is that filter initial state estimates are computed using weighted least squares (WLS) based on at least five observations, and the chances that all five observations misalign is much smaller than four observations misalign.

In the network MFA on local data and network tracks, the spurious track mean ratio goes back to zero on all platforms. And there are no redundant tracks on any platform, as is shown in Figure 9.5.

9.3 Composite Track Accuracy

9.3.1 Composite Track Position Accuracy

- Centralized Architecture

  With the imperfect communication link, the accuracy of the centralized architecture of Ship 4 tracking Fighter 4 is shown in Figure 9.6. Compared to the case with the perfect communication link (Figure 8.4), the accuracy is

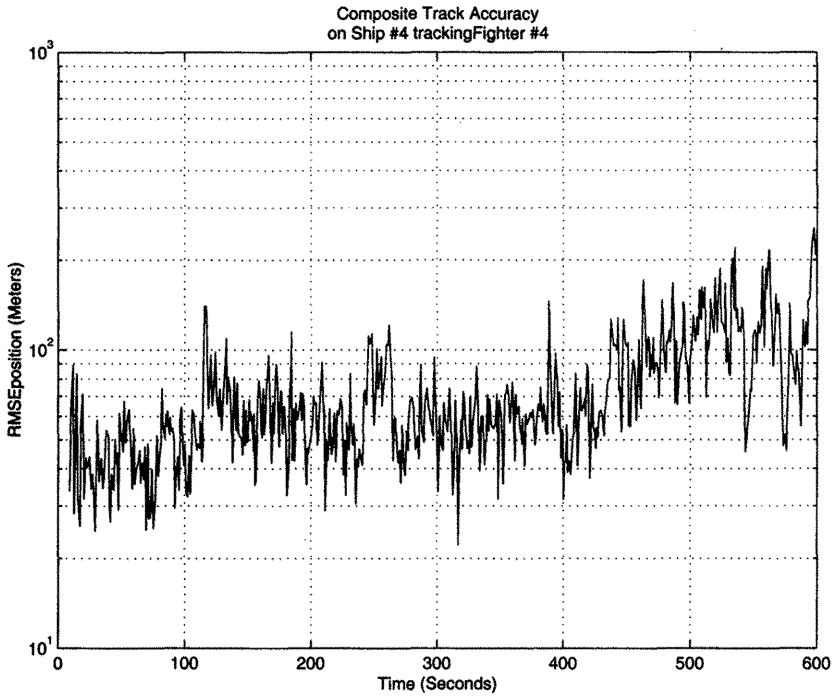


Figure 9.6: Centralized Architecture

worse. However, the accuracy in the centralized architecture is still regarded as the lower bound of all the network architectures proposed.

- Network MFA Centralized

  Figure 9.7 illustrates the composite track position RMSE of Ship 4 tracking Fighter 4. Figure 9.8 shows the accuracy from 400 To 600 seconds of the centralized architecture versus the network MFA centralized. It can be concluded that the differences are negligible. The network MFA centralized architecture actually puts a centralized composite tracker on each platform. And each composite tracker tracks independently of each other. Thus, the accuracy performance should be close to the centralized architecture.

- Network MFA on Local Data and Network Tracks

  In the network MFA on local data and network tracks, the composite track position accuracy is shown in Figure 9.9. The errors are huge at the tracking

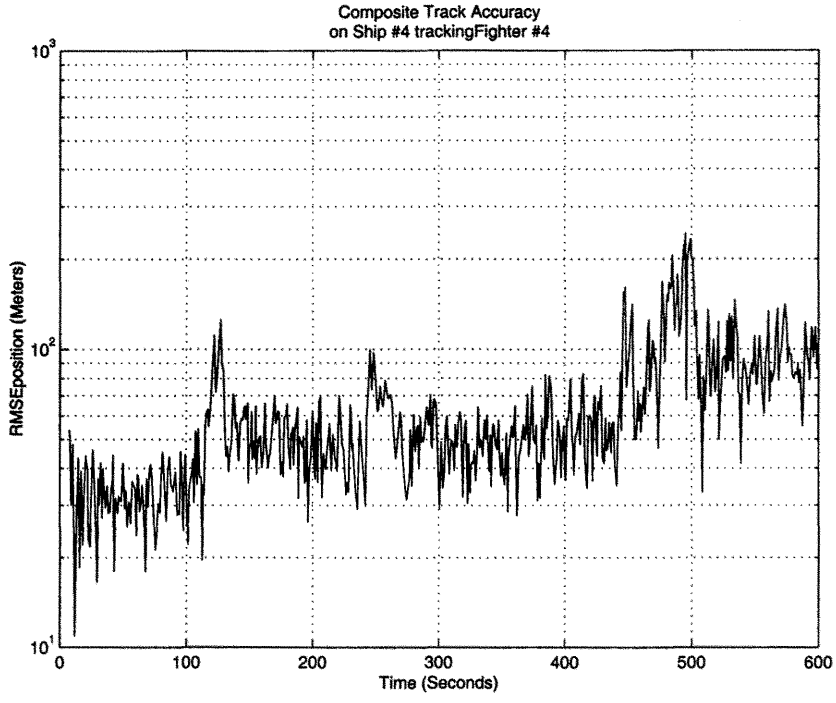


Figure 9.7: Network MFA Centralized

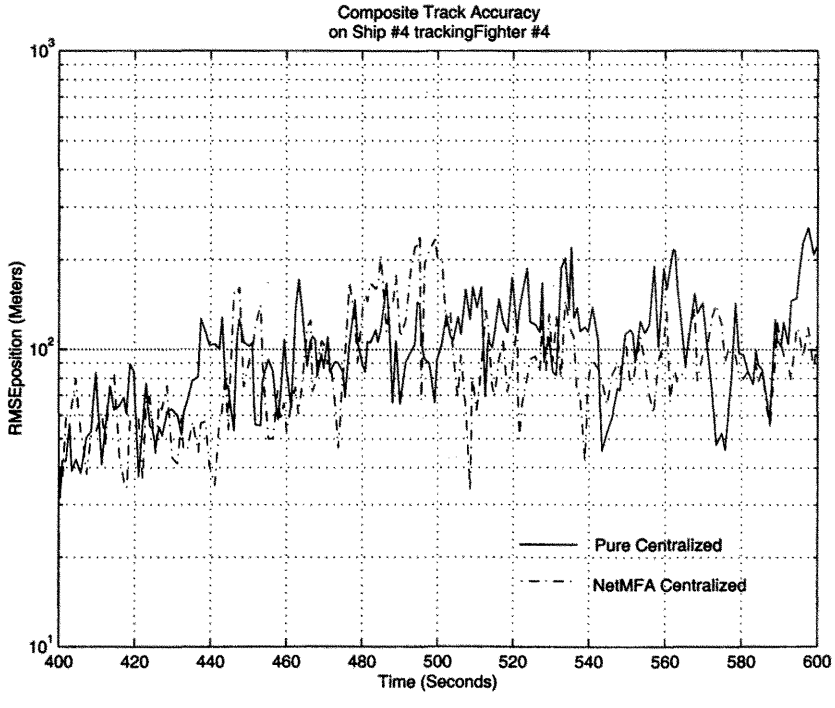


Figure 9.8: Comparison between Centralized and Network MFA Centralized

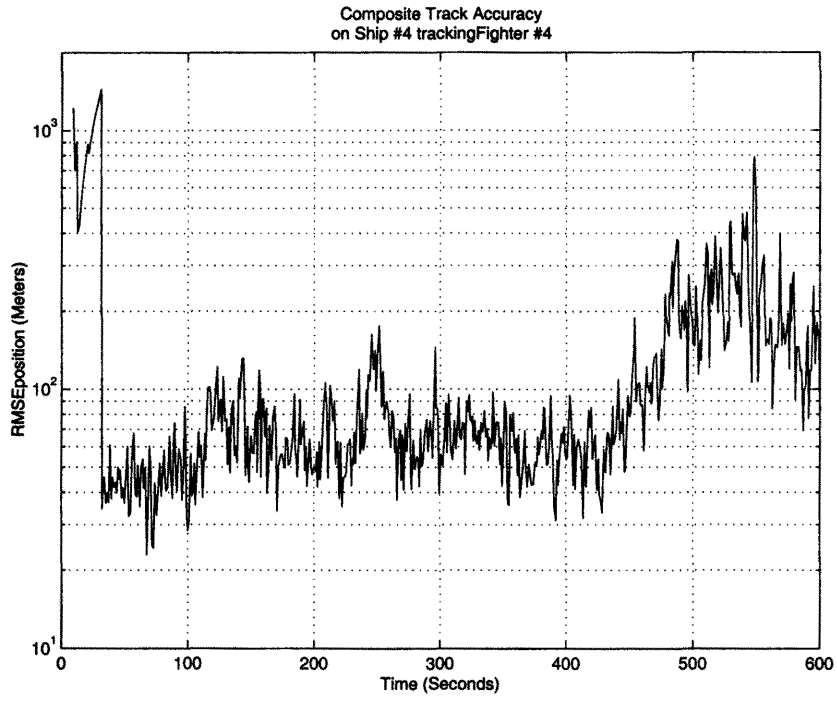


Figure 9.9: Network MFA on Local Data and Network Tracks

initiation step, which is again due to the fact that the composite tracker initiate new tracks on local data only. The accuracy improves after remote information comes in. However, compared with the centralized architecture, the accuracy is still worse, as is shown in Figure 9.10.

After 460 seconds, when all four fighters merge together, it can be shown that the accuracy of the network MFA on local data and network tracks becomes much worse than the centralized case. This is due to the delays in the network. The processing delays from when fixed data association decisions are made on a local frame and broadcast until the remote platforms process the corresponding AMRFrame are significant. Thus, for a metrics scoring request, the composite tracker has less information available to make the prediction.

- Network MFA on All Data and Network Tracks

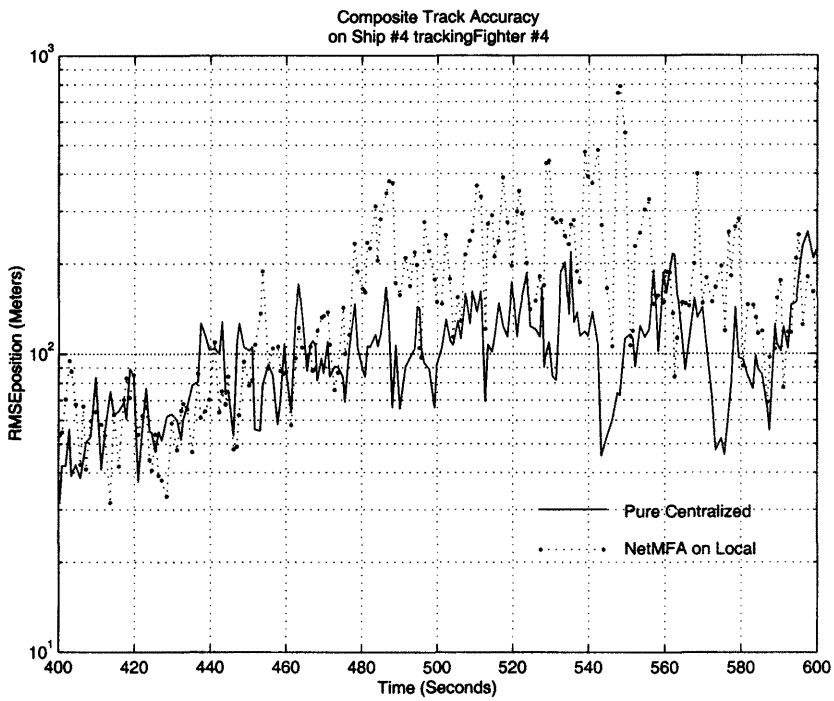


Figure 9.10: Comparison between Centralized and Network MFA on Local Data

Figure 9.11 shows the composite tracker position accuracy of the Network MFA on all data and network tracks. In the track initiation step, the accuracy is close to what's being achieved in the centralized architecture. However, the accuracy is a little worse during the turns and after the four fighters merge together. The reason is that processing time necessary for decisions on local frames to be used by remote composite trackers increases under the imperfect communication links.

Predict Using "soft" vs. "hard" vs. "softPlus"

Figure 9.13 shows the composite track position accuracy of Ship 4 tracking Fighter 4 in the network MFA centralized architecture. However, it can be shown that when the *predictFrom* option is set to be "softPlus", where all available information is used to estimate the target positions, the accuracy improves significantly.

The differences using "soft" and "hard" are negligible, though the "soft" is a little bit better than the "hard" at certain times. This is due to the fact that the

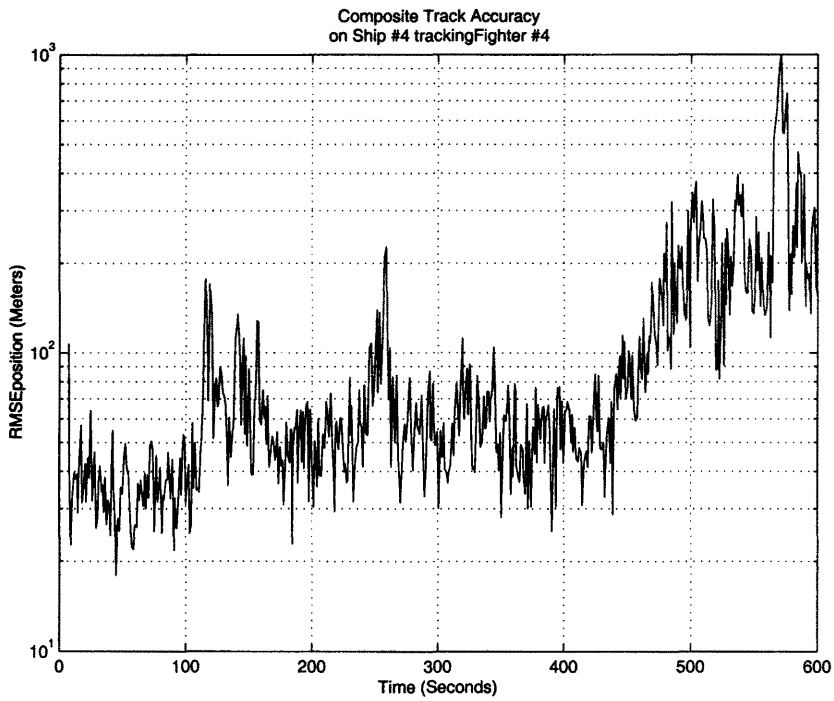


Figure 9.11: Network MFA on All Data and Network Tracks

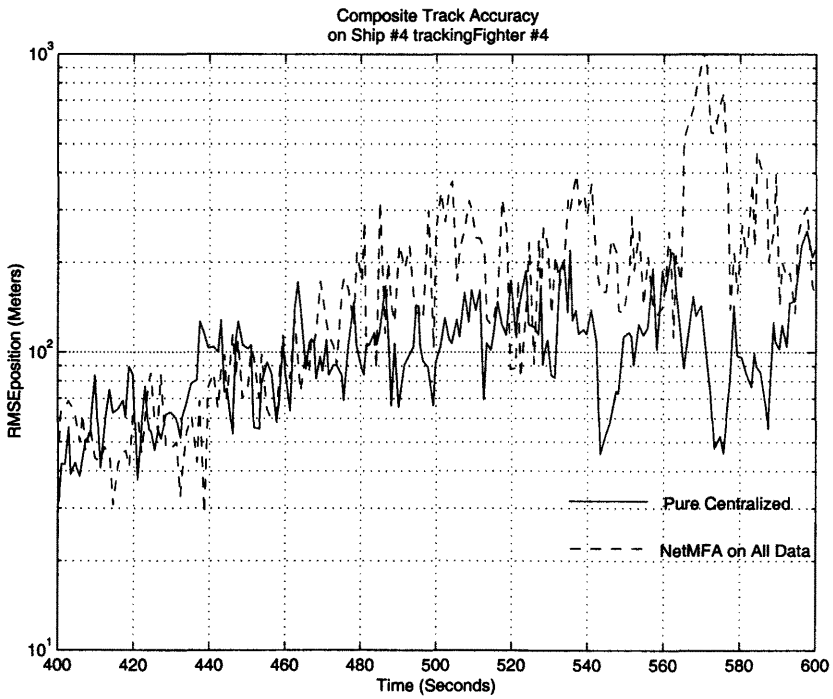


Figure 9.12: Comparison between Centralized and Network MFA on All Data

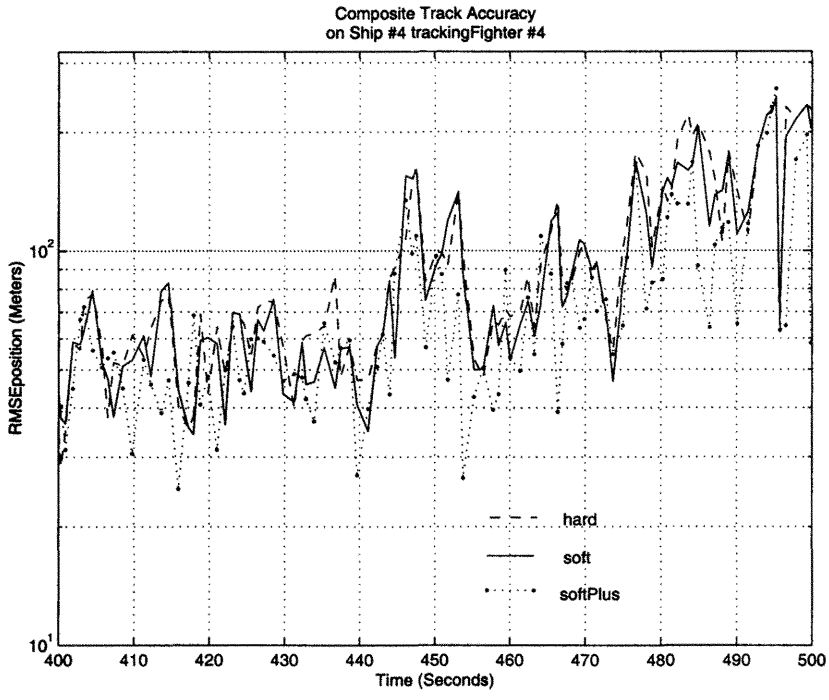


Figure 9.13: Network MFA Centralized

composite trackers are using a 6/2 window, and when a metrics scoring request occurs, there is only one frame in the track extension window. With only one additional frame, the improvements are really small. With a longer extension window, *e.g.* a 6/4 window, the "soft" option should be able to give more accurate state estimations than the "hard" option.

From the view point of processing load, the "hard" option is the cheapest. Every time there is a metrics scoring request, it just searches through the existing firm decision tracks. New data association is not necessary at all.

For both network MFA on local data and on all data, the most recent "soft" decisions might be changed with the remote information. Thus, whenever there is a metrics call, a new data association problem is set up and solved.

The "softPlus" option is the most expensive. All unfinished frames are used, and with the imperfect communication link, each composite tracker may have up to 6 or 7 unfinished frames. All those frames are appended at the end of the

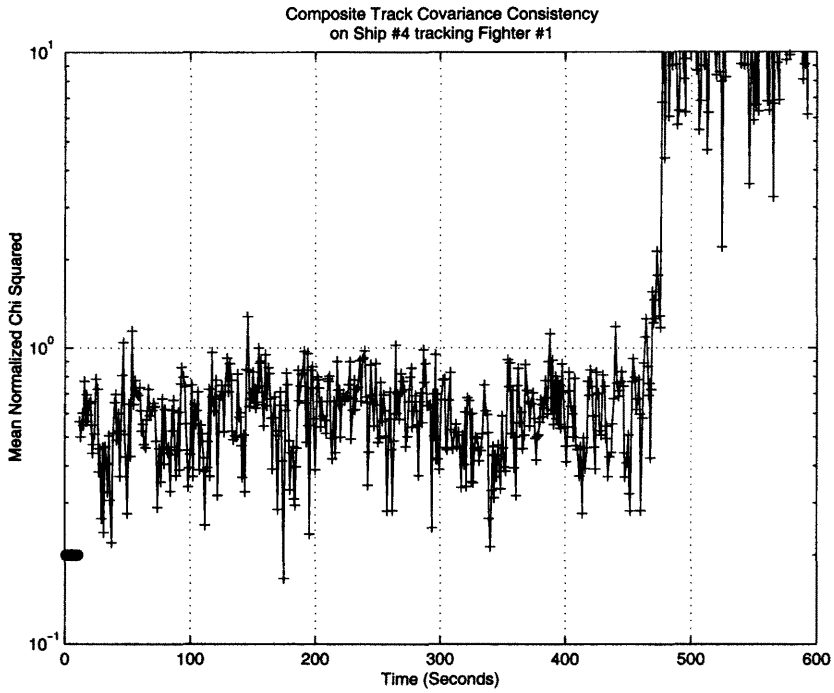


Figure 9.14: Network MFA on All Data and Network Tracks

window, and an $(M + p)$ dimensional assignment problem is set up and solved, where p is the number of unfinished frames.

9.3.2 Composite Track Covariance Consistency

The composite track covariance consistency in the imperfect communication link is close to that of the perfect communication link case. Figure 9.14 shows the covariance consistency of Ship 4 tracking Fighter 1 in the Network MFA on all data and network tracks.

With the presence of random communication delays, observations associated with each track may be out of order. However, with a refiltering window length up to 20, the composite tracker is able to handle the out of order data optimally, and give a consistent set of state estimates.

9.4 Cross-platform Commonality History

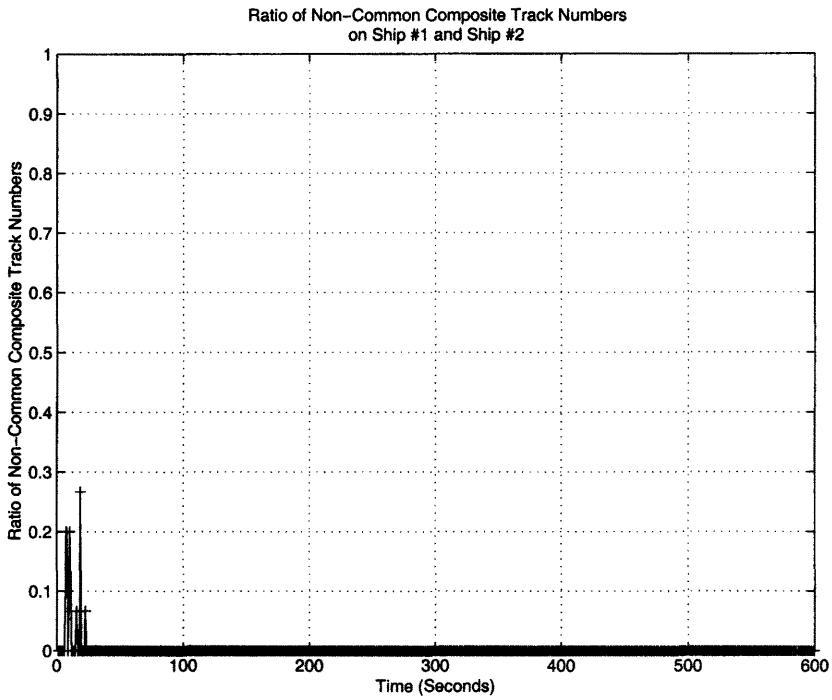


Figure 9.15: Centralized Architecture

9.4.1 Centralized Architecture

For the centralized architecture, the ratio of non-common composite track numbers is shown in Figure 9.15, and the composite track state estimate differences are shown in Figure 9.16.

