



**GAUSSIAN MIXTURE REDUCTION
FOR TRACKING MULTIPLE MANEUVERING TARGETS
IN CLUTTER**

THESIS

Jason L. Williams, Flight Lieutenant, RAAF

AFIT/GE/ENG/03-19

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, United States Department of Defense, United States Government, Royal Australian Air Force, Australian Department of Defence, Australian Commonwealth Government, or the corresponding agencies of any other government, or any other defense organization.

AFIT/GE/ENG/03-19

GAUSSIAN MIXTURE REDUCTION
FOR TRACKING MULTIPLE MANEUVERING TARGETS
IN CLUTTER

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Jason L. Williams, BE(Electronics)(Hons) BInfTech *QUT*
Flight Lieutenant, RAAF

March, 2003

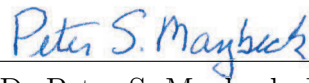
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

GAUSSIAN MIXTURE REDUCTION
FOR TRACKING MULTIPLE MANEUVERING TARGETS
IN CLUTTER

Jason L. Williams, BE(Electronics)(Hons) BInfTech *QUT*

Flight Lieutenant, RAAF

Approved:



Dr Peter S. Maybeck, Ph.D
Thesis Advisor

5 mar 03

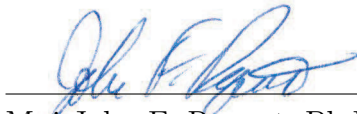
Date



Maj Roger L. Claypoole, Jr, Ph.D
Committee Member

5 MAR 03

Date



Maj John F. Raquet, Ph.D
Committee Member

5 MAR 03

Date

Acknowledgments

Faith is to believe what you do not yet see; the reward for this faith is to see what you believe.

St. Augustine

The past 18 months have presented a truly unique opportunity to study challenging problems in a world-class institution. I am greatly indebted to the Royal Australian Air Force and the United States Air Force for making this wonderful opportunity possible.

I offer the sincerest of thanks to Professor Peter Maybeck, my thesis advisor, teacher and mentor. Your input, guidance, encouragement and support over the past 18 months have been nothing short of astounding. To my committee members, Major Roger Claypoole and Major John Raquet, many thanks for your valuable input and encouragement. To my sponsor, Mr Stan Musick, AFRL/SNAT, thanks for your perspective.

To my friends in the Control and Navigation sequence, Craig, Tina, Terry, Alec, Jae and Aydin, thanks for helping to keep me sane through a gruelling academic load. We've made it! To Flight Lieutenant Ngoya Pepela, whom I now count more as a brother than a friend, thanks for everything.

To my family, thanks for your continual support over this time. To my wife, you have walked every step of this journey by my side. This achievement is as much yours as it is mine. Thanks for your patience, encouragement, support and love.

To the Lord God, thanks for creating me, saving me, leading me and sustaining me. Thanks for answering my every prayer. I now look back on the moments of despondency when I felt far from your presence, and see that You were only leading me to better things. May I trust in You always.

Jason L. Williams

Table of Contents

	Page
Acknowledgments	iii
List of Figures	viii
List of Tables	xiv
List of Abbreviations	xv
Notation	xvi
Abstract	xix
I. Introduction	1-1
1.1 Motivation	1-2
1.2 Research Goal	1-6
1.3 Assumptions	1-6
1.4 Thesis Organization	1-7
II. Background	2-1
2.1 Introduction	2-1
2.2 Tracking Filters	2-1
2.2.1 Introduction	2-1
2.2.2 Ad Hoc Techniques	2-1
2.2.3 Kalman Filter	2-3
2.2.4 Nonlinear Filters	2-10
2.3 Gaussian Mixtures	2-12
2.4 Multiple Model Adaptive Estimation	2-15

	Page
2.4.1 Non-Switching Models	2-17
2.4.2 Switching Models	2-22
2.4.3 First-Order Generalized Pseudo-Bayesian Estimator	2-24
2.4.4 Second-Order Generalized Pseudo-Bayesian Estimator	2-27
2.4.5 Interacting Multiple Model Estimator	2-30
2.4.6 Summary	2-33
2.5 Data Association	2-35
2.5.1 Measurement Gating	2-37
2.5.2 Association Event Probability	2-38
2.5.3 Forming Joint Hypotheses	2-45
2.5.4 Joint Target State	2-47
2.5.5 State Update	2-51
2.5.6 Global Nearest Neighbor	2-53
2.5.7 Probabilistic Data Association	2-55
2.5.8 Correlation Between Targets	2-57
2.5.9 Maximum Likelihood Methods	2-61
2.5.10 Multiple Hypothesis Tracking	2-61
2.5.11 Controlling the Number of Hypotheses	2-63
2.5.12 Multidimensional Techniques	2-68
2.5.13 Interacting Multiple Model–Multiple Hypothesis Tracker	2-69
2.5.14 Summary	2-72
2.6 Optimization Methods	2-72
III. Analysis	3-1
3.1 Introduction	3-1

	Page
3.2 PDA Bias and Coalescence	3-1
3.3 Gaussian Mixture Reduction	3-10
3.3.1 Cost Measures	3-13
3.3.2 Analysis of Integral Square Difference Measure	3-19
3.3.3 Iterative Optimization	3-24
3.3.4 Initialization Algorithm	3-33
3.4 Summary	3-42
IV. Simulation Results	4-1
4.1 Introduction	4-1
4.2 Initialization Algorithm	4-1
4.3 Iterative Optimization	4-7
4.4 Single Target in Clutter	4-10
4.4.1 Comparison with Pruning Algorithm	4-16
4.4.2 Comparison with Salmond's Joining and Clus-	
tering Algorithms	4-20
4.4.3 Comparison with Lainiotis Algorithm	4-36
4.4.4 Comparison with Iterative Optimization Algo-	
rithm	4-39
4.4.5 Comparison with PDA Algorithm	4-42
4.5 Multiple Targets in Clutter	4-47
4.6 Single Maneuvering Target	4-47
4.7 Summary	4-53
V. Conclusions and Recommendations	5-1
5.1 Restatement of Research Goal	5-1
5.2 Summary of Results	5-1
5.2.1 Single Target Tracking Performance	5-1
5.2.2 Multiple Target Tracking Performance	5-2

	Page
5.2.3 Maneuvering Target Tracking Performance . .	5-2
5.3 Significant Contributions of Research	5-3
5.4 Recommendations for Future Investigations	5-5
Appendix A. Derivations	A-1
A.1 Product of Two Gaussians of Same Dimension	A-1
A.2 Modified Gating Algorithm	A-5
A.3 Switching Bayesian Transition Probability	A-7
Appendix B. Matrix Reference Manual	B-1
Appendix C. Source Code	C-1
C.1 ISDInit.c	C-2
Bibliography	BIB-1
Vita	VITA-1

List of Figures

Figure		Page
1.1	The difficulty of data association: the origin of each measurement is not known, hence the system does not know which measurement belongs to which target, and which measurements are false alarms (due to radar clutter).	1-2
1.2	An example of a Gaussian mixture: the individual weighted Gaussian component PDFs are shown using dashed lines; the overall PDF (the sum of the components) is shown using a solid line.	1-3
1.3	Approximating a Gaussian mixture using fewer mixture components. The original mixture is shown in the top left figure, alongside approximations using four components (top right), three components (bottom left) and two components (bottom right).	1-4
2.1	Block diagram of non-switching multiple model estimation algorithm.	2-20
2.2	MMAE probability flow with and without a lower probability bound. Note the logarithmic scale used in each of the plots. .	2-21
2.3	Block diagram of full order Markov switching estimator. . .	2-25
2.4	Block diagram of GPB-1 algorithm.	2-27
2.5	Block diagram of GPB-2 algorithm.	2-29
2.6	Block diagram of IMM algorithm.	2-34
2.7	The data association problem: how to update target state given a series of unlabelled measurements.	2-36
2.8	Measurement gating in a multiple target environment.	2-37
2.9	Pseudocode to form all joint association events for two targets.	2-46
2.10	Pseudocode to form all joint association events for an arbitrary number of targets.	2-48

Figure		Page
2.11	One-dimensional multiple target data association example. . .	2-57
2.12	Correlation arising due to combining of hypotheses.	2-59
2.13	The impact of forcing independence between targets in a multiple hypothesis system: resultant joint target PDF contains a hypothesis for each pairing of hypotheses from each target, rather than only the actual joint hypotheses as shown in Figure 2.12(a).	2-69
2.14	Gradient of the cost function indicating the direction of the minimum.	2-73
2.15	Operation of the Newton-Raphson algorithm: each step moves to be minimum of the local approximating parabola.	2-76
3.1	Pairs of equally valid tracking solutions in joint target state space.	3-7
3.2	Coalesced joint target state estimate and covariance using JPDA algorithm.	3-8
3.3	Joint target state PDF, (a) disallowing correlation between targets, and (b) allowing correlation between targets (correlation coefficient = -0.9).	3-10
3.4	Joint target state snapshot from Monte Carlo simulation. . .	3-11
3.5	Comparison of various even nonlinearities.	3-16
3.6	Block diagram of proposed Gaussian mixture reduction initialization algorithm.	3-36
3.7	Elements of ISD cost function. Each square represents a multivariate Gaussian evaluation to measure the similarity of the respective components of the two mixtures. Shaded squares represent the components that need to be re-evaluated if the second component in the reduced mixture is modified.	3-38
3.8	Execution times for various implementations for evaluating the match between all pairings of 500 randomly generated four-dimensional Gaussian multivariate PDFs.	3-42

Figure		Page
3.9	Execution times for various implementations of cost function Gaussian mixture reduction initialization algorithm to simplify 60-component four-dimensional Gaussian mixture to 10 components.	3-43
4.1	Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using the ISD initialization algorithm.	4-3
4.2	Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using Salmond's joining algorithm.	4-5
4.3	Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using Salmond's clustering algorithm.	4-6
4.4	Iterative optimization of a 3-component approximation (shown in solid line) to a 5-component Gaussian mixture (shown in dashed line). The top left figure shows the starting point for the optimization, calculated using the ISD initialization algorithm. Remaining figures show the refined solution after 1, 2, and 12 gradient iterations.	4-8
4.5	Cost function trajectory and step size adjustment. The top figure shows the cost reduction as the PDF approximation is optimized iteratively, while the bottom figure shows the gradient step size adaptation.	4-9
4.6	Iterative optimization of a 3-component approximation (shown in solid line) to a 5-component Gaussian mixture (shown in dashed line). The top left figure shows the starting point for the optimization, calculated using the ISD initialization algorithm. The remaining figures show the refined solution after 4, 9, and 29 gradient iterations.	4-10
4.7	Cost function trajectory and step size adjustment. The top figure shows the cost reduction as the PDF approximation is optimized iteratively, while the bottom figure shows the gradient step size adaptation.	4-11

Figure		Page
4.8	Average track life for various merging and pruning algorithms.	4-15
4.9	Performance of ISD initialization algorithm compared to the standard MHT pruning algorithm.	4-16
4.10	Performance of 25-component ISD initialization algorithm compared to 25-component pruning algorithm.	4-18
4.11	Performance of 25-component ISD initialization algorithm compared to 100-component pruning algorithm.	4-19
4.12	Performance of ISD initialization algorithm compared to Salmond joining algorithm.	4-22
4.13	Performance of 5-component ISD initialization algorithm compared to 5-component Salmond joining algorithm.	4-23
4.14	Performance of 30-component ISD initialization algorithm compared to 30-component Salmond joining algorithm.	4-24
4.15	Performance of 35-component ISD initialization algorithm compared to 35-component Salmond joining algorithm.	4-25
4.16	Performance of ISD initialization algorithm compared to Salmond clustering algorithm.	4-27
4.17	Performance of ISD initialization algorithm compared to Salmond clustering and joining algorithms with the maximum number of hypotheses spawned by any parent hypothesis limited to 50.	4-28
4.18	Average track life for scenario using extended clutter population region.	4-29
4.19	Histogram of track life for ISD initialization and Salmond's joining and clustering algorithms, utilizing 25, 30 and 35 mixture components. Plots labelled "ECPR" describe the Monte Carlo simulations utilizing the extended clutter population region; the remaining plots describe the original scenario.	4-31
4.20	Performance of ISD initialization algorithm compared to Salmond joining and clustering algorithms, with clutter population region increased in size by ten times in both x and y axis directions.	4-32

Figure		Page
4.21	Performance of 25-component ISD initialization algorithm compared to 25-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.	4-33
4.22	Performance of 30-component ISD initialization algorithm compared to 30-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.	4-34
4.23	Performance of 35-component ISD initialization algorithm compared to 35-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.	4-35
4.24	Performance of ISD initialization algorithm compared to modified Lainiotis algorithm.	4-38
4.25	Performance of ISD initialization algorithm compared to same algorithm utilizing iterative optimization to refine the approximation.	4-40
4.26	Comparison of track life for simulations of 10-component ISD initialization algorithm, and the same algorithm utilizing iterative optimization to refine the approximation.	4-41
4.27	Performance of PDA compared to other algorithms using a single Gaussian component.	4-43
4.28	Performance of PDA compared to other algorithms using a single Gaussian component.	4-45
4.29	Histograms of track life for PDA algorithm and single-component ISD initialization algorithm.	4-46
4.30	RMS position and velocity error of system utilizing Bayesian switching model approximation. Filter-predicted RMS error shown in dashed line.	4-51
4.31	RMS position and velocity error of IMM system. Filter-predicted RMS error shown in dashed line.	4-52

Figure		Page
A.1	Measurement gating: the gating equation describes an ellipse as shown in (a); the smallest circle enclosing the ellipse is shown in (b); the square aligned with coordinate axes enclosing the circle is shown in (c).	A-8

List of Tables

Table		Page
4.1	Parameters of the one-dimensional Gaussian mixture used to test the initialization algorithm.	4-1
4.2	Reduction steps for Gaussian mixture example.	4-2
4.3	Number of Monte Carlo simulations run for each algorithm and number of mixture components.	4-14
4.4	Parameters for maneuvering target scenario.	4-49

List of Abbreviations

Abbreviation	Page
PDF Probability Density Function	2-4
EKF Extended Kalman Filter	2-10
MMAE Multiple Model Adaptive Estimator	2-17
GPB-1 First-Order Generalized Pseudo-Bayesian	2-24
GPB-2 Second-Order Generalized Pseudo-Bayesian	2-27
IMM Interacting Multiple Model	2-30
GNN Global Nearest Neighbor	2-54
PDA Probabilistic Data Association	2-55
JPDA Joint Probabilistic Data Association	2-55
JPDAC Joint Probabilistic Data Association Coupled	2-60
CPDA Coupled Probabilistic Data Association	2-60
MHT Multiple Hypothesis Tracker	2-61
SB-MHT Structured Branching Multiple Hypothesis Tracker	2-62
PMHT Probabilistic Multiple Hypothesis Tracker	2-69
ISD Integral Square Difference	3-15
EM Expectation Maximization	3-18
RMS Root-Mean-Square	4-50
SPRT Sequential Probability Ratio Test	5-6

Notation

Notation	Usage
$\mathbf{z}, \mathbf{Z}$, etc.	vectors are shown in boldface italic text
$\mathbf{P}, \mathbf{H}$, etc.	matrices are shown in boldface roman text
$\hat{\mathbf{x}}$, etc.	estimates are indicated using the ‘hat’ augmentation
$\mathbf{x}(t)$	a continuous-time signal, where the indexing t is a continuous variable representing the time in seconds
$\mathbf{x}(k)$	a discrete-time signal, where the indexing k is the sample number, and the k -th sample is taken at time t_k
$\hat{\mathbf{x}}(k k-1)$	the estimate of the signal at sample k , using information only up to the $(k-1)$ -th measurement
$\hat{\mathbf{z}}(k k-1)$	the predicted value of the measurement at sample k , using information only up to the $(k-1)$ -th measurement
$\mathbf{Z}^k$	the entire measurement history from sample 1 to sample k
$\mathbf{Z}_k$	<i>all</i> measurements provided to the system in the k -th set of measurements (i.e., the k -th scan)
$\mathbf{z}_j(k)$	the j -th measurement from the k -th set of measurement (i.e., the k -th scan)
$P\{\cdot\}$	the probability of the discrete event specified in $\{\cdot\}$
$f\{\cdot\}$	the probability density function of the continuous parameter specified in the argument $\{\cdot\}$
$\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}\}$	denotes a Gaussian probability density function for variable $\mathbf{x}$, distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{P}$: $\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}\} = 2\pi\mathbf{P} ^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{P}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \right\}$
$E\{\cdot\}$	the expectation operation, finding the expected value of the argument: $E\{\cdot\} = \int \{\cdot\} f\{\cdot\} d\{\cdot\}$

Notation	Usage
N_f	the number of filters (or models) in the system
M_j	the event in which model j is in force in a non-switching multiple model system (no time argument is supplied, as this is the non-switching case, in which the model in force does not change with time)
$M_{k,j}$	the event in which model j is in force at sample k in a switching multiple model system
$M^{k,l}$	the l -th model history event in a switching multiple model system — consists of a single time step event (e.g., $M_{k,j}$) for each sample time from 1 to k
$\hat{\mathbf{x}}_j, \mathbf{P}_j$	the state estimate and covariance of the j -th filter in a multiple model system, or of the j -th target
$\hat{\mathbf{x}}^j, \mathbf{P}^j$	the modified state of the j -th model after mixing; provided as the input to the j -th filter in the IMM algorithm
$\hat{\mathbf{X}}, \mathbb{P}$	the <i>joint</i> state estimate of <i>all</i> targets, and the covariance of the joint state estimate
$N_m(k)$	the number of measurements in the k -th set (i.e., the k -th scan)
N_t	the number of targets under track
$\theta_{ji}(k)$	a single measurement association event, indicating the association of measurement j with target i at sample k
$\Theta_l(k)$	the l -th joint association event for measurement set k , containing a single measurement event for each of the $N_m(k)$ measurements received in the k -th scan
$\Psi_u(k)$	the u -th association <i>history</i> event, which contains a joint association event for each scan from 1 to k

Notation	Usage
$N_h(k)$	the number of association hypotheses in the tracking system after incorporation of the k -th set of measurements
$\mathbf{\Omega}_{N_h}(k)$	the full parameters (weights, means, covariances) of the N_h association hypotheses after incorporation of the k -th set of measurements
$N_r(k)$	the number of association hypotheses at the end of the k -th processing cycle, after hypothesis reduction has been applied
$\bar{\mathbf{\Omega}}_{N_r}(k)$	the parameters of the reduced set of N_r hypotheses

Abstract

The problem of tracking multiple maneuvering targets in clutter naturally leads to a Gaussian mixture representation of the Probability Density Function (PDF) of the target state vector. State-of-the-art Multiple Hypothesis Tracking (MHT) techniques maintain the mean, covariance and probability weight corresponding to each hypothesis, yet they rely on *ad hoc* merging and pruning rules to control the growth of hypotheses. This thesis investigates the performance benefit achievable by applying a structured cost function-based approach to the hypothesis control problem.

A new cost function, the Integral Square Difference (ISD) cost, is proposed for measuring the difference between the full target state PDF and a reduced-order approximation. The ISD cost function is physically meaningful, and, unlike any previously proposed cost function, it is also mathematically tractable, requiring neither numerical integration nor approximation for evaluation. A reduction algorithm is proposed which selects components for merging or pruning to minimize the increase in the ISD cost. This solution is used directly, and also as the starting point for an iterative gradient-based optimization.

The performance of the ISD-based algorithm for tracking a single target in heavy clutter is compared to that of Salmond's joining filter, which previously had provided the highest performance in the scenario examined. For a large number of mixture components, it is shown that the ISD algorithm outperforms the joining filter remarkably, yielding an average track life more than double that achievable using the joining filter. The results indicate that the tracking performance of the ISD-based filter in heavy clutter is significantly higher than achievable using any previously published algorithm.

GAUSSIAN MIXTURE REDUCTION FOR TRACKING MULTIPLE MANEUVERING TARGETS IN CLUTTER

I. Introduction

In their early inception, radar systems were able to track a single target in a clutter-free environment. The limited surveillance capability essentially presented the operator with a raw display of measurements, leaving it up to the human to interpret the display and infer information such as velocity. Early tracking radars illuminated the target continually to ensure that knowledge of the target position did not deteriorate, causing loss of track. This prevented the radar from performing any other tasks at the same time, such as tracking multiple targets or maintaining surveillance capability during track.

The vast body of theory of stochastic estimation developed since the 1960s has enabled revolutionary changes to the design and capability of radar systems. Modern radar systems, such as those utilizing Electronically Scanned Array (ESA) antenna technology [5, 7, 51], can perform multiple functions at once, simultaneously providing high quality tracking estimates for some targets while maintaining wide-area surveillance of the entire operational theater. Virtually every modern surveillance radar operates in Track While Scan (TWS) mode, in which the radar continually searches for existence of new targets, but once detected, targets are also tracked using data filtering techniques [5:3].

The essential function of a modern radar system is to maintain as much knowledge of the target state¹ as possible. The success of modern tracking techniques is

¹i.e., position, velocity, acceleration, etc.

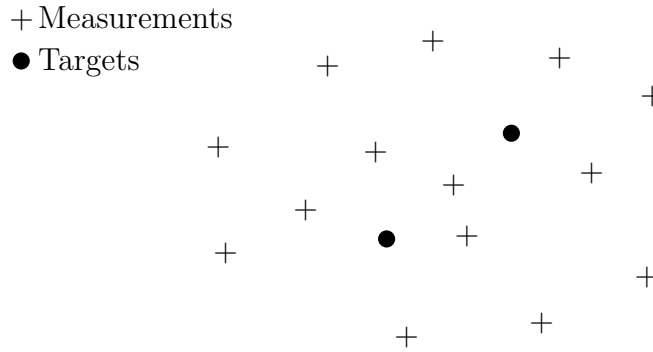


Figure 1.1. The difficulty of data association: the origin of each measurement is not known, hence the system does not know which measurement belongs to which target, and which measurements are false alarms (due to radar clutter).

largely determined by their ability to compute and store the Probability Density Function (PDF) of target state in an efficient manner. This study cuts to the core of this problem: how can the PDF of the target state vector be reduced in complexity such that the system remains computationally tractable, while causing the smallest possible change in the overall structure of the PDF.

1.1 Motivation

In order to maintain knowledge of a target's kinematic state, a radar system must be able to update its target state model using the radar detections produced during each scan interval. Data association algorithms are the tools utilized to perform this update. The difficulty in such algorithms is illustrated in Figure 1.1: the system is provided with a set of detections, each of which indicates the possible presence of a target. However, the system *does not know* which measurement belongs to which target, or which measurements are actually false alarms (the result of radar clutter), hence the best way to update the state estimates for each target using the measurements is unclear.