The non-common composite track numbers at the beginning of the simulation are due to the fact the communication delays in the network cause track state messages arrive on different platforms at different times.

The composite track state estimate differences performance achieved in the centralized architecture is regarded as the lower bound of all the network tracking architectures we propose.

9.4.2 Network MFA Centralized

The network MFA centralized architecture puts a centralized composite tracker on each platform. Observations are broadcast to all remote platforms, so compos-

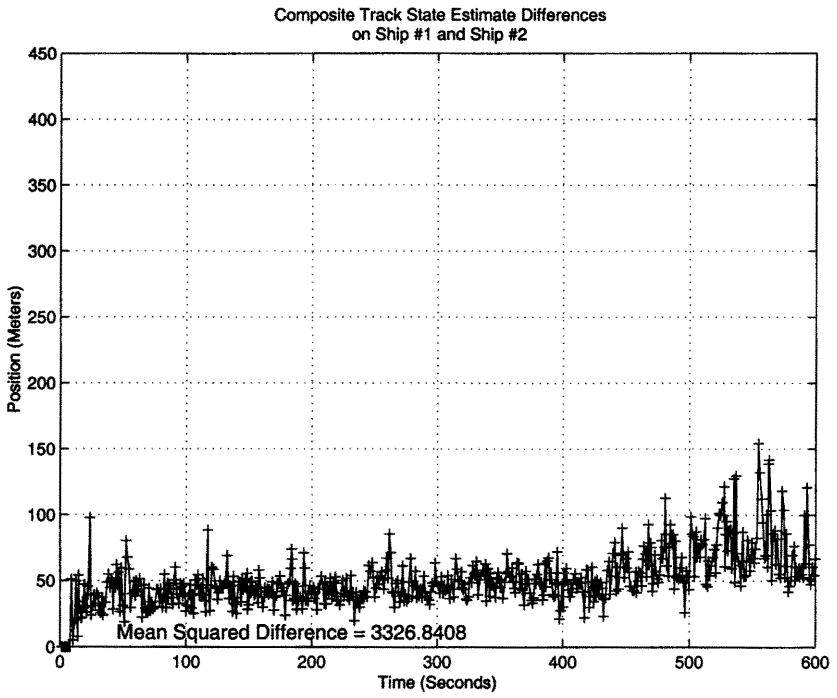


Figure 9.16: Centralized Architecture

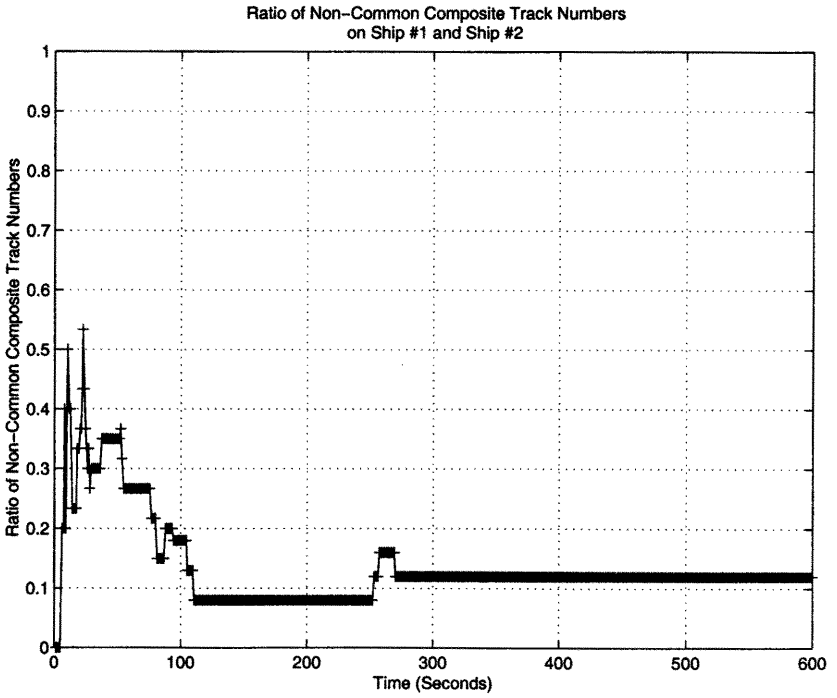


Figure 9.17: Network MFA Centralized

ite trackers in the network have access to all the data. However, each composite tracker makes its own tracking decisions regardless of the other trackers in the network.

In this architecture, the composite trackers have the same track ID bank and no track numbering scheme is applied to correlate the tracks across the network. Whenever a new track initiates, the composite tracker assigns the next available track ID to it, regardless of what the track ID is used to represent it on the remote platforms. Therefore, when there is no communication delay, the composite trackers are still capable of using the same track IDs for the same truth object. With communication delays and lost data, the same truth object may be represented using different track IDs on different platforms.

As is shown in Figure 9.17, the ratio of non-common composite track numbers stays relatively high during the entire simulation. However, the composite track state estimate differences defined in Section 8.4.2 are computed based on the state estimates for composite tracks with the same active composite track ID. Since the same track ID on different platforms is used to represent different truth objects, this metrics is not even applicable for the network MFA centralized architecture.

9.4.3 Network MFA on Local Data and Network Tracks

Figure 9.18 shows the ratio of non-common composite track numbers in the network MFA on local data and network tracks. The peak at the beginning of the simulation is caused by different track initiation times on different platforms. The non-common composite track numbers afterwards are caused by redundant tracks initiated on some platforms.

Figure 9.19 shows the composite track state estimate differences in network MFA on local data and network tracks.

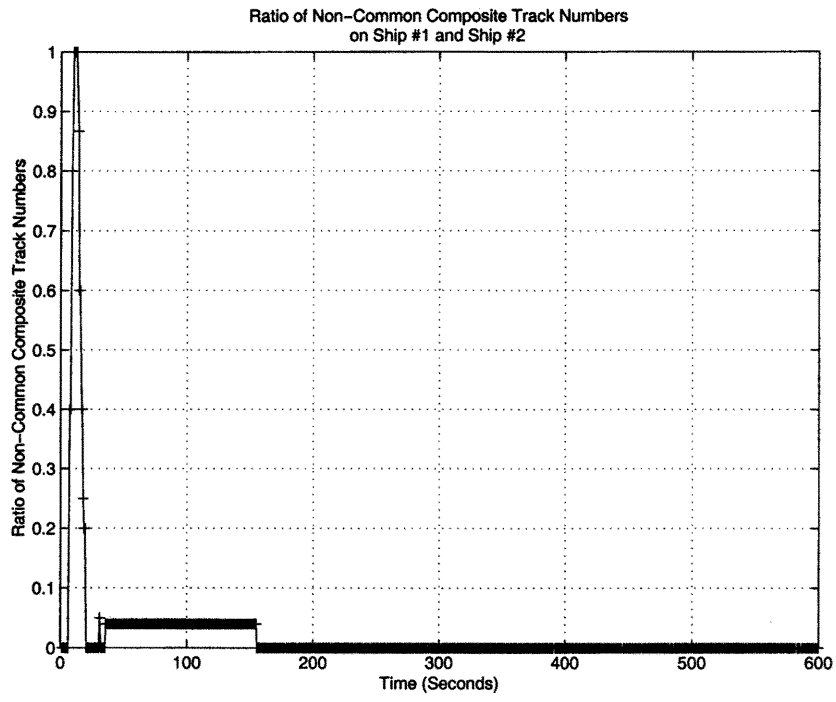


Figure 9.18: Network MFA on Local Data and Network Tracks

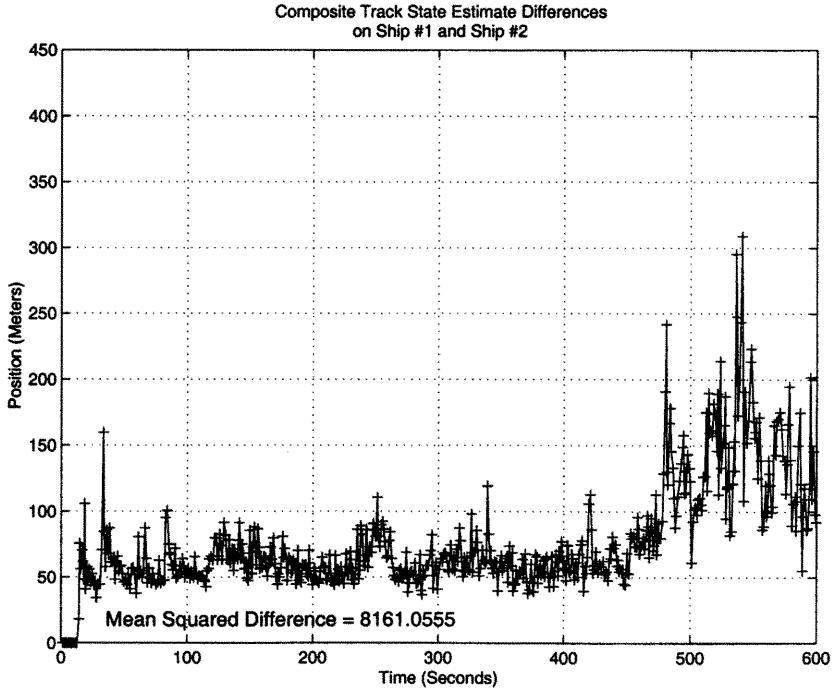


Figure 9.19: Network MFA on Local Data and Network Tracks

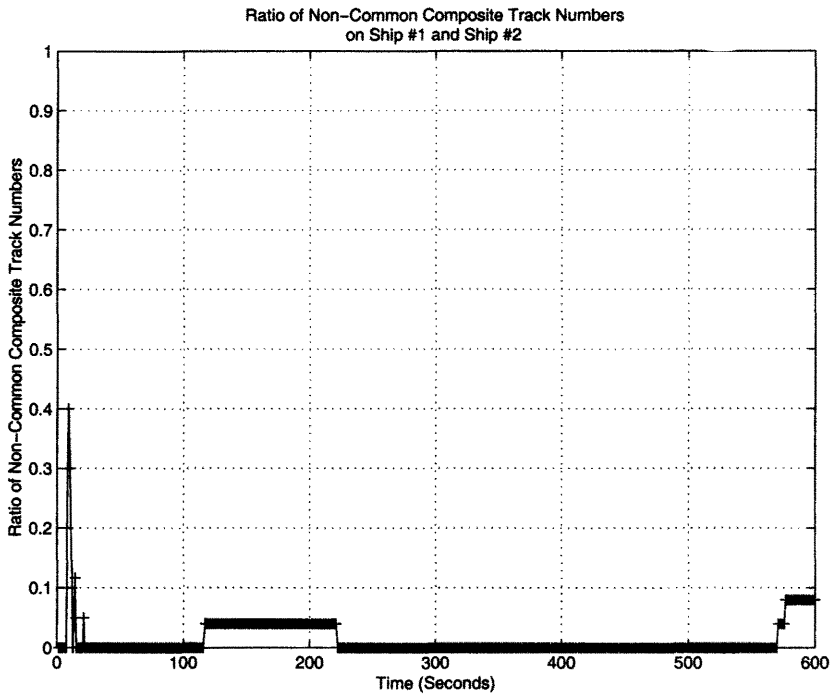


Figure 9.20: Network MFA on All Data and Network Tracks

9.4.4 Network MFA on All Data and Network Tracks

Figure 9.20 shows the ratio of non-common composite track number for network MFA on all data and network tracks. The peak at the beginning of the simulation is lower than that of the network MFA on local data, which shows that initiating tracks with all data is superior to initiating tracks with local data only. The non-common composite track numbers afterwards are caused by the redundant tracks initiated on Ship 2.

Figure 9.21 shows the composite track state estimate differences in network MFA on all data and network tracks. The differences are smaller than the network MFA on local data and network tracks before the fighters merge together.

9.5 Communication Loading

The Communication loading levels in all four architectures are similar to what's shown in Figure 8.15 to 8.18.

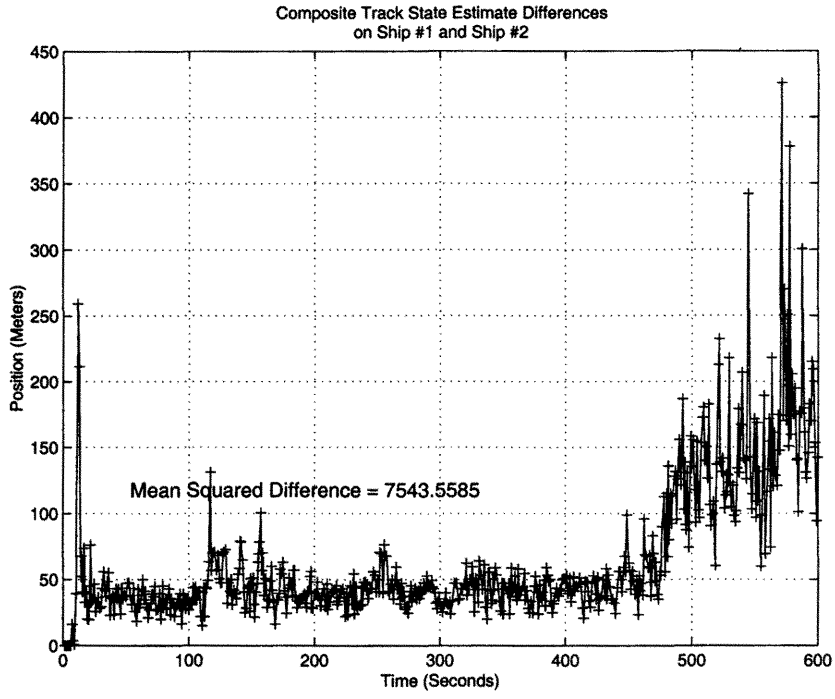


Figure 9.21: Network MFA on All Data and Network Tracks

9.6 Conclusions and Discussion

The metrics that measure track continuity such as spurious track mean ratio, redundant track ratios, track breakage, track completeness remain nearly perfect even with moderate communication delays and lost data; however, most other metrics degenerate somewhat. The centralized tracking architecture remains the best overall architecture except for communication loading. The delays cause non-common composite track numbers for the Network MFA Centralized to degenerate significantly and this is primarily due to the lack of any mechanism to coordinate tracks from platform to platform. All other metrics perform well for this architecture.

In both network MFA on local data and network MFA on all data, specific methods were developed to ensure cross-platform commonality history metrics. These include track numbering schemes and track correlation for initiating tracks as well as data association rules that yield a consistent set of tracks N frames back.

Network MFA on local data and network tracks demands the lowest communication bandwidth. However, the composite trackers initiate new tracks based on local data only, which may cause bigger estimation errors in the initialization step, and tracks may not be initiated fast enough.

Network MFA on all data and network tracks is more expensive in the sense of communication and processing loadings. However, it generally provides a more consistent set of tracks across the network.

9.7 Future Directions of Research and Recommendations

1. Adaptive Push-Pull Scheme

   The network tracking system is a time-varying system. Instead of broadcasting all message to remote platforms, the composite tracker can push the messages that are of interest to a particular tracker. On the other hand, the tracker can request information from remote trackers based on the tracking performance.

2. Data Compression: Tracklets

   Currently, information passed around the network consists of observations, AMRs, or track states. In the future, Tracklets [53, 31, 30] can be used for data compression purposes. A Tracklet is a track computed so that its errors are not cross-correlated with the errors of any other data in the system for the same apparent target. A Tracklet is like a large (typically 6−dimensional) measurement and is equivalent to a track based on only the most recent measurements, typically 6 to 30 measurements.

3. Hybrid System

   The network architectures proposed are: network MFA centralized, network MFA on local data and network tracks, and network MFA on all data and network tracks. In the future, instead of using one architecture in the network,

the tracking system can be a hybrid system that contains some combinations of these architectures.

4. Biases and Registration Errors

   Future network tracking architectures should be able to estimate and remove sensor biases and registration errors [8]. If uncorrected, the registration errors can lead to large tracking errors and potentially to the formation of multiple tracks (ghosts) on the same target. Once estimated, the biases will be used to transform the measurement data.

Part II

Feature Aided Tracking

Chapter 10

INTRODUCTION

10.1 Statement of the Problem

The central problem in any multi-target/multi-sensor surveillance system is the data association problem of partitioning the observations into tracks and false alarms. Current methods for multi-target tracking generally fall into two categories: sequential and deferred logic. Deferred logic considers several frames of observations all at once in making data association decisions. The principal deferred logic method used is called multiple hypothesis tracking (MHT), in which one uses a sliding window of size N, builds a tree of possibilities, assigns a likelihood score to each track, develops an intricate pruning logic, and then solves the data association problem.

Future multi-target/multi-sensor surveillance systems will provide a variety of feature and attribute data as well as kinematical data. However, the term *Feature Aided Tracking* includes both *feature* and *attribute* data. The potential use of the fused information is in determining target type and identity, assisting kinematic estimation of the target dynamics through maneuver detection and improved model selection, and aiding data association. The proposed research program is focused on the use of features and attributes to improve the likelihood ratio scores, and thus improve data association.

There are three types of data available:

- Kinematical

  Measurements of the target's position and its derivatives. Examples of kinematical measurements are range, azimuth, elevation and Doppler.

- Features

  Characteristics of a target from a continuous sample space. Features can be measured directly or computed based on a number of measured quantities. Examples of features include estimated target dimension, radar cross section, and other target signature data.

- Attribute:

  Characteristics of a target from a discrete sample space. Examples of attributes include target type, type of radar systems used by a target, and number of engines on an airplane.

10.2 Overview

This research program undertook an investigation into the theory and practice of the use of features and attributes in the data association process. A key step is the derivation of generalized likelihood ratios combining both kinematical data and features/attributes.

Chapter 11 explains the mathematical formulation of the data association problem of partitioning reports into tracks and false alarms and the likelihood ratio scores developed for kinematical measurements.

For kinematical and feature measurements, Kalman filtering techniques are proposed to be the best. For cases when only kinematical measurements are available, the likelihood ratio calculation is presented in Chapter 11. When feature measurements are available, the discussion is divided into two cases: independent

features and features cross-correlated with kinematical measurements. Both cases are presented in Chapter 12.

For attribute measurements, techniques used for attribute tracking are largely an issue of the nature of the available *a priori* information about target attributes, target behavior, and the attribute measurement process. Bayesian reasoning (Chapter 13) and evidential reasoning(Chapter 14) are the two most important techniques.

Bayesian reasoning is a classical approach. Theoretical studies for one attribute or multiple attributes are given in Chapter 13. Formal Bayesian processing, unfortunately, requires a degree of *a priori* knowledge (i.e., complete knowledge of transitional and *a priori* densities) that is difficult to obtain. Evidential reasoning has been proposed as an alternative to Bayesian processing by many researchers. Its attractiveness rests on its less stringent needs for statistical information. In a multi-sensor scenario, different sensors report their data in different frames of discernment, evidential reasoning is claimed to be able to combine multiple sensor data for attribute problems. We start with one attribute under a complete probability model in Chapter 14, and prove the exact correspondence between evidential reasoning and Bayesian reasoning. Under the circumstances when only partial probability models are available, the generalized likelihood ratio is derived instead. Further investigations of the correspondence between a generalized likelihood ratio and the probabilities will be carried out. We plan to apply similar techniques in multiple attribute situations.

Measures of merit and performance are developed to demonstrate and evaluate the effectiveness of the use of features and attributes in tracking. Some commonly used measures of merits are: tracking (kinematical) accuracy, track continuity, track purity, probability of false tracks, number of missed tracks and time to detect a track (track initiation).

Simulations of one feature (range extent) and one attribute (to be determined) will be carried out using our multi-target/multi-sensor tracker to demonstrate the performance improvements obtained through the use of features and attributes.

10.3 Literature Review

Multiple hypothesis tracking has been popularized by the fundamental work of Reid [62]. These works are further discussed in the books of Blackman and Popoli [8], Bar-Shalom and Li [5], which also serve as excellent introductions to the field of multi-target tracking and multi-sensor data fusion. Poore [59] has formulated sensor fusion problems in terms of these multidimensional assignment problems.

For kinematical tracking, model selection is essential to the performance. Different dynamic models such as the nearly constant velocity model, the nearly constant acceleration model, the Singer model, the 2-D turning model and the 3-D turning model can be found in [3, 70, 69, 68, 67, 81, 9]. Standard Kalman filtering techniques [1, 3], square root filters [26] and IMM techniques [3, 14] are thoroughly investigated. Different IMM architectures that are commonly used in tracking systems are introduced in [28, 81, 10, 9, 19]. Bar-Sharlom [5] introduces the idea of augmenting the kinematical state with the feature states.

Oliver Drummond [33] summarizes different types of measurements and describes how different gating techniques are used to eliminate unlikely measurement-track pairs, thus aiding the data association problem. Blackman [8] gives an overview of the various methods used in tracking in the presence of feature and attribute information.

The classical approaches based on Bayesian and maximum likelihood require the most detailed knowledge. The Bayesian network [25] is regarded as computationally efficient [23, 22, 35, 43, 50]. Among the various algorithms available in the literature, the symbolic probabilistic inference [65] is regarded as best.

The non-traditional techniques require significantly less information. The voting and set intersection techniques require the least *a priori* knowledge [8]. The fact that no probabilistic information is utilized makes the description of the algorithm very intuitive. The evidential reasoning (Dempster-Shafer reasoning) [17, 18, 16] requires some prior information and maintains belief measures (mass assignments) over multiple propositions. The implementation of evidential reasoning is relatively straightforward if it is assumed that basic mass assignment information is available. There is difficulty with evidential reasoning if one wants the basic mass assignments and the resulting support and plausibility to have any relationship to probability. Several practical approaches are discussed in the literature for determining and combining mass values for identification problems. [17]

Chapter 11

LIKELIHOOD RATIO SCORES CALCULATION

The goal of the tracking problem is to determine the number of targets, which measurements go with which targets and which are false, (i.e., the data association problem), and to estimate the state of each target at some set of times given a sequence of measurements that emanate from that target.

In the MHT method, a sliding window of length N is used. N frames of measurements (or reports) inside the window are processed together. A frame (sometimes called a proper frame) of data is a set of measurements which contains at most one observation from each target. Irrevocable decisions are made for the first frame of data (measurements).

11.1 Data Association Problem Formulation

Let $Z(k)$ denote a frame of M_k reports $\{z_{i_k}^k\}_{i_k=1}^{M_k}$ and let Z^N denote the cumulative data set of N such sets.

$$Z(k) = \{z_{i_k}^k\}_{i_k=1}^{M_k} \quad \text{and} \quad Z^N = \{Z(1), \ldots, Z(N)\}, \tag{11.1}$$

The data association problem in multitarget tracking and multisensor data fusion is generally posed as

$$\text{Maximize} \left\{ \frac{P(\Gamma = \gamma \mid Z^N)}{P(\Gamma = \gamma^0 \mid Z^N)} \mid \gamma \in \Gamma^* \right\} \tag{11.2}$$

where Z^N represents N data sets (11.1), γ is a partition of indices of the data, Γ^* is the finite collection of all such partitions, Γ is a discrete random element defined on Γ^*, γ^0 is a reference partition, and $P(\Gamma = \gamma \mid Z^N)$ is the *a posterior* probability of a partition γ being true given the data Z^N.