A Gaussian mixture, consisting of a weighted sum of Gaussian PDFs, each with different means and covariances, is the natural form of the PDF of target

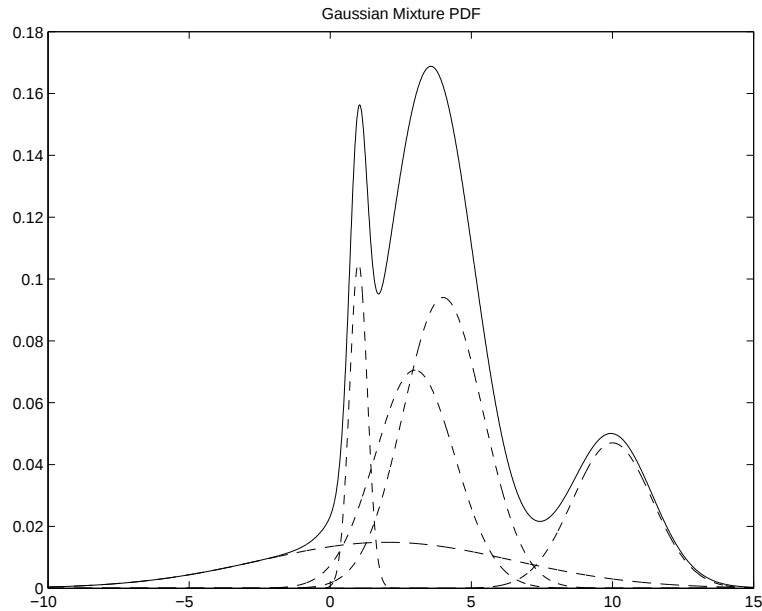


Figure 1.2. An example of a Gaussian mixture: the individual weighted Gaussian component PDFs are shown using dashed lines; the overall PDF (the sum of the components) is shown using a solid line.

state in this problem. Using such a structure, a mixture component is created for every possible association event,² with the mean and covariance calculated *assuming* that the particular hypothesis is true, and the weight calculated to represent the *probability* that the particular hypothesis is true. An example of a Gaussian mixture is shown in Figure 1.2, with the individual weighted component PDFs shown using dashed lines, and the overall PDF (the sum of the components) shown using a solid line.

The difficulty of data association is that every association hypothesis from the previous processing cycle must be paired with every association event from the current set of measurements, and a new association hypothesis must be created for each pairing. For example, if a system commences with a single association hypothesis, and the first set of measurements received gives rise to 20 possible association events,

²i.e., every possible pairing of targets and measurements.

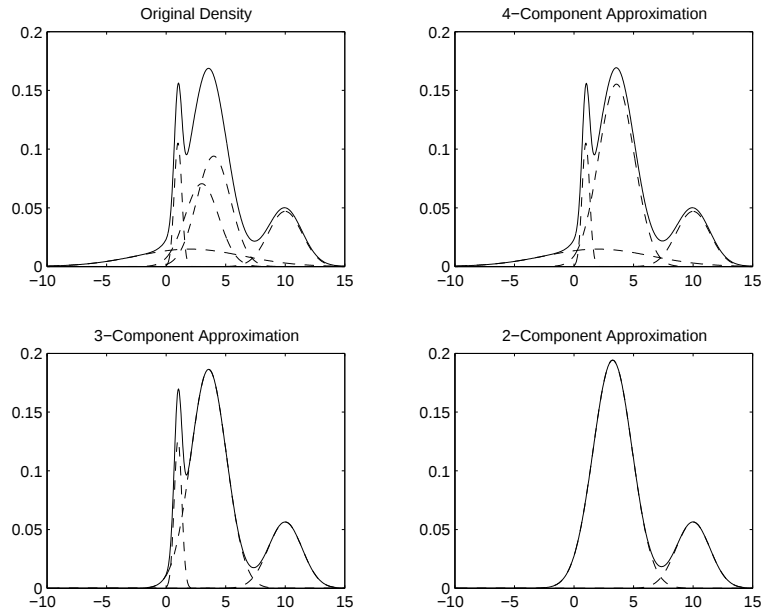


Figure 1.3. Approximating a Gaussian mixture using fewer mixture components. The original mixture is shown in the top left figure, alongside approximations using four components (top right), three components (bottom left) and two components (bottom right).

then there will be 20 association hypotheses. If the following set of measurements produces 30 possible association events, then the number of hypotheses will increase to 600: one for each pairing of previous hypothesis and new association event. Each hypothesis will require a corresponding Gaussian mixture component, each with a different mean, covariance and probability weight; and the number of components will grow exponentially with time. It is therefore necessary to employ methods of reducing the complexity of the mixture while maintaining its overall form as well as possible. Typically many of the hypotheses are very similar, or contribute a very small probability weight, hence it is possible to reduce the number of mixture components without modifying the PDF structure significantly. This is illustrated in Figure 1.3, which shows optimized approximations of the PDF of Figure 1.2 (which has five mixture components) using four, three and two mixture components.

The Multiple Hypothesis Tracker (MHT) [4:334–340, 6:283–300, 7:360–369, 40] is the state-of-the-art tracking algorithm for modern civilian and military radar systems. The algorithm directly maintains the Gaussian mixture representation of the target state PDF, retaining multiple association hypotheses, each represented by a mixture component, with a probability weight, mean vector and covariance matrix. The concept of the MHT is to provide a deferred decision-making structure, such that target-measurement association decisions, which are uncertain at a given processing cycle, can be made at a later time after further information has been received [8]. Although the correct hypothesis may not be the most likely at a given instant in time, as more sets of measurements are received, hypotheses due to random clutter will become less likely, making the correct hypothesis comparatively more likely. Since the number of hypotheses grows exponentially with time, any practical implementation must apply some form of simplification to the PDF, most commonly performed by merging similar hypotheses together, and deleting (pruning) unlikely hypotheses. The effectiveness of the deferred decision-making structure is completely dependent on whether the correct hypothesis remains in the Gaussian mixture when clarifying measurements are received, a function which is determined purely by the merging and pruning logic.

Few mixture reduction algorithms have been published in open literature. Salmond [44–47] proposes an algorithm which *combines* the mixture components which are *closest* in the sense of a given *ad hoc* distance measure. Salmond notes that the ideal solution would be to search for the solution which optimizes a meaningful cost function, but concludes that the computational expense of such an undertaking would be problematic. Fifteen years later, the simulations once performed on a Cray 1S supercomputer can now be run on a common home computer, so it would seem appropriate to challenge such assumptions.

The essence of this study is to reduce a Gaussian mixture model from a larger number of components to a smaller number of components, while modifying the PDF

as a whole as little as possible, as illustrated in Figure 1.3. Although the application of interest in this study is target tracking, the algorithm is equally relevant to any of the rich class of research areas to which the Gaussian mixture has been applied. This includes a wide variety of statistical classification problems such as speech and speaker recognition [14, 41], lip recognition [14, 55], face recognition [14] and image segmentation [14, 58], to name a few.

1.2 *Research Goal*

The goal of this study is to develop techniques of maintaining the PDF of joint target state with higher fidelity than allowed by existing methods. The major focus of the research is to concentrate on the *efficiency* of the representation in order to provide the best description of the target distribution using the most compact set of parameters possible.

Considering this focus on efficiency, the study will commence by examining the difficulties associated with algorithms based on Probabilistic Data Association (PDA) [2, 4], which provide an extremely compact representation of the target state, retaining only a single Gaussian component. Subsequently, methods will be developed to reduce the number of components in a Gaussian mixture while effecting the smallest change possible in the overall PDF structure. In this way, these techniques will provide a generalization of PDA which will provide the best representation possible of the target state PDF for a given number of Gaussian mixture components.

1.3 *Assumptions*

The assumptions made in this study arise out of the probabilistic model of joint association events presented in Section 2.5.2. Firstly, the problem of target existence is not addressed, and the algorithms developed assume knowledge of the number of targets present. Through this assumption, we concentrate the research effort on developing the best possible method of maintaining knowledge of the target state

PDF; target existence considerations can be incorporated later as presented in [42]. Secondly, the measurement model assumes that each measurement belongs to one target and one target only, hence ignoring the possible case in which two targets are within the same radar resolution cell and provide a single merged measurement. While this assumption will be violated in situations in which targets are extremely closely-spaced, such cases can be handled as an exception (as presented in [27, 28]), and the overall solution form is unchanged. Finally, the measurement model assumes that each target gives rise to no more than one measurement, hence ignoring the possible case in which a large, near target spans multiple resolution cells and gives rise to several measurements. Again, extensions can be derived to handle such cases explicitly, and the overall form of the solution remains unchanged.

The simulations presented in Chapter IV also assume a linear Cartesian measurement model in order to employ the standard Kalman filter, as presented in Section 2.2.3. The linear measurement model was selected in order to concentrate the study on the impact of data association; the linear model could be replaced by a nonlinear polar measurement model (as provided by conventional radar systems) by replacing the Kalman filters in the structure with extended Kalman filters, as discussed in Section 2.2.4. The results presented in [44] utilize a linear measurement model for similar reasons.

1.4 *Thesis Organization*

Chapter II examines the background of target tracking, comparing and contrasting current techniques and highlighting areas of possible improvement. Section 2.2 reviews the basic structures of single-target non-maneuvering tracking filters, presenting the background of the Kalman filter. Section 2.3 briefly considers the theory and practice of Gaussian mixture models, which are utilized throughout the remainder of the thesis. Section 2.4 outlines the extensions which have been made to the Kalman filter to address the issue of maneuvering targets, deriving the algo-

rithm which is currently considered state-of-the-art, the Interacting Multiple Model (IMM) estimator [3:461–465, 10]. Section 2.5 derives in detail the probabilistic model of joint association events, and then uses this model to highlight the differences and similarities of the current generation of data association algorithms. Section 2.5.13 briefly outlines some of the methods which have been proposed in open literature to combine the maneuvering techniques of Section 2.4 with the data association techniques of Section 2.5 to aid tracking of multiple maneuvering targets in clutter. Finally, Section 2.6 briefly reviews some of the techniques of numerical optimization which will be utilized in Chapter III.

Chapter III commences by analyzing some of the difficulties observed with the Joint Probabilistic Data Association (JPDA) [2, 4] and Coupled Probabilistic Data Association (CPDA) [12] algorithms, providing new insight into the target bias and track coalescence phenomena examined in [12, 16]. Subsequently, an algorithm is developed for reducing the number of components in a Gaussian mixture, starting in Section 3.3.1 by considering possible cost measures which could be used to measure the deviation from the original PDF caused by the reduction. Section 3.3.2 analyzes our chosen cost function, the Integral Square Difference (ISD) cost, in detail before Section 3.3.3 applies the iterative optimization techniques presented in Section 2.6 to find the set of parameters for the reduced PDF which minimizes the cost. Section 3.3.4 proposes an algorithm which can be used to initialize the iterative optimization, a function which is very important, considering the multi-modal structure of the cost function.

Chapter IV commences by testing the initialization and iterative optimization techniques on a simple one-dimensional problem, providing a graphical demonstration of their operation. Sections 4.4 to 4.6 then present results of computer simulations applying the algorithm to the problems of tracking a single target in clutter, tracking multiple targets in clutter and tracking a single maneuvering target.

Chapter V concludes by summarizing the results and suggesting areas for further investigation.

II. Background

2.1 Introduction

The discipline of target tracking spans a wide range of theory, incorporating basic state estimation, multiple model estimation techniques, and data association methods. Almost all modern tracking systems utilize the Kalman filter as the central tool for state estimation; this method is described in Section 2.2. When a target is maneuvering, the success of the standard Kalman filter can be limited severely. Alternative structures using several Kalman filters in parallel have proven successful in this problem; these are described in Section 2.4. The ambiguity of the origin of measurements is unavoidable in tracking systems; the most common methodologies of dealing with this problem are discussed in Section 2.5. Section 2.3 briefly outlines the Gaussian mixture, which is the statistical model that arises naturally in these problems, and Section 2.6 reviews the theory of iterative optimization which will be employed in Chapter III.

2.2 Tracking Filters

2.2.1 Introduction. The following sections review the fundamental tools of target tracking, most importantly, the Kalman filter. The material presented is not intended to be a complete coverage of the topic area; rather, major outcomes are stated and pointers are given to useful reference material that describe the field in more detail.

2.2.2 Ad Hoc Techniques. The concept that commenced the revolution in surveillance radar performance was that of incorporating dynamics models into the tracking system. By utilizing the equations of Newtonian dynamics, such models make it possible to predict the future location of the target, freeing the radar to perform other tasks between updates. The dynamics models can be based on simple

constant velocity assumptions, such as:

$$\begin{aligned}x(k) &= x(k-1) + v(k-1) \cdot [t_k - t_{k-1}] \\v(k) &= v(k-1)\end{aligned}\tag{2.1}$$

where x is the position of the target, v is the velocity and $[t_k - t_{k-1}]$ is the time difference between the k -th and $(k-1)$ -th measurement instants. If the velocity of the target is not well modelled as constant, then constant acceleration models can be employed such as:

$$\begin{aligned}x(k) &= x(k-1) + v(k-1) \cdot [t_k - t_{k-1}] + \frac{1}{2} a(k-1) \cdot [t_k - t_{k-1}]^2 \\v(k) &= v(k-1) + a(k-1) \cdot [t_k - t_{k-1}] \\a(k) &= a(k-1)\end{aligned}\tag{2.2}$$

where a is the acceleration of the target. Early tracking methods based upon these models estimated the position, velocity, and where necessary, acceleration, of the target using a weighted average of the current measurement and the value predicted using Eq. (2.1) or (2.2). The tracking filter based on the constant velocity assumption is referred to as the α - β tracker, and operates as follows [50:260]:

$$\begin{aligned}\hat{x}(k|k) &= \hat{x}(k|k-1) + \alpha[z(k) - \hat{x}(k|k-1)] \\ \hat{v}(k|k) &= \hat{v}(k|k-1) + \frac{\beta}{t_k - t_{k-1}} [z(k) - \hat{x}(k|k-1)]\end{aligned}\tag{2.3}$$

where $z(k)$ is the k -th measurement of target position, occurring at time t_k . The notation $\hat{x}(k|k-1)$ is used to represent the estimated position of the target at time t_k , predicted using Eq. (2.1) and measurements up to time t_{k-1} , and $\hat{x}(k|k)$ represents the estimated position of the target after incorporation of the measurement $z(k)$. Similarly, $\hat{v}(k|k-1)$ is the estimated velocity before incorporation of the new measurement, and $\hat{v}(k|k)$ is the estimated velocity after incorporation of the mea-

surement. The α - β tracker receives its name from the coefficients, α and β , that are used as weighting factors to perform the updates. If α and β are zero, then the system relies purely on the predictions provided by the dynamics model embedded in the system. Conversely, if α and β are one, then the system ignores the system's dynamics model, and relies purely on the latest measurement. Thus, by adjusting α and β , the designer has a trade-off between the weight that the system places on the past measurements, as propagated through the dynamics model, and the weight that the system places on the newly introduced measurement.

The α - β - γ tracker operates similarly, incorporating an additional weighting factor to aid in estimation of the acceleration (which is assumed constant) [6:21]:

$$\begin{aligned}\hat{x}(k|k) &= \hat{x}(k|k-1) + \alpha[z(k) - \hat{x}(k|k-1)] \\ \hat{v}(k|k) &= \hat{v}(k|k-1) + \frac{\beta}{t_k - t_{k-1}} [z(k) - \hat{x}(k|k-1)] \\ \hat{a}(k|k) &= \hat{a}(k|k-1) + \frac{\gamma}{(t_k - t_{k-1})^2} [z(k) - \hat{x}(k|k-1)]\end{aligned}\tag{2.4}$$

These equations operate as per the α - β tracker with the prediction between measurement intervals performed using the constant acceleration model of Eq. (2.2), and γ representing the weighting coefficient used to update the acceleration estimate of the model.

The α - β and α - β - γ trackers can exhibit very good performance, provided that the weighting coefficients are selected carefully [3:288–289]. However, rules for selecting these coefficients are largely *ad hoc*, relying more on trial and error than on mathematical theory.

2.2.3 Kalman Filter. The Kalman filter is the tool which provides a mathematical basis for *ad hoc* methods such as the α - β and α - β - γ trackers, and at the same time gives a mechanism for calculating the optimal values of the weighting

coefficients. The filter is based on the simple linear state model:¹

$$\begin{aligned}\mathbf{x}(k) &= \mathbf{\Phi}(k, k-1)\mathbf{x}(k-1) + \mathbf{G}_d(k-1)\mathbf{w}(k-1) \\ \mathbf{z}(k) &= \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k)\end{aligned}\tag{2.5}$$

where $\mathbf{x}(k)$ is the true state of the system at sample instant k ; $\mathbf{z}(k)$ is the noise corrupted measurement supplied to the estimator at time instant k ; $\mathbf{\Phi}$, $\mathbf{G}_d$ and $\mathbf{H}$ are known system matrices; and $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are two mutually independent white Gaussian noise processes (also independent of prior knowledge of $\mathbf{x}$) such that:

$$\begin{aligned}E\{\mathbf{w}(k)\mathbf{w}(l)\} &= \mathbf{Q}_d(k) \delta_{kl} \\ E\{\mathbf{v}(k)\mathbf{v}(l)\} &= \mathbf{R}(k) \delta_{kl}\end{aligned}$$

where δ_{kl} is the Kronecker delta function (unity when $k = l$, zero otherwise).

If the prior knowledge of $\mathbf{x}$ indicates that it follows a Gaussian Probability Density Function (PDF) with mean $\hat{\mathbf{x}}(k-1|k-1)$ and covariance $\mathbf{P}(k-1|k-1)$:

$$\begin{aligned}f\{\mathbf{x}(k-1)|\mathbf{Z}^{k-1}\} &= |2\pi\mathbf{P}(k-1|k-1)|^{-\frac{1}{2}} \cdot \\ &\quad \cdot \exp\{-\frac{1}{2}[\mathbf{x}(k-1) - \hat{\mathbf{x}}(k-1|k-1)]^T \cdot \\ &\quad \cdot \mathbf{P}(k-1|k-1)^{-1}[\mathbf{x}(k-1) - \hat{\mathbf{x}}(k-1|k-1)]\} \\ &\triangleq \mathcal{N}\{\mathbf{x}(k-1); \hat{\mathbf{x}}(k-1|k-1), \mathbf{P}(k-1|k-1)\}\end{aligned}$$

then it can be shown that the PDF of the state of $\mathbf{x}$ propagated forward to sample period k remains Gaussian [34:208–209]:

$$f\{\mathbf{x}(k)|\mathbf{Z}^{k-1}\} = \mathcal{N}\{\mathbf{x}(k); \hat{\mathbf{x}}(k|k-1), \mathbf{P}(k|k-1)\}$$

¹Shown in discrete-time form, even though there will inevitably be an underlying continuous-time system.

where the mean $\hat{\mathbf{x}}(k|k-1)$ and covariance $\mathbf{P}(k|k-1)$ are described by the Kalman filter propagation equations:

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= \Phi(k, k-1)\hat{\mathbf{x}}(k-1|k-1) \\ \mathbf{P}(k|k-1) &= \Phi(k, k-1)\mathbf{P}(k-1|k-1)\Phi(k, k-1)^T + \\ &\quad + \mathbf{G}_d(k-1)\mathbf{Q}_d(k-1)\mathbf{G}_d(k-1)^T\end{aligned}\tag{2.6}$$

Although Kalman first derived the measurement update algorithm using insights from orthogonal projection [24], it is easily understood for the case of Gaussian PDFs by applying Bayes' rule:

$$\begin{aligned}f\{\mathbf{x}(k)|\mathbf{Z}^k\} &= f\{\mathbf{x}(k)|\mathbf{z}(k), \mathbf{Z}^{k-1}\} \\ &= \frac{f\{\mathbf{x}(k), \mathbf{z}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|\mathbf{x}(k), \mathbf{Z}^{k-1}\}f\{\mathbf{x}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}}\end{aligned}\tag{2.7}$$

The first term in the numerator of Eq. (2.7) represents the PDF of the measurement vector conditioned on the *true value* of the target state vector, as well as the previous measurement history. Given the relationship of Eq. (2.5), this PDF will thus be Gaussian with a mean of $\mathbf{H}\mathbf{x}(k)$ (the *true* value of the state vector), and covariance $\mathbf{R}$. The latter term in the numerator of Eq. (2.7) represents the knowledge of the current state conditioned on the previous measurements: this will be the Gaussian function propagated from the previous processing cycle with parameters as per Eq. (2.6). The denominator of Eq. (2.7) is simply the marginal density of the measurements, calculated as the integral of the numerator over all $\mathbf{x}$:

$$\begin{aligned}f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\} &= \int_{-\infty}^{\infty} f\{\mathbf{x}(k), \mathbf{z}(k)|\mathbf{Z}^{k-1}\}d\mathbf{x}(k) \\ &= \int_{-\infty}^{\infty} f\{\mathbf{z}(k)|\mathbf{x}(k), \mathbf{Z}^{k-1}\}f\{\mathbf{x}(k)|\mathbf{Z}^{k-1}\}d\mathbf{x}(k)\end{aligned}$$

where the vector limits $(-\infty, \infty)$ remind us that the integration is to be performed over every element of the vector $\mathbf{x}(k)$.

Collecting these results, Eq. (2.7) can (after much algebra) be simplified to be a Gaussian PDF, with mean $\hat{\mathbf{x}}(k|k)$ and covariance $\mathbf{P}(k|k)$ [34:212–217]:

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \mathcal{N}\{\mathbf{x}(k); \hat{\mathbf{x}}(k|k), \mathbf{P}(k|k)\}$$

where the mean $\hat{\mathbf{x}}(k|k)$ and covariance $\mathbf{P}(k|k)$ are described by the Kalman filter measurement update equations:

$$\begin{aligned}\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)] \\ \mathbf{P}(k|k) &= \mathbf{P}(k|k-1) - \mathbf{K}(k)\mathbf{H}(k)\mathbf{P}(k|k-1)\end{aligned}\tag{2.8}$$

with $\mathbf{K}(k)$ referred to as the Kalman filter gain:

$$\mathbf{K}(k) = \mathbf{P}(k|k-1)\mathbf{H}(k)^T[\mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)^T + \mathbf{R}(k)]^{-1}\tag{2.9}$$

The form of the measurement update expression in Eq. (2.8) gives rise to the definition of the filter residual, also referred to as the innovation [34:218]:

$$\boldsymbol{\nu}(k) = \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)\tag{2.10}$$

Considering the form of Eq. (2.10), the residual consists of the difference between the *actual* measurement, and the *best prediction* of the measurement: hence it embodies the *new information* that the measurement provides to the system. If the assumptions of the algorithm summarized in Eq. (2.5) are satisfied, then the residual

series should possess the following properties [34:228–229]:

$$\begin{aligned}
\text{Zero-mean} & : E\{\boldsymbol{\nu}(k)\} = \mathbf{0} \\
\text{White} & : E\{\boldsymbol{\nu}(k)\boldsymbol{\nu}(l)^T\} = \mathbf{0}, \quad k \neq l \\
\text{Covariance} & : E\{\boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T\} \triangleq \mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)^T + \mathbf{R}(k) \\
\text{Gaussian} & : f\{\boldsymbol{\nu}(k)\} = \mathcal{N}\{\boldsymbol{\nu}(k); \mathbf{0}, \mathbf{S}(k)\}
\end{aligned} \tag{2.11}$$

If the residual series does not display these characteristics, then there is a high probability that the assumptions of the algorithm have not been satisfied. There are two common causes of this in the target tracking application: target maneuver and measurement ambiguity. If the maneuver that the target is exhibiting does not match the maneuver described in the Kalman filter dynamics model (i.e., the matrix Φ being used in the Kalman filter does not match the true Φ matrix), then the mismatch will produce a residual series which does not possess the characteristics described in Eq. (2.11). Methods of dealing with the problems caused by target maneuver are described in Section 2.4. Similarly, if the system uses the *incorrect* measurement then the residual series will not possess the characteristics of Eq. (2.11). This is common in radar systems which produce several measurements during each processing interval (some due to different targets, some due to clutter), but only one measurement is correct, and the system *does not know* which is the correct measurement. Methods for dealing with the problems of measurement uncertainty and data association are described in Section 2.5.

Following from the properties of Eq. (2.11), the likelihood that a given residual vector will occur is described by the PDF:

$$\mathcal{N}\{\boldsymbol{\nu}(k); \mathbf{0}, \mathbf{S}(k)\} = |2\pi\mathbf{S}(k)|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}\boldsymbol{\nu}(k)^T \mathbf{S}(k)^{-1} \boldsymbol{\nu}(k) \right\} \tag{2.12}$$

Following from this expression, we can define a region in $\boldsymbol{\nu}(k)$ space such that the probability that a valid residual (resulting from the correct measurement and a

correctly modelled system) is outside of this region is very small. For example, if we want the region $\mathbb{V}$ to contain the most likely set of residuals such that the probability that a residual lies within $\mathbb{V}$ is P_g :²

$$P_g = \int_{\boldsymbol{\nu}(k) \in \mathbb{V}} |2\pi \mathbf{S}(k)|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} \boldsymbol{\nu}(k)^T \mathbf{S}(k)^{-1} \boldsymbol{\nu}(k) \right\} d\boldsymbol{\nu}(k) \quad (2.13)$$

then the region will be defined by:

$$\mathbb{V} = \{ \boldsymbol{\nu}(k) : \boldsymbol{\nu}(k)^T \mathbf{S}(k)^{-1} \boldsymbol{\nu}(k) \leq \gamma \} \quad (2.14)$$

where γ is a threshold calculated to produce the desired probability P_g . There are many interpretations of the distance measure defined by $\boldsymbol{\nu}(k)^T \mathbf{S}(k)^{-1} \boldsymbol{\nu}(k)$. If the covariance $\mathbf{S}(k)$ is an identity matrix, then the distance measure reduces to the norm of the residual, or alternatively the Euclidean distance between the measurement and the predicted value of the measurement. In the multidimensional case, it seems appropriate then to weight each principal axis according to the relative level of certainty in that direction. Thus the inclusion of the predicted covariance modifies the Euclidean distance inner product to a generalized inner product utilizing appropriate weightings. The impact of the covariance inverse is to make the elements of the vector independent, thus the resultant test statistic is distributed according to a χ^2 PDF, for which the number of degrees of freedom is the number of measurement variables.

The Kalman filter is the *optimal* solution (according to almost any error criterion) to the tracking problem if the system is linear and known, with additive white Gaussian noise. The Kalman filter is also the optimal *linear* linear solution (according to the minimum variance unbiased criterion) for any linear tracking problem, regardless of the characteristics of the noise process [34:235].

²As discussed in Section 2.5.1, P_g represents the probability that a measurement will lie within a gating region.

There is much more to be said about the Kalman filter; this section has merely stated the equations of the discrete-time variant. There are many topics which should be understood in order to address the target tracking problem, particularly generation of the discrete-time model for an underlying continuous-time system, and modelling of time-correlated noise processes. Countless books have been written on these subjects, and the interested reader is directed to the thorough coverage of the area which can be found in [34].

The continuous-time system underlying the discrete-time system of Eq. (2.5) will be of the form:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \\ \mathbf{z}(k) &= \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k)\end{aligned}\tag{2.15}$$

where $\mathbf{x}(k) \triangleq \mathbf{x}(t_k)$ for notational convenience, and $\mathbf{w}(t)$ is a continuous-time white noise process such that:

$$E\{\mathbf{w}(t)\mathbf{w}(t')\} = \mathbf{Q}(t) \delta(t - t')$$

and $\delta(t - t')$ is the Dirac delta function:

$$\begin{aligned}\delta(t) &= 0, \quad t \neq 0 \\ \int_{-\infty}^{\infty} \delta(t) dt &= 1\end{aligned}$$

The Kalman filter propagation equations for this system are equivalent to Eq. (2.6) [34:209]:

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= \mathbf{\Phi}(t_k, t_{k-1})\hat{\mathbf{x}}(k-1|k-1) \\ \mathbf{P}(k|k-1) &= \mathbf{\Phi}(t_k, t_{k-1})\mathbf{P}(k-1|k-1)\mathbf{\Phi}(t_k, t_{k-1})^T + \\ &\quad + \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}(\tau)^T\mathbf{\Phi}(t_k, \tau)^T d\tau\end{aligned}\quad (2.16)$$

which gives rise to the definition of $\mathbf{G}_d$ and $\mathbf{Q}_d$ matrices which satisfy:

$$\mathbf{G}_d(k-1)\mathbf{Q}_d(k-1)\mathbf{G}_d(k-1)^T \triangleq \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}(\tau)^T\mathbf{\Phi}(t_k, \tau)^T d\tau$$

The *state transition matrix* $\mathbf{\Phi}(t|t_0)$ is the solution of the deterministic differential equation [34:40]:

$$\frac{d\mathbf{\Phi}(t, t_0)}{dt} = \mathbf{F}(t)\mathbf{\Phi}(t, t_0)$$

from the initial condition $\mathbf{\Phi}(t_0, t_0) = \mathbf{I}$, and $\mathbf{\Phi}(k|k-1) \triangleq \mathbf{\Phi}(t_k|t_{k-1})$ for notational convenience. If the dynamical system is time invariant such that $\mathbf{F}(t) = \mathbf{F}$, then $\mathbf{\Phi}(t|t_0)$ can be shown to be [34:42]:

$$\mathbf{\Phi}(t|t_0) = \exp(\mathbf{F} \cdot [t - t_0])$$

where $\exp(\cdot)$ denotes the *matrix* exponential operation.

2.2.4 Nonlinear Filters. While the body of knowledge for dealing with linear systems is very extensive, few systems are truly linear in reality, but rather they can be modelled as linear within certain operating regions. When it is necessary to operate outside these regions, linear techniques break down and more advanced nonlinear estimator forms become necessary. The most common nonlinear estimation algorithm is the Extended Kalman Filter (EKF) [3, 35].

The EKF is designed to address nonlinearities of the following form:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}[\mathbf{x}(t), t] + \mathbf{G}(t)\mathbf{w}(t) \\ \mathbf{z}(k) &= \mathbf{h}[\mathbf{x}(k), k] + \mathbf{v}(k)\end{aligned}\tag{2.17}$$

where the first equation is in continuous-time form, similarly to Eq. (2.15), as nonlinear systems cannot in general be converted to equivalent discrete-time systems. The vector function $\mathbf{f}[\mathbf{x}(t), t]$ represents the nonlinear dynamics model of the system, while the vector function $\mathbf{h}[\mathbf{x}(k), k]$ represents the nonlinear measurement model of the system. The noise processes for both equations remain additive, this being the major restriction of the EKF technique.

Using the EKF, time propagation between measurement samples $(k - 1)$ (occurring at time t_{k-1}) and k (occurring at time t_k) must be performed using numerical integration of the following expressions [35:44–45]:

$$\begin{aligned}\dot{\hat{\mathbf{x}}}(t|t_{k-1}) &= \mathbf{f}[\hat{\mathbf{x}}(t|t_{k-1}), t] \\ \dot{\mathbf{P}}(t|t_{k-1}) &= \mathbf{F}[\hat{\mathbf{x}}(t|t_{k-1}), t]\mathbf{P}(t|t_{k-1}) + \\ &\quad + \mathbf{P}(t|t_{k-1})\mathbf{F}[\hat{\mathbf{x}}(t|t_{k-1}), t]^T + \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}(t)^T\end{aligned}\tag{2.18}$$

where the matrix $\mathbf{F}[\hat{\mathbf{x}}(t|t_{k-1}), t]$ represents the linearization of the vector function $\mathbf{f}[\hat{\mathbf{x}}(t|t_{k-1}), t]$ with respect to the parameter $\hat{\mathbf{x}}(t|t_{k-1})$, reevaluated over each numerical integration step:

$$\mathbf{F}[\hat{\mathbf{x}}(t|t_{k-1}), t] \triangleq \frac{\partial \mathbf{f}[\hat{\mathbf{x}}(t|t_{k-1}), t]}{\partial \hat{\mathbf{x}}(t|t_{k-1})}$$

The measurement update equation for the EKF is very similar to the standard Kalman filter measurement update form of Eq. (2.8) [35:44]:

$$\begin{aligned}\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)\{\mathbf{z}(k) - \mathbf{h}[\hat{\mathbf{x}}(k|k-1), k]\} \\ \mathbf{P}(k|k) &= \mathbf{P}(k|k-1) - \mathbf{K}(k)\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k]\mathbf{P}(k|k-1)\end{aligned}\tag{2.19}$$

where the Kalman filter gain remains as per the standard Kalman filter:

$$\begin{aligned}\mathbf{K}(k) &= \mathbf{P}(k|k-1)\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k]^T \cdot \\ &\quad \cdot \{\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k]\mathbf{P}(k|k-1)\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k]^T + \mathbf{R}(k)\}^{-1}\end{aligned}\tag{2.20}$$

and the matrix $\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k]$ is the linearization of the vector function $\mathbf{h}[\hat{\mathbf{x}}(k|k-1), k]$ with respect to the parameter $\hat{\mathbf{x}}(k|k-1)$:

$$\mathbf{H}[\hat{\mathbf{x}}(k|k-1), k] \triangleq \frac{\partial \mathbf{h}[\hat{\mathbf{x}}(k|k-1), k]}{\partial \hat{\mathbf{x}}(k|k-1)}$$

If either the dynamics model or the measurement model is linear, then the standard Kalman filter equations may be used for that portion of the processing cycle. In this study, we will neglect the effect of nonlinearities in order to concentrate purely on the impact of the problems caused by target maneuver and data association.

2.3 Gaussian Mixtures

The Gaussian mixture is a powerful modelling tool for characterizing the PDF of variables which follow complicated multi-modal distributions. The basic form of

a Gaussian mixture containing N components is:

$$f\{\mathbf{x}\} = \sum_{i=1}^N p_i \mathcal{N}\{\mathbf{x}; \hat{\mathbf{x}}_i, \mathbf{P}_i\} \quad (2.21)$$

where $\{p_i\}$ are the relative weights of each Gaussian component ($p_i \geq 0 \ \forall \ i$, $\sum_i^N p_i = 1$), $\{\hat{\mathbf{x}}_i\}$ are the means of each component, and $\{\mathbf{P}_i\}$ are the covariances.

As will be seen in the following sections, Gaussian mixture models arise naturally as the solution to several problems in target tracking, including maneuvering target tracking and data association. In the coming sections it will often be necessary to calculate the overall mean and overall covariance of a Gaussian mixture. The overall mean can be calculated using the expectation operation:

$$\begin{aligned} \boldsymbol{\mu}_c &= E\{\mathbf{x}\} = \int_{-\infty}^{\infty} \left\{ \sum_{i=1}^N p_i \mathbf{x} \mathcal{N}\{\mathbf{x}; \hat{\mathbf{x}}_i, \mathbf{P}_i\} \right\} d\mathbf{x} \\ &= \sum_{i=1}^N p_i \int_{-\infty}^{\infty} \mathbf{x} \mathcal{N}\{\mathbf{x}; \hat{\mathbf{x}}_i, \mathbf{P}_i\} d\mathbf{x} \\ &= \sum_{i=1}^N p_i \hat{\mathbf{x}}_i \end{aligned} \quad (2.22)$$

The overall covariance can be calculated similarly:

$$\begin{aligned} \mathbf{P}_c &= E\{(\mathbf{x} - \boldsymbol{\mu}_c)(\mathbf{x} - \boldsymbol{\mu}_c)^T\} = E\{\mathbf{x}\mathbf{x}^T\} - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T \\ &= \int_{-\infty}^{\infty} \left\{ \sum_{i=1}^N p_i \mathbf{x}\mathbf{x}^T \mathcal{N}\{\mathbf{x}; \hat{\mathbf{x}}_i, \mathbf{P}_i\} \right\} d\mathbf{x} - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T \\ &= \sum_{i=1}^N p_i \int_{-\infty}^{\infty} \mathbf{x}\mathbf{x}^T \mathcal{N}\{\mathbf{x}; \hat{\mathbf{x}}_i, \mathbf{P}_i\} d\mathbf{x} - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T \\ &= \sum_{i=1}^N p_i (\mathbf{P}_i + \hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^T) - \boldsymbol{\mu}_c \boldsymbol{\mu}_c^T \\ &= \sum_{i=1}^N p_i [\mathbf{P}_i + (\hat{\mathbf{x}}_i - \boldsymbol{\mu}_c)(\hat{\mathbf{x}}_i - \boldsymbol{\mu}_c)^T] \end{aligned} \quad (2.23)$$

The major theme of this thesis is to simplify a Gaussian mixture with many components to a reduced form with fewer components. One of the common building blocks for performing this function will be to merge two similar mixture components together. In order to maintain the same overall mean and covariance for the mixture, the parameters of the combined component will be [6:293]:

$$\begin{aligned}
\text{Weight} &: p_c = p_1 + p_2 \\
\text{Mean} &: \boldsymbol{\mu}_c = \frac{1}{p_1 + p_2} \{p_1 \hat{\mathbf{x}}_1 + p_2 \hat{\mathbf{x}}_2\} \\
\text{Covariance} &: \mathbf{P}_c = \frac{1}{p_1 + p_2} \left\{ p_1 \mathbf{P}_1 + p_2 \mathbf{P}_2 + \frac{p_1 p_2}{p_1 + p_2} (\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)^T \right\}
\end{aligned} \tag{2.24}$$

The derivation of $\boldsymbol{\mu}_c$ in Eq. (2.24) is obvious following from Eq. (2.22); $\mathbf{P}_c$ in Eq. (2.24) is derived from Eq. (2.23) by:

$$\begin{aligned}
\mathbf{P}_c &= \frac{\sum_{i=1}^2 p_i [\mathbf{P}_i + (\hat{\mathbf{x}}_i - \boldsymbol{\mu}_c)(\hat{\mathbf{x}}_i - \boldsymbol{\mu}_c)^T]}{p_1 + p_2} \\
&= [p_1 \mathbf{P}_1 + p_1 (\hat{\mathbf{x}}_1 - \boldsymbol{\mu}_c)(\hat{\mathbf{x}}_1 - \boldsymbol{\mu}_c)^T + \\
&\quad + p_2 \mathbf{P}_2 + p_2 (\hat{\mathbf{x}}_2 - \boldsymbol{\mu}_c)(\hat{\mathbf{x}}_2 - \boldsymbol{\mu}_c)^T] / (p_1 + p_2)
\end{aligned} \tag{2.25}$$

Expanding $\boldsymbol{\mu}_c$ using Eq. (2.24):

$$\begin{aligned}
\hat{\mathbf{x}}_1 - \boldsymbol{\mu}_c &= \hat{\mathbf{x}}_1 - (p_1 \hat{\mathbf{x}}_1 + p_2 \hat{\mathbf{x}}_2) / (p_1 + p_2) \\
&= (p_1 \hat{\mathbf{x}}_1 + p_2 \hat{\mathbf{x}}_1 - p_1 \hat{\mathbf{x}}_1 - p_2 \hat{\mathbf{x}}_2) / (p_1 + p_2) \\
&= p_2 (\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2) / (p_1 + p_2)
\end{aligned} \tag{2.26}$$

Similarly:

$$\begin{aligned}
\hat{\mathbf{x}}_2 - \boldsymbol{\mu}_c &= \hat{\mathbf{x}}_2 - (p_1\hat{\mathbf{x}}_1 + p_2\hat{\mathbf{x}}_2)/(p_1 + p_2) \\
&= (p_1\hat{\mathbf{x}}_2 + p_2\hat{\mathbf{x}}_2 - p_1\hat{\mathbf{x}}_1 - p_2\hat{\mathbf{x}}_2)/(p_1 + p_2) \\
&= -p_1(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)/(p_1 + p_2)
\end{aligned} \tag{2.27}$$

Substituting these expressions into Eq. (2.25) we obtain:

$$\begin{aligned}
\mathbf{P}_c &= [p_1\mathbf{P}_1 + p_1p_2^2(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)^T/(p_1 + p_2)^2 + \\
&\quad + p_2\mathbf{P}_2 + p_2p_1^2(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)^T/(p_1 + p_2)^2]/(p_1 + p_2) \\
&= [p_1\mathbf{P}_1 + p_2\mathbf{P}_2 + \frac{p_1p_2(p_1 + p_2)}{(p_1 + p_2)^2}(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)^T]/(p_1 + p_2) \\
&= [p_1\mathbf{P}_1 + p_2\mathbf{P}_2 + \frac{p_1p_2}{p_1 + p_2}(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2)^T]/(p_1 + p_2)
\end{aligned} \tag{2.28}$$

which matches the result shown in Eq. (2.24).

2.4 Multiple Model Adaptive Estimation

The techniques described in Section 2.2 are extremely effective at tracking moving objects when the assumptions of the algorithm are satisfied. However, one assumption inherent to those methods is that the dynamics model of the target is *known* for all time. In any scenario of practical interest, this will never be the case, and thus any claim of optimality (or even near-optimality) is lost.

In the target tracking application, there are two fundamental *classes* of models: maneuvering and non-maneuvering. Non-maneuvering models are used to exploit the fact that most aircraft fly along straight paths most of the time: such knowledge brings intrinsic certainty into the estimation problem, which can be used to reduce the bandwidth of the tracking filter and greatly increase the precision of state estimates.

Maneuvering models are needed in an estimation system to describe the motion of the target when it is *not non-maneuvering*. The rich variety of maneuvers that a target may exhibit (particularly in a military setting) comparatively raises the level of uncertainty in the system. While some segments may be able to be predicted with some accuracy for short periods of time, to some extent the maneuver will need to be modelled as a noise process with appropriately selected strength and bandwidth (through an appropriately designed noise process shaping filter).

Fundamentally, target trackers may use two strategies to adapt to changing dynamics models. The first approach is to use the measurement to *estimate* the unknown maneuver parameters (often using batch processing or sliding window-type methods), and then correct the state estimates using these parameters. The disadvantage of this method is that it is slow to adapt to change: it takes several sample periods after the onset of a maneuver to be able to estimate the maneuver parameters with any level of accuracy. Iterative reprocessing of the last measurement or the last N measurements can reduce this lag, but not without a significant increase in computational complexity.

The second approach for adapting to changing target dynamics is to use a parallel bank of non-adaptive estimators, each tuned to a different operating condition (e.g., type and level of maneuver, etc.), and then to combine the outputs into a single weighted average estimate based on the apparent performance of each elemental filter. This latter architecture has a major advantage over the former in regards to adaptation speed: the question of “what is the maneuver?” has effectively been changed to “is the maneuver best represented by model a, model b, or model c?”, thereby re-posing the question as a *detection* problem, rather than an *estimation* problem. Virtually all multiple model techniques share this same basic architecture, and differ only in the manner in which the model weights are calculated, and in the mixing of model-conditioned estimates between processing cycles.

2.4.1 Non-Switching Models. The “basic” form of Multiple Model Adaptive Estimator (MMAE) [3, 37] is derived when one assumes that the model in force does not change with time. While at first this may seem to defeat the purpose of employing multiple model methods, the result reveals insight into *ad hoc* modifications which can be used to transform the structure into an effective adaptive algorithm.

2.4.1.1 Calculation of Model Probabilities. The event M_j is defined to represent the condition that dynamics model j is in force. No time argument is required, as the model is assumed not to switch with time. The *a posteriori* probability that model j is in force conditioned on the measurement history up to and including sample time k is represented by:

$$\mu_j(k) \triangleq P\{M_j|\mathbf{Z}^k\} \quad (2.29)$$

Expanding $\mathbf{Z}^k$ in Eq. (2.29) into the combination of the previous measurement history $\mathbf{Z}^{k-1}$ combined with the current measurement $\mathbf{z}(k)$, and then using Bayes’ rule in both $\mathbf{z}(k)$ and M_j yields:

$$\begin{aligned} \mu_j(k) &= P\{M_j|\mathbf{Z}^{k-1}, \mathbf{z}(k)\} \\ &= \frac{f\{M_j, \mathbf{z}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|M_j, \mathbf{Z}^{k-1}\}P\{M_j|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \end{aligned} \quad (2.30)$$

where the notation $P\{\cdot\}$ refers to the probability of a discrete event, whereas $f\{\cdot\}$ refers to the density function of a continuous variable. The denominator in Eq. (2.30) can be expanded using the total probability expansion over all models:

$$f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\} = \sum_{i=1}^{N_f} f\{\mathbf{z}(k)|M_i, \mathbf{Z}^{k-1}\}P\{M_i|\mathbf{Z}^{k-1}\} \quad (2.31)$$

where N_f is the number of hypothesized models (and thus the number of elemental filters in the structure). This gives the following recursive equation for the model probabilities $\mu_j(k)$:

$$\mu_j(k) = \frac{f\{\mathbf{z}(k)|M_j, \mathbf{Z}^{k-1}\}\mu_j(k-1)}{\sum_{i=1}^{N_f} f\{\mathbf{z}(k)|M_i, \mathbf{Z}^{k-1}\}\mu_i(k-1)} \quad (2.32)$$

The representation of the parameter space by a number of discrete models is effectively an employment of the total probability theorem, representing the PDF of the new measurement as a weighted sum over each possible model. The total probability theorem rule requires two characteristics of the event space partitioning: the partitions should be *mutually exclusive* (they should not overlap within the space), and they should be *complete* (they should span the entire event space). Although these requirements will seldom be met in practical multiple model filter designs, they remain valuable design guidelines: that the models chosen should be distinct, such that a single model should be predominantly responding at any one time, and complete, such that the elemental filters adequately model all possible hypotheses that may feasibly occur.

The calculation of $f\{\mathbf{z}(k)|M_j, \mathbf{Z}^{k-1}\}$ is a simple matter for the non-switching model case; this density function represents the match between the incoming measurement and the previous measurement history assuming that model j has been in force throughout. Assuming the standard linear Kalman filter measurement model, this is evaluated as:

$$f\{\mathbf{z}(k)|M_j, \mathbf{Z}^{k-1}\} = \mathcal{N}\{\mathbf{z}(k); \mathbf{H}_j \hat{\mathbf{x}}_j(k|k-1), \mathbf{H}_j \mathbf{P}_j(k|k-1) \mathbf{H}_j^T + \mathbf{R}_j\} \quad (2.33)$$

where $\hat{\mathbf{x}}_j(k|k-1)$ and $\mathbf{P}_j(k|k-1)$ are the state estimate and covariance of the filter for model j , and $\mathbf{H}_j$ and $\mathbf{R}_j$ are the measurement matrix and measurement covariance under model j .