A partition of the cumulative data set Z^N is defined as follows.

$$I^N = \{I(1), I(2), \ldots, I(N)\} \text{ where } I(k) = \{i_k\}_{i_k=0}^{M_k} \tag{11.3}$$

denote the indices in the data sets (11.1). The *zero index, which stands for a dummy report* z_0^k is added for notational convenience in representing tracks. The dummy report z_0^k serves several purposes in the representation of missing data, false reports, initiating tracks, and terminating tracks.

A partition γ of I^N and the collection of all such partitions Γ^* is defined by

$$\begin{aligned}
\gamma &= \{\gamma_1, \ldots, \gamma_{n(\gamma)} \mid \gamma_i = \{i_1, \cdots, i_N\} \neq \{0, \cdots, 0\}, \\
&\qquad \text{for } i = 1, \cdots, n(\gamma)\}, && (11.4) \\
\gamma_i \cap \gamma_j &= \emptyset, \quad \text{for } i \neq j && (11.5) \\
I^N &= \cup_{j=1}^{n(\gamma)} \gamma_j, && (11.6) \\
\Gamma^* &= \{\gamma \mid \gamma \text{ satisfies (11.4)} - \text{(11.6)}\}. && (11.7)
\end{aligned}$$

Here, γ_i in (11.4) will be called a track, so that $n(\gamma)$ denotes the number of tracks (or elements) in the partition γ. $\gamma_i \neq \emptyset$ for each i in (11.4) means that a track can't be made up of dummy reports $z_0^k (k = 1, \ldots, N)$ only. Equation (11.6) can be reformulated as $i_k \neq j_k$, for $k = 1, \cdots, N$, where $\gamma_i = \{i_1, \cdots, i_N\}$ and $\gamma_j = \{j_1, \cdots, j_N\}$. It can be interpreted as saying that no observation can belong to two tracks.

Given a partition $\gamma \in \Gamma^*$,

$$Z_\gamma = \{Z_{\gamma_1}, \ldots, Z_{\gamma_{n(\gamma)}}\} \tag{11.8}$$

where

$$\gamma_i = \quad (i_1, \ldots, i_N) \tag{11.9}$$

$$Z_{\gamma_i} = \quad Z_{i_1 \cdots i_n} \equiv \left(z_{i_1}^1, \ldots, z_{i_N}^N\right) \tag{11.10}$$

The partition γ^0 of the data in which all reports are declared to be false reports is defined by

$$Z_{\gamma^0} = \{Z_{0\cdots 0 i_k 0 \cdots 0} \equiv \left(z_0^1, \ldots, z_0^{k-1}, z_{i_k}^k, z_0^{k+1}, \ldots, z_0^N\right) \mid i_k = 1, \ldots, M_k; k = 1, \ldots, N\}. \tag{11.11}$$

The independence assumptions can be summarized as:

$$p(Z^N \mid \Gamma = \gamma) = \quad \prod_{\gamma_i \in \gamma} p(Z_{\gamma_i} \mid \Gamma = \gamma) \tag{11.12}$$

$$p(Z_{\gamma_i} \mid \Gamma = \gamma) = \quad p(Z_{\gamma_i} \mid \Gamma = \omega)$$

$$\text{for all } \gamma_i \in \gamma \text{ and } \gamma_i \in \omega, \text{such that } \gamma, \omega \in \Gamma^* \tag{11.13}$$

$$P_\Gamma(\Gamma = \gamma) = \quad C \prod_{i=1}^{n(\gamma)} G(\gamma_i) \tag{11.14}$$

where C is a constant independent of the partition $\gamma \in \Gamma^*$ and G is a probability distribution on the set of tracks γ_i in (11.4). We also note that since for each $\gamma \in \Gamma^*$, Z_γ corresponds to a partition of the data into $n(\gamma)$ feasible tracks of data, assumption (11.12) says that the $n(\gamma)$ tracks of data, $\{Z_{\gamma_1}, \ldots, Z_{\gamma_{n(\gamma)}}\}$, are independent if γ is the true partition. Of course, there may be dependence between the reports within a single track. Equation (11.13) further states that these tracks are independent across all partitions of the data.

Based on Bayes' formula,

$$P(\Gamma = \gamma \mid Z^N) = \quad \tfrac{1}{p(Z^N)} p(Z^N \mid \Gamma = \gamma) P_\Gamma(\Gamma = \gamma) \tag{11.15}$$

$$= \quad \tfrac{1}{p(Z^N)} \left[\prod_{i=1}^{n(\gamma)} p(Z_{\gamma_i})\right] P_\Gamma(\Gamma = \gamma) \tag{11.16}$$

$$= \quad \tfrac{C}{p(Z^N)} \prod_{\gamma_i \in \gamma} p(Z_{\gamma_i}) G(\gamma_i) \tag{11.17}$$

To derive the assignment problem, observe that for $\gamma_i = (i_1 \cdots i_n)$ as in (11.10) and the reference partition (11.11), the expansion (11.17) can be written as

$$\frac{P(\Gamma = \gamma \mid Z^N)}{P(\Gamma = \gamma^0 \mid Z^N)} \equiv L_\gamma \equiv \prod_{(i_1 \cdots i_N) \in \gamma} L_{i_1 \cdots i_N} \tag{11.18}$$

where

$$L_{i_1 \cdots i_N} = \frac{p(Z_{i_1 \cdots i_N}) G(Z_{i_1 \cdots i_N})}{\prod_{k=1, i_k \neq 0}^{N} p(Z_{0 \cdots 0 i_k 0 \cdots 0}) G(Z_{0 \cdots 0 i_k 0 \cdots 0})}. \tag{11.19}$$

Here the index i_k in the denominator corresponds to the k^{th} index of $Z_{i_1 \cdots i_N}$ in the numerator.

Next define

$$c_{i_1 \cdots i_N} = -\ln L_{i_1 \cdots i_N}, \tag{11.20}$$

so that

$$-\ln \left[\frac{P(\gamma | Z^N)}{P(\gamma^0 | Z^N)} \right] = \sum_{(i_1, \ldots, i_N) \in \gamma} c_{i_1 \cdots i_N}. \tag{11.21}$$

Define a $0-1$ variable

$$z_{i_1 \cdots i_N} = \begin{cases} 1 & \text{if } (i_1, \ldots, i_N) \in \gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{11.22}$$

An equivalent characterization of a partition is that it is a solution to the equations

$$\sum_{(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N) = (0, \ldots, 0)}^{(M_1, \ldots, M_{k-1}, M_{k+1}, \ldots, M_N)} z_{i_1 \ldots i_N} = 1 \text{ for } i_k = 1, \ldots, M_k \text{ and } k = 1, \ldots, N. \tag{11.23}$$

The data association problem (11.2), formulated for tracking, can be further posed as an N-dimensional assignment problem.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i_1=0}^{M_1} \cdots \sum_{i_N=0}^{M_N} c_{i_1 \cdots i_N} z_{i_1 \cdots i_N} \\
\text{Subject To :} \quad & \sum_{i_2=0}^{M_2} \cdots \sum_{i_N=0}^{M_N} z_{i_1 \cdots i_N} = 1, \quad i_1 = 1, \ldots, M_1, \\
& \sum_{i_1=0}^{M_1} \cdots \sum_{i_{k-1}=0}^{M_{k-1}} \sum_{i_{k+1}=0}^{M_{k+1}} \cdots \sum_{i_N=0}^{M_N} z_{i_1 \cdots i_N} = 1, \qquad (11.24) \\
& \qquad \text{for} \quad i_k = 1, \ldots, M_k \text{ and } k = 2, \ldots, N-1, \\
& \sum_{i_1=0}^{M_1} \cdots \sum_{i_{N-1}=0}^{M_{N-1}} z_{i_1 \cdots i_N} = 1, \quad i_N = 1, \ldots, M_N, \\
& z_{i_1 \cdots i_N} \in \{0, 1\} \text{ for all } i_1, \ldots, i_N.
\end{aligned}
$$

11.2 Likelihood Ratio Calculation

From the discussion in the previous section, computation of the likelihood ratio $L_{i_1 \cdots i_N}$ (or the score $c_{i_1 \cdots i_N}$) is an essential part of the N-dimensional assignment problem. In this section, recursive formulas are derived which fit in the sliding window techniques perfectly.

Denote the k^{th} set of data reports as $Z(k) = \{z_i^k\}_{i_k=1}^{M_k}$ and the cumulative set of k such data sets as $Z^k = \{Z(1), \ldots, Z(k)\}$ so that $Z^k = Z(k) \cup Z^{k-1}$. The customary notation here [62] is to define $\Omega^k = \{\Omega_j^k \mid j = 0, 1, \ldots, I_k\}$ as the set of hypotheses about the feasible partitions of the cumulative set of measurements Z^k into tracks and false alarms. In the notation of the previous section, each Ω_j^k represents the event that a partition $\gamma^k \in \Gamma^{k*}$ is true. (Here, the dependence of γ on the number of scans k has been explicitly included as a superscript.) The hypothesis Ω_0^k is that all reports are false. Let $\Omega_{\psi_l}^{k-1}$ denote that specific hypothesis in Ω^{k-1} that produces Ω_l^k, and let $\psi_l(k)$ denote the hypothesis that indicates the specific status of all targets postulated by $\Omega_{\psi_l}^{k-1}$ at the scan time t_k and the specific

origin of all reports received at scan time t_k. Thus

$$\Omega_l^k = \psi_l(k) \cup \Omega_{\psi_l}^{k-1} \text{ and } Z^k = Z(k) \cup Z^{k-1} \tag{11.25}$$

Using Bayes' rule, one can write

$$P(\Omega_l^k|Z^k) = p(Z(k)|\Omega_l^k, Z^{k-1})P(\psi_l(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1})P(\Omega_{\psi_l}^{k-1}|Z^{k-1}) \left[\frac{p(Z^{k-1})}{p(Z^k)}\right] \tag{11.26}$$

Thus, the data association problem posed in (11.2) can be reformulated using a recursive formula

$$\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)} = \frac{p(Z(N)|\Omega_l^N, Z^{N-1})P(\psi_l(N)|\Omega_{\psi_l}^{N-1}, Z^{N-1})}{p(Z(N)|\Omega_0^N, Z^{N-1})P(\psi_0(N)|\Omega_{\psi_0}^{N-1}, Z^{N-1})} \left[\frac{P(\Omega_{\psi_l}^{N-1}|Z^{N-1})}{P(\Omega_{\psi_0}^{N-1}|Z^{N-1})}\right] \tag{11.27}$$

For ease of reference, we tabulate the following list of definitions:

P_χ^k is the probability of termination on scan k;

P_d^k is the probability of detection on scan k;

$z_{i_k}^k$ is measurement i_k from scan k;

δ^k is the number of measurements on scan k
originating from previously established tracks;

ν^k is the number of new targets detected on scan k
with an associated probability mass function $\mu_\nu^k(\nu^k)$;

f^k is the number of false alarms on scan k
with an associated probability mass function $\mu_f^k(f^k)$;

M_k $= \nu^k + f^k + \delta^k$, is the total number of measurements on scan k ;

τ^k is the number of targets that were extended from scan $k-1$ to scan k;

χ^k is the number of terminated (and nondetected) targets on scan k;

$$f_i^k = \begin{cases} 1, & \text{if } z_i^k \text{ is a false alarm;} \\ 0, & \text{otherwise;} \end{cases}$$

$$\nu_i^k = \begin{cases} 1, & \text{if } z_i^k \text{ is a new target;} \\ 0, & \text{otherwise;} \end{cases}$$

$$\delta_i^k = \begin{cases} 1, & \text{if } z_i^k \text{ belongs to an existing track;} \\ 0, & \text{otherwise;} \end{cases}$$

$$\Delta_{ij} = \begin{cases} 1, & \text{i = j;} \\ 0, & \text{otherwise.} \end{cases}$$

(11.28)

Thus, $\tau^k - \chi^k - \delta^k$ is the number of missed detections on scan k (non-terminated target but not detected), and the total number of targets that exist after scan k is $\tau^{k+1} = \tau^k - \chi^k + \nu^k$. In addition, let $p_t^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ represent the likelihood that the report $z_{i_k}^k$ originated from a previously established target, $p_\nu^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ rep-

resent the likelihood that the report originated from a new source; and, $p_f^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ represent the likelihood the report represents a false alarm.

Based on the detailed derivation in Appendix A, we arrive at the following expression for the likelihood ratio:

$$\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)} = \left\{\frac{\nu^N!f^N!}{M_N!}\frac{\mu_f^N(f^N)\mu_\nu^N(\nu^N)}{\mu_f^N(M_N)\mu_\nu^N(0)}\right\} \left\{(P_\chi^N)^{\chi^N}\left[(1-P_\chi^N)(1-P_d^N)(1-P_m^N)\right]^{\tau^N-\delta^N-\chi^N}\right\} \prod_{i=1}^{M_N}\left\{\left[(1-P_\chi^N)P_d^N\frac{p_t^N(z_{i_N}^N|\Omega_l^N,Z^{N-1})}{p_f^N(z_{i_N}^N|\Omega_0^N,Z^{N-1})}\right]^{\delta_{i_N}^N}\left[\frac{p_\nu^N(z_{i_N}^N|\Omega_l^N,Z^{N-1})}{p_f^N(z_{i_N}^N|\Omega_0^N,Z^{N-1})}\right]^{\nu_{i_N}^N}\right\} \frac{P(\Omega_{\psi_l}^{N-1}|Z^{N-1})}{P(\Omega_0^{N-1}|Z^{N-1})}. \quad (11.29)$$

If it is further assumed that $\mu_f^k(f^k) = \exp(-\lambda_f^k)\frac{(\lambda_f^k)^{f^k}}{f^k!}$ and $\mu_\nu^k(\nu^k) = \exp(-\lambda_\nu^k)\frac{(\lambda_\nu^k)^{\nu^k}}{\nu^k!}$ are Poisson probability mass functions(for $k = 1, \ldots, N$), then

$$\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)} = \frac{P(\Omega_{\psi_l}^{N-1}|Z^{N-1})}{P(\Omega_0^{N-1}|Z^{N-1})}\left\{(P_\chi^N)^{\chi^N}\left[(1-P_\chi^N)(1-P_d^N)\right]^{\tau^N-\delta^N-\chi^N}\right\} \prod_{i=1}^{M_N}\left\{\left[\frac{(1-P_\chi^N)P_d^N p_t^N(z_{i_N}^N|\Omega_l^N,Z^{N-1})}{\lambda_f^N p_f^N(z_{i_N}^N|\Omega_0^N,Z^{N-1})}\right]^{\delta_{i_N}^N}\left[\frac{\lambda_\nu^N p_\nu^N(z_{i_N}^N|\Omega_l^N,Z^{N-1})}{\lambda_f^N p_f^N(z_{i_N}^N|\Omega_0^N,Z^{N-1})}\right]^{\nu_{i_N}^N}\right\} \quad (11.30)$$

The above equation can be formulated equivalently as

$$\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)} = \prod_{\{i_1 i_2 \cdots i_N\}\in\gamma} L_{i_1 i_2 \cdots i_N}. \quad (11.31)$$

where $L_{i_1 i_2 \cdots i_N}$ is the likelihood ratio of the track $Z_{i_1 \cdots i_N} \equiv \{z_{i_1}^1, \ldots, z_{i_N}^N\}$.

Define

$$P_\phi^k = \begin{cases} P_\chi^k, & \text{if track } Z_{i_1 \cdots i_N} \text{ terminates at scan } k; \\ (1-P_\chi^k)(1-P_d^k), & \text{if track } Z_{i_1 \cdots i_N} \text{ has a missed detection on scan } k; \\ 1, & \text{otherwise.} \end{cases} \quad (11.32)$$

Then

$$L_{i_1 i_2 \cdots i_N} \equiv L(Z_{i_1 i_2 \cdots i_N}) \equiv L(z_{i_1}^1, \ldots, z_{i_N}^N)$$
$$= \prod_{k=1}^{N} \left\{ P_\phi^k \right\}^{\Delta_{0 i_k}} \left\{ \left[\frac{(1 - P_\chi^k) P_d^k p_t^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\delta_{i_k}^k} \left[\frac{\lambda_\nu^k p_\nu^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\nu_{i_k}^k} \right\}^{(1 - \Delta_{0 i_k})} \tag{11.33}$$

provided at least two of the indices in $\{i_1, i_2, \ldots, i_N\}$ are nonzero and

$$L_{0 \cdots 0 i_k 0 \cdots 0} \equiv 1 \text{ provided } \{0, \ldots, 0, i_k, 0, \ldots, 0\} \in \gamma$$

The likelihood ratios and the scores can be written recursively as:

$$L_{i_1 i_2 \cdots i_N} = \prod_{k=1}^{N} L_{i_k} = L_{i_1 i_2 \cdots i_{N-1}} L_{i_N} \tag{11.34}$$

$$c_{i_1 i_2 \cdots i_N} = \sum_{k=1}^{N} c_{i_k} = c_{i_1 i_2 \cdots i_{N-1}} + c_{i_N} \tag{11.35}$$

with

$$L_{i_k} = \left\{ P_\phi^k \right\}^{\Delta_{0 i_k}} \left\{ \left[\frac{(1 - P_\chi^k) P_d^k p_t^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\delta_{i_k}^k} \left[\frac{\lambda_\nu^k p_\nu^k(z_{i_k}^k | Z_{i_1 \cdots i_N})}{\lambda_f^k p_f^k(z_{i_k}^k | Z_{0 \cdots 0 i_k 0 \cdots 0})} \right]^{\nu_{i_k}^k} \right\}^{(1 - \Delta_{0 i_k})}$$
$$\text{for } k = 1, 2, \cdots, N. \tag{11.36}$$

Whenever a new scan of data (indexed by scan $k+1$, $k = 1, 2, \cdots$) is brought into the sliding window for processing, one can recursively update $L_{i_1 \cdots i_k i_{k+1}}$ or $c_{i_1 \cdots i_k i_{k+1}}$ by calculating $L_{i_{k+1}}$ or $c_{i_{k+1}}$ using (11.34) or (11.35) respectively.

11.3 Kinematical Measurements

Consider the probability density function $p_t^k(z_{i_k}^k | Z_{i_1 \cdots i_k})$ that an observation $z_{i_k}^k$ emanates from the track $Z_{i_1 \cdots i_k}$. If the observation $z_{i_k}^k$ contains only position or velocity information of the target (*i.e.* kinematical measurements only), the dynamics of the target is modeled by a discrete state transition equation.

$$\boldsymbol{x}_k = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}) + \boldsymbol{G}_{k-1} \boldsymbol{w}_{k-1} \tag{11.37}$$

where $\boldsymbol{x}_k$ is the state vector at time t_k, $\boldsymbol{x}_k$ usually contains position, velocity and acceleration information about the target. The measurement equation is

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{11.38}$$

A Kalman filter (in Appendix B) is used to recursively estimate the state vector $\hat{\boldsymbol{x}}_{k|k}$ based on the cumulative observation set $\boldsymbol{z}^k = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_k\}$. For a linear system, where, $\boldsymbol{f}_{k-1}(\cdot)$ and $\boldsymbol{h}_k(\cdot)$ are both linear functions, and under the assumption that the plant noise $\boldsymbol{w}_k$ and the measurement noise $\boldsymbol{v}_k$ are white Gaussian, then one can conclude that

$$\text{if } \; p(\boldsymbol{x}_{k-1}|\boldsymbol{z}^{k-1}) \sim \boldsymbol{N}(\hat{\boldsymbol{x}}_{k-1|k-1}; \boldsymbol{P}_{k-1|k-1}), \quad \text{then } \; p(\boldsymbol{x}_k|\boldsymbol{z}^k) \sim \boldsymbol{N}(\hat{\boldsymbol{x}}_{k|k}; \boldsymbol{P}_{k|k}). \tag{11.39}$$

which means that for a linear system, the Kalman filter is optimal in the **Minimum Variance (MV)** sense. For a nonlinear system, the extended Kalman filter (in Appendix B) is used in state estimation which is optimal in the **Linear Minimum Variance (LMV)** sense.

11.3.1 Dynamic Models

For kinematical tracking, model selection is essential to performance. The most commonly used dynamic models in the literature are:

- Nearly Constant Velocity Model (NCV)

  The NCV model is a white noise acceleration model, in which the acceleration is assumed to be zero mean white Gaussian [5]. This is the most popular dynamic model in tracking. Proper settings of the noise levels is an important part of model design. Small process noise will model the air turbulence, winds aloft changes and so forth, while a somewhat larger process noise is used to cover slow turns as well as small linear acceleration so as to ease the burden of modeling a broad range of maneuvers.

- Nearly Constant Acceleration Model (NCA)

  In the NCA model, changes in acceleration are modeled by a zero mean white noise. [5]

- Singer Model

  In the Singer model [70, 69, 68, 67], the acceleration is assumed to be a first-order Markov process.

- Nearly Constant Turn Rate Model

  The nearly constant rate turn model is usually formulated in the horizontal plane (2-D space) [9]. The turning rate is assumed to be a Gaussian random variable, where the variance models the fluctuation in the turn rate. The model is nonlinear.

11.3.2 Likelihood calculation for a single Kalman Filter

In the likelihood ratio score calculation, it is always assumed that $z_{i_k}^k$ obeys a normal distribution with mean $\hat{\boldsymbol{z}}_{k|k-1}$ and with covariance matrix $\boldsymbol{S}_k$, as discussed in Appendix B. Mathematically,

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) \sim \mathcal{N}(z_{i_k}^k; \hat{\boldsymbol{z}}_{k|k-1}; \boldsymbol{S}_k) \tag{11.40}$$

or

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = \frac{1}{\sqrt{|2\pi \boldsymbol{S}_k|}} \exp\{-\frac{1}{2}(z_{i_k}^k - \hat{\boldsymbol{z}}_{k|k-1})' \boldsymbol{S}_k^{-1} (z_{i_k}^k - \hat{\boldsymbol{z}}_{k|k-1})\} \tag{11.41}$$

Similarly, the probability density function $p_f^k(z_{i_k}^k|Z_{0\cdots 0i_k})$ that an observation $z_{i_k}^k$ is a false alarm is assumed to be a uniform distribution over the entire surveillance region.

11.3.3 Likelihood calculation in IMM

In situations where a target abruptly performs maneuvers and changes its type of movement, the use of one particular type of Kalman Filter will not lead to good estimates. The interactive multiple model (IMM) algorithm is now regarded as the superior technique for tracking maneuvering targets. The IMM algorithm is decision-making free, it undergoes a soft switching of models and can give good estimates even at the critical maneuver periods. A detailed discussion of the IMM algorithms can be found in appendix B.4.

A unique feature of the IMM approach is the manner in which the state estimates and the covariance matrices from these multiple models are combined according to a Markov model for the transition between target maneuver states. The total number of target maneuver models is defined to be r and is typically around three.

The likelihood $p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k})$ in the IMM architecture is the weighted sum of likelihood functions Λ_k^j for each individual filter as in equation (B.33).

$$\begin{aligned} p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) &= \sum_{j=1}^{r} p_t^k(z_{i_k}^k|m_{k-1}^j, Z_{i_1\cdots i_{k-1}})\, p(m_{k-1}^j|Z_{i_1\cdots i_{k-1}}) \\ &= \sum_{j=1}^{r} \mu_{k-1}^j \Lambda_k^j \\ &= \sum_{j=1}^{r} \mu_{k-1}^j \frac{1}{\sqrt{|2\pi \boldsymbol{S}_k^j|}} \exp\{-\frac{1}{2}(z_{i_k}^k - \hat{\boldsymbol{z}}_{k|k-1}^j)'(\boldsymbol{S}_k^j)^{-1}(z_{i_k}^k - \hat{\boldsymbol{z}}_{k|k-1}^j)\} \end{aligned} \tag{11.42}$$

Chapter 12

FEATURES

12.1 Measurement Space Decomposition

Chapter 10 describes the mathematical formulation of the data association problem and explains the likelihood ratio scores used. With the presence of a variety of feature and attribute data as well as kinematical data, the proper fusion of feature and attribute information into the likelihood ratio scores (equation 11.36) can improve the data association problem, and thus improve the performance of the surveillance system.

Denote the entire measurement space to be $\mathcal{Z}$, then

$$\mathcal{Z} = \mathcal{Z}^G \cup \mathcal{Z}^F \cup \mathcal{Z}^A$$

with $\mathcal{Z}^G$ the kinematic (geolocational) measurement space, $\mathcal{Z}^F$ the feature measurement space, and $\mathcal{Z}^A$ the attribute measurement space.

Features are characteristics of a target that are from a continuous sample space. Features can be measured directly or computed based on a number of measured quantities. Examples of features include estimated target dimension, radar cross section, and other target signature data. Due to the fact that features are similar to kinematic measurements, they can be processed in much the same way, i.e., using filtering and estimation techniques.

12.2 Features Independent of Kinematic Measurements

In cases when feature measurements are statistically independent of kinematical measurements, separate Kalman filters can be used for kinematical and feature data. Suppose the feature state vector at time t_k is denoted by $\mathbf{x}_k^F$. The dynamic behavior of the feature can be modeled by the feature state dynamic equation:

$$\mathbf{x}_k^F = \mathbf{F}_{k-1}\mathbf{x}_{k-1}^F + \mathbf{w}_{k-1}, \tag{12.1}$$

where $\mathbf{F}_{k-1}$ is the state transition matrix for feature state vectors, and the noise term $\mathbf{w}_{k-1}$ models uncertainties in the model. For static features, $\mathbf{F}_k = \mathbf{I}$, for $\forall\, k$.

Denote the feature measurements at time t_k as $\mathbf{z}_{i_k}^{F_k} \in \mathcal{Z}^F$. The measurement equation has the form:

$$\mathbf{z}_{i_k}^{F_k} = \mathbf{h}(\mathbf{f}_k) + \mathbf{v}_k \tag{12.2}$$

With an initial state estimate $\hat{\mathbf{x}}_0^F$ and covariance matrix $\mathbf{P}_{0|0}$, one can initiate the Kalman filter to estimate $\hat{\mathbf{x}}_{k|k}^F$ based on the feature measurements $\{\mathbf{z}_{i_0}^{F_0}, \cdots, \mathbf{z}_{i_k}^{F_k}\}$. For track extension, if the measurements taken in different spaces are statistically independent, then the probability in equation 11.36 can be rewritten as

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = p_t^k(z_{i_k}^{G_k}|Z_{i_1\cdots i_k})\, p_t^k(z_{i_k}^{F_k}|Z_{i_1\cdots i_k})\, p_t^k(z_{i_k}^{A_k}|Z_{i_1\cdots i_k}). \tag{12.3}$$

Similarly,

$$p_f^k(z_{i_k}^k|Z_{0\cdots 0i_k}) = p_f^k(z_{i_k}^{G_k}|Z_{0\cdots 0i_k})\, p_f^k(z_{i_k}^{F_k}|Z_{0\cdots 0i_k})\, p_f^k(z_{i_k}^{A_k}|Z_{0\cdots 0i_k}). \tag{12.4}$$

As per the discussion on kinematical tracking (chapter 11), the likelihood for feature measurements obeys a Gaussian distribution centered at the predicted feature measurement with a covariance that is the innovation covariance matrix given in each iteration of Kalman filtering.

$$p_t^{F_k}(z_{i_k}^{F_k}|Z_{i_1\cdots i_k}^F) \sim \mathcal{N}(z_{i_k}^{F_k}, \hat{\boldsymbol{z}}_{k|k-1}^F; \boldsymbol{S}_k) \tag{12.5}$$

or

$$p_t^{F_k}(z_{i_k}^{F_k}|Z_{i_1\cdots i_k}^F) = \frac{1}{\sqrt{|2\pi \boldsymbol{S}_k|}}\exp\{-\frac{1}{2}(z_{i_k}^{F_k} - \hat{\boldsymbol{z}}_{k|k-1}^F)'\boldsymbol{S}_k^{-1}(z_{i_k}^{F_k} - \hat{\boldsymbol{z}}_{k|k-1}^F)\} \quad (12.6)$$

The probability density function $p_f^{F_k}(z_{i_k}^k|Z_{0\cdots 0i_k}^F)$ that an observation $z_{i_k}^{F_k}$ is a false alarm is assumed to be uniformly distributed over the entire feature space.

12.3 Features Cross-Correlated with Kinematical Measurements

When feature measurements are cross-correlated with kinematic measurements, they should be processed as a single vector. The state vector used in Kalman filtering is a concatenation of the kinematic states and features states. The state equation and the measurement equation should include both kinematic and feature information.

The likelihood in equation 12.3 will now be treated as

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = p_t^k(z_{i_k}^{G\cup F_k}|Z_{i_1\cdots i_k})\, p_t^k(z_{i_k}^{A_k}|Z_{i_1\cdots i_k}) \quad (12.7)$$

12.3.1 Using a Single Kalman Filter

If only one dynamic model is used for kinematic and feature data, then

$$p_t^k(z_{i_k}^{G\cup F_k}|Z_{i_1\cdots i_k}) = \frac{1}{\sqrt{|2\pi \boldsymbol{S}_k|}}\exp\{-\frac{1}{2}(z_{i_k}^{G\cup F_k} - \hat{\boldsymbol{z}}_{k|k-1}^{G\cup F_k})'\boldsymbol{S}_k^{-1}(z_{i_k}^{G\cup F_k} - \hat{\boldsymbol{z}}_{k|k-1}^{G\cup F_k})\} \quad (12.8)$$

where $z_{i_k}^{G\cup F_k}$ is the concatenation of kinematic and feature measurements, $\hat{\boldsymbol{z}}_{k|k-1}^{G\cup F_k}$ is the predicted measurements (kinematic and feature), and $\boldsymbol{S}_k$ is the corresponding innovation covariance matrix.

This processing is dealing with a higher dimensional space, thus it is computationally more intensive.

12.3.2 IMM Approach I

As discussed in Chapter 11, when processing kinematical measurements in situations where a target abruptly performs maneuvers and changes its type of movement, the use of one particular type of Kalman Filter will not lead to good estimates. The interactive multiple model (IMM) algorithm is currently regarded as the best technique for tracking maneuvering targets.

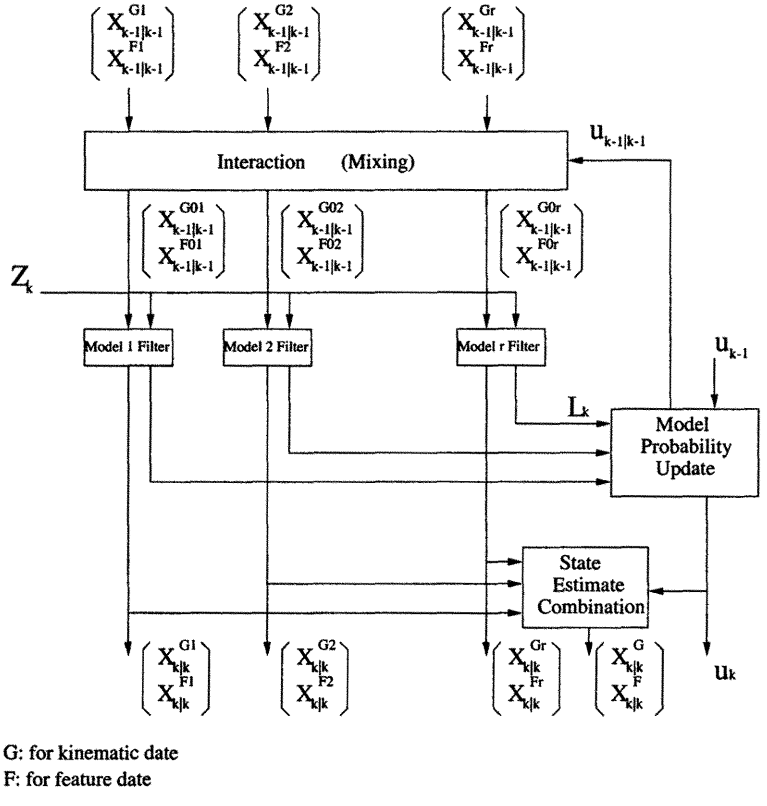


Figure 12.1: Structural diagram for IMM with features: Approach I

When the Kalman state vector contains kinematic states augmented with feature states, one can still use the IMM approach on the augmented state vector. The state estimates and the covariance matrices from selected multiple models (for both kinematic and feature dynamics) are combined according to a Markov model for the transition between target maneuver states as before. The structural diagram is shown in Figure 12.1.

12.3.3 IMM Approach II

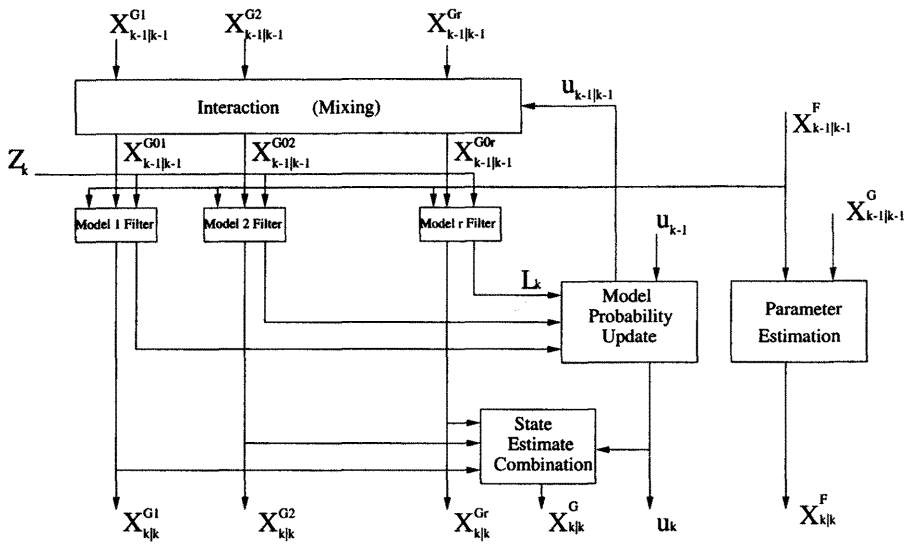


Figure 12.2: Structural diagram for IMM with features: Approach II

Generally, a single state transition equation for feature data should suffice, since we have better *a priori* knowledge of target feature dynamics. In this case, an IMM architecture as shown in figure 12.2 is proposed. There, only the kinematic states are combined over multiple models according to a Markov transition matrix. The feature states are estimated using parameter estimation techniques.

Chapter 13

ATTRIBUTES: BAYESIAN REASONING

Attributes are characteristics of a target that come from a discrete sample space. They are generally independent of the kinematical and feature measurements. Attribute tracking can be described as the process of combining information collected over time from one or more sources to refine our knowledge about possibly evolving attributes of a target or group of targets.

Techniques used for attribute tracking are largely an issue of the nature of the available *a priori* information about target attributes, target behavior, and the attribute measurement process.

Classical approaches based on Bayesian and maximum likelihood require the most detailed knowledge. Complete legitimate probability models for both the *a priori* and transitional densities are a prerequisite for the application of Bayesian processing. Maximum likelihood processing only requires a legitimate probability for the transitional density, with the assumption of a uniform *a priori.*

13.1 Single Attribute

One static attribute denoted by $x_i \in \mathcal{X}$ is considered. When the full probabilistic model is known, including $P(x_i)$, the *a priori* distribution, and $p(z|x_i)$, which is called the transitional distribution of observation z given truth x_i, Bayesian reasoning is typically denoted as

$$P(x_i|z) = \frac{p(z|x_i)P(x_i)}{\sum_{x_j \in \mathcal{X}} p(z|x_j)P(x_j)}. \tag{13.1}$$

Assume that all the observations are statistically independent, otherwise, one needs to pre-whiten the observation stream as explained in [8]. For the probability term in (12.3) associated with attributes,

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = p_t^k(z_{i_k}^k|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}) = \sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ P(x_i|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}), \quad (13.2)$$

where $p(z_{i_k}^k|x_i)$ is a transitional probability that depends on the measurement process modeling procedure. The probability $p(x_i|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1})$ can be computed recursively using:

$$P(x_i|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}) = \frac{p(z_{i_{k-1}}^{k-1}|x_i)\ P(x_i|z_{i_1}^1,\cdots,z_{i_{k-2}}^{k-2})}{\sum_{x_m\in\mathcal{X}} p(z_{i_{k-1}}^{k-1}|x_m)\ P(x_m|z_{i_1}^1,\cdots,z_{i_{k-2}}^{k-2})}, \quad (13.3)$$

with initialization

$$P(x_i|z_{i_1}^1) = \frac{p(z_{i_1}^1|x_i)P(x_i)}{\sum_{x_m\in\mathcal{X}} p(z_{i_1}^1|x_m)P(x_m)}. \quad (13.4)$$

For the probability term in (12.4) associated with attributes,

$$p_t^k(z_{i_k}^k|Z_{0\cdots 0i_k}) = p_t^k(z_{i_k}^k) = \sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ P(x_i). \quad (13.5)$$

13.2 Multiple Attributes

Suppose r attributes are considered at the same time, and

$$x_j^i \in \mathcal{X}^i,\ \ i = 1,\cdots,r$$

13.2.1 Case I: Statistically Independent

If the r attributes are statistically independent,

$$P(x_{j_1}^1, x_{j_2}^2,\cdots,x_{j_r}^r) = P(x_{j_1}^1)\ P(x_{j_2}^2)\cdots P(x_{j_r}^r), \quad (13.6)$$

and the measurement $z_{i_k}^k$ can divided into r statistically independent components

$$z_{i_k}^k = [\ z_{i_k}^k(1),\cdots,z_{i_k}^k(r)\ ]^t \quad (13.7)$$

$$p(z_{i_k}^k) = p(z_{i_k}^k(1))\cdots p(z_{i_k}^k(r)) \quad (13.8)$$

such that

$$p(z_{i_k}^k(p)|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r) = p(z_{i_k}^k(p)|x_{j_p}^p) \qquad p = 1, \cdots, r \tag{13.9}$$

then the transitional probability $p(z_{i_k}^k|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)$ can be simplified to

$$\begin{aligned} p(z_{i_k}^k|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r) =& \frac{P(x_{j_1}^1|z_{i_k}^k(1)) \cdots P(x_{j_r}^r|z_{i_k}^k(p))P(z_{i_k}^k)}{P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)} \\ =& \frac{1}{c} p(z_{i_k}^k|x_{j_1}^1) \cdots p(z_{i_k}^k|x_{j_r}^r), \end{aligned} \tag{13.10}$$

where $c = p(z_{i_k}^k)$ is a normalizing constant.

The probability term associated with attributes in the likelihood function (12.3) can be written as:

$$\begin{aligned} p_t^k(z_{i_k}^k|Z_{i_1 \cdots i_k}) &= p_t^k(z_{i_k}^k|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) \\ &= \prod_p p_t^k(z_{i_k}^k(p)|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) \\ &= \prod_p \Big\{ \sum_{x_{j_p}^p \in \mathcal{X}^p} p_t^k(z_{i_k}^k(p)|x_{j_p}^p) P(x_{j_p}^p|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) \Big\} \\ &= \prod_p \Big\{ \sum_{x_{j_p}^p \in \mathcal{X}^p} p_t^k(z_{i_k}^k(p)|x_{j_p}^p) P(x_{j_p}^p|z_{i_1}^1(p), \cdots, z_{i_{k-1}}^{k-1}(p)) \Big\} \end{aligned} \tag{13.11}$$

for which $P(x_{j_p}^p|z_{i_1}^1(p), \cdots, z_{i_{k-1}}^{k-1}(p))$, $p = 1, \cdots, r$ can be computed recursively as in (13.3) for the single attribute case.

The likelihood for false alarms (12.4) can be written

$$\begin{aligned} p_t^k(z_{i_k}^k|Z_{0 \cdots 0 i_k}) &= p_t^k(z_{i_k}^k) \\ &= \frac{1}{c} \sum_{x_{j_1}^1 \in \mathcal{X}^1} p(z_{i_k}^k(1)|x_{j_1}^1) P(x_{j_1}^1) \cdots \sum_{x_{j_r}^r \in \mathcal{X}^r} p(z_{i_k}^k(r)|x_{j_r}^r) P(x_{j_r}^r). \end{aligned} \tag{13.12}$$

The normalizing constant c will cancel upon computing the likelihood ratio $\dfrac{p_t^k(z_{i_k}^k|Z_{i_1 \cdots i_k})}{p_t^k(z_{i_k}^k|Z_{0 \cdots 0 i_k})}$.

From (13.11) and (13.12), it can be concluded that if the attributes are statistically independent, then one can compute the likelihood of each attribute individually, and multiply them together, that is

$$\frac{p_t^k(z_{i_k}^k|Z_{i_1 \cdots i_k})}{p_t^k(z_{i_k}^k|Z_{0 \cdots 0 i_k})} = \prod_{p=1}^r \frac{p_t^{k_p}(z_{i_k}^k(p)|Z_{i_1 \cdots i_k})}{p_t^{k_p}(z_{i_k}^k(p)|Z_{0 \cdots 0 i_k})} \tag{13.13}$$

where

$$\frac{p_t^{k_p}(z_{i_k}^k(p)|Z_{i_1\cdots i_k})}{p_t^{k_p}(z_{i_k}^k(p)|Z_{0\cdots 0i_k})} = \frac{\sum_{x_{j_p}^p \in \mathcal{X}^p} p(z_{i_k}^k(p)|x_{j_p}^p) P(x_{j_p}^p|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1})}{\sum_{x_{j_p}^p \in \mathcal{X}^p} p(z_{i_k}^k(p)|x_{j_p}^p) P(x_{h_p}^p)} \tag{13.14}$$

13.2.2 Case II: Statistically Correlated

If the attributes $x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r$ are statistically correlated, or each component of the measurement z is dependent all all the attributes, then they needs to be processed as a whole.

When the full probabilistic model is known, including $P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)$, the joint *a priori* distribution, and $p(z|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)$, which is called the transitional distribution of observation z given truth $x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r$, Bayesian reasoning is typically denoted as

$$P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r|z) = \frac{p(z|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r) P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)}{\sum_{x_{m_1}^1 \in \mathcal{X}^1, \cdots, x_{m_r}^r \in \mathcal{X}^r} p(z|x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r) P(x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r)} \tag{13.15}$$

The probability term in (12.3) associated with attributes is

$$\begin{aligned} p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) =& p_t^k(z_{i_k}^k|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) \\ =& \sum_{x_{j_1}^1 \in \mathcal{X}^1, \cdots, x_{j_r}^r \in \mathcal{X}^r} p(z_{i_k}^k|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)\ P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) \end{aligned} \tag{13.16}$$

where $p(z_{i_k}^k|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)$ is a transitional probability that depends on the measurement process modeling procedure. The *a posterior* probability $P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1})$ can be computed recursively using:

$$\begin{aligned} &P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) = \\ &\frac{p(z_{i_{k-1}}^{k-1}|x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)\ P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r|z_{i_1}^1, \cdots, z_{i_{k-2}}^{k-2})}{\sum_{x_{m_1}^1 \in \mathcal{X}^1, \cdots, x_{m_r}^r \in \mathcal{X}^r} p(z_{i_{k-1}}^{k-1}|x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r)\ P(x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r|z_{i_1}^1, \cdots, z_{i_{k-2}}^{k-2})}, \end{aligned} \tag{13.17}$$

with initialization

$$P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r | z_{i_1}^1) = \frac{p(z_{i_1}^1 | x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r) P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r)}{\sum_{x_{m_1}^1 \in \mathcal{X}^1, \cdots, x_{m_r}^r \in \mathcal{X}^r} p(z_{i_1}^1 | x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r) P(x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r)}. \quad (13.18)$$

The probability term in (12.4) associated with attributes is

$$p_t^k(z_{i_k}^k | Z_{0 \cdots 0 i_k}) = p_t^k(z_{i_k}^k) = \sum_{x_{m_1}^1 \in \mathcal{X}^1, \cdots, x_{m_r}^r \in \mathcal{X}^r} p(z_{i_k}^k | x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r) \, P(x_{m_1}^1, x_{m_2}^2, \cdots, x_{m_r}^r) \quad (13.19)$$

13.2.3 Case III: General Case

Generally, among the r attributes, some may be statistically independent, and some maybe cross correlated. One can divide them into subsets of attributes, such that those subsets (*i.e.* $\{x_{j_1}^1\}, \cdots, \{x_{j_{r_1}}^{r_1}\}, \{x_{j_{r_1+1}}^{r_1+1}, \cdots, x_{j_{r_2}}^{r_2}\} \cdots \{x_{j_{r_M+1}}^{r_M+1}, \cdots, x_{j_{r_{(M+1)}}}^{r_{(M+1)}}\}$) are mutually independent. Mathematically,

$$P(x_{j_1}^1, x_{j_2}^2, \cdots, x_{j_r}^r) = P(x_{j_1}^1) \cdots P(x_{j_{r_1}}^{r_1}) P(x_{j_{r_1+1}}^{r_1+1}, \cdots, x_{j_{r_2}}^{r_2}) \cdots P(x_{j_{r_M+1}}^{r_M+1}, \cdots, x_{j_{r_{(M+1)}}}^{r_{(M+1)}}) \quad (13.20)$$

with $r_{(M+1)} = r$. The likelihood ratio can then be decomposed into several smaller components.

$$\frac{p_t^k(z_{i_k}^k | Z_{i_1 \cdots i_k})}{p_t^k(z_{i_k}^k | Z_{0 \cdots 0 i_k})} = \prod_{j=1}^{r_1} \frac{p_t^{k_j}(z_{i_k}^k | Z_{i_1 \cdots i_k})}{p_t^{k_j}(z_{i_k}^k | Z_{0 \cdots 0 i_k})} \prod_{j=1}^{M} \frac{p_t^{k_{r_j+1, \cdots, r_{(j+1)}}}(z_{i_k}^k | Z_{i_1 \cdots i_k})}{p_t^{k_{r_j+1, \cdots, r_{(j+1)}}}(z_{i_k}^k | Z_{0 \cdots 0 i_k})} \quad (13.21)$$

13.3 Implementation

13.3.1 Single Attribute

For a single attribute, the computation of likelihood ratio is relatively simple. Each potential track will keep a vector of *a posteriori* probabilities $P(x_i | Z_{i_1 \cdots i_k})$, for $i = 1, \cdots, |\mathcal{X}|$. Each time a new frame of data is brought into the window, the vector is updated according to equation (13.3).