2.4.1.2 Calculation of Combined Estimate. The central conditional mean estimate is formed as a weighted average of the elemental filter estimates using the model probabilities $\mu_j(k)$ as the weights:

$$\hat{\mathbf{x}}(k|k) = \sum_{j=1}^{N_f} \mu_j(k) \hat{\mathbf{x}}_j(k|k) \quad (2.34)$$

Though generally not required, the covariance of this estimate can also be formed using a weighted average, but adding the correction term which takes into account the spreading introduced by the different estimates:

$$\mathbf{P}(k|k) = \sum_{j=1}^{N_f} \mu_j(k) \{ \mathbf{P}_j(k|k) + [\hat{\mathbf{x}}_j(k|k) - \hat{\mathbf{x}}(k|k)][\hat{\mathbf{x}}_j(k|k) - \hat{\mathbf{x}}(k|k)]^T \} \quad (2.35)$$

Figure 2.1 shows the structure of the non-switching MMAE algorithm. The model-conditioned estimates calculated by each elemental filter at each processing cycle are passed directly into the same filter at the following processing cycle, as it is assumed that the model in force does not change with time. The overall combined estimate is calculated as a weighted average, as shown in Eqs. (2.34) and (2.35).

2.4.1.3 Ad Hoc Modifications. The assumption of the algorithm that the model in force *will not change* is evident in the form of Eq. (2.32): the recursive nature of the formulation implies that the certainty of the system will grow with time, as the *a priori* model probabilities are multiplied by the model-conditioned measurement density function at each processing cycle. To illustrate the difficulty that this can cause, consider a model which consistently performs poorly (as it is badly mismatched to the true system). As time progresses, the probability of the model decreases exponentially, as the certainty with which the model is rejected grows. If the true system switches models (e.g., a non-maneuvering target com-

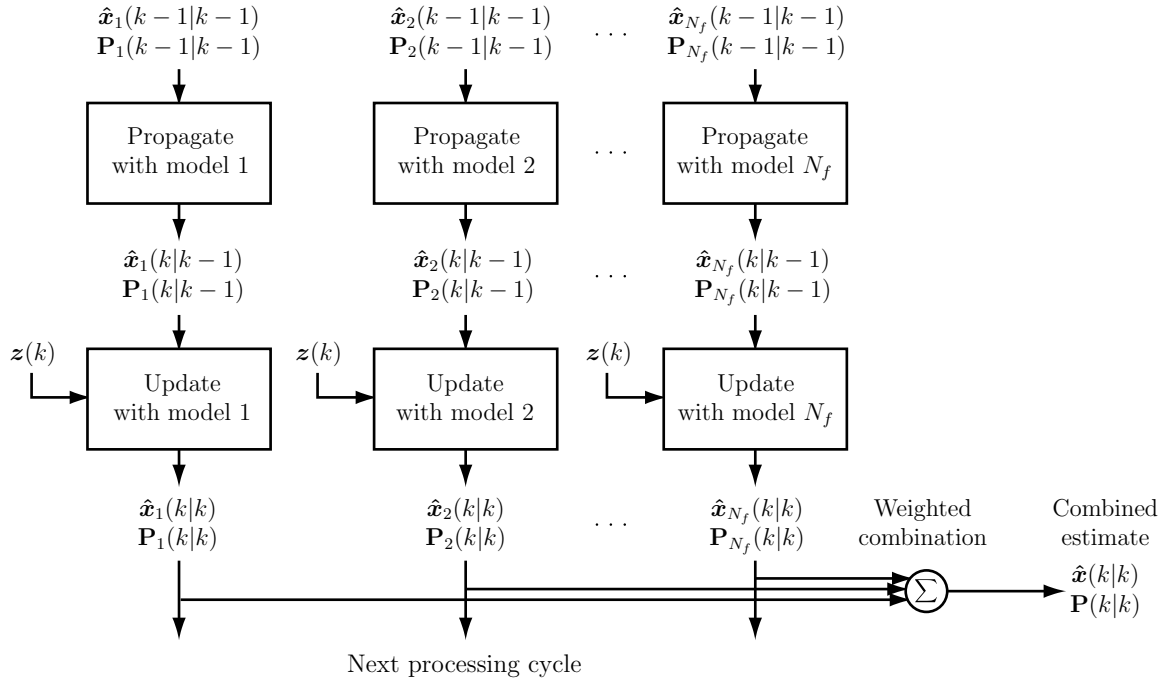


Figure 2.1. Block diagram of non-switching multiple model estimation algorithm.

mences an evasive maneuver), then many orders of magnitude of certainty with which the new model was being *rejected* (through the recursion) must be overcome before a significant amount of probability will return to it. If the probability of the model had decreased to such a level that a numerical underflow condition had occurred and the value had been rounded to zero, then the model probability will *never* recover.

An obvious method of overcoming this difficulty is to impose a lower bound on the model probabilities such that any probabilities that fall below the bound are increased back to that level. The level of the bound can be adjusted experimentally, providing a trade-off between speed of adaptation, and level of certainty accrued by the estimator. Higher bounds will increase the agility of probability flow between models while making the system more susceptible to incorrect probability flow due to noise, whereas lower bounds will slow the adaptation process while providing more

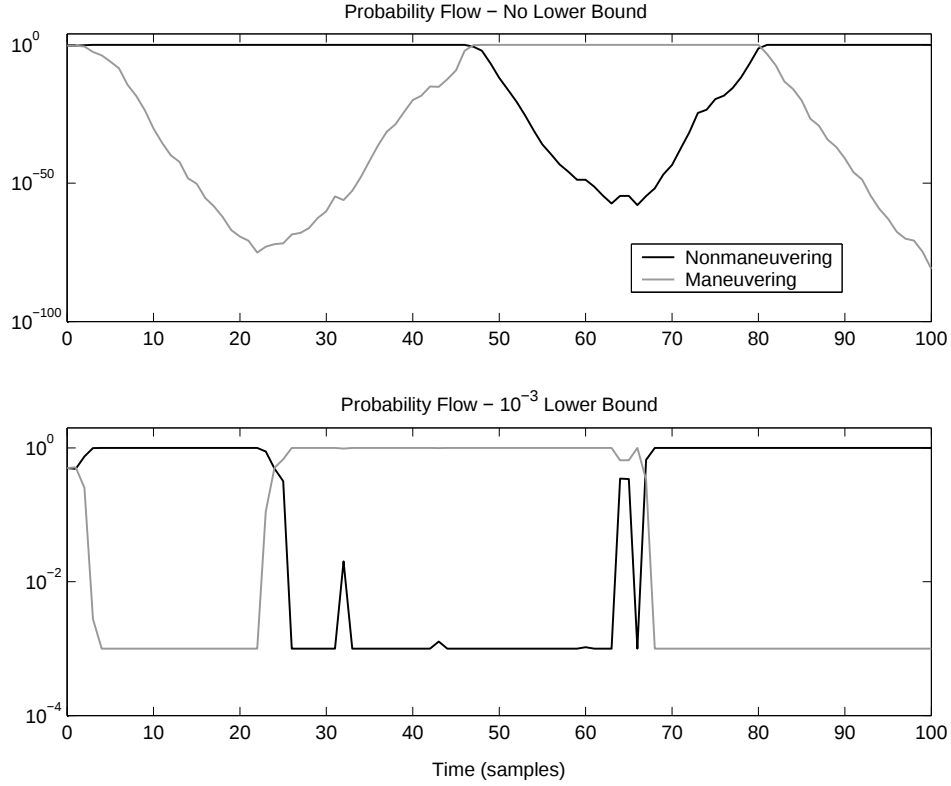


Figure 2.2. MMAE probability flow with and without a lower probability bound. Note the logarithmic scale used in each of the plots.

robustness against noise. An example of this is shown in Figure 2.2, in which the non-maneuvering model is in effect from samples 0 to 20 and from 61 to 100, and the maneuvering model is in effect from samples 21 to 60. The top diagram shows the difficulty experienced when no lower bound is applied: by the time the maneuvering model comes into force (at the 21st sample), its probability has reduced to 10^{-75} , and it takes nearly 30 samples for this probability to recover and return to being competitive with the non-maneuvering model. The bottom diagram shows the same scenario, but with a lower bound of 10^{-3} applied to the model probabilities. The use of the lower bound reduces the time required to respond to the model switch to around five sample periods.

The second *ad hoc* modification required to use the algorithm in practice is to monitor the estimates of badly-performing models for divergence, and reinitialize them if this is sensed to occur. Traditionally the trigger used for this has been the normalized residual, $\boldsymbol{\nu}_j^T \mathbf{S}_j^{-1} \boldsymbol{\nu}_j$, as utilized in Eq. (2.14). As discussed in Section 2.2.3, this value provides an indication of the match between the measurement and the value predicted by the model, hence if this value is large (above some threshold), then the model can be assumed to have diverged and should be reinitialized using the combined estimate from the non-divergent filters. An alternative trigger which could be used for reinitializing models is to restart them whenever the lower probability bound is applied [57]. In this way, elemental filters which are not contributing to the overall estimate are continually reset such that they are ready when the respective model comes into force.

2.4.2 Switching Models. To allow for model switching, the model in force is permitted to change at any sample instant, and *model history* events are used to characterize the transitional behavior of the system with time. Such events take the form:³

$$M^{k,l} = \left\{ M_{1,m_{1l}}, M_{2,m_{2l}}, \dots, M_{k,m_{kl}} \right\} \quad (2.36)$$

The notation is interpreted as meaning that the l -th possible model history at time k consists of model m_{1l} at sample time 1, model m_{2l} at sample time 2, etc., where each m_{jl} is the index to a model number between 1 and N_f .

If transitions are allowed to any of the N_f models at any sample instant, then every model history event at time k will give rise to N_f new events at time $(k + 1)$, hence the number of possible model histories increases exponentially with time according to N_f^k . The PDF of the target state conditioned on the measurements

³Note that superscripts are used to indicate a model *history* event, whereas subscripts indicate a single time step event.

must then be calculated as a total probability expansion over all model history events:

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{l=1}^{N_f^k} f\{\mathbf{x}(k)|M^{k,l}, \mathbf{Z}^k\} P\{M^{k,l}|\mathbf{Z}^k\} \quad (2.37)$$

The model history probability $P\{M^{k,l}|\mathbf{Z}^k\}$ is expanded as:

$$\begin{aligned} P\{M^{k,l}|\mathbf{Z}^k\} &= P\{M^{k,l}|\mathbf{z}(k), \mathbf{Z}^{k-1}\} \\ &= \frac{f\{M^{k,l}, \mathbf{z}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\} P\{M^{k,l}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\} P\{M_{k,j}, M^{k-1,l'}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\} P\{M_{k,j}|M^{k-1,l'}, \mathbf{Z}^{k-1}\} P\{M^{k-1,l'}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \end{aligned} \quad (2.38)$$

where l is the index of the current model history (between 1 and N_f^k), l' is the index of the previous model history (between 1 and N_f^{k-1}), and j is the index of the current model (between 1 and N_f) hypothesized by the model history event $M^{k,l}$. The denominator is expanded as a total probability expansion over all model history events as in Eq. (2.37).

The method commonly used to evaluate model history event probabilities is to assume that the model transition process is a Markov process, such that the probability of transition depends *only* on the previous model number $m_{k-1,l'}$, and

not on the prior model history or prior measurements:

$$\begin{aligned}
P\{M_{k,j}|M^{k-1,l'}, \mathbf{Z}^{k-1}\} &= P\{M_{k,j}|M^{k-1,l'}\} \\
&= P\{M_{k,j}|M_{1,m_{1,l'}}, M_{2,m_{2,l'}}, \dots, M_{k-1,m_{k-1,l'}}\} \\
&= P\{M_{k,j}|M_{k-1,m_{k-1,l'}}\} \\
&\triangleq p_{m_{k-1,l'},j}(k)
\end{aligned} \tag{2.39}$$

Thus $p_{m_{k-1,l'},j}(k)$ is the probability of transitioning from model index $m_{k-1,l'}$ at sample time $(k-1)$ to model index j at sample time k , where each index is a model number between 1 and N_f .

While the assumption of Eq. (2.39) provides a mechanism for computation of the model history probability, the conditioning in Eq. (2.38) of the new measurement probability on the model history still produces an exponentially increasing number of hypotheses with time, hence further approximation (such as combining branches) is required. The most commonly used algorithms are described in the following sections. The structure of the full order Bayesian switching estimator is shown in Figure 2.3. The diagram demonstrates the growing number of filters which is required: the output of every filter at the current processing cycle must be processed in the following processing cycle using every model, hence the number of filtering operations at the k -th cycle is N_f^k .

2.4.3 First-Order Generalized Pseudo-Bayesian Estimator. The First-Order Generalized Pseudo-Bayesian (GPB-1) estimator [3:454–456] limits the memory of the model history events by combining the estimates from all models into a single estimate at the end of each processing cycle. At the start of each processing cycle, the information carried forward from the previous measurement interval is a single combined estimate: any conditioning on previous model history events has been discarded. Hence the PDF of the estimate is modified from the switching model

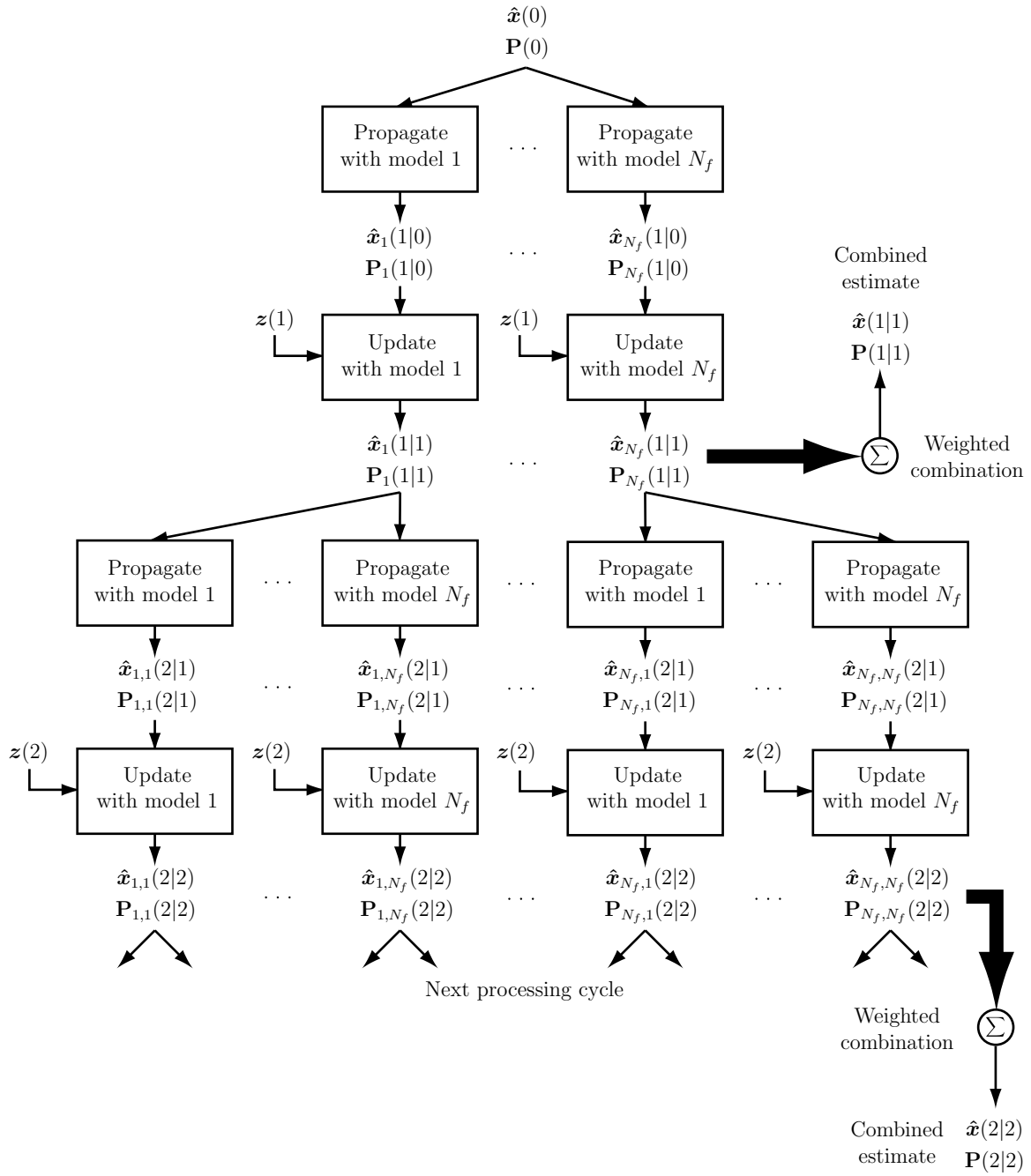


Figure 2.3. Block diagram of full order Markov switching estimator.

of Eq. (2.37):

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{l=1}^{N_f^k} f\{\mathbf{x}(k)|M^{k,l}, \mathbf{Z}^k\} P\{M^{k,l}|\mathbf{Z}^k\}$$

to the simplified version:

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\} \quad (2.40)$$

where the total probability expansion over the entire model history event $M^{k,l}$ is replaced by expansion over the single most recent model event $M_{k,j}$. Expanding Eq. (2.40), we further approximate that the previous measurement history $\mathbf{Z}^{k-1}$ is adequately represented by the *single* estimate and covariance from the previous processing cycle, $\{\hat{\mathbf{x}}(k-1|k-1), \mathbf{P}(k-1|k-1)\}$:

$$\begin{aligned} f\{\mathbf{x}(k)|\mathbf{Z}^k\} &= \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\} \\ &= \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{z}(k), \mathbf{Z}^{k-1}\} P\{M_{k,j}|\mathbf{Z}^k\} \\ &= \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{z}(k), \hat{\mathbf{x}}(k-1|k-1), \mathbf{P}(k-1|k-1)\} \cdot \\ &\quad \cdot P\{M_{k,j}|\mathbf{Z}^k\} \end{aligned} \quad (2.41)$$

In effect, the approximations of Eq. (2.40) and (2.41) mean to say that the entire model transition history and measurement history are representable through the single estimate from the previous processing cycle. Once the conditional model probability in Eq. (2.41) has been evaluated using the developments of Eq. (2.38) and (2.39), the combined estimate is then calculated as per Eqs. (2.34) and (2.35) and the cycle repeats.

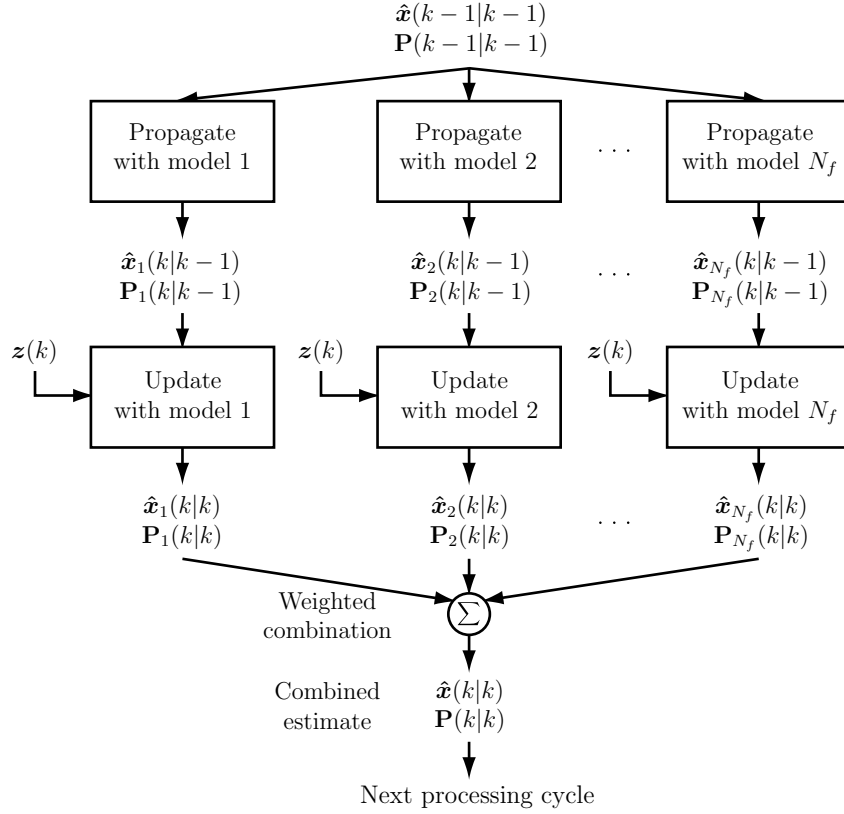


Figure 2.4. Block diagram of GPB-1 algorithm.

The structure of the GPB-1 algorithm is shown in Figure 2.4. The outputs of all filters are merged into a single estimate at each processing cycle, which is used as the input to each of the filters at the next processing cycle, providing a very coarse approximation of the optimal system shown in Figure 2.3.

2.4.4 Second-Order Generalized Pseudo-Bayesian Estimator. The Second-Order Generalized Pseudo-Bayesian (GPB-2) estimator [3:457–460] operates on similar principles to the first-order variant, except that the memory is allowed to extend an additional processing cycle. Again, the PDF of the estimate is modified from the full order switching model of Eq. (2.37):

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{l=1}^{N_f^k} f\{\mathbf{x}(k)|M^{k,l}, \mathbf{Z}^k\} P\{M^{k,l}|\mathbf{Z}^k\}$$

to the simplified version, which this time incorporates the *previous* model $M_{k-1,i}$ in addition to the current model $M_{k,j}$:

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k-1,i}, M_{k,j}, \mathbf{Z}^k\} P\{M_{k-1,i}, M_{k,j}|\mathbf{Z}^k\} \quad (2.42)$$

Manipulating Eq. (2.42) and assuming that the history $\{M_{k-1,i}, \mathbf{Z}^{k-1}\}$ is adequately represented by the combined estimates from the i -th model in the previous processing cycle $\{\hat{\mathbf{x}}_i(k-1|k-1), \mathbf{P}_i(k-1|k-1)\}$, and that (according to the Markov model) the model transition depends only on the previous model, and not on the measurement history:

$$\begin{aligned} f\{\mathbf{x}(k)|\mathbf{Z}^k\} &= \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{z}(k), \{M_{k-1,i}, \mathbf{Z}^{k-1}\}\} \cdot \\ &\quad \cdot P\{M_{k,j}|M_{k-1,i}, \mathbf{Z}^k\} P\{M_{k-1,i}|\mathbf{Z}^k\} \\ &= \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{z}(k), \hat{\mathbf{x}}_i(k-1|k-1), \mathbf{P}_i(k-1|k-1)\} \cdot \\ &\quad \cdot P\{M_{k,j}|M_{k-1,i}\} P\{M_{k-1,i}|\mathbf{Z}^k\} \end{aligned} \quad (2.43)$$

Hence the operation of the GPB-2 algorithm is such that the estimate from each model in the previous processing cycle is processed using each dynamics model, giving N_f^2 total elemental filters. At the end of each processing cycle, the N_f^2 estimates are combined down to N_f estimates, combining estimates from different models in the *previous* processing cycle to leave *one estimate* for each model in the *latest* processing cycle. This is illustrated in Figure 2.5, which shows the structure of the algorithm. Comparing the structure to the GPB-1 algorithm shown in Figure 2.4, the GPB-2 algorithm uses N_f^2 filters, thus it is able to maintain N_f estimates and propagate each estimate with each of the N_f filters at each processing interval, rather than collapsing the PDF of target state down to a single estimate at each processing interval.

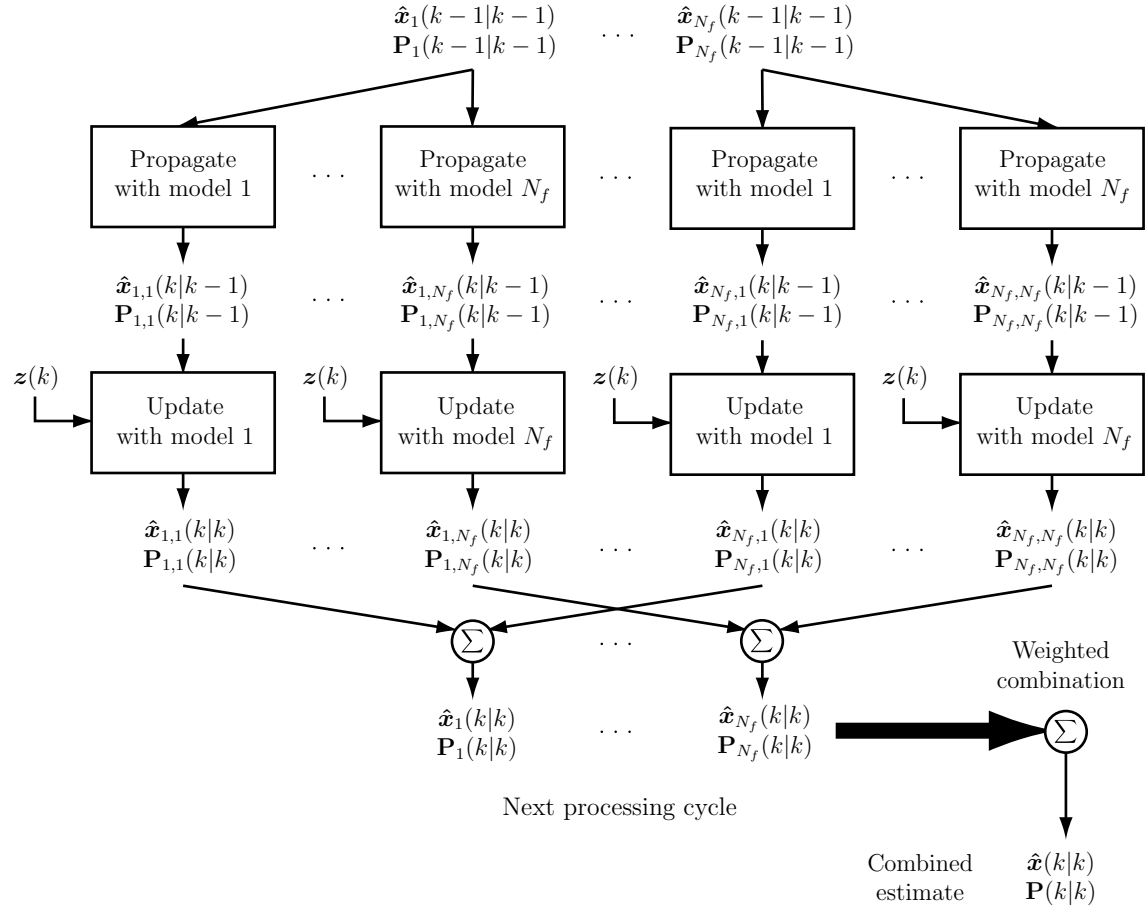


Figure 2.5. Block diagram of GPB-2 algorithm.

2.4.5 *Interacting Multiple Model Estimator.* The Interacting Multiple Model (IMM) estimator [3:461–465, 10] is a methodology which achieves comparable performance to the GPB-2 estimator using only N_f elemental filters, rather than N_f^2 as required by the latter. The algorithm can be derived by considering the limitations inherent to the problem: if only N_f elemental filters are allowable, then the input to the j -th filter should be the best estimate of the state at time instant $(k-1)$, conditioned on the event that model j is in force *at time instant k* (the new sample time), $f\{\mathbf{x}(k-1)|M_{k,j}, \mathbf{Z}^{k-1}\}$. Using this expression as a starting point, we follow a single iteration of the algorithm, through to the calculation of the same function at the following sample period.

Following a standard Kalman filter propagate-update cycle at the k -th sample time, the output of the j -th elemental filter will be $f\{\mathbf{x}(k)|M_{k,j}, \mathbf{Z}^k\}$. The requirement for the IMM algorithm is thus to combine the estimates from the N_f elemental filters to calculate the inputs $f\{\mathbf{x}(k)|M_{k+1,i}, \mathbf{Z}^k\}$ of each elemental filter for the next processing cycle.

The overall PDF formed using the information from all N_f filters represents the total information contained by the system at time k :

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\} \quad (2.44)$$

The goal of the intermixing is thus to massage Eq. (2.44) into the expansion necessary at the input to the next processing cycle:

$$f\{\mathbf{x}(k)|\mathbf{Z}^k\} = \sum_{i=1}^{N_f} f\{\mathbf{x}(k)|M_{k+1,i}, \mathbf{Z}^k\} P\{M_{k+1,i}|\mathbf{Z}^k\} \quad (2.45)$$

The last factor in Eq. (2.45) is easily evaluated using the Markov assumption as per Eq. (2.39):

$$\begin{aligned}
P\{M_{k+1,i}|\mathbf{Z}^k\} &= \sum_{j=1}^{N_f} P\{M_{k+1,i}|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\} \\
&= \sum_{j=1}^{N_f} P\{M_{k+1,i}|M_{k,j}\} P\{M_{k,j}|\mathbf{Z}^k\}
\end{aligned} \tag{2.46}$$

Note that if $\mathbf{T}(k+1|k)$ is the Markov transition matrix such that:

$$\{\mathbf{T}\}_{ij} = P\{M_{k+1,i}|M_{k,j}\}$$

then Eq. (2.46) is simply a matrix multiplication of $\mathbf{T}(k+1|k)$ by the vector with elements that are the probabilities $P\{M_{k,j}|\mathbf{Z}^k\}$, yielding the vector of components that represent the probabilities $P\{M_{k+1,i}|\mathbf{Z}^k\}$.

The leading factor in the sum of Eq. (2.45) is then expanded using the total probability theorem over the previous model index j :

$$f\{\mathbf{x}(k)|M_{k+1,i}, \mathbf{Z}^k\} = \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k+1,i}, M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|M_{k+1,i}, \mathbf{Z}^k\} \tag{2.47}$$

where the *backward* transition probabilities are calculated by:

$$\begin{aligned}
P\{M_{k,j}|M_{k+1,i}, \mathbf{Z}^k\} &= \frac{P\{M_{k,j}, M_{k+1,i}|\mathbf{Z}^k\}}{P\{M_{k+1,i}|\mathbf{Z}^k\}} \\
&= \frac{P\{M_{k+1,i}|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\}}{P\{M_{k+1,i}|\mathbf{Z}^k\}} \\
&= \frac{P\{M_{k+1,i}|M_{k,j}, \mathbf{Z}^k\} P\{M_{k,j}|\mathbf{Z}^k\}}{\sum_{n=1}^{N_f} P\{M_{k+1,i}|M_{k,n}, \mathbf{Z}^k\} P\{M_{k,n}|\mathbf{Z}^k\}}
\end{aligned} \tag{2.48}$$

According to the Markov assumption, the transition probability $P\{M_{k+1,i}|M_{k,j}, \mathbf{Z}^k\}$ does not depend on the measurement history $\mathbf{Z}^k$, hence this conditioning is dropped.

Assuming that the estimator history $\mathbf{Z}^k$ is adequately modelled by the N_f estimates from the previous processing cycle (each estimate conditioned on a different model $M_{k,j}$), Eq. (2.47) is then approximated by a single Gaussian density:⁴

$$\begin{aligned} f\{\mathbf{x}(k)|M_{k+1,i}, \mathbf{Z}^k\} &\approx \sum_{j=1}^{N_f} f\{\mathbf{x}(k)|M_{k+1,i}, \hat{\mathbf{x}}_j(k|k), \mathbf{P}_j(k|k)\} P\{M_{k,j}|M_{k+1,i}, \mathbf{Z}^k\} \\ &\approx \mathcal{N}\{\mathbf{x}(k); \hat{\mathbf{x}}^i(k|k), \mathbf{P}^i(k|k)\} \end{aligned} \quad (2.49)$$

where the mean and variance of the Gaussian are given by:

$$\begin{aligned} \hat{\mathbf{x}}^i(k|k) &= \sum_{j=1}^{N_f} P\{M_{k,j}|M_{k+1,i}, \mathbf{Z}^k\} \hat{\mathbf{x}}_j(k|k) \\ \mathbf{P}^i(k|k) &= \sum_{j=1}^{N_f} P\{M_{k,j}|M_{k+1,i}, \mathbf{Z}^k\} \{\mathbf{P}_j(k|k) + \\ &\quad + [\hat{\mathbf{x}}_j(k|k) - \hat{\mathbf{x}}^i(k|k)][\hat{\mathbf{x}}_j(k|k) - \hat{\mathbf{x}}^i(k|k)]^T\} \end{aligned} \quad (2.50)$$

The *a posteriori* model probabilities $P\{M_{k,j}|\mathbf{Z}^k\}$ required for Eq. (2.48) are calculated recursively using the expressions:

$$\begin{aligned} P\{M_{k,j}|\mathbf{Z}^k\} &= P\{M_{k,j}|\mathbf{z}(k), \mathbf{Z}^{k-1}\} \\ &= \frac{f\{M_{k,j}, \mathbf{z}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\ &= \frac{f\{\mathbf{z}(k)|M_{k,j}, \mathbf{Z}^{k-1}\} P\{M_{k,j}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \end{aligned} \quad (2.51)$$

⁴Note that $\{\hat{\mathbf{x}}_j(k|k), \mathbf{P}_j(k|k)\}$ is taken to refer to the filter estimate at the output of the previous processing cycle, while $\{\hat{\mathbf{x}}^i(k|k), \mathbf{P}^i(k|k)\}$ represents the *mixed* estimates to be provided at the input to the next processing cycle.

As discussed in Eq. (2.46), $P\{M_{k,j}|\mathbf{Z}^{k-1}\}$ can be expanded using the total probability theorem as:

$$\begin{aligned} P\{M_{k,j}|\mathbf{Z}^{k-1}\} &= \sum_{i=1}^{N_f} P\{M_{k,j}|M_{k-1,i}, \mathbf{Z}^{k-1}\} P\{M_{k-1,i}|\mathbf{Z}^{k-1}\} \\ &= \sum_{i=1}^{N_f} P\{M_{k,j}|M_{k-1,i}\} P\{M_{k-1,i}|\mathbf{Z}^{k-1}\} \end{aligned} \quad (2.52)$$

where the assumption that the model transition probability does not depend on the measurement history is again invoked.

Thus by substituting in Eq. (2.52) and expanding the denominator using the total probability theorem, Eq. (2.51) becomes:

$$P\{M_{k,j}|\mathbf{Z}^k\} = \frac{f\{z(k)|M_{k,j}, \mathbf{Z}^{k-1}\} \sum_{i=1}^{N_f} P\{M_{k,j}|M_{k-1,i}\} P\{M_{k-1,i}|\mathbf{Z}^{k-1}\}}{\sum_{n=1}^{N_f} f\{z(k)|M_{k,n}, \mathbf{Z}^{k-1}\} P\{M_{k,n}|\mathbf{Z}^{k-1}\}} \quad (2.53)$$

(Note that the denominator is simply the scaling factor necessary to ensure that the conditional model probabilities sum to unity.)

As per the preceding multiple model techniques, the combined estimate is calculated at each processing cycle to give the output of the estimator. A block diagram of the IMM algorithm is shown in Figure 2.6. The structure is very similar to the non-switching MMAE structure shown in Figure 2.1: there are N_f filters, each of which is supplied with a different input. However, rather than passing the output of each filter directly into the same filter at the next processing cycle, the algorithm mixes the estimates according to the Markov transition model in order to allow the system to react to changes to the model in force.

2.4.6 Summary. The previous sections have presented the commonly used multiple model estimation structures. The traditional MMAE is based on the as-

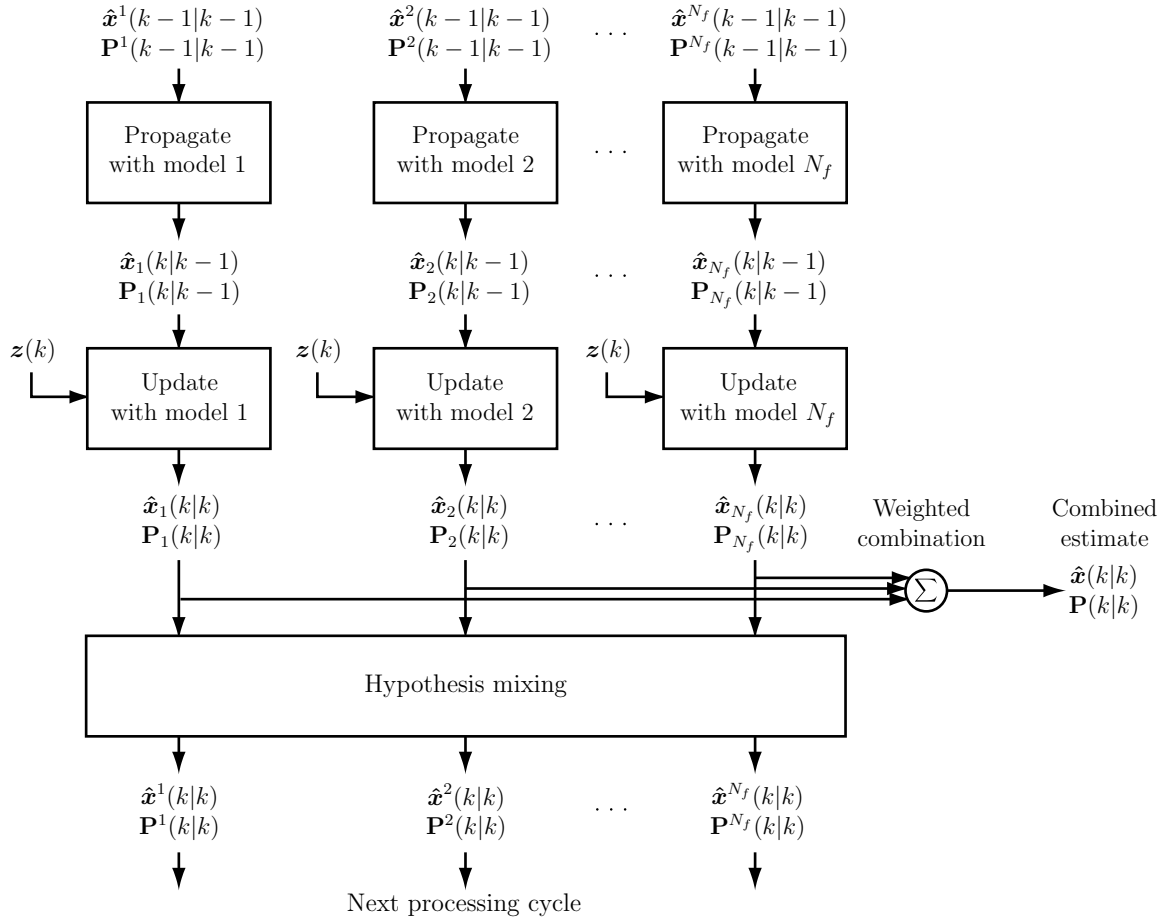


Figure 2.6. Block diagram of IMM algorithm.

sumption that the model in force does not change with time; *ad hoc* modifications extend the algorithm to provide adequate performance in a switching model environment. The switching model estimators all utilize a Markov model for transition probabilities; the most commonly used algorithm is the IMM, which provides similar performance to the GPB-2 at a fraction of the computational cost.

2.5 Data Association

Surveillance radar systems⁵ typically operate by steering the radar beam in a repetitive scan pattern, such as a circular scan (in which the radar antenna is rotated around 360° in the horizontal plane at a constant rate), a sector scan (in which the antenna is moved forwards and backwards across a fixed horizontal arc) or a two-dimensional raster scan (effectively a number of sector “scan bars”, separated in the vertical plane). At the end of each scan interval, a series of radar detections will have been made, which indicate the possible presence of a target at a particular location. The data supplied with each measurement may include angle (azimuth and/or elevation), range and Doppler shift, each of which will be corrupted by noise.

At the same time, the radar is maintaining a track file, containing a listing of known targets alongside state information such as location, velocity and acceleration, and possibly identification information. Fundamentally, the role of the data association algorithm is to determine how to update the existing tracks using the incoming block of measurements. The difficulty is that the measurements are not labelled: the radar system does not know to which target the measurements belong, or whether they belong to a target at all (i.e., they may be false detections, such as those caused by radar clutter). This is illustrated in Figure 2.7: the solution of how to update the target states (as illustrated by the solid dots) for the given measurements (illustrated by the plus marks) is neither obvious nor simple.

⁵Especially mechanically scanned radar systems.

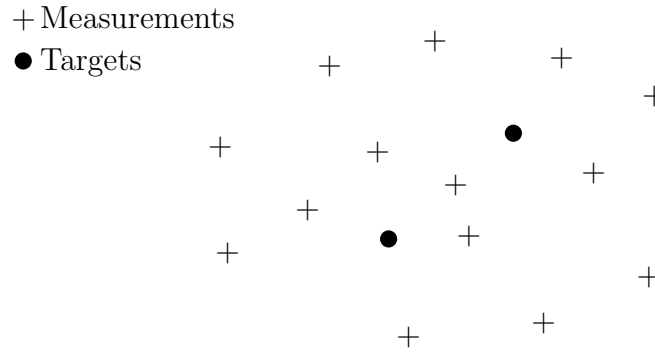


Figure 2.7. The data association problem: how to update target state given a series of unlabelled measurements.

The performance of the radar system as a whole is impacted greatly by the data association algorithm. The data association algorithm handles tasks from initial track forming (when targets are first detected), to track update (maintaining accurate state estimates while targets are under track), and finally track deletion. The characteristics of the data association algorithm are able to change the ability of the system to reject false measurements, the accuracy of the track maintained by the system, and the likelihood of loss of track (when the estimate deviates unrecoverably from the actual target position, or the system incorrectly declares that the target no longer exists). Maintaining continuity of tracks is a high priority for radar systems, as this provides a much clearer view to the radar operator and systems that use the radar output, and it helps to keep the link between target position data and identification information.

The following sections derive the probabilistic model utilized in almost every modern tracking algorithm, and then they describe the different approximations applied by the various techniques. While the initial descriptions of each of the conventional techniques lead to terribly inefficient implementations, they are in fact algebraically equivalent to the more efficient implementations presented in the various references given. Although manner of presentation is unlike any reference of which the author is aware, it is our opinion that it leads to a clearer understanding

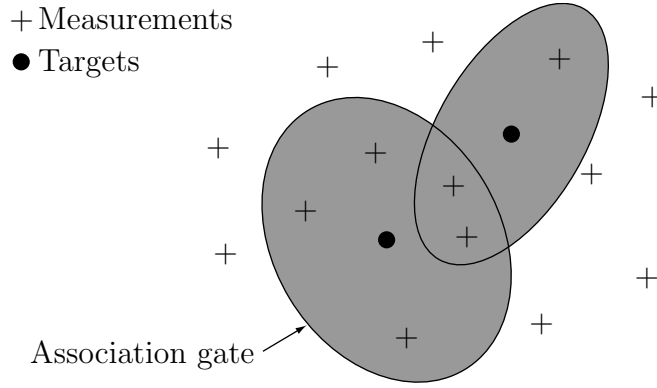


Figure 2.8. Measurement gating in a multiple target environment.

of the approximations inherent to the various techniques used in modern tracking systems, and the consequent strengths and weaknesses of the algorithms.

2.5.1 Measurement Gating. Measurement gating is a technique used in virtually all data association algorithms to avoid the computation time of processing association possibilities which are kinematically impossible or statistically improbable. The concept of measurement gating is that, if a measurement is not within some predefined distance of a track (i.e., within the track's association gate), then that measurement-track pairing is extremely unlikely to be correct, and thus it is not considered for association. This is illustrated in Figure 2.8: only those measurements within the shaded region around each target are processed. In order to have the data association algorithm consider all plausible assignment options, the association gate is generally selected to be quite large, typically designed to incorporate *at least* 98% of the hypervolume under the PDF of the predicted location. This hypervolume is the probability that the target-originated measurement will fall within the gate, and hence it is denoted as P_g . Throughout this document the terms *measurement gate* and *association gate* will be used interchangeably.

The calculations for measurement gating are performed using the expression of Eq. (2.14). If $\hat{z}_i(k|k-1)$ is the predicted location of the measurement belonging to

target i , and $\mathbf{S}_i(k)$ is the covariance of the residual formed using the measurement belonging to target i , then the j -th measurement will be considered for association with target i if:⁶

$$[\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)]^T \mathbf{S}_i(k)^{-1} [\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)] \leq \gamma \quad (2.54)$$

where γ is the threshold calculated from the desired value of the probability that the correct measurement is in the gate (P_g) using χ^2 tables as described in [4:95–96].

Measurement gating also provides a mechanism to break the data association problem up into manageable portions, or clusters. A cluster contains all targets which have common measurements within their association gates. For example, if there are four targets, and targets 1 and 2 share a measurement (as illustrated by the measurements contained within both association gates in Figure 2.8), targets 2 and 3 share a measurement and target 4 does not share any measurements, then targets 1, 2 and 3 will be in one cluster and target 4 will be in its own cluster. When targets do not share measurements, the separate clusters may be processed as independent tracking problems, greatly reducing the number of joint hypotheses, as introduced below.

2.5.2 Association Event Probability. The basis of each of the data association algorithms discussed in this chapter is the probabilistic model for joint association events, such as that described in [2, 4, 6, 7]. The model is derived in detail in the next pages, followed by descriptions of the approximations utilized by the various conventional tracking algorithms.

We use the notation $\Theta_l(k)$ to denote the l -th joint association event at sample period k . Each joint association event represents a hypothesis on the origin of each

⁶i.e., the j -th measurement is inside target i 's association gate.

measurement.⁷ For example, a typical joint hypothesis might be:

$$\Theta_l(k) = \{\theta_{12}, \theta_{21}, \theta_{34}, \theta_{40}\}$$

where the elemental event θ_{ji} represents the association of measurement j with target i , and θ_{j0} represents the event that no target is associated with measurement j , indicating that measurement j is clutter-originated. In the example above, measurement 1 has been associated with target 2, measurement 2 with target 1, measurement 3 with target 4, and measurement 4 is the result of clutter. When combined with the knowledge of the number of targets present at time k , knowledge of which targets have and have not been detected is implicitly contained in the event; in the example above targets 1, 2 and 4 were detected, thus if there were four targets under track at time k then target 3 was missed.