13.3.2 Multiple Attributes

When multiple attributes are present in the tracking system, the computation is much more complicated. The Bayesian Network (BN) is used to model or to encode probabilistic relationships among distributions so that one can compute the association likelihood as well as *a posteriori* probabilities efficiently.

The BN is an annotated acyclic directed graph, each node in the graph is a random element, and the arc between two nodes indicates a potential stochastic dependence between the two random elements by the two nodes. The qualitative relationships represented by incoming arcs to a state node is quantified as a conditional probability distribution. For our tracking system, the BN is used to relate the target attribute states to measurements at the sensors.

In addition to the convenient and flexible representation, a major benefit of using BNs is the existence of powerful probabilistic inference algorithms developed recently. The symbolic probabilistic inference (SPI) algorithm is regarded as the most flexible and efficient.

Chapter 14

SINGLE ATTRIBUTE: DEMPSTER-SCHAFER REASONING

Non-traditional techniques of attribute tracking require significantly less information. Those that require the least *a priori* knowledge are the voting and set intersection techniques. The fact that no probabilistic information is utilized makes the algorithms very intuitive. The set intersection approach is really a process of elimination, while the voting approach is a process of inclusion.

Evidential reasoning (Dempster-Shafer reasoning) requires little more information than set intersection and voting techniques and can be considered a weighted version of them. With proper settings, evidential reasoning is closely related to probabilities.

14.1 Complete Probability Models

In a full probabilistic model, it is possible to choose the evidential masses with an exact correspondence between evidential reasoning and Bayesian reasoning. In this case, the *a priori* distribution $P(x_i)$, and the transitional probability (likelihood) $p(z|x_i)$ for all $x_i \in \mathcal{X}$ are known.

14.1.1 Mass Assignment

- A Priori Mass Assignment

Under a complete probability model, the *a priori* focal elements X_i are simply the singletons $x_i \in \mathcal{X}$ for which $P(x_i) \neq 0$. The basic *a priori* mass assignment is:

$$m_x(X_i) = m_x(x_i) = P(x_i). \tag{14.1}$$

Denote the frame of discernment as Θ, then

$$\sum_{X_i \subseteq \Theta} m_x(X_i) = \sum_{x_i \in \mathcal{X}} m_x(x_i) = 1. \tag{14.2}$$

- Transitional Mass Assignment

The procedure for transitional basic mass assignment $m_z(\cdot)$ is more complicated. It depends on the observation z.

1. Order the singletons x_i in descending value according to the likelihood function $l_z(x_i) = p(z|x_i)$ so that

$$l_z(x_1) \geq \cdots l_z(x_M) \tag{14.3}$$

where M is the cardinality of the set $\mathcal{X}$.

2. Define the sequence of scaled likelihood values:

$$\{r_1 \geq \cdots \geq r_M \mid r_i = \frac{l_z(x_i)}{l_z(x_1)}\} \tag{14.4}$$

3. Let $P \leq M$ be the cardinality of the set of r values in which repeated values have been discarded. This set defines a sequence of unique values q_j

$$\{q_1 > \cdots > q_P\} \tag{14.5}$$

4. Define the $P \leq M$ telescoping sets $Y_j \subseteq \Theta$

$$Y_j = \{x_i \mid \frac{l_z(x_i)}{l_z(x_1)} \geq q_j\} \tag{14.6}$$

Note that it is the nature of these telescoping sets that

$$x_i \in Y_k \text{ implies } x_i \in Y_j, \forall j > k \tag{14.7}$$

5. Assign $m_z(Y_j)$

$$m_z(Y_j) = \begin{cases} q_j - q_{j+1}, & \text{if } j < P \\ q_P, & \text{if } j = P \end{cases} \tag{14.8}$$

so,

$$\sum_{j=1}^{P} m_z(Y_j) \quad = \quad 1 \tag{14.9}$$

Based on the telescoping nature of Y_j and the transitional mass assignment described above, the following formulas can be derived.

$$\sum_{k|Y_k \cap x_i \neq \emptyset} m_z(Y_k) = \frac{l_z(x_i)}{l_z(x_1)} = \frac{1}{l_z(x_1)} p(z|x_i), \quad \forall x_i \subseteq \Theta \tag{14.10}$$

$$\sum_{k|Y_k \cap X \neq \emptyset} m_z(Y_k) = \frac{1}{l_z(x_1)} \max_{\{i|x_i \in X\}} p(z|x_i), \quad X \subseteq \Theta \tag{14.11}$$

$$\sum_{k|Y_k \cap X = X} m_z(Y_k) = \frac{1}{l_z(x_1)} \min_{\{i|x_i \in X\}} p(z|x_i), \quad X \subseteq \Theta \tag{14.12}$$

14.1.2 Mass Combination

The *a posterior* mass is computed by combining the *a priori* mass and the transitional mass using *Demspter's rule of combination* (Appendix C.1).

$$\begin{aligned} m_{x|z}(x_i) =& \frac{1}{1-\kappa} \sum_{X_j, Y_k | X_j \cap Y_k = x_i} m_x(X_j) m_z(Y_k), \\ \kappa =& \sum_{X_j, Y_k | X_j \cap Y_k = \emptyset} m_x(X_j)\, m_z(Y_k) \end{aligned} \tag{14.13}$$

Since all the focal elements in the *a priori* mass assignment are singletons, all the focal elements in the *a posteriori* mass assignment are singletons as well.

Given a sequence of observations $\{z_{i_1}^1, \cdots, z_{i_N}^k, \cdots\}$, one can start by combining the *a priori* mass function $m_x(x_i)$ and the transitional mass function $m_{z_1}(Y_j^1)$ based on the first measurement $z_{i_1}^1$ using formula (14.13). Call that $m_{x|z}(X_i)$.

Then we can recursively combine the mass functions $m_{x|z_1,\cdots,z_{N-1}}(X_i)$ and $m_{z_N}(Y_j^N)$ based on the N-th observation $z_{i_N}^N$ using

$$m_{x|z_1,\cdots,z_{N-1},z_N}(x_i) = \frac{1}{1-\kappa} \sum_{X_j,Y_k^N|X_j\cap Y_k^N=x_i} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N)$$

$$\kappa = \sum_{X_j,Y_k^N|X_j\cap Y_k^N=\emptyset} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N) \tag{14.14}$$

The generalized likelihood function is constructed using the *a posteriori* mass function instead of the *a posteriori* probability.

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = p_t^k(z_{i_k}^k|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}) = \sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ m_{x|z_1,\cdots,z_{k-1}}(x_i) \tag{14.15}$$

The generalized likelihood function for false alarms is now formulated as:

$$p_t^k(z_{i_k}^k|Z_{0\cdots 0i_k}) = p_t^k(z_{i_k}^k) = \sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ m_x(x_i) \tag{14.16}$$

Thus, the generalized likelihood ratio is

$$\frac{p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k})}{p_t^k(z_{i_k}^k|Z_{0\cdots 0i_k})} = \frac{\sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ m_{x|z_1,\cdots,z_{k-1}}(x_i)}{\sum_{x_i\in\mathcal{X}} p(z_{i_k}^k|x_i)\ m_x(x_i)} \tag{14.17}$$

14.1.3 Correspondence Between Bayesian and Evidential Reasoning

Under a complete probability model with the mass assigned as above, it can be proved using mathematical induction that evidential reasoning is exactly the same as Bayesian reasoning. The generalized likelihood ratio equals the traditional likelihood ratio, which can be further reduced due to the fact that the *a posteriori* mass and the *a posteriori* probability are the same.

$$P(x_i|z_{i_1}^1,\cdots,z_{i_k}^k) = m_{x|z_1,\cdots,z_k}(x_i) \tag{14.18}$$

For $k = 1$, according to equation 14.13

$$\begin{aligned}
m_{x|z}(x_i) =& \frac{1}{1-\kappa} \sum_{X_j, Y_k | X_j \cap Y_k = x_i} m_x(X_j) m_z(Y_k) \\
=& \frac{\sum_{X_j, Y_k | X_j \cap Y_k = x_i} m_x(X_j) m_z(Y_k)}{1 - \sum_{X_j, Y_k | X_j \cap Y_k = \emptyset} m_x(X_j)\, m_z(Y_k)} \\
=& \frac{\sum_{X_j, Y_k | X_j \cap Y_k = x_i} m_x(X_j) m_z(Y_k)}{\sum_{X_j, Y_k | X_j \cap Y_k \neq \emptyset} m_x(X_j)\, m_z(Y_k)} \\
=& \frac{m_x(x_i) \sum_{Y_k | x_i \cap Y_k \neq \emptyset} m_z(Y_k)}{\sum_{x_j} m_x(x_j) \sum_{Y_k | x_j \cap Y_k \neq \emptyset} m_z(Y_k)} \\
=& \frac{m_x(x_i) l_z(x_i)}{\sum_{x_j} m_x(x_j) l_z(x_j)} \\
=& \frac{P(x_i) p(z_{i_1}^1 | x_i)}{\sum_{x_j} P(x_j) p(z_{i_1}^1 | x_j)} \\
=& P(x_i | z)
\end{aligned} \tag{14.19}$$

If for $k-1$, it is assumed that

$$P(x_i | z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) = m_{x|z_1,\cdots,z_{k-1}}(x_i) \tag{14.20}$$

then for k

$$\begin{aligned}
m_{x|z_1,\cdots,z_k}(x_i) =& \frac{\sum_{X_j, Y_k^N | X_j \cap Y_k^N = x_i} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N)}{1 - \sum_{X_j, Y_k^N | X_j \cap Y_k^N = \emptyset} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N)} \\
=& \frac{m_{x|z_1,\cdots,z_{k-1}}(x_i) \sum_{Y_k^N | x_i \cap Y_k^N \neq \emptyset} m_{z_N}(Y_k^N)}{\sum_{x_j} m_{x|z_1,\cdots,z_{k-1}}(x_j) \sum_{Y_k^N | x_j \cap Y_k^N \neq \emptyset} m_{z_N}(Y_k^N)} \\
=& \frac{m_{x|z_1,\cdots,z_{k-1}}(x_i) l_z(x_i)}{\sum_{x_j} m_{x|z_1,\cdots,z_{k-1}}(x_j) l_z(x_j)} \\
=& \frac{p(z_{i_k}^k | x_i) P(x_i | z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1})}{\sum_{x_j} p(z_{i_k}^k | x_j) P(x_j | z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1})} \\
=& P(x_i | z_{i_1}^1, \cdots, z_{i_k}^k)
\end{aligned} \tag{14.21}$$

This completes the proof that Dempster-Schafer reasoning is identical to Bayesian reasoning if the recommended procedure for basic mass assignment under a complete probability model is followed. The combined *a posteriori* mass function $m_{x|z_1,\cdots,z_k}(x_i)$ is just the *a posteriori* probability.

14.2 Partial a Priori Probability Models

When only partial prior probability models are available, that is, the *a priori* focal elements are J subsets $X_j \subseteq \Theta$ with $\mathcal{P}$- distribution $\mathcal{P}(X_j)$ (Appendix C). The traditional Bayesian reasoning method won't work. In order to be able to use Bayesian reasoning techniques, one needs to make assumptions about mass distributions within each $\mathcal{P}(X_j)$. One feasible assumption is that the mass is distributed uniformly within each X_j, thus

$$P(x_i) = \sum_{j \mid x_i \in X_j} \frac{\mathcal{P}(X_j)}{|X_j|}, \tag{14.22}$$

where $|X_j|$ is the cardinality of the set X_j. If the actual distribution is not uniform, the assumption might introduce huge errors to the tracking system. However, evidential reasoning techniques don't need to make those assumptions.

The *a priori* mass assignments are straight forward:

$$m_x(X) = \begin{cases} \mathcal{P}(X_j), & \text{for } X = X_j \text{ is a focal element} \\ 0, & \text{otherwise} \end{cases} \tag{14.23}$$

Since the complete transitional probability model is available, one can use the recommended transitional mass assignment as described in the previous section for $m_z(Y_k)$. The mass combination is done according to *Dempster's Rule*:

$$\begin{aligned} m_{x|z}(X_i) &= \frac{1}{1-\kappa} \sum_{X_j, Y_k | X_j \cap Y_k = X_i} m_x(X_j) m_z(Y_k) \\ \kappa &= \sum_{X_j, Y_k | X_j \cap Y_k = \emptyset} m_x(X_j)\, m_z(Y_k) \end{aligned} \tag{14.24}$$

Again, we can recursively combine the mass functions of $m_{x|z_1,\cdots,z_{N-1}}(X_i)$ and $m_{z_N}(Y_j^N)$ based on the N-th observation $z_{i_N}^N$ by

$$\begin{aligned} m_{x|z_1,\cdots,z_{N-1},z_N}(X_i^N) &= \frac{1}{1-\kappa} \sum_{X_j, Y_k^N | X_j \cap Y_k^N = X_i^N} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N) \\ \kappa &= \sum_{X_j, Y_k^N | X_j \cap Y_k^N = \emptyset} m_{x|z_1,\cdots,z_{N-1}}(X_j) m_{z_N}(Y_k^N) \end{aligned} \tag{14.25}$$

The difference is that now the focal elements of $m_{x|z}(\cdot), \cdots, m_{x|z_1,\cdots,z_{N-1},z_N}(\cdot), \cdots$ are not singletons any more, they are just subsets of Θ.

According to the discussion in Section (6.1), the generalized likelihood function associated with the attribute term in equation (12.3) can be constructed as follows:

$$p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) = p_t^k(z_{i_k}^k|z_{i_1}^1, \cdots, z_{i_{k-1}}^{k-1}) = \sum_{X_i \in \mathcal{X}} p(z_{i_k}^k|X_i)\ m_{x|z_1,\cdots,z_{k-1}}(X_i),$$

$$p(z_{i_k}^k|X_i) = \sum_{x \in X_i} p(z_{i_k}^k|x). \tag{14.26}$$

Here, $p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k})$ does not have the traditional meaning of a probability any more.

14.3 Partial Transitional Probability Models

Under a partial transitional probability model, $p(Z_j^i|x_i)$, for any $i = 1, \cdots, m$, $j = 1, \cdots, n_i$, $Z_j^i \subseteq \mathcal{Z}$ are given, where $\mathcal{Z}$ is the entire attribute measurement space. One can not compute the *a posteriori* mass function $m_{x|z_1,\cdots,z_k}(X_i)$ recursively as explained before.

14.3.1 Approaches to Computing A Posteriori Mass

Blackman's book [8] recommends the following two approaches to compute $m_{x|z_1,\cdots,z_k}(X_i)$:

- Power set approach

  Define the state of nature to be that attribute type x_i generates a measurement that is in Z_j^i. Thus, the total number of possible states are $\prod_{j=1}^m n_j$. The power set approach assumes uniform *a priori* distribution of all the states of nature. For each subspace $Z(k)$ of $\mathcal{Z}$, the approach considers all the possible states and defines $m_{x|Z}(X_j(k)|Z(k))$, for $j = 1, \cdots, j_k$. The subspace $Z(k)$'s are generally a partition of $\mathcal{Z}$, and equals the set intersection of certain Z_j^i's.

- Typical Sequence approach

  The concept of a typical sequence is defined in terms of a J-element partition, $\mathcal{U}_i$, given the true target attribute x_i.

$$\mathcal{U}_i = \{a_j, \cdots, a_1 \mid x_i\}$$

  The probabilities of the event partitions $a_{j|i}$ are

$$P(a_j|x_i) \;=\; P_{j|i}$$

  For n repeated independent trials, the elements $a_{j|i}$ of $\mathcal{U}_i$ form sequences of the form

$$\{a_{j|i} \text{ occurs } n_j \text{ times in a specific order}\}$$

  with the probability of each sequence given by

$$P_{1|i}^{n_1} \cdots P_{j|i}^{n_j} \cdots P_{J|i}^{n_J}$$

  where $n_1 + \cdots + n_J = n$.

  The sequence is called *typical* for x_i if $n_j \approx nP_{j|i}$. All other sequences are called *rare*.

  In our problem of a partial transitional probability model, the a_j's are just subspace Z_j^i's. Thus, the a_j's are not necessarily disjoint. Given a measurement z_k, heuristic methods are developed to choose the a_j such that $z_k \in a_j$ and maximizing the typicality of the given measurement sequence $z_1, \cdots, z_N$.

  Thus, the typical sequence approach computes the typicality of a given sequence of measurements assuming given attribute x_i, for $i = 1, \cdots, m$. The *a posterior* focal elements and the corresponding mass assignments are all based on that.

14.3.2 Generalized Likelihood

In partial transitional probability models, the generalized likelihood function is not even determined. We can only find the maximum and minimum of a generalized likelihood:

$$\begin{aligned}\max\ p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) &= \max\ p_t^k(z_{i_k}^k|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}) \\ &= \sum_{x_i|x_i\in\mathcal{X}} \{\{\max_{Z_p^i|z_{i_k}^k\in Z_p^i} p(Z_p^i|x_i)\}\{\sum_{X_j|x_i\in X_j} m_{x|z_1,\cdots,z_{k-1}}(X_j)\}\}\end{aligned} \tag{14.27}$$

$$\begin{aligned}\min\ p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k}) &= \min\ p_t^k(z_{i_k}^k|z_{i_1}^1,\cdots,z_{i_{k-1}}^{k-1}) \\ &= \sum_{x_i|x_i\in\mathcal{X}} \{\{\min_{Z_p^i|z_{i_k}^k\in Z_p^i} p(Z_p^i|x_i)\}\quad m_{x|z_1,\cdots,z_{k-1}}(x_i)\}\end{aligned} \tag{14.28}$$

Again, the correspondence between $p_t^k(z_{i_k}^k|Z_{i_1\cdots i_k})$ and the probability theory is uncertain.

Chapter 15

SIMULATION: RANGE EXTENT

Range extent is a feature provided by modern surveillance systems. It is the length that the target is projected onto the range direction.

For simplification, the following $2-D$ problem is considered. A Cartesian East-North system is fixed at the sensor location. The sensor measurements are range (r), azimuth (a), Doppler $(\dot{r})$, and range extent (d). The target is not regarded as a point target anymore. It is assumed to be a rectangle of dimension $W \times L$. Target velocity is assumed to be parallel to the long side L (Figure 15.1).

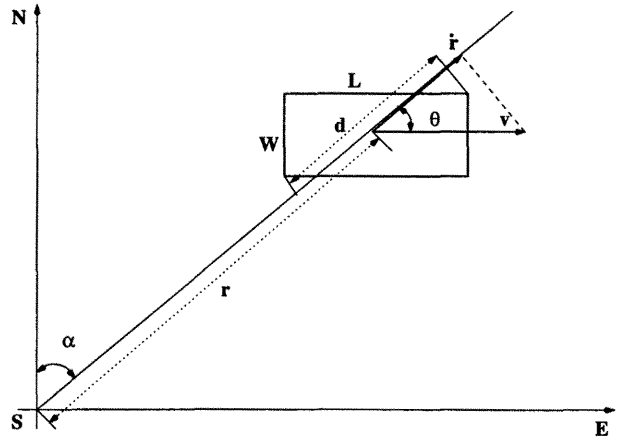


Figure 15.1: 2-D measurements

15.1 Problem Formulation

A nearly constant velocity model is used to describe the target's dynamics, and the target state at time t_k is chosen to be $\mathbf{X_k} = [x_k, \dot{x}_k, y_k, \dot{y}_k]^T$. Given an

additional feature measurement (range extent), the measurement vector is $\mathbf{z_k} = [r_k, a_k, \dot{r}_k, d_k]^T$. The dynamic state vector is augmented with two feature states, width (W) and length (L). So, $\mathbf{X_k} = [x_k, \dot{x}_k, y_k, \dot{y}_k, w_k, l_k]$. The state space is modeled by

$$\mathbf{X_{k+1}} = \mathbf{F_k X_k} + \mathbf{w_k}, \tag{15.1}$$

$$\mathbf{z_k} = \mathbf{h(X_k)} + \mathbf{v_k}, \tag{15.2}$$

where

$$\mathbf{F_k} = \begin{bmatrix} 1 & \Delta t_k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t_k & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{15.3}$$

and $\mathbf{w_k}$, $\mathbf{v_k}$ are white Gaussian noise with zero mean. The covariance matrices are

$$\mathbf{Q_k} = \mathbf{E}[\mathbf{w_k w_k^T}] = \begin{bmatrix} q_x \frac{\Delta t_k^3}{3} & q_x \frac{\Delta t_k^2}{2} & 0 & 0 & 0 & 0 \\ q_x \frac{\Delta t_k^2}{2} & q_x \Delta t_k & 0 & 0 & 0 & 0 \\ 0 & 0 & q_y \frac{\Delta t_k^3}{3} & q_y \frac{\Delta t_k^2}{2} & 0 & 0 \\ 0 & 0 & q_y \frac{\Delta t_k^2}{2} & q_y \Delta t_k & 0 & 0 \\ 0 & 0 & 0 & 0 & \epsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & \epsilon \end{bmatrix}$$

and

$$\mathbf{R_k} = \mathbf{E}[\mathbf{v_k v_k^T}] = \begin{bmatrix} \sigma_r^2 & 0 & 0 & 0 \\ 0 & \sigma_a^2 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{r}}^2 & 0 \\ 0 & 0 & 0 & \sigma_d^2 \end{bmatrix}$$

In the first matrix, q_x, q_y are parameters that indicate noise levels in the NCV model and ϵ is some small number that one sets to make sure $\mathbf{Q_k}$ is positive definite. In the second, $\sigma_r^2, \sigma_a^2, \sigma_{\dot{r}}^2$ and σ_d^2 are measurement error variances that are decided by sensor characteristics. The measurement function $\mathbf{h(X_k)} = [h_r(\mathbf{X_k}), h_a(\mathbf{X_k}), h_{\dot{r}}(\mathbf{X_k}), h_d(\mathbf{X_k})]^T$

is nonlinear:

$$h_r(\mathbf{X_k}) = \sqrt{x_k^2 + y_k^2}, \tag{15.4}$$

$$h_a(\mathbf{X_k}) = \arctan\left(\frac{x_k}{y_k}\right) \tag{15.5}$$

$$h_{\dot{r}}(\mathbf{X_k}) = \frac{x_k\dot{x}_k + y_k\dot{y}_k}{\sqrt{x_k^2 + y_k^2}}, \tag{15.6}$$

$$h_d(\mathbf{X_k}) = w_k \sin\theta_k + l_k \cos\theta_k, \tag{15.7}$$

where $\theta_k = \arctan\left(\frac{\dot{x}_k}{\dot{y}_k}\right) - \arctan\left(\frac{x_k}{y_k}\right)$.