The requirements placed on joint association events provide a mechanism to embed physically meaningful stipulations into the probabilistic model. The most common requirements used are that each measurement can be associated with no more than one target, and each target can be associated with no more than one measurement. These requirements overlook possibilities in which two targets are within the same radar resolution cell and produce a single merged measurement, and possibilities in which a target is close enough that it occupies multiple radar resolution cells and produces multiple measurements. However, they also preclude associations which are obviously invalid, such as two broadly spaced targets giving rise to a single common measurement, or a single target giving rise to two broadly spaced measurements, and where necessary the previously discussed possibilities may be handled as exceptions.

⁷Only those measurements inside the union of the measurement gates of each target in the cluster are considered.

Following the development of [4:314–317], probability of a joint association event can be evaluated using two successive applications of Bayes' rule:

$$\begin{aligned}
P\{\Theta_l(k)|\mathbf{Z}^k\} &= P\{\Theta_l(k)|\mathbf{Z}_k, N_m(k), \mathbf{Z}^{k-1}\} \\
&= \frac{f\{\mathbf{Z}_k, \Theta_l(k)|N_m(k), \mathbf{Z}^{k-1}\}}{f\{\mathbf{Z}_k|N_m(k), \mathbf{Z}^{k-1}\}} \\
&= \frac{f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\}P\{\Theta_l(k)|N_m(k), \mathbf{Z}^{k-1}\}}{f\{\mathbf{Z}_k|N_m(k), \mathbf{Z}^{k-1}\}} \quad (2.55)
\end{aligned}$$

where $N_m(k)$ is the number of measurements in the combined gating region at scan k , which is inherent in the knowledge of the measurements themselves.

The leading term in the numerator of Eq. (2.55) amounts to the *a priori* likelihood of the measurements received in scan k , conditioned on the past measurements ($\mathbf{Z}^{k-1}$), the number of measurements in the current cycle ($N_m(k)$) and the joint association event ($\Theta_l(k)$). The notation of the capital $\mathbf{Z}_k$ is used to represent the *joint* state of all measurements, rather than the marginal PDF of a single measurement. If the j -th measurement is hypothesized to be clutter-originated (such that $\theta_{j0} \in \Theta_l$), then its PDF is modelled as uniform within the combined measurement gate. Denoting $\mathbb{V}(k)$ as the union of the measurement gating regions of all targets in the cluster, and $V(k)$ as the volume of this region, the individual measurement PDFs of clutter-originated measurements can be evaluated as:⁸

$$f\{\mathbf{z}_j(k)|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} = \begin{cases} V^{-1} & : \quad \mathbf{z}_j(k) \in \mathbb{V}(k) \\ 0 & : \quad \mathbf{z}_j(k) \notin \mathbb{V}(k) \end{cases} \quad (2.56)$$

The evaluation of the components of this PDF which are target-originated is discussed in Section 2.5.4.

⁸Note that the measurement is guaranteed to be within the combined association region by the prior application of gating, hence the second case is defunct.

The second term in the numerator of Eq. (2.55), $P\{\Theta_l(k)|N_m(k), \mathbf{Z}^{k-1}\}$, is the probability of the joint association event $\Theta_l(k)$ for the *current* scan, conditioned only on the number of measurements in the association gate ($N_m(k)$) and the measurement history *prior* to the current sample period. In the absence of any information about the value of the current measurements, the prior measurement history is assumed to contain no information about the current association event such that:

$$P\{\Theta_l(k)|N_m(k), \mathbf{Z}^{k-1}\} = P\{\Theta_l(k)|N_m(k)\}$$

This prior event probability is evaluated by considering the target detections, missed detections and clutter measurements hypothesized in the event $\Theta_l(k)$. Denoting $\boldsymbol{\delta}(\Theta_l)$ as the vector of target detection indicators,⁹ and $\phi(\Theta_l)$ as the number of measurements originating from clutter,¹⁰ both of which are intrinsic in the knowledge of the association event $\Theta_l(k)$:

$$\begin{aligned} P\{\Theta_l(k)|N_m(k)\} &= P\{\Theta_l(k), \boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)|N_m(k)\} \\ &= P\{\Theta_l(k)|\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l), N_m(k)\}P\{\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)|N_m(k)\} \end{aligned} \quad (2.57)$$

The first term in Eq. (2.57) is evaluated by assuming that all joint association events that contain the same set of detected targets and the same number of clutter measurements are equally likely. The count of such events is the number of permutations possible when selecting $\psi(\Theta_l) = \sum_i \delta_i(\Theta_l) = N_m(k) - \phi(\Theta_l)$ (the number of detected targets) measurements out of $N_m(k)$ (the total number of measurements). This is the classic “balls out of an urn” problem without replacement and considering

⁹i.e., the i -th element of $\boldsymbol{\delta}$ is ‘1’ if target i is hypothesized as being detected in event $\Theta_l(k)$, or ‘0’ if target i is hypothesized to have been missed in event $\Theta_l(k)$.

¹⁰i.e., the total number of measurements minus the number of targets hypothesized as having been detected.

order, and is evaluated as [32:44]:

$$P_{\psi(\Theta_l)}^{N_m(k)} = \frac{N_m(k)!}{[N_m(k) - \psi(\Theta_l)]!} = \frac{N_m(k)!}{\phi(\Theta_l)!}$$

Subsequently the probability of each equally likely event is:

$$P\{\Theta_l(k)|\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l), N_m(k)\} = \left(\frac{N_m(k)!}{\phi(\Theta_l)!}\right)^{-1} = \frac{\phi(\Theta_l)!}{N_m(k)!}$$

In the traditional development of the algorithm, as presented in [2:226] and [4:315], the second term in Eq. (2.57) is evaluated by assuming independence among $\boldsymbol{\delta}(\Theta_l)$ and $\phi(\Theta_l)$ such that:

$$P\{\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)|N_m(k)\} = P\{\boldsymbol{\delta}(\Theta_l)\}P\{\phi(\Theta_l)\} \quad (2.58)$$

Strictly, this assumption of independence is invalid when conditioned on the number of measurements $N_m(k)$, because once given the target detection vector $\boldsymbol{\delta}(\Theta_l)$ and the number of measurements $N_m(k)$, one implicitly knows $\phi(\Theta_l)$ by the relationship:

$$\phi(\Theta_l) = N_m(k) - \sum_i \delta_i(\Theta_l)$$

However, one can arrive at essentially the same result by applying Bayes' rule twice to remove the conditioning, resulting in:

$$\begin{aligned} P\{\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)|N_m(k)\} &= \frac{P\{\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l), N_m(k)\}}{P\{N_m(k)\}} \\ &= \frac{P\{N_m(k)|\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)\}P\{\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)\}}{P\{N_m(k)\}} \\ &= \frac{P\{\boldsymbol{\delta}(\Theta_l)\}P\{\phi(\Theta_l)\}}{P\{N_m(k)\}} \end{aligned} \quad (2.59)$$

where $P\{N_m(k)|\boldsymbol{\delta}(\Theta_l), \phi(\Theta_l)\}$ is cancelled in the final step as it will evaluate to unity for any consistent association event, and the independence of $\boldsymbol{\delta}(\Theta_l)$ and $\phi(\Theta_l)$ is

assumed this time *without conditioning on* $N_m(k)$. The denominator term $P\{N_m(k)\}$ may be evaluated through the total probability expansion:

$$\begin{aligned} P\{N_m(k)\} &= \sum P\{N_m(k)|\boldsymbol{\delta}, \phi\} P\{\boldsymbol{\delta}, \phi\} \\ &= \sum P\{\boldsymbol{\delta}\} P\{\phi\} \end{aligned}$$

where the sum is over all possible $\{\boldsymbol{\delta}, \phi\}$ such that $\phi + \sum_i \boldsymbol{\delta}_i = N_m(k)$. However, since the denominator is identical for all association events, the term will contribute a constant scaling factor to all terms, which will be cancelled when the association events are normalized to sum to unity.

The *a priori* probability of the target detection vector $P\{\boldsymbol{\delta}(\Theta_l)\}$ is evaluated by assuming independence between each of the target detection possibilities; e.g., if the target detection vector proposes that ψ of the N_t targets were detected, then:

$$P\{\boldsymbol{\delta}\} = P_{dg}^\psi (1 - P_{dg})^{N_t - \psi} \quad (2.60)$$

where P_{dg} is the probability that any one of the N_t targets will be detected, and that the resulting measurement is within the association gate. If P_d is the target detection probability and P_g is the probability that the target-oriented measurement is within the association gate, then:

$$P_{dg} = P_d P_g$$

and thus:

$$P\{\boldsymbol{\delta}\} = (P_d P_g)^\psi (1 - P_d P_g)^{N_t - \psi} \quad (2.61)$$

The *a priori* probability of the number of clutter measurements $P\{\phi(\Theta_l)\}$ is evaluated utilizing a Poisson model [4:135] with parameter λV , where λ represents the density of false measurements within the validation region (i.e., the expected

number of clutter detections per unit hypervolume in measurement space), and V is the hypervolume of the combined gating region for all targets:

$$P\{\phi\} = \frac{(\lambda V)^\phi}{\phi!} e^{-\lambda V}$$

Collecting terms from Eq. (2.55) we arrive at the following expression: (time arguments are omitted where unambiguous)

$$\begin{aligned} P\{\Theta_l(k)|\mathbf{Z}^k\} &= \frac{f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}P\{\Theta_l|\boldsymbol{\delta}, \phi, N_m\}P\{\boldsymbol{\delta}\}P\{\phi\}}{f\{\mathbf{Z}_k|N_m, \mathbf{Z}^{k-1}\}P\{N_m\}} \\ &= \frac{1}{c}f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}\frac{\phi!}{N_m!} \cdot \frac{(\lambda V)^\phi}{\phi!} e^{-\lambda V} P_{dg}^\psi (1 - P_{dg})^{N_t - \psi} \\ &= \frac{1}{c'}f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}(\lambda V)^\phi P_{dg}^\psi (1 - P_{dg})^{N_t - \psi} \end{aligned} \quad (2.62)$$

where c is the denominator of the first expression in Eq. (2.62), evaluated as the sum of all numerators using the total probability expansion and the simplifications of Eqs. (2.57)–(2.59):

$$\begin{aligned} c &= f\{\mathbf{Z}_k|N_m(k), \mathbf{Z}^{k-1}\}P\{N_m(k)\} \\ &= \sum_l f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}P\{\Theta_l|N_m, \mathbf{Z}^{k-1}\}P\{N_m\} \\ &= \sum_l f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}P\{\Theta_l|N_m\}P\{N_m\} \\ &= \sum_l f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}P\{\Theta_l|\boldsymbol{\delta}, \phi, N_m\} \frac{P\{\boldsymbol{\delta}\}P\{\phi\}}{P\{N_m\}} P\{N_m\} \\ &= \sum_l f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}P\{\Theta_l|\boldsymbol{\delta}, \phi, N_m\}P\{\boldsymbol{\delta}\}P\{\phi\} \end{aligned} \quad (2.63)$$

As the above expression (after applying the summation) is the same for all association events, the term c merely functions as a normalization constant, ensuring that the probabilities of all joint association events sum to unity. The constant is modified to c' after the incorporation of the Poisson exponential term $e^{-\lambda V}$ and the factorial

of the number of measurements $N_m!$:

$$c' = \frac{cN_m!}{e^{-\lambda V}}$$

The volume of the combined validation region for the targets, $V(k)$, is not easily calculated. However, considering that Eq. (2.62) contains a V^ϕ term (arising from the Poisson model for clutter), and that $f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\}$ will include a V^{-1} term for each measurement believed to be the result of clutter (as discussed in Eq. (2.56)), of which there are ϕ , the $V(k)$ terms will cancel, leaving only terms involving the clutter density λ .

Eq. (2.62) can be easily modified to admit the case of different detection probabilities for each target (as is motivated physically by the variation of detection probability with radar return power level and signal to noise ratio). However, the simplification should be a fair approximation when considering targets of similar physical size (and radar cross section) within a relatively small cluster.

2.5.3 Forming Joint Hypotheses. As will be described in Section 2.5.5, to update the estimate of the target state, all joint association events must be formed. If there is only a single target in the cluster, then this is a simple task, and the association events are that the target is associated with each measurement in the gate, or that the target is not associated with any measurement (i.e., it is hypothesized to have been missed).

If there are two targets in a cluster, then the number of joint association hypotheses is roughly squared compared to the single target case. If the results of the measurement gate tests are stored in the “**valid**” matrix such that the (i, j) entry is a binary flag indicating whether measurement j is inside the association region for target i , then the pseudocode in Figure 2.9 will form all joint events.

```

% Loop for first target -- associate measurement zero (missed
% detection), then each of the measurements
for m1 = 0 to numMeas do

    % Check that measurement m1 is inside association gate for
    % target 1
    if m1 = 0 or valid(1,m1) then

        % Loop for second target - associate measurement zero (missed
        % detection), then each of the measurements
        for m2 = 0 to numMeas do

            % Check that measurement m2 is inside association gate for
            % target 2 and that measurement m2 is not associated with
            % target 1
            if m2 = 0 or (valid(2,m2) and m1 != m2) then

                % Create a new association event
                associate target 1 with measurement m1
                and target 2 with measurement m2

            endif
        endfor
    endif
endfor
endif
endfor

```

Figure 2.9. Pseudocode to form all joint association events for two targets.

In the case of N_t targets, one level of **for** loop will be required for each target in the cluster being processed. The easiest way of implementing this for the general case will be using recursion. The pseudocode shown in Figure 2.10 recursively generates all joint association events for an arbitrary number of targets. The operation of the code is to associate each target with every possible measurement recursively until all targets have measurements associated with them, at which stage the joint event is finalized and stored in whichever form is required for the algorithm being utilized. As association events are formed, the target-measurement pairings for the event are progressively collected in the “**assoc**” structure, which will contain the complete association list for the event when the recursion reaches its stopping point.

2.5.4 Joint Target State. In the previous development, the *a priori* probability of the measurements was expressed using the *joint* PDF, and little further attention was paid to its evaluation. If the state vectors of the targets are assumed independent, then the PDF of the joint target state (which is required to perform a Kalman filter measurement update) can be expressed as:

$$f\{\mathbf{X}(k)|\mathbf{Z}^{k-1}\} = \prod_{i=1}^{N_t} f\{\mathbf{x}_i(k)|\mathbf{Z}^{k-1}\} \quad (2.64)$$

where the PDF of the state of target i is assumed Gaussian with mean $\hat{\mathbf{x}}_i(k|k-1)$ and covariance $\mathbf{P}_i(k|k-1)$. In this case, the *a priori* knowledge of the measurement vectors, conditioned on an association event (as required for Eq. (2.62)), is also independent and can be expressed as:

$$f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} = \prod_{j=1}^{N_m} f\{\mathbf{z}_j(k)|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} \quad (2.65)$$

where the PDF of the measurement associated with target i is Gaussian with mean $\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)$ and covariance $\mathbf{H}\mathbf{P}_i(k|k-1)\mathbf{H}^T + \mathbf{R}$, and the PDFs of measurements hypothesized to be the result of clutter are uniform as per Eq. (2.56). In this case,

```

function associate(freeMeasurements, freeTargets, assoc)
% Function is called initially with freeMeasurements containing a
% listing of all measurements in the cluster, freeTargets containing
% a listing of all targets in the cluster, and assoc containing an
% empty list which will be used to construct the association events
% recursively

if freeTargets list is empty then
    % Association event is complete: store in appropriate global
    % structure
    calculate probability of joint event described by assoc
    add assoc to the list of joint association events
else
    % Select the first free target to associate, delete from free list
    t = first entry in freeTargets
    newFreeTargets = freeTargets with first entry deleted

    % Associate target with measurement zero -- i.e., hypothesize
    % that a missed detection occurred for the target
    assocNew = assoc with appended entry (t,0)
    associate(freeMeasurements, newFreeTargets, assocNew)

    % Associate all remaining measurements with target
    for j = 1 to length of freeMeasurements do
        m = j-th element of freeMeasurements

        % Check that measurement is inside target's association gate
        if valid(t,m) then
            % Create new list of free measurements
            newFreeMeasurements =
                freeMeasurements with j-th element deleted

            % Create updated association list
            assocNew = assoc with appended entry (t,m)
            associate(newFreeMeasurements, newFreeTargets, assocNew)
        endif
    endfor
endif
endif

```

Figure 2.10. Pseudocode to form all joint association events for an arbitrary number of targets.

the estimates conditioned on a given association event can be updated from sample period $(k - 1)$ to sample period k with the standard Kalman filter update equation of Eq. (2.8), using the measurement assigned to the target in the association event $\Theta_l(k)$ and the measurement matrix $\mathbf{H}$. Targets which are hypothesized to have been missed under the association event are left unchanged.

In the case in which target state vectors are correlated,¹¹ the entire update must be performed in one step, using an augmented measurement matrix $\mathbb{H}$. As an example, consider the same joint association event used in Section 2.5.2:

$$\Theta_l(k) = \{\theta_{12}, \theta_{21}, \theta_{34}, \theta_{40}\}$$

where we recall that the elemental event θ_{ji} represents the association of measurement j with target i , such that our sample event indicates that measurement 1 has been associated with target 2, measurement 2 with target 1, measurement 3 with target 4 and measurement 4 is the result of clutter, and there were four targets under track, hence target 3 was missed. The joint target state is in block form, with a single block for each target:

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \\ \mathbf{x}_3(k) \\ \mathbf{x}_4(k) \end{bmatrix}$$

¹¹The motivation for admitting correlation between targets will become apparent in Section 2.5.8.

Similarly, the joint measurement vector is in block form, with a single block for each measurement:

$$\mathbf{Z}_k = \begin{bmatrix} \mathbf{z}_1(k) \\ \mathbf{z}_2(k) \\ \mathbf{z}_3(k) \\ \mathbf{z}_4(k) \end{bmatrix}$$

Given that measurement 4 is hypothesized to be the result of clutter, under the association event $\Theta_l(k)$ we discard it, defining the modified target-originated measurement vector as:

$$\mathbf{Z}_k' = \begin{bmatrix} \mathbf{z}_1(k) \\ \mathbf{z}_2(k) \\ \mathbf{z}_3(k) \end{bmatrix}$$

Denoting $\mathbf{H}$ as the matrix which describes the relationship between the a single measurement and the state of a single target, the block measurement matrix $\mathbb{H}$ which describes the relationship between the joint measurement vector and the joint target state vector for this association event is:

$$\mathbb{H}(\Theta_l(k)) = \begin{bmatrix} \mathbf{0} & \mathbf{H} & \mathbf{0} & \mathbf{0} \\ \mathbf{H} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H} \end{bmatrix}$$

such that:

$$\mathbf{Z}_k' = \mathbb{H}(\Theta_l(k))\mathbf{X}(k) + \mathbf{V}(k)$$

and thus the standard Kalman filter update expression of Eq. (2.8) is employed using these augmented structures to find the updated joint target state conditioned on the particular association event $\Theta_l(k)$. To evaluate Eq. (2.62) when correlation exists between targets, we use the expression:

$$f\{\mathbf{Z}_k|\Theta_l, N_m, \mathbf{Z}^{k-1}\} = \mathcal{N}\{\mathbf{Z}_k'; \mathbb{H}\hat{\mathbf{X}}, \mathbb{H}\mathbb{P}\mathbb{H}^T + \mathbb{R}\}V^{-\phi} \quad (2.66)$$

where $\mathbb{P}$ is the matrix containing the covariance of the joint target state estimate $\hat{\mathbf{X}}$, $\mathbb{R}$ is the block-diagonal matrix containing the covariance of the augmented measurement noise $\mathbf{V}(k)$, and V is the volume of the combined gating region.¹² The latter term incorporates the uniform density of clutter-originated measurements (of count ϕ), as discussed in Eq. (2.56).

2.5.5 State Update. The PDF of the joint target state stored by the tracking system at the end of sample period $(k-1)$ is denoted by $f\{\mathbf{X}(k-1)|\mathbf{Z}^{k-1}\}$. Assuming that the prior state PDF is a single Gaussian function, the standard linear propagation model presented in Section 2.2.3 can be used to propagate the PDF to the k -th sample period, resulting in $f\{\mathbf{X}(k)|\mathbf{Z}^{k-1}\}$. This expression then has the measurements from the k -th sample period incorporated, resulting in the new PDF $f\{\mathbf{X}(k)|\mathbf{Z}^k\}$, which is the same as the original PDF except one sample period later, thus the process is able to be repeated recursively. The probability of the joint association event, as developed in Section 2.5.2, is utilized to perform this state update using the total probability expansion:

$$f\{\mathbf{X}(k)|\mathbf{Z}^k\} = \sum_l f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k)\} P\{\Theta_l(k)|\mathbf{Z}^k\} \quad (2.67)$$

The expression $f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k)\}$ represents the updated joint target state conditioned on the new measurement history and a specific association event. If the prior target state density was a single Gaussian PDF, then this is easily calculated using the standard Kalman filter update equations as per Eq. (2.8), with the augmented joint measurement matrix $\mathbb{H}$, as described in Section 2.5.4.

Even if the original joint target density was a single Gaussian PDF, the updated density of Eq. (2.67) is a Gaussian mixture, with one component for each joint

¹²Note the difference in notation between the boldface $\mathbf{V}(k)$, which is the vector of the augmented measurement noise for all target-oriented measurements, and V (not boldface), which is the scalar volume of the combined gating region.

association event. Accordingly, some means is necessary to perform this measurement update when the input is a Gaussian mixture, rather than a single Gaussian function. If each component of the Gaussian mixture at the input of the update cycle is interpreted as the result of an earlier association hypothesis, denoted $\Psi_u(k-1)$,¹³ then the PDF at sample period $(k-1)$ can be expanded as:

$$f\{\mathbf{X}(k-1)|\mathbf{Z}^{k-1}\} = \sum_u f\{\mathbf{X}(k-1)|\mathbf{Z}^{k-1}, \Psi_u(k-1)\} P\{\Psi_u(k-1)|\mathbf{Z}^{k-1}\} \quad (2.68)$$

Each component of the PDF in Eq. (2.68) can be propagated using the same linear propagation equations as a single Gaussian function to find the set of component Gaussian functions $\{f\{\mathbf{X}(k)|\mathbf{Z}^{k-1}, \Psi_u(k-1)\}\}$. The state update will then be performed by modifying the PDF update equation of Eq. (2.67) to:

$$\begin{aligned} f\{\mathbf{X}(k)|\mathbf{Z}^k\} &= \sum_l \sum_u f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k), \Psi_u(k-1)\} \cdot \\ &\quad \cdot P\{\Theta_l(k)|\mathbf{Z}^k, \Psi_u(k-1)\} P\{\Psi_u(k-1)|\mathbf{Z}^{k-1}\} \end{aligned} \quad (2.69)$$

In this expression the term $f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k), \Psi_u(k-1)\}$ once again represents the update of a *single* Gaussian PDF using a single association event, hence the standard Kalman filter update equation can be used, and the result is again a single Gaussian. The result of Eq. (2.69) is therefore another Gaussian mixture, with a single component for each $\{\Psi_u(k-1), \Theta_l(k)\}$ pair, i.e., each previous hypothesis is updated using each current association event. The number of components in the new mixture is equal to the number of previous components multiplied by the number of current association hypotheses.

¹³The notation $\Psi_u(k-1)$ is used to distinguish the previous hypotheses, which are association *histories*, from the latest *single-event* association hypotheses, $\Theta_l(k)$. For example, a single association history event $\Psi_u(k-1)$ may consist of the history $\{\Theta_5(1), \Theta_{16}(2), \Theta_7(3), \dots\}$, in which the element $\Theta_l(k)$ indicates that the association history hypothesizes the joint association event Θ_l at sample time k . The variable ‘ u ’ was chosen arbitrarily for the index so as not to conflict with other notation in this document.

The double summation of Eq. (2.69) can be expressed as an equivalent single summation for propagation to the next processing cycle. The new mixture will then be represented as:

$$f\{\mathbf{X}(k)|\mathbf{Z}^k\} = \sum_{u'} f\{\mathbf{X}(k)|\mathbf{Z}^k, \Psi_{u'}(k)\} P\{\Psi_{u'}(k)|\mathbf{Z}^k\} \quad (2.70)$$

where the new indexing u' covers all new mixture components, with:

$$\begin{aligned} f\{\mathbf{X}(k)|\mathbf{Z}^k, \Psi_{u'}(k)\} &= f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k), \Psi_u(k-1)\} \\ P\{\Psi_{u'}(k)|\mathbf{Z}^k\} &= P\{\Theta_l(k)|\mathbf{Z}^k, \Psi_u(k-1)\} P\{\Psi_u(k-1)|\mathbf{Z}^{k-1}\} \end{aligned}$$

This strategy is the optimal Bayesian data association solution, and the transition from Eq. (2.68) to Eqs. (2.69) and (2.70) reveals the major problem associated with it: at each sample period, the previous number of hypotheses is *multiplied* by the number of joint association hypotheses in the current sample period, hence the number of hypotheses required to be maintained grows *exponentially*, with the rate of growth according to the number of joint association hypotheses in each sample period. Thus the optimal Bayesian solution is clearly intractable, and some form of simplification will be necessary to reduce the number of components in the Gaussian mixture to a manageable level.

2.5.6 Global Nearest Neighbor. Possibly the easiest way of addressing the problem of the increasing number of hypotheses would be simply to take the Gaussian mixture component corresponding to the *most likely* hypothesis and discard the rest of the mixture, leaving only a single component. At each sample period, the PDF of target state propagated from the previous sample period will be a single Gaussian PDF, hence the update process consists of calculating the probability of all joint association events, and then updating the joint target state with the most likely hypothesis. This algorithm is referred to as Global Nearest Neighbor

(GNN), to indicate that the best global (i.e., joint) association hypothesis is to be selected [7:338–342].

If only a single target is present, then the equations of the joint association events will be very similar to each other, and the algorithm can be simplified to the standard Nearest Neighbor (i.e., not global). The simplified algorithm performs the association of the measurement with the smallest distance, according to the Mahalanobis distance measure, similar to the exponent of a Gaussian PDF:

$$d_j^2 = (\mathbf{z}_j - \hat{\mathbf{z}})^T \mathbf{S}^{-1} (\mathbf{z}_j - \hat{\mathbf{z}}) \quad (2.71)$$

where $\mathbf{z}_j$ is the j -th measurement, $\hat{\mathbf{z}}$ is the predicted measurement for the single target, and $\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}$ is the predicted covariance of the residual formed with the correct measurement. Comparing Eq. (2.71) with Eq. (2.14) reveals that the square of the Mahalanobis distance is actually the normalized residual quadratic, which, in Section 2.4, was used to indicate how well the tracking model matched the measurements, and is now used to indicate how well the measurements match the tracking model.

Nearest neighbor association techniques are sometimes referred to as *hard assignment* methods, indicating that *hard* decisions have been made: the system *assigns* target-measurement associations, and progresses in processing *assuming* that the assignments were indeed correct. The following sections describe techniques which use probabilistically weighted (*soft*) decisions. The performance of hard assignment methods is very limiting. As highlighted by Streit and Luginbuhl [53:1], the hard decisions associated with techniques such as GNN introduce opportunities for decision mistakes, and hence necessarily increase estimation error. Intuitively, one can see that much of the information carried by the joint target PDF is being discarded, hence logically one would expect the success of the method to be limited.

2.5.7 Probabilistic Data Association. The single target Probabilistic Data Association (PDA) algorithm [2:163–170] and its multiple target extension, the Joint Probabilistic Data Association (JPDA) algorithm [4:310–319], are two more techniques which reduce the joint target state PDF down to a single mixture component at the end of each sample period. Rather than taking the *most likely* association hypothesis at each processing interval, these techniques take the *weighted average of all association hypotheses*.

Thus, the approximation inherent to the PDA/JPDA algorithm is:

$$\begin{aligned} f\{\mathbf{X}(k)|\mathbf{Z}^k\} &= \sum_l f\{\mathbf{X}(k)|\mathbf{Z}^k, \Theta_l(k)\} P\{\Theta_l(k)|\mathbf{Z}^k\} \\ &\approx \mathcal{N}\{\mathbf{X}(k); \hat{\mathbf{X}}(k|k), \mathbb{P}(k|k)\} \end{aligned} \quad (2.72)$$

where $\hat{\mathbf{X}}(k|k)$ is the weighted average of the means of the Gaussian sum as according to Eq. (2.22), and $\mathbb{P}(k|k)$ is the weighted average of the covariances of the Gaussian sum as according to Eq. (2.23):

$$\begin{aligned} \hat{\mathbf{X}}(k|k) &= \sum_l P\{\Theta_l(k)|\mathbf{Z}^k\} \hat{\mathbf{X}}(k|k, \Theta_l(k)) \\ \mathbb{P}(k|k) &= \sum_l P\{\Theta_l(k)|\mathbf{Z}^k\} [\mathbf{P}(k|k, \Theta_l(k)) + \\ &\quad + (\hat{\mathbf{X}}(k|k, \Theta_l(k)) - \hat{\mathbf{X}}(k|k))(\hat{\mathbf{X}}(k|k, \Theta_l(k)) - \hat{\mathbf{X}}(k|k))^T] \end{aligned}$$

Unless it is explicitly prevented, the combined covariance of Eq. (2.72) will have correlation between targets, as induced by the “spreading of the means” terms on the final line of the above expression. The implication of this is discussed further in Section 2.5.8; the JPDA algorithm discards any correlation between targets in order to reduce computational complexity.

The equations above represent one possible method of implementing the JPDA algorithm. However, because all estimates are combined into a single overall mean

and covariance at each sample period, and correlation terms are discarded, the Kalman weights and covariances for a given target under each event will be identical, and thus the implementation can be optimized substantially.

The common implementation of the JPDA algorithm uses the following algebraically equivalent equations to update the state estimate and covariance of target i [4]:

$$\begin{aligned}\hat{\mathbf{x}}_i(k|k) &= \hat{\mathbf{x}}_i(k|k-1) + \mathbf{K}_i(k)\bar{\boldsymbol{\nu}}_i(k) \\ \mathbf{P}_i(k|k) &= \mathbf{P}_i(k|k-1) - \alpha_i \mathbf{K}_i(k) \mathbf{H} \mathbf{P}_i(k|k-1) + \tilde{\mathbf{P}}_i(k)\end{aligned}\quad (2.73)$$

where $\bar{\boldsymbol{\nu}}_i(k)$ represents the combined residual for target i and $\tilde{\mathbf{P}}_i(k)$ represents the spreading of the variance due to the combination of multiple Gaussian components:

$$\begin{aligned}\bar{\boldsymbol{\nu}}_i(k) &= \sum_{j=1}^{N_m} \beta_{ji} \boldsymbol{\nu}_{ji}(k) \\ \boldsymbol{\nu}_{ji}(k) &= \mathbf{z}_j(k) - \mathbf{H} \hat{\mathbf{x}}_i(k|k-1) \\ \alpha_i &= \sum_{j=1}^{N_m} \beta_{ji} \\ \tilde{\mathbf{P}}_i(k) &= \sum_{j=1}^{N_m} \beta_{ji} \boldsymbol{\nu}_{ji}(k) \boldsymbol{\nu}_{ji}(k)^T - \bar{\boldsymbol{\nu}}_i(k) \bar{\boldsymbol{\nu}}_i(k)^T\end{aligned}$$

$\boldsymbol{\nu}_{ji}(k)$ represents the residual formed with measurement j and target i , and β_{ji} is the combined probability of all events in which measurement j is associated with target i :

$$\beta_{ji} = \sum_{l: \theta_{ji} \in \Theta_l(k)} P\{\Theta_l(k) | \mathbf{Z}^k\}$$

The PDA/JPDA algorithm has been applied to a vast array of problems in open literature, and has proven itself to be very effective in less demanding tracking environments (for example, [4:320–327]). In more demanding tracking problems (such as high clutter density and targets which remain close for extended periods of

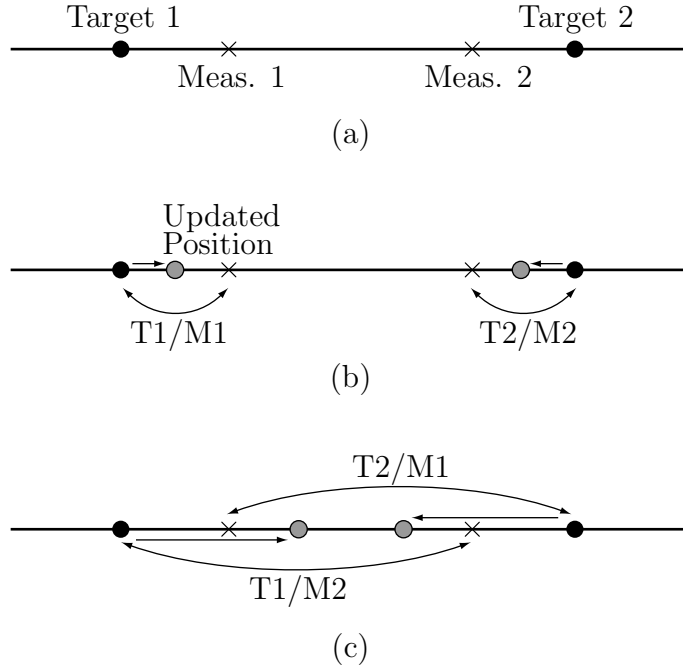


Figure 2.11. One-dimensional multiple target data association example.

time), the simplification applied to the joint target state PDF begins to prove too much [27], and the more detailed representations described in the following sections are necessary.

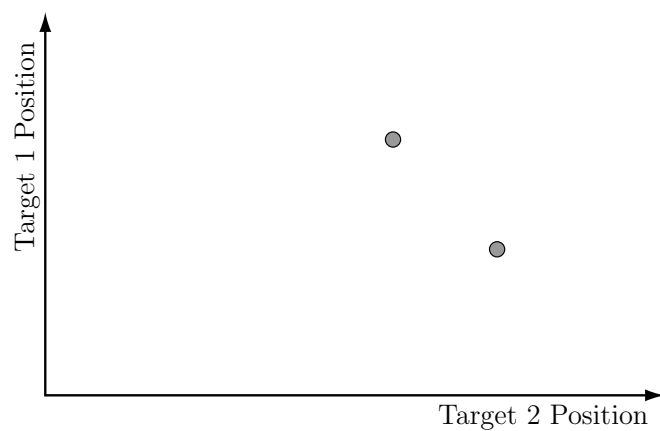
2.5.8 Correlation Between Targets. Although it initially seems unusual to allow correlation to develop between the state estimates of two physically independent targets, detailed consideration of the joint PDF of target state reveals the motivation for doing so, and the potential benefit that may be obtained. The following one-dimensional tracking example, illustrated in Figures 2.11 and 2.12, helps to explain.

Figure 2.11(a) shows the *a priori* position of the two targets, marked by ‘•’, and the two newly received measurements, marked by ‘x’. Considering only association events in which each target is associated with a single measurement, there are two possible associations: either each target will be associated with the mea-

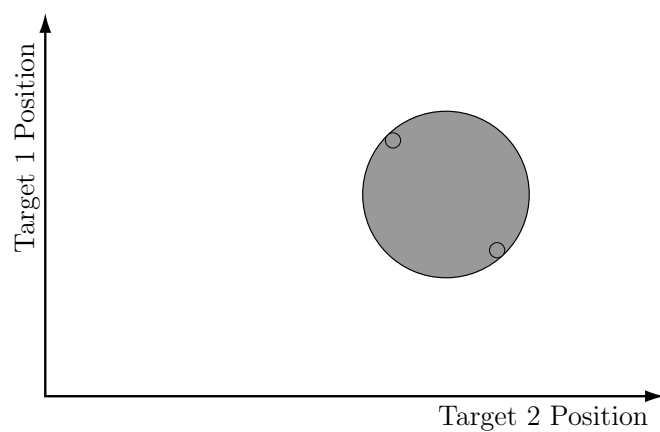
surement closer to it, or each target will be associated with the measurement farther from it. The first joint association event is illustrated in Figure 2.11(b), in which the rounded arrows indicate the associations, and the gray dots indicate the updated state of the targets, moved toward the measurements used to update them. The second association event is illustrated similarly in Figure 2.11(c): the greater disparity between each target-measurement pairing will tend to produce a larger update in the position of each target; in practice the probability of this event will be smaller as the associations are less likely.

The updated states corresponding to the two association events of Figures 2.11(b) and (c) are illustrated in joint target space in Figure 2.12(a). The updated state corresponding to each possible joint association event maps to a point in the joint target space, and the resultant joint target PDF will consist of a Gaussian sum with weighted Gaussian functions at each of these points, and different covariance matrices determining the spread about these points. Under the approximation of JPDA, these joint hypotheses are to be represented by a single Gaussian PDF. If this simplified PDF is forced to be independent between targets, then the resultant function will be as illustrated in Figure 2.12(b): the coordinates of the covariance must be aligned with the target state coordinate systems, hence a broad approximation is necessary, representing a great loss of information. If correlation is allowed between targets, then the covariance takes the form illustrated in Figure 2.12(c): the marginal covariance in each target coordinate system remains identical, but the high degree of correlation *between* the coordinate systems greatly increases the information retained.

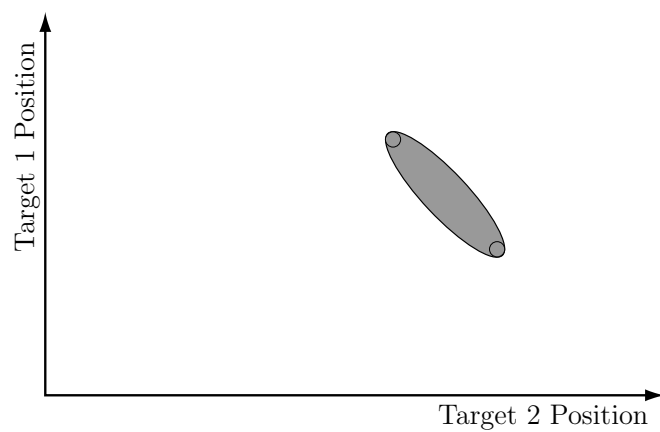
The intuitive understanding of the benefit of allowing covariance such as that illustrated in Figure 2.12(c) is this: if later measurements confirm that target 2 was further in fact to the right, then this indicates that target 1 was actually further to the left. Likewise, if later measurements indicate that target 1 was further to the left, then this serves to confirm that target 2 was actually further to the right.



(a)



(b)



(c)

Figure 2.12. Correlation arising due to combining of hypotheses.

In this way, correlation between hypotheses allows later measurements to resolve uncertainties left over from earlier processing cycles, much like the deferred decision-making capability of the Multiple Hypothesis Tracker described in Section 2.5.10.

Joint Probabilistic Data Association Coupled (JPDAC) and Coupled Probabilistic Data Association (CPDA) are two extensions of JPDA which allow correlation to develop between targets for periods during which targets are in the same region (i.e., cluster). JPDAC [4:328–329] was the initial implementation of the concept, allowing correlation to develop between state estimates in a cluster containing two targets. In the reference cited, however, there is no mention of how to approach joint association events in which detection of one or both of the targets is hypothesized to have been missed. Such events may be of minor concern if the probability of detection is close to unity and the association gate is selected to be very large. However, if the probability of detection becomes significantly less than unity, such an omission can have a devastating impact on the performance of the system.

The CPDA algorithm described in [9, 12] is a full implementation of the approximation of Eq. (2.72) admitting correlation between targets. Implementation of this algorithm directly (without calculating the full mean and covariance individually for each hypothesis before merging) is rather difficult, and necessitates the somewhat opaque notation found in these articles.

Correlation in PDA implementations was initially developed to fix the problem of tracks belonging to nearby targets tending to coalesce into a single track midway between the two. However, as discussed in [12], both JPDAC and CPDA perform more poorly in this respect than the original uncorrelated JPDA algorithm. The explanation of this phenomenon provided in the cited article is that CPDA develops strong correlation between the targets, hence it tends to keep the two tracks together between competing measurements. However, as illustrated in the example of Figure 2.12, the coupling which develops between targets is almost guaranteed to be *negative*

correlation, hence such an explanation would seem unsatisfactory. The causes of bias and coalescence in PDA algorithms are examined in depth in Section 3.2.

2.5.9 Maximum Likelihood Methods. The systems described in [25, 26] bear much resemblance to the CPDA technique described above: they both reduce the distribution of the target to a single Gaussian mixture, and they both explicitly model the correlation which develops between targets. Rather than using the Gaussian function with the parameters derived as the weighted mean over all possible associations, these methods instead select the state estimate as the value which *maximizes* the likelihood of receiving the measurements (considering all possible associations), with the covariance evaluated using the Fisher information matrix. Other novel inclusions of these techniques are that they consider association over several sets of measurements [25], and that they propose an approximation of the Kronecker delta function which avoids the necessity of generating all joint association events [26].

2.5.10 Multiple Hypothesis Tracking. The Multiple Hypothesis Tracker (MHT) is an algorithm that has been discussed in literature in many different forms, starting with [40] and [49]. The basic concept of the algorithm is to maintain hypotheses for every plausible association event; each hypothesis consists of the probability of the event, and the mean and covariance of the target state conditioned on the event. In this way, the algorithm essentially maintains the Gaussian mixture representation of the PDF of target state as developed in Section 2.5.5. To alleviate the exponential explosion of hypotheses, pruning and merging algorithms are applied to the hypothesis tree to eliminate those hypotheses that become implausible, and merge those which produce similar results.

The original presentation of the multiple target algorithm [40] proceeds in a measurement oriented manner, whereby the algorithm is driven by considering the possible origins of each measurement individually. The growth of association hypotheses inherently follows a tree structure [6:285], in which each leaf node indicates

a hypothesis for the existence and location of the targets at the current time instant. Measurements from the same scan are processed together to avoid assigning two measurements to a target. When considering each measurement, there are always at least two possibilities: the measurement may be a false alarm (as from clutter), or it may represent a new target, hence each measurement generates at least two new nodes for each entry in the hypothesis list. For hypotheses that contain existing tracks, the measurement may also represent a continuation of each of these (assuming that the measurement gate is satisfied), hence more hypotheses can potentially be generated.

A more readily understood track-oriented development of the MHT algorithm is presented in [30], and similarly in [8], where it was termed the Structured Branching Multiple Hypothesis Tracker (SB-MHT). The structure of this algorithm is to create single-target hypotheses for each of the possible measurements with which a target can be associated at each processing cycle. One accounts for joint hypotheses by maintaining lists of compatible single-target hypotheses, providing an efficient means of keeping track of a large number of joint hypotheses, each pointing to a series of single target hypotheses containing the target parameters.

The key step in ensuring the performance and computability of an MHT algorithm is efficient hypothesis pruning and merging algorithms, yet the majority of these are based largely on *ad hoc* methods. Several different pruning methods are suggested in [6:291], such as deleting those hypotheses whose probabilities are less than a certain threshold, retaining the N_h most likely hypotheses, or retaining the most likely hypotheses such that the total probability of the set retained is greater than some threshold (close to unity). Merging of hypotheses may be performed on the basis of shared measurements over a period of time (e.g., if the associations in two hypotheses are identical over the last three scans, they are merged), or after direct state comparison. Merging may be performed either using replacement (deleting

the lower probability hypothesis and adding its probability to the other), or by some form of weighted averaging.

2.5.11 Controlling the Number of Hypotheses. Of the algorithms discussed, the MHT is the only algorithm which is able to maintain more than a single association hypothesis between measurement intervals. Although the MHT represents the state-of-the-art in modern target tracking, the most vital task to the algorithm, selection of which hypotheses to retain, is largely *ad hoc* in most implementations. Most commonly, a Maximum Likelihood strategy is adopted whereby the N_h most likely tracks (or all tracks with probabilities that exceed a given threshold) are maintained, and the remainder deleted.

Few merging strategies are discussed in open literature: the most common is *n*-scan merging [49], which merges the hypotheses that incorporate identical measurement histories over the last *n* processing cycles, effectively limiting the maximum number of processing cycles for which decision making can be deferred. As the length of the memory is increased, the merging has less impact: the average number of hypotheses will increase exponentially with the length of the memory, and will soon need to be controlled by deleting less likely hypotheses, returning us largely to the Maximum Likelihood pruning strategy where we started.

Other merging methods based on similarity of target state distributions have been suggested [6, 7, 40] but the little detail given indicates that the simplifications rely on *ad hoc* state comparisons such as [6:293]:

$$\begin{aligned}
|\{\hat{\mathbf{x}}_1\}_i - \{\hat{\mathbf{x}}_2\}_i| &\leq \beta \sqrt{\{\mathbf{P}_1\}_{ii} + \{\mathbf{P}_2\}_{ii}} \quad \forall i \\
\{\mathbf{P}_1\}_{ii} &< \gamma \{\mathbf{P}_2\}_{ii} \quad \forall i \\
\{\mathbf{P}_2\}_{ii} &< \gamma \{\mathbf{P}_1\}_{ii} \quad \forall i
\end{aligned}$$

with $\beta = 0.1$ and $\gamma = 2.0$. Interpreting the algorithm, in order to be merged, the state estimates of the hypotheses must be within roughly 0.1 standard deviations of each other,¹⁴ and the covariance trace elements must differ by no more than a factor of two.

In benign tracking environments, the hypothesis selection strategy is of little importance, as long as the number of hypotheses maintained is adequate to ensure that the correct hypothesis is maintained with a high probability. In more adverse tracking environments, the correct association hypothesis may appear less likely than false hypotheses for several consecutive processing intervals. Hence, to maintain the correct hypothesis, either a much larger number of hypotheses will need to be maintained (increasing exponentially with the number of processing cycles over which the association remains ambiguous), or a more efficient hypothesis selection method will be required.