15.2 Observability Problem

A (deterministic) system is **completely observable** if its initial state can be *fully and uniquely* recovered from a finite number of observations of its output and full knowledge of its input. The system described by equations (15.1) and (15.2) is **locally observable** if the observation matrix

$$\mathbf{Q_{ok}} = \begin{bmatrix} H_k \\ H_k\, F_k \\ \vdots \\ H_k\, F_k^{n_x-1} \end{bmatrix} \tag{15.8}$$

has full rank n_x, where n_x is the dimension of state vector. Actually, the rank of $\mathbf{Q_{ok}}$ for the system described by (15.1) and (15.2) is 5, while $n_x = 6$. This means the system is not observable.

An example problem in which cars are tracked on a highway is considered, in which case, most of the cars are of the same width. In state equation (15.1), assume that the width is known, and augment the dynamic state with length only. Then, $\mathbf{X_k} = [x_k, \dot{x}_k, y_k, \dot{y}_k, l_k]$. The state transition matrix $\mathbf{F_k}$ in (15.1) is

$$\mathbf{F_k} = \begin{bmatrix} 1 & \Delta t_k & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t_k & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{15.9}$$

the covariance matrix of $\mathbf{w_k}$ is

$$\mathbf{Q_k} = \mathbf{E}[\mathbf{w_k}\mathbf{w_k^T}] = \begin{bmatrix} q_x\frac{\Delta t_k^3}{3} & q_x\frac{\Delta t_k^2}{2} & 0 & 0 & 0 \\ q_x\frac{\Delta t_k^2}{2} & q_x\Delta t_k & 0 & 0 & 0 \\ 0 & 0 & q_y\frac{\Delta t_k^3}{3} & q_y\frac{\Delta t_k^2}{2} & 0 \\ 0 & 0 & q_y\frac{\Delta t_k^2}{2} & q_y\Delta t_k & 0 \\ 0 & 0 & 0 & 0 & \epsilon \end{bmatrix}. \tag{15.10}$$

The observation matrix of the above system if of full rank.

15.3 Simulation Results

The above example problem in which cars are tracked on a highway is tested using matlab. Two extended Kalman filters are used for comparison purposes. The first one uses an NCV model in its state equation and the state vector consists of only position and velocity information. The measurements are range, azimuth and Doppler. The other one makes use of range extent and the state vector is augmented with length, while width is assumed to be known. Its state and measurement equations are described by (15.1) and (15.2) with the matrices defined by (15.9) and (15.10).

Assume the car being tracked is of dimension 2×4 meters. Sensor measurements are generated every second. Sensor parameters are given as follows:

$$\begin{aligned} \sigma_r^2 &= 100\ m^2 \\ \sigma_a^2 &= 0.001\ radian^2 \\ \sigma_{\dot{r}}^2 &= 100\ \frac{m^2}{s^2} \\ R_{max} &= 2 \times 10^5\ m \\ R_{max} &= 200\ \frac{m}{s} \\ D_{max} &= 10\ m \end{aligned} \tag{15.11}$$

As discussed in Chapter 11, the likelihood score obeys:

$$c_{i_1 \cdots i_{k-1} i_k} = c_{i_1 \cdots i_{k-1}} + c_{i_k} \tag{15.12}$$

where

$$c_{i_k}^{without} = \ln \frac{\frac{1}{|2\pi \mathbf{S_k}|} \exp\left(-\frac{1}{2} dz_k^T \mathbf{S_k^{-1}} dz_k\right)}{R_{max} \times 2\pi \times \dot{R}_{max}} \tag{15.13}$$

$$c_{i_k}^{with} = \ln \frac{\frac{1}{|2\pi \mathbf{S_k^w}|} \exp\left(-\frac{1}{2} dz_k^{wT} (\mathbf{S_k^w})^{-1} dz_k^w\right)}{R_{max} \times 2\pi \times \dot{R}_{max} \times D_{max}} \tag{15.14}$$

In equation (15.13), $\mathbf{S_k}$ and dz_k are the innovation matrix (3×3) and predicted measurement error (3×1) given by a Kalman filter that does not use range extent and where the state vector is dynamics only. In equation (15.14), $\mathbf{S_k^w}$ and dz_k^w are the innovation matrix (4×4) and predicted measurement error (4×1) given by a Kalman filter that does use range extent and where the state vector is dynamics concatenated with length and width.

The likelihood scores after 100 steps are compared for 20 Monte Carlo runs. Different σ_d values that range from 0.1 to 3 are chosen. Figure (15.2) shows how $\frac{c_{i_{100}}^{with} - c_{i_{100}}^{without}}{c_{i_{100}}^{without}}$ changes with respect to $\frac{\sigma_d}{D_{max}}$.

If the value $\frac{c_{i_{100}}^{with} - c_{i_{100}}^{without}}{c_{i_{100}}^{without}}$ is greater than zero, this means with range extent, we are able to improve our likelihood ratio scores. As can be seen in Figure (15.2), score improvement can be up to 14% under small measurement noises (small σ_d). When $\frac{\sigma_d}{D_{max}}$ increases to around 0.22, updating the filtering with range extent information doesn't help any more. If the range extent measurement is extremely noisy, *i.e.*, $\frac{\sigma_d}{D_{max}} > 0.22$, the likelihood ratio score is even worse than the filter using dynamic information only.

More Monte Carlo simulations were performed for targets with different dimensions. The results are similar to Figure (15.2).

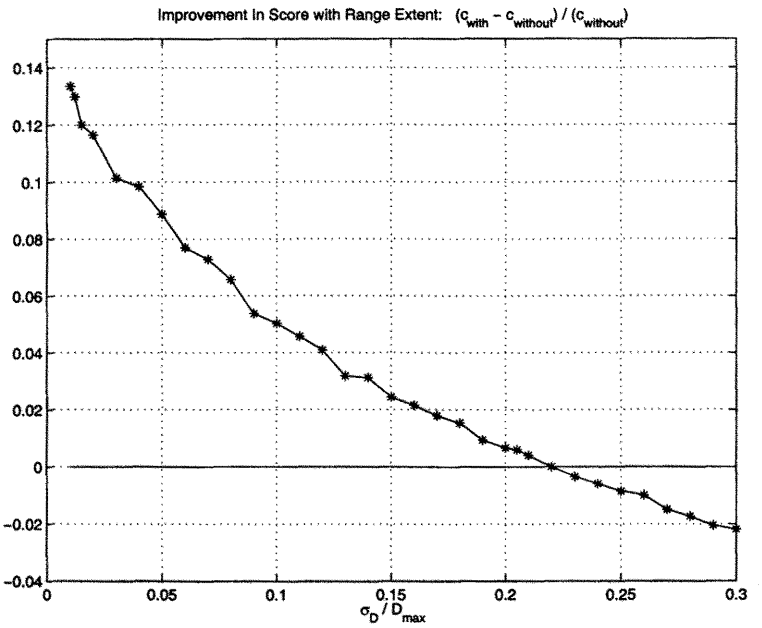


Figure 15.2: Score Improvements Using Range Extent

15.4 Conclusions

The above simulations show that one must assure the new system maintains observability when attempting to augment the dynamic state vector (position and velocity) with feature states. Numerically stable methods should be applied when taking the inverse of the innovation matrix ($\mathbf{S_k}$), if the observation matrix is close to singular, or ill-conditioned.

With relatively accurate feature measurements (small σ_d's), one can improve the likelihood ratio score by augmenting the state and updating the filter with feature measurements. If the feature measurements are relatively noisy (big σ_d's), then the feature information should be discarded.

Bibliography

[1] B.D. Anderson and J.B. Moore. *Optimal Filtering.* Prentice Hall, Englewood Cliffs, NJ, 1979.

[2] Y. Bar-Shalom and K.C. Chang. Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, AES-25:296–300, March 1989.

[3] Yaakov Bar-Shalom. *Multitarget-Multisensor Tracking: Advanced Applications.* Artech House, Norwood, MA, 1990.

[4] Yaakov Bar-Shalom and W. D. Blair. *Multitarget/Multisensor Tracking: Applications and Advances III.* Artech House, Norwood, MA, 2000.

[5] Yaakov Bar-Shalom and X.R. Li. *Multitarget-Multisensor Tracking Principles and Techniques.* YBS Publishing, Storrs, CT, 1995.

[6] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, Upper Saddle River, NJ, 1992.

[7] S. Blackman. *Multiple target tracking with radar application.* Artech House, Norwood, MA, 1986.

[8] S. Blackman and R. Popoli. *Design and analysis of Modern tracking systems.* Artech House, Norwood, MA, 1999.

[9] S.S. Blackman. Interacting multiple models filtering for radar system application. *Technical Report*, 1993.

[10] S.S. Blackman, R.J. Dempster, and S.H. Roszkowski. IMM/MHT applications to radar and IR multitarget tracking. *Proc. of Signal and Data Processing of Small Targets 1997*, SPIE 3163:429–439, July 1997.

[11] W.D. Blair and W. Brandt-Pearce. Tracking multiple unresolved rayleigh targets with a monopulse radar. *Signal and data processing for small targets, SPIE*, April 1996.

[12] W.D. Blair and W. Brandt-Pearce. Statistics of monopulse measurements for tracking targets in the presence of sea-surface induced multipath. *IEEE Aerospace Conference*, March 1998.

[13] W.D. Blair and W. Brandt-Pearce. Statistics of monopulse measurements for tracking targets in the presence of sea-surface induced multipath. *IEEE Aerospace Conference*, March 1998.

[14] H.A.P. Blom. An efficient decision-making-free filter for progresses with abrupt changes. *Proceedings of the International Federation of Automatic Control Symposium on Identification and System Parameter Estimation*, pages 631–636, July 1985.

[15] H.A.P. Blom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transaction on Automatic Control*, 33:780–783, August 1988.

[16] P.L. Bogler. Shafer-Dempster reasoning with application to multisensor target identification systems. *IEEE Trans. on systems, man and cybernetics*, SMC-17:968–977, 1987.

[17] E. Bosse and M. Simard. Managing evidential reasoning for identity information fusion. *Optical Engineering*, pages 391–400, February 1998.

[18] D.M. Buede. Shafer-Dempster and Bayesian reasoning: A response to Shafer -Dempster reasoning with application to multisensor target. *IEEE Trans. on systems, man and cybernetics*, 18:1009–1011, 1988.

[19] M.T. Busch and S.S. Blackman. Evaluation of IMM filtering for an air defense system application. *Signal and Data Processing of Small Targets 1995, SPIE*, 2561:435–447, July 1995.

[20] E. Cassassolles and L. Martinet. Integration of radar measurement attributes in the multiple hypothesis tracker results for track initiation. *Signal and data processing of small targets 1996, SPIE*, 2759:397–403, 1996.

[21] F.R. Castella. Multisensor, multisite tracking filter. *IEE Proc. Radar, Sonar Navigation*, 141:75–83, April 1994.

[22] K.C. Chang, Jun Liu, and Jing Zhou. Bayesian probabilistic inference for target recognition. *Signal processing, sensor fusion and target recognition, V, Proc. SPIE*, 2755:158–165, April 1996.

[23] Kuo-Chu Chang and Robert Fung. Target identification with bayesian networks in a multiple hypothesis tracking system. *Optical Engineering*, 36:684–491, March 1997.

[24] Chee-Yee Chong, Shozo Mori, and Kuo-Chu Chang. Distributed multitarget multisensor tracking. In *Multitarget-Multisensor Tracking: Advanced Applications* [3].

[25] C.Y. Chong and S. Mori. Hierarchical multitarget tracking and classification, a bayesian approach. *Proceedings of the Second International Conference on Information Fusion*, 2:1045–1053, 1999.

[26] C.K. Chui and G. Chen. *Kalman Filtering with real-time applications, 2nd edition.* Springer Series in Information Sciences, Springer-Verlag, New York, 1991.

[27] F. E. Daum and R. J. Fitzgerald. The importance of resolution in multiple target tracking. *Signal and Data Processing of Small Targets*, 329, 1994.

[28] R.J. Dempster and S.S. Blackman. IMM/MHT solution to radar and multisensor benchmark tracking problem. *Signal and Data Processing of Small Targets 1998, SPIE*, 3373, April 1998.

[29] O. E. Drummond. A hybrid fusion algorithm architecture and tracklets. *Signal and Data Processing of Small Targets 1997, SPIE*, 3136:485–502, 1997.

[30] O. E. Drummond. A hybrid sensor fusion algorithm architecture and tracklets. *Signal and Data Processing of Small Targets*, SPIE 3163:485–502, 1997.

[31] O. E. Drummond. Tracklets and a hybrid fusion with process noise. *Signal and Data Processing of Small Targets*, SPIE 3163:512–524, 1997.

[32] O. E. Drummond and S. S. Blackman. Challenges of developing algorithms for multiple sensor, multiple target tracking. *SPIE Proceedings*, 1096:244 – 255, 1989.

[33] Oliver E. Drummond. On features and attributes in multisensor, multitarget tracking. *Proceedings of the American Control Conference*, 2:599–604, 1984.

[34] Oliver E. Drummond. Multiple sensor, multiple target tracking. *SPIE's 44th Annual Meeting, Short Course Notes*, 1999.

[35] Robert Fung and K.C. Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. *Proceeding of 5th Workshop on Uncertainty in AI*, pages 112–117, 1989.

[36] Erich Gamma and Richard Helm. *Design Patterns.* Addison-Wesley, Reading, Massachusetts, 1995.

[37] F.D. Garber and N.F. Chamberlin. Time-domain and frequency-domain feature selection for reliable radar target identification. *IEEE National Radar conference*, pages 79–84, April 1988.

[38] J.L. Gertz. Multisensor surveillance for improved aircraft tracking. *The Lincoln Laboratory Journal*, 2:473–483, 1989.

[39] Jean Gordon and E. H. Shortliffe. A method for managing evidential reasoning in a hierarchical hypothesis space: a retrospective. *Artificial Intelligence*, 59:43–47, 1993.

[40] R.D. Hilton, D.A. Martin, and W.D. Blair. Tracking with time-delayed data in multisensor systems. *NSWCDD/TR-93/351*, August 1993.

[41] Ronald A. Iltis and K.L. Anderson. A consistent estimation criterion for multisensor bearings only tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 32:108–119, January 1996.

[42] Lain-Wen Jang and Jung-Jae Chao. An information fusion algorithm for data association in multitarget tracking. *Proceedings of the First Australian Data Fusion Symposium*, pages 119–124, 1996.

[43] K. Kim. Bayesian inference network: Applications to target tracking. *Signal and Data Processing of Small Targets 1992, SPIE*, 1698:360–371, 1992.

[44] Thiagalingam Kirubarajan, H. Wang, and Y. Bar-Shalom. Efficient multisensor fusion using multidimensional assignment for multitarget tracking. *SPIE Proceedings*, 3374:14–25, 1998.

[45] Jurg Kohlas and Paul-Andre Monney. *A Mathematical Theory of Hints.* Springer-Verlag, Berlin, 1995.

[46] Jeffery R. Layne and David Simon. A multiple model estimator for a tightly coupled HRR automatic target recognition and MTI tracking system. *SPIE Proceedings*, 3721:362–373, 1999.

[47] M.D. Levine. Feature extraction: A survey. *Proceedings of the IEEE*, 57:1391–1407, August 1969.

[48] X. Rong Li and Y. Bar-Shalom. Tracking in clutter with nearest neighbor filters: Analysis and performance. *IEEE Transactions on Aerospace and Electronic Systems*, 32:995–1009, July 1996.

[49] XiaoRong Li and Vesselin P. Jilkov. Survey of maneuvering target tracking: dynamic models. *SPIE Proceedings*, 4048:212 – 235, 2000.

[50] Jun Liu and Kuo-Chu Chang. Feature-based target recognition with a bayesian network. *Optical Engineering*, 35:701–707, March 1996.

[51] Ronald Mahler. An introduction to multisource-multitarget statistics and its applications. Technical report, Lockheed Martin, March 2000.

[52] Kenneth S. Miller and D.M. Leskiw. Nonlinear estimation with radar observations. *IEEE Transactions on Aerospace and Electronic systems*, AES-18:192–200, March 1982.

[53] M. D. Miller and O. E. Drummond. Tracklets and covariance truncation options for theater missile tracking. *Proceedings of the International Conference on Multisource-multisensor Information Fusion*, pages 165–174, July 1998.

[54] Longbin Mo and Yaakov Bar-Shalom. Unbiased converted measurements for tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 34:1023–1027, July 1998.

[55] J. R. Moore and W. D. Blair. Practical aspects of multisensor tracking. In *Multitarget/Multisensor Tracking: Applications and Advances III* [4].

[56] R.L. Moses and J.W. Carl Jr. Autoregressive modeling of radar data with application to target identification. *IEEE Nation radar conference*, pages 220–229, April 1988.

[57] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes, 2nd Edition.* McGrawPHill Book Company, New York, 1984.

[58] John B. Pearson and Edwin B. Stear. Kalman filter applications in airborne radar tracking. *IEEE Transactions on Aerospace and Electronic Systems*, AES-10:319–329, May 1974.

[59] A. B. Poore. Multidimensional assignment formulation of data association problems arising from multitarget tracking and multisensor data fusion. *Computational Optimization and Application*, pages 27–57, 1994.

[60] A. B. Poore and A. J. Robertson III. A new class of lagrangian relaxation based algorithms for a class of multidimensional assignment problems. *Computational Optimization and Applications*, 8(2):129–150, 1997.

[61] Aubrey B. Poore and Xin Yan. Some algorithmic improvements in multiframe most probable hypothesis tracking. *Signal and Data Processing of Small Targets, SPIE*, 1999.

[62] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, AC-24:843–854, 1979.

[63] J.B. Romine and E.W. Kamen. Target maneuver detection using image features. *Signal and data procession for small targets, SPIE*, 1995.

[64] R. Rothrock and O. E. Drummond. Pilot JCTN algorithm benchmarking performance metrics. Technical report, Office of Naval Research, 1999.

[65] Ross D. Shachter, B. D'Ambrosio, and B.A. Del Favero. Symbolic probabilistic inference in belief networks. *Automated Reasoning*, pages 126–131, 1990.

[66] Glenn Shafe and Roger Logan. Implementing Dempster's rule for hierarchical evidence. *Artificial Intelligence*, 33:271–298, 1987.

[67] R.A. Singer. Estimating optimal tracking filter performance for manned maneuvering targets. *IEEE Transactions on Aerospace and Electronics Systems*, AES-6:473–483, July 1970.

[68] R.A. Singer and K.W. Behnke. Real-time tracking filter evaluation and selection for practical applications. *IEEE Transaction on Aerospace and Electronic Systems*, AES-7:100–110, January 1971.

[69] R.A. Singer, R.G. Sea, and K.B. Housewright. Derivation and evaluation of improved tracking filters for use in dense multitarget environments. *IEEE Transactions on Information Theory*, pages 423–432, July 1974.

[70] R.A. Singer and J.J. Stein. An optimal tracking filter for processing sensor data of imprecisely determined origin in surveillance systems. *Proc. IEEE Conf. Decision and Control*, pages 171–175, December 1971.

[71] R.W. Sittler. An optimal data association problem in surveillance theory. *IEEE Transactions on Military Electronics*, MIL-8:125–139, April 1964.

[72] Taek L. Song. Observability of target tracking with bearings only measurements. *IEEE Transaction on Aerospace and Electronic Systems*, 32:1468–1471, October 1996.

[73] W. Richard Stevens. *Unix Network Programming, Networking APIs: Sockets and XTI.* Prentice Hall, Upper Saddle River, NJ, 1998.

[74] Bjarne Stroustrup. *The C++ Programming Language, third edition.* Addison-Wesley, Reading, Massachusetts, 1997.

[75] Bjarne Stroustrup. *The C++ Programming Language, special edition.* Addison-Wesley, Reading, Massachusetts, 2000.

[76] J.G. Teti and R.P. Gorman. A Multifeature decision space approach to radar target identification. *IEEE transactions on Aerospace and Electronic Systems*, 32:480–487, January 1996.

[77] O. Tremois and J.P. Le Cadre. Target motion analysis with multiple arrays: Performance analysis. *IEEE Transactions on Aerospace and Electronics Systems*, 32:1030–1045, July 1996.

[78] G.V. Trunk, S. Brockette, and J.D. Wilson. Tracking with velocity. *Signal and Data Processing of Small Targets 1992, SPIE*, 1698:425–431, 1992.

[79] Mark Walmsley. *Multi-Threaded Programming in C++*. Springer, London, 2000.

[80] Gregory A. Watson, K.S. Anthony, and T.R. Rice. Multisensor, multisite integration for composite tracking of maneuvering targets. *Signal and Data Processing of Small Targets 1998, SPIE*, 3373:308–319, April 1998.

[81] Gregory A. Watson and W.D. Blair. IMM algorithm for tracking targets that maneuver through coordinated turns. *Proc. of Signal and Data Processing of Small Targets 1992*, SPIE 1698:236–247, April 1992.

[82] R. Wu and K.C. Chang. Maneuvering target tracking with colored noise. *IEEE Transactions on Aerospace and Electronics Systems*, 32:1311–1319, October 1996.

Appendix A

PROBABILITY CALCULATIONS

The objective in this appendix is to derive specific formulas for the factors $p(Z(k)|\Omega_l^k, Z^{k-1})$ and $P(\psi_l(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1})$ in equation (11.26). For the first factor, note that each report is assigned to a previous track, to a new source, or to a false alarm. Using the symbols defined in (11.27), the likelihood of the reports $Z(k)$ given the association hypothesis is

$$p(Z(k)|\Omega_l^k, Z^{k-1}) = \prod_{i_k=1}^{M_k} \left\{ \left[p_t^k(z_{i_k}^k|\Omega_l^k, Z^{k-1}) \right]^{\delta_{i_k}^k} \left[p_\nu^k(z_{i_k}^k|\Omega_l^k, Z^{k-1}) \right]^{\nu_{i_k}^k} \left[p_f^k(z_{i_k}^k|\Omega_l^k, Z^{k-1}) \right]^{f_{i_k}^k} \right\} \quad \text{(A.1)}$$

where $p_t^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ represents the likelihood that the report originated from a target maintaining a certain dynamic model, $p_\nu^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ represents the likelihood that the report originated from a new source, and $p_f^k(z_{i_k}^k|\Omega_l^k, Z^{k-1})$ represents the likelihood the report represents a false alarm.

The development of an expression for $P(\psi_l(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1})$ makes extensive use of the notation and definitions in (11.27). Let $\psi_{lN}(k)$ denote the event that postulates the *numbers* $\{f^k, \nu^k, \delta^k, \chi^k\}$ defined in (11.27). Of the M_k reports, ν^k originate from new targets with an associated probability mass function $\mu_\nu^k(\nu^k)$, f_k are false reports with probability mass function $\mu_f^k(f^k)$, and the remaining δ_k reports are associated with existing targets. Of the τ^k targets that exist after scan $k-1$, χ^k of these τ^k targets are terminated and are not observed (on scan k), δ^k of

the $\tau^k - \chi^k$ non-terminated targets are detected, and $\tau^k - \chi^k - \delta^k$ non-terminated targets are not detected. (Thus, the total number of targets that exist after scan k is $\tau_{k+1} = \tau^k - \chi^k + \nu^k$.) Recalling that the binomial coefficient

$$\binom{n}{m} \equiv \frac{n!}{(n-m)!m!}$$

gives the number of ways to arrange n elements taken m at a time, one can show [62]

$$\begin{aligned} P(\psi_{lN}(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1}) = &\left\{\mu_f^k(f^k)\mu_\nu^k(\nu^k)\right\}\left\{\binom{\tau^k}{\chi^k}(P_\chi^k)^{\chi^k}(1-P_\chi^k)^{\tau^k-\chi^k}\right\} \\ &\left\{\binom{\tau^k-\chi^k}{\delta^k}(P_d^k)^{\delta^k}(1-P_d^k)^{\tau^k-\chi^k-\delta^k}\right\}. \end{aligned} \tag{A.2}$$

The number of ways M_k observations can be divided into δ^k detected targets, f^k false reports, and ν^k new targets is

$$\binom{M_k}{\delta^k}\binom{M_k-\delta^k}{\nu^k}\binom{M_k-\delta^k-\nu^k}{f^k} = \binom{M_k}{\delta^k}\binom{M_k-\delta^k}{\nu^k}.$$

Also, the number of ways τ^k targets can be divided into χ^k terminated targets, δ^k extended targets from scan $(k-1)$, and $\tau^k - \chi^k - \delta^k$ missed targets is

$$\binom{\tau^k}{\chi^k}\binom{\tau^k-\chi^k}{\delta^k}\binom{\tau^k-\chi^k-\delta^k}{\tau^k-\chi^k-\delta^k} = \binom{\tau^k}{\chi^k}\binom{\tau^k-\chi^k}{\delta^k}.$$

Let $\psi_{lC}(k)$ denote the event within $\psi_{lN}(k)$ that designates a specific set of δ^k extended targets from scan $(k-1)$, χ^k terminated targets, $\tau^k - \chi^k - \delta^k$ missed targets, δ^k detected targets, f^k false reports, and ν^k new targets. Assuming each such event is equally likely,

$$P(\psi_{lC}(k)|\psi_{lN}(k), \Omega_{\psi_l}^{k-1}, Z^{k-1}) = \left\{\binom{\tau^k}{\chi^k}\binom{\tau^k-\chi^k}{\delta^k}\binom{M_k}{\delta^k}\binom{M_k-\delta^k}{\nu^k}\right\}^{-1} \tag{A.3}$$

From the event $\psi_{lC}(k)$, $\psi_l(k)$ represents a specific hypothesis that assigns a specific set of reports (detected targets on scan k) to the extended targets from

scan $(k-1)$. The number of ways to assign δ^k detections to δ^k targets is $\delta^k!$. Thus, assuming the probability for each such assignment is the same, the probability of the hypothesis $\psi_l(k)$ given $\psi_{lC}(k)$ is

$$P(\psi_l(k)|\psi_{lC}(k), \Omega_{\psi_l}^{k-1}, Z^{k-1}) = \frac{1}{\delta^k!} \tag{A.4}$$

The product of the expressions in (A.2) - (A.4) yields

$$\begin{aligned} P(\psi_l(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1}) =& P(\psi_{lN}(k)|\Omega_{\psi_l}^{k-1}, Z^{k-1}) P(\psi_{lC}(k)|\psi_{lN}(k), \Omega_{\psi_l}^{k-1}, Z^{k-1}) \times \\ & P(\psi_l(k)|\psi_{lC}(k), \Omega_{\psi_l}^{k-1}, Z^{k-1}) \\ =& \left\{ \frac{\nu^k! f^k!}{M_k!} \mu_f^k(f^k) \mu_\nu^k(\nu^k) \right\} \left\{ (P_\chi^k)^{\chi^k} \right\} \times \\ & \left\{ \left[(1-P_\chi^k)(1-P_d^k) \right]^{\tau^k - \delta^k - \chi^k} \left[(1-P_\chi^k) P_d^k \right]^{\delta^k} \right\} \end{aligned} \tag{A.5}$$

Similarly, based on the hypothesis Ω_0^k that all reports are false, one can conclude that:

$$p(Z(k)|\Omega_0^k, Z^{k-1}) = \prod_{i_k=1}^{M_k} \left\{ \left[p_f^k(z_{i_k}^k|\Omega_0^k, Z^{k-1}) \right]^{f_{i_k}^k} \right\}, \tag{A.6}$$

$$\begin{aligned} P(\psi_0(k)|\Omega_0^{k-1}, Z^{k-1}) =& P(\psi_{0N}(k)|\Omega_0^{k-1}, Z^{k-1}) P(\psi_{0C}(k)|\psi_{0N}(k), \Omega_0^{k-1}, Z^{k-1}) \times \\ & P(\psi_0(k)|\psi_{0C}(k), \Omega_0^{k-1}, Z^{k-1}) \\ =& \mu_f^k(f^k) \mu_\nu^k(\nu^k), \end{aligned} \tag{A.7}$$

where $\psi_0(k)$, $\psi_{0N}(k)$ and $\psi_{0C}(k)$ are defined correspondingly.

The substitution of (A.1), (A.5), (A.6), and (A.7) into (11.27) conditioned on Z^N yields the following expression for $\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)}$:

$$\frac{P(\Omega_l^N|Z^N)}{P(\Omega_0^N|Z^N)} = \left\{ \frac{\nu^N! f^N!}{M_N!} \frac{\mu_f^N(f^N)\mu_\nu^N(\nu^N)}{\mu_f^N(M_N)\mu_\nu^N(0)} \right\}$$
$$\left\{ (P_\chi^N)^{\chi^N} \left[(1-P_\chi^N)(1-P_d^N)(1-P_m^N) \right]^{\tau^N - \delta^N - \chi^N} \right\} \times$$
$$\prod_{i=1}^{M_N} \left\{ \left[(1-P_\chi^N) P_d^N \frac{p_t^N(z_{i_N}^N|\Omega_l^N, Z^{N-1})}{p_f^N(z_{i_N}^N|\Omega_0^N, Z^{N-1})} \right]^{\delta_{i_N}^N} \left[\frac{p_\nu^N(z_{i_N}^N|\Omega_l^N, Z^{N-1})}{p_f^N(z_{i_N}^N|\Omega_0^N, Z^{N-1})} \right]^{\nu_{i_N}^N} \right\}$$
$$\frac{P(\Omega_{\psi_l}^{N-1}|Z^{N-1})}{P(\Omega_0^{N-1}|Z^{N-1})}. \quad \text{(A.8)}$$

Appendix B

KALMAN FILTERING

Kalman filtering technique is a *recursive processing* method, which is optimal in the **Linear Minimum Variance(LMV)** sense. For a linear system, and under the assumptions of white Gaussian noise (both plant noise and measurement noise), the state estimate given by the Kalman Filter is the minimum variance estimator.

B.1 Algorithm

The Kalman filter addresses the general problem of trying to estimate the state $\mathbf{x} \in \mathcal{R}^N$ of a discrete-time controlled process that is governed by a linear stochastic difference equation. For a discrete linear time varying system, the **state equation** can be written in the form

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{w}_{k-1}, \tag{B.1}$$

where $\mathbf{w}_k \in \mathcal{R}^P$ denotes the driving noise input to the system. The $N \times N$ matrix $\mathbf{F}$ denotes the (time-dependent) state transition matrix and the $N \times P$ matrix $\mathbf{G}$ the (time-dependent) noise gain matrix.

Let $\hat{\mathbf{x}}_{j|k}$ denote an estimation about $\mathbf{x}$ at time t_j based on measurements up to time t_k. The following basic assumptions are typically made in order to make computations mathematically feasible.

1. The noise processes $\mathbf{w}$ and $\mathbf{v}$ consist of white noise with zero mean, *i.e.*:

$$\boldsymbol{E}[\mathbf{w}_k] = \mathbf{0}, \quad \boldsymbol{E}\left[\mathbf{w}_k\mathbf{w}_l^T\right] = \mathbf{Q}_k\delta(k-l) \quad \text{and} \quad \boldsymbol{E}[\mathbf{v}_k] = \mathbf{0}, \quad \boldsymbol{E}\left[\mathbf{v}_k\mathbf{v}_l^T\right] = \mathbf{R}_k\delta(k-l).$$

2. $\boldsymbol{w}_k$ and $\boldsymbol{v}_l$ are uncorrelated, *i.e.*: $\boldsymbol{E}[\boldsymbol{w}_k \boldsymbol{v}_l^T] = \mathbf{0}$

3. $\boldsymbol{x}_0$, the initial state is a random variable (or vector), satisfying the following properties:

   $\boldsymbol{E}[\boldsymbol{x}_0] = \mu_x(0)$, where μ denotes the mean.

   $\boldsymbol{E}[(\boldsymbol{x}_0 - \mu_x(0))(\boldsymbol{x}_0 - \mu_x(0))^T] = \boldsymbol{P}_{0|0}$, where $\mathbf{P}_{j|k}$ denotes the **conditional error covariance** matrix of $\mathbf{x}_j$ based on estimate $\mathbf{x}_{j|k}$:

$$\mathbf{P}_{j|k} = \mathbf{E}\left[(\mathbf{x}_j - \hat{\mathbf{x}}_{j|k})(\mathbf{x}_j - \hat{\mathbf{x}}_{j|k})^T\right].$$

   Also, for any k,

   $\boldsymbol{E}[\boldsymbol{x}_0 \boldsymbol{w}_k^T] = \mathbf{0}$ and $\boldsymbol{E}[\boldsymbol{x}_0 \boldsymbol{v}_k^T] = \mathbf{0}$.

The Kalman filtering process involves the following steps which are performed recursively in real-time, while new measurements are observed:

1. As initial conditions, set

$$\hat{\boldsymbol{x}}_{0|0} = \mathbf{E}[\mathbf{x}_0] \quad \text{and} \quad \boldsymbol{P}_{0|0} = diag(\alpha, \ldots, \alpha).$$

   Note that if we have more confidence in the initial estimates of $\hat{\boldsymbol{x}}_{0|0}$, we could set α to be smaller.

2. Predict the new state based on the previous estimate:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_{k-1} \hat{\boldsymbol{x}}_{k-1|k-1}, \tag{B.2}$$

   where $\hat{\boldsymbol{x}}_{k|k-1}$ is the predicted state variable.

3. Computation of *a priori* error covariance matrix (error covariance matrix before updating the estimation)

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1} \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_{k-1}^T + \boldsymbol{G}_{k-1} \boldsymbol{Q}_{k-1} \boldsymbol{G}_{k-1}^T. \tag{B.3}$$

4. Compute the Kalman Gain matrix:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\mathbf{S}_k^{-1}, \tag{B.4}$$

where the matrix

$$\mathbf{S}_k = \left[\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k\right]$$

is called the **innovation covariance matrix.**

5. Update the state estimate $\hat{\mathbf{x}}_k$:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{X}}_{k|k-1} + \boldsymbol{K}_k\left[\mathbf{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1}\right]. \tag{B.5}$$

6. Compute the *a posteriori* error covariance matrix (error covariance matrix after updating the estimation)

$$\boldsymbol{P}_{k|k} = \left[\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right]\boldsymbol{P}_{k|k-1}. \tag{B.6}$$

7. Return to step 2).

B.2 Extended Kalman Filter

For the nonlinear model

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}_{k-1}(\mathbf{x}_{k-1}) + \mathbf{w}_{k-1}\\ \mathbf{z}_k &= \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k.\end{aligned} \tag{B.7}$$

We assume additive zero white noise

$$E[\mathbf{w}_k] = 0,\ E[\mathbf{w}_k\mathbf{w}_k^T] = \mathbf{Q}_k\delta_{kj},\ \ E[\mathbf{v}_k] = 0,\ E[\mathbf{v}_k\mathbf{v}_k^T] = \mathbf{R}_k\delta_{kj}, \tag{B.8}$$

where the covariance $\mathbf{Q}$ is related to the noise gain $\mathbf{G}$ through $\mathbf{Q} = \mathbf{G}\mathbf{C}_w\mathbf{G}^T$. We further assume that the measurement noise $\mathbf{v}_k$ is uncorrelated to the process noise $\mathbf{w}_k$.

The main idea behind the extended Kalman Filter is to linearly approximate the nonlinear model. This is done to first-order (resulting in the first-order extended Kalman Filter) by expanding $\mathbf{f}_k(\mathbf{x}_k)$ and $\mathbf{h}_k(\mathbf{x}_k)$ around the best estimates $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k|k-1}$ respectively:

$$\begin{aligned} \mathbf{f}_k(\mathbf{x}_k) &\approx \mathbf{f}_k(\hat{\mathbf{x}}_k) + \mathbf{F}_k(\hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k) \\ \mathbf{h}_k(\mathbf{x}_k) &\approx \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}_k(\hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}). \end{aligned} \tag{B.9}$$

$\mathbf{F}_k$ and $\mathbf{H}_k$ denote the Jacobian matrices:

$$\mathbf{F}_k(\hat{\mathbf{x}}_k) = \left[\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k)\right], \text{ and } \mathbf{H}_k(\hat{\mathbf{x}}_{k|k-1}) = \left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]. \tag{B.10}$$

The extended Kalman Filter algorithm can be summarized as follows.

1. Initial conditions:

$$\hat{\boldsymbol{x}}_{0|0} = \mathbf{E}[\mathbf{x}_0], \quad \hat{\boldsymbol{x}}_{1|0} = \mathbf{f}_0(\mathbf{x}_0), \quad \boldsymbol{P}_{0|0} = \mathrm{Var}[\mathbf{x}_0].$$

2. Given estimate $\hat{\mathbf{x}}_{k-1|k-1}$, evaluate Jacobian $\mathbf{F}_{k-1}\left(\hat{\mathbf{x}}_{k-1|k-1}\right)$.

3. Predict new state: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_k\left(\hat{\mathbf{x}}_{k-1|k-1}\right)$.

4. Given estimate $\hat{\mathbf{x}}_{k|k-1}$, evaluate Jacobian $\mathbf{H}_{k-1}\left(\hat{\mathbf{x}}_{k|k-1}\right)$.

5. Predict covariance:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{B.11}$$

6. Compute the filter gain:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\mathbf{S}_k^{-1}, \tag{B.12}$$

with

$$\mathbf{S}_k = \left[\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k\right].$$

7. Update the state estimate:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \left[\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}\right], \tag{B.13}$$

with

$$\mathbf{z}_k = \mathbf{h}_k(\hat{\boldsymbol{x}}_k) + \mathbf{v}_k, \text{ and } \hat{\boldsymbol{z}}_{k|k-1} = \mathbf{h}_k\left(\hat{\boldsymbol{x}}_{k|k-1}\right).$$

8. Update covariance:

$$\boldsymbol{P}_{k|k} = \left[\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k\right] \boldsymbol{P}_{k|k-1}. \tag{B.14}$$

9. Return to step 2.

B.3 Modifications to Kalman Filtering Algorithm

Notice that the standard Kalman filter algorithm requires the inversion of the $M \times M$ innovation covariance matrix $\mathbf{S}_k$ at each time step. This matrix inversion is computationally expensive. To maintain the real-time capability and the numerical stability of the Kalman filter, it is important to be able to compute the Kalman gain without directly inverting a matrix at each step.

Notice the innovation covariance matrix $\boldsymbol{S}_k$ is a positive definite matrix, so Cholesky decomposition can be applied on it, and one gets:

$$\boldsymbol{S}_k = \boldsymbol{L}_{sk} \boldsymbol{L}_{sk}^T \tag{B.15}$$

where $\boldsymbol{L}_{sk}$ is a lower triangular matrix, thus,

$$\boldsymbol{S}_k^{-1} = \boldsymbol{L}_{sk}^{-T} \boldsymbol{L}_{sk}^{-1}. \tag{B.16}$$

The state update equation can be rewritten as,

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{L}_{sk}^{-T} \boldsymbol{L}_{sk}^{-1} \tilde{\boldsymbol{z}}_k \tag{B.17}$$

One should compute $\boldsymbol{L}_{sk}^{-1} \tilde{\boldsymbol{z}}_k$ first, and since $\boldsymbol{L}_{sk}$ is a lower triangular matrix, forward elimination method can be used. Then when computing $\boldsymbol{L}_{sk}^{-T} (\boldsymbol{L}_{sk}^{-1} \tilde{\boldsymbol{z}}_k)$, $\boldsymbol{L}_{sk}^T$ is an upper triangular matrix, so backward elimination method can be used.

Covariance update can be treated similarly.

B.4 Square Root Filter

A more numerically stable way is to use a square root filter as shown in [26]. Unlike the standard Kalman filtering algorithm, the square root filter operates on state estimate and the square root of the covariance matrix. The square root of a positive definite matrix $\mathbf{A}$ is a lower triangular matrix denoted by $\mathbf{A}^{\frac{1}{2}}$, and $\mathbf{A} = \mathbf{A}^{\frac{1}{2}}\mathbf{A}^{\frac{1}{2}\mathbf{T}}$.

Denote $\boldsymbol{J}_{k|k} = \boldsymbol{P}_{k|k}^{\frac{1}{2}}$ and $\boldsymbol{J}_{k|k-1} = \boldsymbol{P}_{k|k-1}^{\frac{1}{2}}$. The algorithm is as follows:

1. Initial Conditions:

$$\boldsymbol{J}_{0|0} = \boldsymbol{P}_{0|0}^{\frac{1}{2}} \tag{B.18}$$

This can be done using Cholesky decomposition.

2. Prediction:

$$\hat{\boldsymbol{x}}_{k|k-1} = \mathbf{f}_k\left(\hat{\boldsymbol{x}}_{k-1|k-1}\right) \tag{B.19}$$

$$\boldsymbol{J}_{k|k-1} = \left\{\begin{bmatrix}\boldsymbol{F}_{k-1}\boldsymbol{J}_{k-1|k-1} & \boldsymbol{Q}_{k-1}^{\frac{1}{2}}\end{bmatrix}\begin{bmatrix}\boldsymbol{F}_{k-1}\boldsymbol{J}_{k-1|k-1} & \boldsymbol{Q}_{k-1}^{\frac{1}{2}}\end{bmatrix}^T\right\}^{\frac{1}{2}} \tag{B.20}$$

$\boldsymbol{Q}_{k-1}^{\frac{1}{2}}$ can be computed using Cholesky decomposition on $\boldsymbol{Q}_{k-1}$. The matrix $\left[\boldsymbol{F}_{k-1}\boldsymbol{J}_{k-1|k-1},\ \boldsymbol{Q}_{k-1}^{\frac{1}{2}}\right]$ is of size $N \times 2N$. Instead of multiplying things out and take the Cholesky decomposition, QR factorization is used instead.

$$\begin{bmatrix}\boldsymbol{F}_{k-1}\boldsymbol{J}_{k-1|k-1} & \boldsymbol{Q}_{k-1}^{\frac{1}{2}}\end{bmatrix}^T = \boldsymbol{Q}\boldsymbol{R} \tag{B.21}$$

where $\boldsymbol{Q}$ is a $2N \times 2N$ orthogonal matrix, and $\boldsymbol{R}$ is an upper triangular matrix of size $2N \times N$, and $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{R}_1 \\ \mathbf{0}\end{bmatrix}$, where $\boldsymbol{R}_1$ is of size $N \times N$. So one has

$$\begin{aligned}\boldsymbol{J}_{k|k-1} &= \left\{\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\right\}^{\frac{1}{2}} \\ &= \left\{\begin{bmatrix}\boldsymbol{R}_1^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{R}_1 \\ \mathbf{0}\end{bmatrix}\right\}^{\frac{1}{2}} \\ &= \left\{\boldsymbol{R}_1^T\boldsymbol{R}_1\right\}^{\frac{1}{2}} \\ &= \boldsymbol{R}_1^T\end{aligned} \tag{B.22}$$

3. Measurement Prediction

$$\tilde{\boldsymbol{z}}_k = \boldsymbol{z}_k - \mathbf{h_k}(\hat{\boldsymbol{x}}_{\mathbf{k|k-1}}) \tag{B.23}$$

4. State update

$$\boldsymbol{M}_k = \{\left[\boldsymbol{H}_k\boldsymbol{J}_{k-1|k} \quad \boldsymbol{R}_k^{\frac{1}{2}}\right]\left[\boldsymbol{H}_k\boldsymbol{J}_{k-1|k} \quad \boldsymbol{R}_k^{\frac{1}{2}}\right]^T\}^{\frac{1}{2}} \tag{B.24}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{J}_{k-1|k}\boldsymbol{J}_{k-1|k}^T\boldsymbol{H}_k^T\boldsymbol{M}_k^{-T}\boldsymbol{M}_k^{-1}\tilde{\boldsymbol{z}}_k \tag{B.25}$$

Suppose

$$\left[\boldsymbol{H}_k\boldsymbol{J}_{k-1|k} \quad \boldsymbol{R}_k^{\frac{1}{2}}\right]^T = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}_1\\ \mathbf{0}\end{bmatrix} \tag{B.26}$$

then,

$$\boldsymbol{M}_k = \boldsymbol{R}_1^T \tag{B.27}$$

And one can still use forward or backward elimination method when taking the inverse of $\boldsymbol{M}_k$ and $\boldsymbol{M}_k^T$.

5. Square Root of the Covariance update

$$\boldsymbol{J}_{k|k} = \boldsymbol{J}_{k-1|k}\left[\boldsymbol{I} - \boldsymbol{J}_{k-1|k}^T\boldsymbol{H}_k^T\boldsymbol{M}_k^{-T}(\boldsymbol{M}_k + \boldsymbol{R}_k^{\frac{1}{2}})^{-1}\boldsymbol{H}_k\boldsymbol{J}_{k-1|k}\right] \tag{B.28}$$

B.5 Interactive Multiple Models (IMM)

Any Kalman filter model is usually optimal only for a certain type of movement. Thus, in situations where an object abruptly performs maneuvers and changes its type of movement, the use of one particular type of Kalman filter will not lead to good estimations. Many Kalman filter models for different types of behavior exist. Thus, to solve the filtering problem in scenarios with objects which change behavior, multiple model filtering has been developed. Amongst these, the Interacting Multiple Model (IMM) is now accepted as the best-cost-effective implementation [3], [5], [8].

In IMM filtering, state estimates and covariance matrices from multiple models are combined according to a Markov model for the transition between target maneuver states. We denote with r the total number of target maneuver models. As a result, the IMM algorithm requires r filters to operate in parallel. The state estimate is the Gaussian mixture of the output of these r filters.

Assume the following system model:

$$\mathbf{x}_k = \mathbf{F}^{m_{k-1}}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{w}_k^{m_{k-1}} \tag{B.29}$$

$$\mathbf{z}_k = \mathbf{H}^{m_k}\mathbf{x}_k + \mathbf{v}_k^{m_k}, \tag{B.30}$$

where m_k denotes the model at time k, $m_k \in \{m_1, \cdots, m_r\}$. Let m_k^j denote that model j is in effect at time k.

It is assumed, that the model transitions form a Markov process with *a priori* known **model transition probabilities**

$$p_{ij} = P\left\{m_k^j \mid m_{k-1}^i\right\}.$$

p_{ij} gives the probability that the system will be described by model m_j at time k under the condition that it was described by model m_i at time $k-1$. It is $\sum_{j=1}^{r} p_{ij} = 1, \ \forall i$ and the p_{ij} are state independent.

Next, let μ_k^j denote the conditional probability that m_k^j is correct at time k based on the past observations (**model probability**):

$$\mu_k^j = P\left\{m_k^j \mid \mathbf{z}_k, \mathbf{z}_{k-1}, \cdots, \mathbf{z}_0\right\}.$$

The **mixing probability** μ_{k-1}^{ij} is defined as

$$\mu_{k-1}^{ij} := \mu_{k-1|k-1}^{ij} = P\left\{m_{k-1}^i \mid m_k^j, \mathbf{z}_{k-1}, \cdots, \mathbf{z}_0\right\},$$

and denotes the probability that mode m_i was in effect at time $k-1$ given that m_j is in effect at time k conditioned on the measurement $\mathbf{z}_{k-1}$ and past measurements [3].

Assuming a Markov process for the model transition probabilities, the mixing probabilities can be computed from the model probabilities through [3]:

$$\mu_{k-1}^{ij} = \frac{p_{ij}\mu_{k-1}^{i}}{c_{k-1}^{j}}, \quad c_{k-1}^{j} = \sum_{i=1}^{r} p_{ij}\mu_{k-1}^{i}. \tag{B.31}$$

The c^j play the role of normalizing constants. Given the filter estimates $\hat{\mathbf{x}}_{k-1|k-1}^{i}$, the covariance matrices $\mathbf{P}_{k-1|k-1}^{i}$ for model i and the mixing probabilities μ_{k-1}^{ij} at time $k-1$, the new filtered state estimates and covariance matrices produced by the **mixing process** are as follows:

$$\begin{aligned}
\hat{\mathbf{x}}_{k-1|k-1}^{0j} &= \sum_{i=1}^{r} \mu_{k-1}^{ij}\hat{\mathbf{x}}_{k-1|k-1}^{i} \\
\mathbf{P}_{k-1|k-1}^{0j} &= \sum_{i=1}^{r} \mu_{k-1}^{ij}\left[\mathbf{P}_{k-1|k-1}^{i} + \mathbf{d}_{k-1}^{i}\left(\mathbf{d}_{k-1}^{i}\right)^{T}\right], \quad j = 1, \cdots, r,
\end{aligned} \tag{B.32}$$

where $\mathbf{d}$ denotes the difference between estimates before and after mixing:

$$\mathbf{d}_{k-1}^{i} := \hat{\mathbf{x}}_{k-1|k-1}^{i} - \hat{\mathbf{x}}_{k-1|k-1}^{0j}.$$

$\hat{\mathbf{x}}^{0j}$ and $\mathbf{P}^{0j}$ are the new mixed initial conditions for the filter matched to model j. In the next step, these estimates are used as inputs to the corresponding filters. The filters use the mixed estimates to produce new estimates of their model probability. A Gaussian statistic is assumed. The **likelihood function** Λ_k^j corresponding to model j at time k can then be expressed in the form

$$\Lambda_k^j = \mathcal{N}\left(\mathbf{z}_k\,;\ \hat{\mathbf{z}}_{k|k-1}^{j},\ \mathbf{S}_k^j\right) = \mathcal{N}\left(\mathbf{z}_k\,;\ \mathbf{H}^j\hat{\mathbf{x}}_{k|k-1}^{j},\ \mathbf{H}^j\mathbf{P}_{k|k-1}^{j}\left(\mathbf{H}^j\right)^{T} + \mathbf{R}^j\right). \tag{B.33}$$

This is frequently called **mode-matching** [3].

With the updated likelihood functions, the model probabilities can be updated using Bayes formula. The **model probability update** can be written in the form

$$\mu_k^j = \frac{\Lambda_k^j\, c_{k-1}^j}{c}, \quad j = 1, \cdots, r, \quad \text{with the normalizing constant } c = \sum_{j=1}^{r} \Lambda_k^j c_{k-1}^j. \tag{B.34}$$

Given the updated model probabilities μ_k^j and the updated state estimates $\hat{\mathbf{x}}_{k|k}^j$ and covariances $\mathbf{P}_{k|k}^j$, these model-conditioned estimates can be combined through the mixture equations

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \sum_{j=1}^{r} \mu_k^j \, \hat{\mathbf{x}}_{k|k}^j \\ \mathbf{P}_{k|k} &= \sum_{j=1}^{r} \mu_k^j \left(\mathbf{P}_{k|k}^j + \mathbf{d}_k^j (\mathbf{d}_k^j)^T \right), \end{aligned} \tag{B.35}$$

where now $\mathbf{d}_k^j = \hat{\mathbf{x}}_{k|k}^j - \hat{\mathbf{x}}_{k|k}$. This **combination of estimate and covariance** is not part of the recursion algorithm, it can be performed upon request.

See Fig. B.1 for a graphical representation of the IMM algorithm. In Fig. B.2 the algorithm is described for $r = 2$.

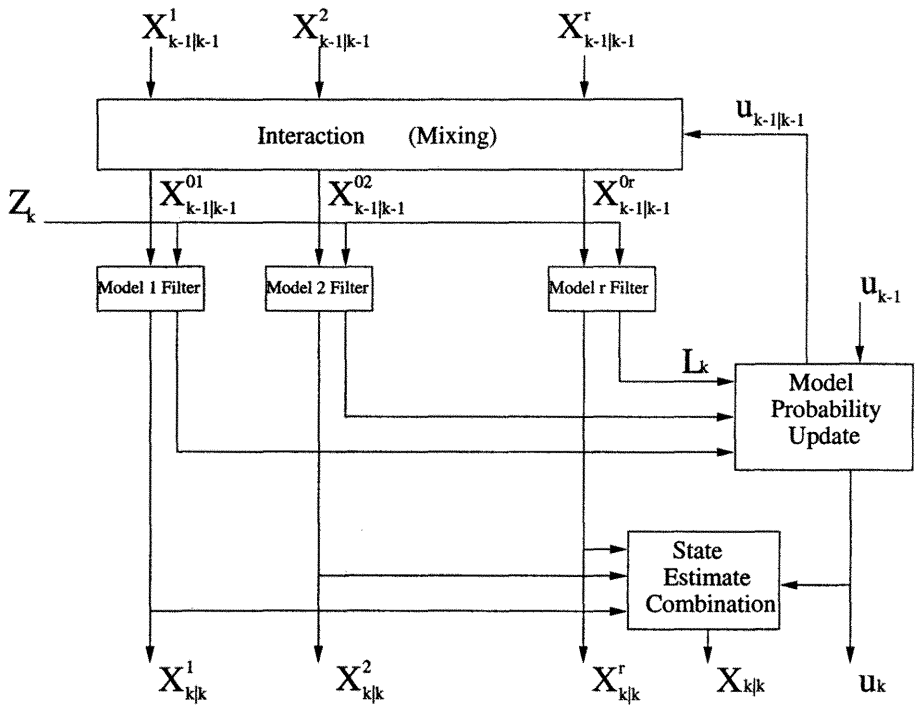


Figure B.1: Structural diagram for r model IMM.

Suppose that we are asked to give the best state estimate at any given time t. Let t_k be the closest time to t, such that $t_k \leq t$ and we have an observation $\mathbf{z}_k$ at time t_k. Then the best state estimate and error covariance can be obtained using

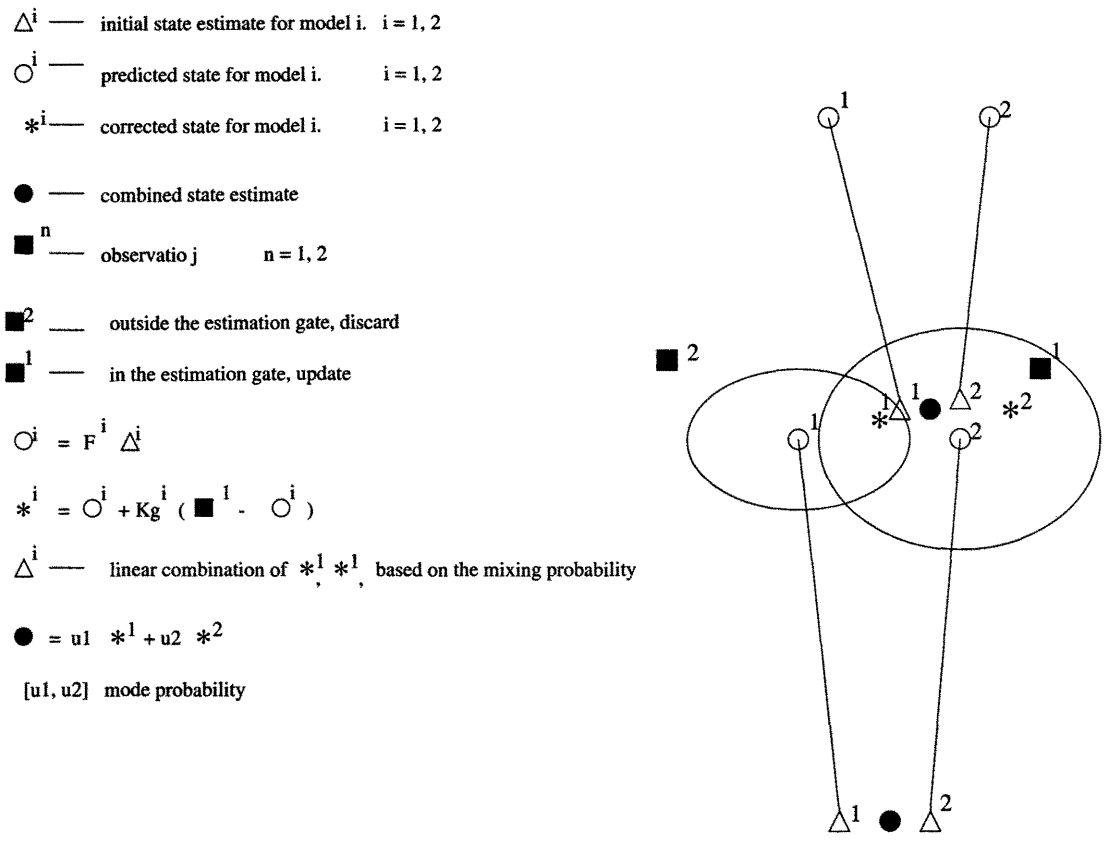


Figure B.2: Illustrations of the IMM2 algorithm.

the following formula:

$$\hat{\mathbf{x}}_{t|k} = \sum_{j=1}^{r} \mu_k^j \, \hat{\mathbf{x}}_{t|k}^j \tag{B.36}$$

$$\mathbf{P}_{t|k} = \sum_{j=1}^{r} \mu_k^j \left(\mathbf{P}_{t|k}^j + \mathbf{d}_k^j (\mathbf{d}_k^j)^T \right), \tag{B.37}$$

where now $\mathbf{d}_k^j = \hat{\mathbf{x}}_{t|k}^j - \hat{\mathbf{x}}_{t|k}$ and $\hat{\boldsymbol{x}}_{t|k}^j$ is the predicted state estimate at time t using model j and μ_k^j is the mode probability for model j given all the observations up to time t_k.

The characteristics of the IMM algorithm can be summarized as follows:

- Decision-making free

  No maneuvering detection decision is needed in IMM algorithms. The output estimate is a combination of r mode-matched filters weighted by the

mode-probability. In other words, the maneuvering detection is automatically represented in the change of mode-probability.

- Soft-switching

  The IMM algorithms undergo a soft switching of models according to the latest updated mode probabilities.

- Estimation error reduction

  The IMM can give good estimates even at the critical maneuver periods (onset and termination). In the case that the target doesn't maneuver (it follows exactly one model), the IMM can still give approximately the same results as using only one model matched Kalman filter since the unmatched models contribute very little to both the output estimates and the combined initial estimates.

The basic IMM algorithm is summarized as follows:

1. Compute the mixing probabilities

   Given the model probabilities μ^j and the model transition probabilities p_{ij}, compute the model mixing probabilities μ^{ij} from (B.31).

2. Mix the initial estimates

   Given the filter estimates $\hat{\mathbf{x}}^j$, the covariance matrices $\mathbf{P}^j$ and the mixing probabilities μ^{ij}, compute the mixed state estimates $\hat{\mathbf{x}}^{0j}$ and $\mathbf{P}^{0j}$ through (B.32).

3. Update the model probabilities

   (a) Predict.

(b) Given the mixed initial estimates $\hat{\mathbf{x}}^{0j}$ and $\mathbf{P}^{0j}$ predict the respective model estimates $\hat{\mathbf{x}}^{j}$ and $\mathbf{P}^{j}$ using the normal Kalman filter equations.

(c) Given the predicted estimates $\hat{\mathbf{x}}^{j}$ and $\mathbf{P}^{j}$ update the model likelihood functions Λ^{j} through (B.33).

(d) Using the updated likelihood function, update the model probabilities μ_j according to (B.34).

4. Update model state estimates

   Using the normal Kalman filter equations, update the respective model estimates.

   Return to 1) if no output is required.

5. Estimate and covariance combination

   Given the updated model probabilities μ^{j}, the updated state estimates $\hat{\mathbf{x}}^{j}$ and covariances $\mathbf{P}^{j}$, combine these estimates according to (B.35).

B.6 Square Root IMM

Instead of operating on state estimates and their covariance matrices, the IMM algorithm can be used to operate on state estimates and the square root of the covariance matrices.

The square root filtering algorithm can be used for updating the single filters. However, the IMM algorithm need to mix the initial state and its covariance according to the mixing probabilities. The initial state can be mixed as explained in the previous section, it will be explained here how to mix the square root of the covariance matrix. Since we have

$$\mathbf{P}^{0j}_{k-1|k-1} = \sum_{i=1}^{r} \mu^{ij}_{k-1} \left[\mathbf{P}^{i}_{k-1|k-1} + \mathbf{d}^{i}_{k-1} \left(\mathbf{d}^{i}_{k-1} \right)^{T} \right] , \quad j = 1, \cdots, r, \tag{B.38}$$

then

$$\begin{aligned}\boldsymbol{J}^{0j}_{k-1|k-1} =& \{\sum_{i=1}^{r} \mu^{ij}_{k-1} \left[\boldsymbol{P}^{i}_{k-1|k-1} + \boldsymbol{d}^{i}_{k-1} \left(\boldsymbol{d}^{i}_{k-1}\right)^{T}\right]\}^{\frac{1}{2}} \\ =& \{\left[\mu^{1j}_{k-1}\boldsymbol{J}^{1}_{k-1|k-1} \quad \mu^{1j}_{k-1}\boldsymbol{d}^{1}_{k-1} \quad \cdots \quad \mu^{rj}_{k-1}\boldsymbol{J}^{r}_{k-1|k-1} \quad \mu^{rj}_{k-1}\boldsymbol{d}^{r}_{k-1}\right] \\ & \left[\mu^{1j}_{k-1}\boldsymbol{J}^{1}_{k-1|k-1} \quad \mu^{1j}_{k-1}\boldsymbol{d}^{1}_{k-1} \quad \cdots \quad \mu^{rj}_{k-1}\boldsymbol{J}^{r}_{k-1|k-1} \quad \mu^{rj}_{k-1}\boldsymbol{d}^{r}_{k-1}\right]^{T}\}^{\frac{1}{2}} \\ =& \boldsymbol{R}_1\end{aligned} \tag{B.39}$$

where $\boldsymbol{R}_1$ is defined by

$$\left[\mu^{1j}_{k-1}\boldsymbol{J}^{1}_{k-1|k-1} \quad \mu^{1j}_{k-1}\boldsymbol{d}^{1}_{k-1} \quad \cdots \quad \mu^{rj}_{k-1}\boldsymbol{J}^{r}_{k-1|k-1} \quad \mu^{rj}_{k-1}\boldsymbol{d}^{r}_{k-1}\right]^{T} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix} \tag{B.40}$$

Appendix C

EVIDENTIAL REASONING

C.1 Introduction

Given a frame of discernment Θ made up of elements (or atoms) x_i, the evidential reasoning scheme can be described as a method for combining two basic mass assignments $m_{\mathcal{A}}(A_i)$ and $m_{\mathcal{B}}(B_j)$ over the frame of discernment Θ to produce a combined basic mass assignment $m_{\mathcal{C}}(C) = m(C|\mathcal{A},\mathcal{B})$ according to a formula known as *Dempster's rule of combination*:

$$m_{\mathcal{C}}(C) = m(C|\mathcal{A},\mathcal{B}) = \frac{1}{1-\kappa} \sum_{i,j|A_i \cap B_j = C} m_{\mathcal{A}}(A_i)\, m_{\mathcal{B}}(B_j) \tag{C.1}$$

where κ is a measure of inconsistency between the basic mass assignments $m_{\mathcal{A}}(A_i)$ and $m_{\mathcal{B}}(B_j)$, it is defined by

$$\kappa = \sum_{i,j|A_i \cap B_j = \emptyset} m_{\mathcal{A}}(A_i)\, m_{\mathcal{B}}(B_j) \tag{C.2}$$

The $A_i \in \Theta$ are called the focal elements of the basic mass assignment $m_{\mathcal{A}}(\cdot)$ such that $m_{\mathcal{A}}(A_i) \neq 0$. Mass assigned to a focal element A_i should be thought of as free to move among the atoms $x_i \in A_i$. The same comments hold for $m_{\mathcal{B}}(\cdot)$ and $m_{\mathcal{C}}(\cdot)$.

In addition to basic mass assignments, evidential reasoning introduces the concepts of support $Spt(X)$ and plausibility $Pls(X)$ of any proposition $X \in \Theta$.

$$\begin{aligned} Spt(X|\mathcal{A},\mathcal{B}) &= \frac{1}{1-\kappa} \sum_{i,j|A_i \cap B_j \subseteq X} m_{\mathcal{A}}(A_i)\, m_{\mathcal{B}}(B_j) \\ &= \sum_{C|C \in \mathcal{C},\, C \in X} m(C|\mathcal{A},\mathcal{B}) \end{aligned} \tag{C.3}$$

and

$$Pls(X|\mathcal{A},\mathcal{B}) = \frac{1}{1-\kappa} \sum_{i,j|A_i \cap B_j \cap X \neq \emptyset} m_{\mathcal{A}}(A_i)\, m_{\mathcal{B}}(B_j)$$
$$= \sum_{C|C \in \mathcal{C},\, C \cap X \neq \emptyset} m(C|\mathcal{A},\mathcal{B})$$
$$= 1 - Spt(\bar{X}|\mathcal{A},\mathcal{B}) \tag{C.4}$$

The support of proposition X is the evidence directly assigned to X or any subset of X (i.e., any proposition which implies X). The plausibility of X is the sum of all the mass assigned to propositions which have a non-null intersection with X (i.e., any proposition which does not contradict X).

C.2 Partial Probability Models

A partial probability model ($\mathcal{P}$-probability) allows us to denote that the exact distribution (as required by conventional complete probability model) of the probability among the individual singletons is unknown. Assume that all that is know is the partial distribution ($\mathcal{P}$-distribution), denoted $\mathcal{P}(a_j)$ of the probability mass among J disjunctions $a_j \subseteq \Theta$. The only constraints placed on $\mathcal{P}(a_j)$ is that $\mathcal{P}(a_j) \geq 0, \quad \forall a_j \quad$ and

$$1 = \sum_{j=1}^{J} \mathcal{P}(a_j). \tag{C.5}$$

Notice that a_j may not be disjoint. $\mathcal{P}(a_j)$ is the mass assigned to the entire set a_j, it does not specify how to assign the mass among the elements $x_i \in A_j$. Thus, there are many complete (i.e. conventional) probability distributions that would be consistent with the partial information of (C.5). One can express all possible consistent complete probability models by a single *parametric model.* The model is parameterized on a matrix of parameter α_{ij}. We denote this matrix $[\alpha_{ij}]$

$$P(x_i|\,[\alpha_{ij}]) = \sum_{j=1}^{J} \mathcal{P}(a_j)\alpha_{ij} \tag{C.6}$$

where $\alpha_{ij} \geq 0$ and

$$\sum_i \alpha_{ij} = 1 \quad \forall j \tag{C.7}$$

and

$$\alpha_{ij} = \begin{cases} 0, & \text{if } x_i \notin a_j \\ \alpha_{ij}, & \text{else the fraction of mass } \mathcal{P}(a_j) \text{ is to be assigned to } x_i \end{cases} \tag{C.8}$$

C.3 $\mathcal{P}$-Probability Models and Evidential Reasoning

In the case of a complete probability model, we can form a set $\hat{X}$ by collecting the x_i for which $P(x_i|z)$ is largest such that

$$P_c = \sum_{i|x_i \in \hat{X}} P(x_i|z) \tag{C.9}$$

where P_c is the desired rate of being correct in the sense that $x_t \in \hat{X}$ for P_c of the trials.

In the case of $\mathcal{P}$-probability models, the experiment can be defined as follows. As a result of the information z, a set $\mathcal{A}$ of subsets A_j of Θ and their rates of occurrence $m_{\mathcal{A}}(A_j)$ are specified.

Unlike complete probability models, it is not possible to calculate the probability, P_c, that any set estimate $\hat{X}$ is correct in the sense that it will contain x_t in P_c fraction of the trials. It is possible, however, to calculate the maximum and the minimum value that P_c could have, as the $\mathcal{P}$-probability law is varied over all the complete set of probability models as described in (C.6).

If one assigns

$$m_{\mathcal{A}}(A_j) = \mathcal{P}(A_j) \tag{C.10}$$

then, to describe a $\mathcal{P}$-probability using evidential reasoning terminology, one has

$$\begin{aligned} \min_{[\alpha_{ij}]} P_c &= \min_{[\alpha_{ij}]} \sum_{i|x_i \in \hat{X}} P(x_i|[\alpha_{ij}]) \\ &= \sum_{j|A_j \subseteq \hat{X}} m_{\mathcal{A}}(A_j) \\ &= Spt(\hat{X}|\mathcal{A}) \end{aligned} \tag{C.11}$$

Similarly,

$$\begin{aligned}
\max_{[\alpha_{ij}]} P_c &= \max_{[\alpha_{ij}]} \sum_{i|x_i \in \hat{X}} P(x_i|[\alpha_{ij}]) \\
&= \sum_{j|A_j \cap \hat{X} \neq \emptyset} m_{\mathcal{A}}(A_j) \\
&= Pls(\hat{X}|\mathcal{A}) \qquad \text{(C.12)}
\end{aligned}$$