2.5.11.1 Early Methods. The approach of the early methods proposed in [1] and [31] appears on the surface to be very similar to that detailed in Section 3.3.4. Alspach [1] selects the Kolmogorov variational distance as the cost function, defined as:

$$J_K = \int |f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} - f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}| d\mathbf{X}(k) \quad (2.74)$$

where $f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}$ represents the full target state PDF, containing $N_h(k)$ hypotheses, and $f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}$ represents the reduced PDF, containing $N_r(k)$ components ($N_r < N_h$), which is being fitted to the full PDF.¹⁵ The algorithm continues

¹⁴The comparison is performed in standard coordinates rather than rotated principal coordinates, in order to avoid the computational loading associated with a matrix inverse for each pair of mixture components.

¹⁵i.e., $\boldsymbol{\Omega}_{N_h}(k)$ represents the full parameters of the distribution (containing N_h mixture component weights, means and covariances), and $\bar{\boldsymbol{\Omega}}_{N_r}(k)$ represents the equivalent reduced set of parameters for N_r mixture components.

by merging and pruning mixture components until the cost exceeds a certain threshold.

The method of Lainiotis and Park [31] uses the Bhattacharyya coefficient as the similarity measure:

$$J_B = \int \sqrt{f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}}d\mathbf{X}(k) \quad (2.75)$$

Computation of these functions for Gaussian mixtures is a formidable task, and the various implementations of [1, 31, 56] rely heavily on mathematical approximations to be able to evaluate the functions without explicit numerical integration. One of the assumptions invoked by Alspach [1] is that all components have the same covariance. If different hypotheses in the filter propose that the target has and has not had missed detections, then the resultant covariance matrices of the mixture components will be different, hence this approximation is undesirable. Furthermore, once mixture components are merged, the covariance matrices will be modified by the spreading terms of Eq. (2.24), again making the assumption of identical covariance matrices problematic.

Lainiotis [31] uses mathematical approximations to evaluate the cost of merging and deleting components. As discussed in [31:625], the Bhattacharyya coefficient between the original Gaussian mixture, and the same mixture with a single component deleted is bounded below by:

$$\rho_a \geq 1 - \frac{1}{2}p_n \quad (2.76)$$

where p_n is the weight of the deleted component. Similarly, the Bhattacharyya coefficient between the original Gaussian mixture and the same mixture with a single pair of components merged is bounded below by:

$$\rho_a \geq 1 - (p_i + p_j)\sqrt{1 - \rho_{i,j}^2} \quad (2.77)$$

where p_i and p_j are the weights of the two components to be merged, and $\rho_{i,j}$ is the Bhattacharyya coefficient between the two components to be merged (which, unlike the Bhattacharyya coefficient between two Gaussian *mixtures*, is easily evaluated). Using the expressions in Eqs. (2.76) and (2.77), the algorithm operates by merging and deleting components which produce a worst-case reduction in the Bhattacharyya coefficient that is smaller than a given threshold.¹⁶

2.5.11.2 Mixture Reduction Algorithm. In the context of the problem of tracking a single target in clutter, Salmond proposed two algorithms [44–47] for reducing the number of hypotheses by systematically merging hypotheses based on certain similarity criteria. The focus of the study was to produce algorithms which were computationally feasible using the hardware available at the time.

The first algorithm is referred to as the *joining algorithm*. The operation of the algorithm is to merge pairs of mixture components successively until the desired level of reduction has been achieved. The distance measure utilized to gauge the similarity of hypotheses i and j is a Mahalanobis-type distance measure:

$$d_{ij}^2 = \frac{p_i p_j}{p_i + p_j} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)^T \mathbf{P}^{-1} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j) \quad (2.78)$$

where the covariance $\mathbf{P}$ is the combined covariance for the entire mixture, as in Eq. (2.23):

$$\begin{aligned} \mathbf{P} &= \sum_{i=1}^{N_h} p_i [\mathbf{P}_i + (\hat{\mathbf{x}}_i - \boldsymbol{\mu})(\hat{\mathbf{x}}_i - \boldsymbol{\mu})^T] \\ \boldsymbol{\mu} &= \sum_{i=1}^{N_h} p_i \hat{\mathbf{x}}_i \end{aligned}$$

The leading fraction in Eq. (2.78) provides a weighting which tends to favor merging hypotheses that carry lower probability weight over those with higher probability.

¹⁶Separate thresholds are used for merging and deleting.

The term acts as a smooth interpolation of the minimum of the two probabilities, and may also be expressed as $(p_i^{-1} + p_j^{-1})^{-1}$.

The algorithm functions by calculating the distance between all pairs of hypotheses, and merging the pair with the smallest distance. The operation continues until the minimum distance is above a threshold:

$$T = 0.001 \dim(\mathbf{x})$$

which was determined based on visual inspection. The threshold is designed to ensure that the mixture structure is not modified beyond an acceptable level. If the desired level of reduction has not been achieved when this threshold is reached, then the operation continues until the mixture has been simplified to the desired number of components.

The second algorithm proposed is the *clustering algorithm*, which combines mixtures into groups (clusters) rather than pairs. The algorithm operates by selecting a principal component for a cluster, denoted as component c (initially the component with the largest probability weight), and merging all components that are within a certain distance of the principal component. The distance measure used is the alternative definition:

$$D_i^2 = \frac{p_i p_c}{p_i + p_c} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_c)^T \mathbf{P}_c^{-1} (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_c) \quad (2.79)$$

which normalizes using the covariance $\mathbf{P}_c$ of the principal component, rather than the total mixture covariance as in Eq. (2.78). Considering the measure of Eq. (2.79) as the normalized distance of the i -th component mean from the principal component, the threshold T_1 used for the distance test can be based on a χ^2 test [7:429], with the recommended value:

$$T_1 = 0.05 T_1'$$

where T_1' is such that $\{D_i^2 : D_i^2 < T_1'\}$ contains 99% of the χ^2 PDF, where the number of degrees of freedom is the number of states.

The clustering algorithm continues iteratively, selecting the largest unclustered component as the principal component of the new cluster at each stage. If the process completes before the desired amount of reduction has been achieved, the algorithm is repeated with a larger threshold. The computational load of the clustering algorithm is significantly lower than that of the joining algorithm, as at each stage the distance of each component to the principal component is calculated, rather than the distance between every pair of components.

In [38], Pao extends Salmond's work to admit the case of multiple sensors and multiple targets. This extension is analogous to the extension from PDA to JPDA: while the probabilistic model is updated to account for joint association events probabilities, it does not maintain correlation between target estimates, and it does not maintain lists of compatible tracks, hence it intrinsically forces independence between target estimates. For example, if the hypotheses for two targets are forced to be independent, then the PDF of joint target state will contain elements for each pairing of hypotheses from the two targets, as illustrated in Figure 2.13, rather than restricting the uncertainty to the actual joint hypotheses as illustrated in Figure 2.12(a).

2.5.12 Multidimensional Techniques. The techniques described thus far have one aspect in common: they all process one frame of data (i.e., the measurements resulting from a single complete radar scan) at a time. An alternative approach which has gained popularity recently is to use multiple frames of data at once, resolving measurement uncertainty using a sequence of data rather than a single scan frame.

Multidimensional assignment is a recently developed extension of the GNN algorithm described in Section 2.5.6, in which multiple sets of data (either multiple

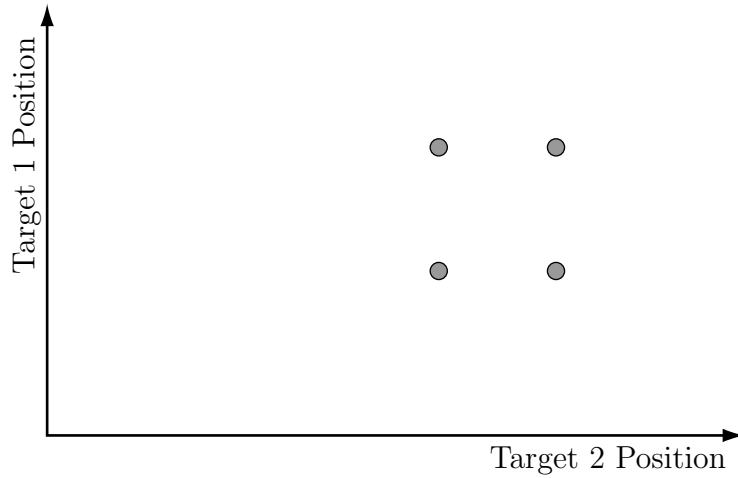


Figure 2.13. The impact of forcing independence between targets in a multiple hypothesis system: resultant joint target PDF contains a hypothesis for each pairing of hypotheses from each target, rather than only the actual joint hypotheses as shown in Figure 2.12(a).

scans from a single sensor, or data from multiple sensors) are simultaneously considered for association. As with GNN, the technique uses hard assignment, with the assignment selected after a global optimization considering all possible joint association events over all data sets. As one would expect, such techniques are computationally demanding, but recent algorithms such as Lagrangian relaxation [39] appear to provide a near-optimal solution for a more acceptable computational burden [7].

Other recent suggestions include multiple-scan JPDA [43], in which joint association events over several scans are probabilistically averaged in one step, and the Probabilistic Multiple Hypothesis Tracker (PMHT) [53], in which measurement-target association probabilities for multiple scans are estimated from a block of data using the Estimation-Maximization (EM) algorithm.

2.5.13 Interacting Multiple Model–Multiple Hypothesis Tracker. In the preceding sections, algorithms were described which are able to track maneuvering targets in situations in which the measurement is of known origin, as were algorithms

which are able to track multiple non-maneuvering targets in clutter, with measurements of unknown origin. The obvious extension of these two separate developments is to unify the two to create a system which is able to track maneuvering targets in the presence of clutter, with measurements of unknown origin.

The systems described in [15, 17] represent a unification of the two preferred techniques from each section: the IMM filter for maneuvering target tracking, and the MHT for data association. As opposed to the strategy proposed in [30:53], the IMM-based approach maintains multiple state estimates within a single hypothesis branch, thus limiting another source of exponentially increasing hypotheses.

The method described in [17] uses IMM only for state prediction and update, and utilizes the single combined IMM estimate for measurement gating and hypothesis likelihood evaluation. Gating is performed to validate each measurement j for consideration with each hypothesis i using the combined estimate from the IMM for the hypothesis. Thus the standard gating equation is used:

$$d_{j,i}^2 = (\mathbf{z}_j - \hat{\mathbf{z}}_i(k|k-1))^T \mathbf{S}_i^{-1} (\mathbf{z}_j - \hat{\mathbf{z}}_i(k|k-1)) \leq \gamma \quad (2.80)$$

where $\mathbf{z}_j$ is the j -th measurement, $\hat{\mathbf{z}}_i$ is the predicted measurement for hypothesis i , and $\mathbf{S}_i$ is the covariance of the residual formed using these two. The details of the algorithm are omitted from [17], however if the models in the IMM differ only in model dynamics (such that the measurement models are identical), then the elements

in Eq. (2.80) could be evaluated using:

$$\begin{aligned}
\hat{\mathbf{z}}_i(k|k-1) &= \mathbf{H}\hat{\mathbf{x}}_i(k|k-1) \\
&= \mathbf{H} \sum_{m=1}^{N_f} \mu_{i,m}(k|k-1) \hat{\mathbf{x}}_{i,m}(k|k-1) \\
\mathbf{S}_i &= \mathbf{H} \mathbf{P}_i(k|k-1) \mathbf{H}^T + \mathbf{R} \\
\mathbf{P}_i(k|k-1) &= \sum_{m=1}^{N_f} \mu_{i,m}(k|k-1) \{ \mathbf{P}_{i,m}(k|k-1) \\
&\quad + [\hat{\mathbf{x}}_{i,m}(k|k-1) - \hat{\mathbf{x}}_i(k|k-1)][\cdot]^T \} \quad (2.81)
\end{aligned}$$

where $\hat{\mathbf{x}}_{i,m}(k|k-1)$, $\mathbf{P}_{i,m}(k|k-1)$ and $\mu_{i,m}(k|k-1)$ are the predicted estimate, covariance and probability of the m -th model from the IMM for hypothesis i , calculated as described in Section 2.4.5. Using the expressions in Eq. (2.81) for the combined predicted measurement, the measurement-to-track association likelihoods can be calculated identically to the single-model case, as per Section 2.5.10.

The IMM-MHT can also be evaluated using a slightly different approach, as described in [15]. Rather than performing measurement gating and hypothesis probability calculation using a single combined estimate, the alternative strategy modifies the gating to be based on the lowest distance of the IMM filters (i.e., the filter demonstrating the best match):

$$\min_m d_{j,i,m} \leq \gamma \quad (2.82)$$

where $d_{j,i,m}$ is the Mahalanobis distance between the j -th measurement and the measurement predicted by the m -th model for the i -th track.

The measurement-to-track association likelihood proposed by [15] also differs from the technique in [17], utilizing a weighted average of the match likelihoods for each of the IMM models, rather than a single match likelihood to the combined IMM

estimate:

$$\Lambda = \frac{P_d}{\lambda} \sum_{m=1}^{N_f} \mu_{i,m} |2\pi \mathbf{S}_{i,m}|^{-\frac{1}{2}} \exp(-\frac{1}{2} d_{j,i,m}^2) \quad (2.83)$$

where $\mu_{i,m}$ is the probability of the m -th model for the hypothesis i , $d_{j,i,m}$ is the Mahalanobis distance from the m -th model of hypothesis i to measurement j (similar to Eq. (2.80)), $\mathbf{S}_{i,m}$ is the covariance of the residual formed from measurement j and the measurement prediction from model i , P_d is the probability of detection, and λ is the false alarm density.

2.5.14 Summary. The previous sections have described the techniques commonly used to address the problem of the ambiguity of measurement origin in tracking systems. The probabilistic model for association events presented in Section 2.5.2 is utilized by the majority of the data association algorithm in common use; the various approximations applied by these algorithms were described in the pursuing sections. Section 2.5.12 briefly discussed some of the recent developments which aim to consider the association of several frames of data at once, while Section 2.5.13 outlined a technique which combines the IMM algorithm with the MHT to be able to track maneuvering targets in the presence of clutter.

2.6 Optimization Methods

In Chapter III we will define a cost function that describes the fidelity of the representation of the target state probability density provided by a reduced order PDF. The goal of this study will then be to maximize the fidelity of the simplified representation by minimizing the value of the cost function. If the cost function were simple in form, it might be possible to solve exactly for the PDF parameters which produce the minimum cost solution. However, in this problem, any meaningful cost function will be highly nonlinear, and numerical optimization procedures will be unavoidable.

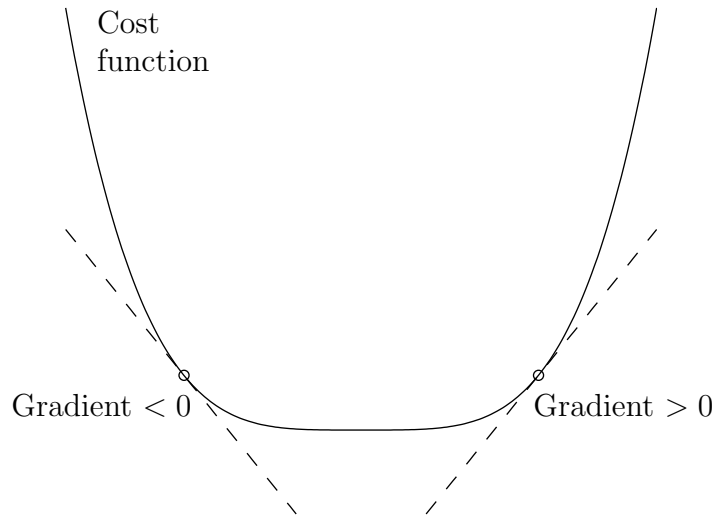


Figure 2.14. Gradient of the cost function indicating the direction of the minimum.

Numerical optimization involves techniques which iteratively converge on an optimal solution that cannot be found exactly using analytic methods. In the context of this thesis, we will be seeking the minimum value of the cost function, and the iterative techniques employed will be designed to descend as close to the minimum as possible, in as few steps as possible.

Gradient techniques [36:33] are numerical optimization methods which use the first derivative (gradient) of the cost function to step iteratively towards the minimum. Their operation in a one-dimensional problem is illustrated in Figure 2.14: if the gradient is positive, then the cost function is increasing to the right, hence the minimum must be to the left; conversely, if the gradient is negative, then the cost function is increasing to the left, hence minimum must be to the right.

The update step for the standard gradient algorithm is described by the following equation [36:33]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \frac{\mathbf{g}_f(\mathbf{x}_k)}{\|\mathbf{g}_f(\mathbf{x}_k)\|} \quad (2.84)$$

where $\mathbf{g}_f(\mathbf{x})$ is the gradient of the cost function $f(\mathbf{x})$:¹⁷

$$\mathbf{g}_f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \nabla f(\mathbf{x})$$

and s_k is the scalar step size for the k -th iteration of the algorithm. The update moves a distance of s_k in parameter space, in the direction of the negative of the gradient vector. The step size provides a trade-off between the speed of convergence and the accuracy of the final result. Using a large step size at the beginning of the search assists in increasing the rate of convergence; reducing the step size as the search progresses helps to refine the solution to provide a very accurate final result, and avoid overshooting the solution. A gradient algorithm step should be guaranteed to *decrease* the value of the cost function, hence if the cost function value *increases*, then the step size was too large, and should be reduced. If several sequential steps produced by the algorithm move in the same or a very similar direction, then the step size should be increased; conversely, if sequential steps move in the opposite direction, then the step size is too large and should be decreased. If the step size is close to its optimal value, then the gradient vector should be approximately orthogonal to the value at the previous step. One *ad hoc* algorithm for step size control based on these observations calculates the angle between successive gradient vectors β_k [36:40]:

$$\begin{aligned} \cos \beta_k &= \frac{\mathbf{g}_f(\mathbf{x}_k)^T \mathbf{g}_f(\mathbf{x}_{k-1})}{\|\mathbf{g}_f(\mathbf{x}_k)\| \cdot \|\mathbf{g}_f(\mathbf{x}_{k-1})\|} \\ s_{k+1} &= [1 + 0.9 \cos \beta_k] s_k \end{aligned} \tag{2.85}$$

The Newton-Raphson method operates similarly to the gradient technique, but uses information provided by the second derivative to converge on the minimum value at a much faster rate close to the solution. The standard Newton-Raphson

¹⁷For convenience we choose to define the derivative of a scalar with respect to a vector as a column vector, as opposed to the convention that this result is a row vector.

step (for the vector parameter case) is given by [36:55]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}(\mathbf{x}_k)^{-1} \mathbf{g}_f(\mathbf{x}_k) \quad (2.86)$$

where $\mathbf{A}(\mathbf{x}_k)$ is the Hessian matrix:

$$\begin{aligned} \{\mathbf{A}(\mathbf{x}_k)\}_{ij} &= \frac{\partial^2 f(\mathbf{x}_k)}{\partial \{\mathbf{x}_k\}_i \partial \{\mathbf{x}_k\}_j} \\ &= \{\mathbf{A}(\mathbf{x}_k)\}_{ji} \end{aligned}$$

The operation of the Newton-Raphson algorithm is to step to the minimum of the parabola which approximates the cost function at the current point. If the cost function in the region of the current parameter value is well approximated by the second order Taylor series terms (i.e., the locally fitted parabola), then the result of the step will move very close to the solution. This is illustrated in Figure 2.15: the cost function shown is x^4 ; the step illustrated moves from the original parameter value to the minimum of the parabola with first and second derivatives that match those of the cost function at the original point.

One obvious requirement of the Newton-Raphson method is that the Hessian matrix must be non-singular. If this is not the case, then $\mathbf{A}(\mathbf{x}_k)^{-1}$ will not be able to be evaluated, and the technique cannot be used.

Even if the full Hessian is not calculated, it can be beneficial to utilize *some* of the information from the second derivative matrix in the computation. For example, if a diagonal weighting matrix is utilized in order to force the cost function contours in the local region to be roughly circular, then the resulting weighted gradient step will move directly toward the solution, overshooting less and taking fewer steps to converge [36:34]. Thus, using a Hessian matrix with only diagonal terms, or a block-diagonal Hessian matrix, may speed convergence somewhat.

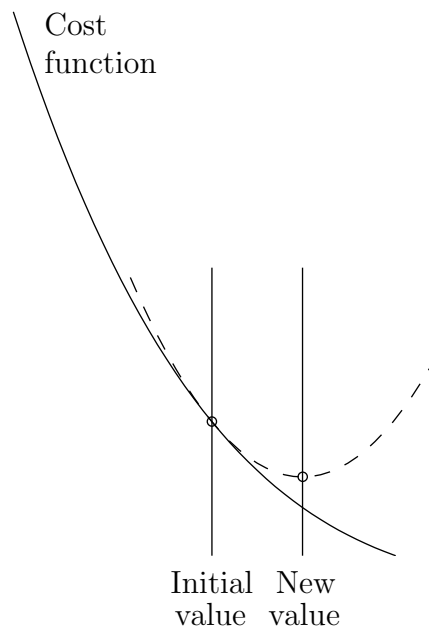


Figure 2.15. Operation of the Newton-Raphson algorithm: each step moves to be minimum of the local approximating parabola.

The various techniques described in these sections will converge to a minimum provided that the algorithm is commenced from within the region of convergence. The gradient method is guaranteed to converge to a local optimum from anywhere in the search space. The region of convergence for the Newton-Raphson method is a finite-sized convergence ball; diagonal or block-diagonal Newton-Raphson approximations will be somewhere between the two. If the cost function has a single global minimum and it increases from that point in every direction, then the solution found by the iterative algorithm is guaranteed to be the global minimum. The cost function forms defined in Chapter III do not have this characteristic, but rather they are extremely multi-modal, with many maxima and minima. In this situation, the algorithms will converge (assuming that the starting point supplied to the algorithm is inside the region of convergence) to a *local* minimum (most likely the minimum closest to the starting point given to the algorithm), and there is no guarantee that this point will indeed represent the *global* minimum.

III. Analysis

3.1 Introduction

As outlined in Section 1.2, the goal of this study is to develop techniques which are able to maintain a high fidelity representation of the PDF of target state, focusing on the efficiency of this representation. The most compact PDF representation in common use is that of a single Gaussian function; Section 3.2 examines some of the difficulties which are commonly experienced using such a coarse approximation.

Section 3.3 then develops an algorithm which aims to provide the best possible representation of the target state PDF using any given number of components in a Gaussian mixture. The algorithm is based on the minimization of a cost function; possible selections for this cost function are considered in Section 3.3.1. The cost function selected for our algorithm, the Integral Square Difference (ISD) cost, is examined in detail in Section 3.3.2, before iterative optimization techniques are applied in Section 3.3.3. Finally, it is apparent that iterative optimization of such a multi-modal function is highly dependent on the starting point provided to the algorithm; a methodology for deriving a near-optimal starting point is developed in Section 3.3.4.

3.2 PDA Bias and Coalescence

The JPDA algorithm is computationally desirable when compared to more modern MHT algorithms, as the tracking system is required to maintain only a single Gaussian PDF rather than a Gaussian mixture with a component corresponding to each hypothesis, with the number of hypotheses growing at an exponential rate. However, the original formulation of the JPDA algorithm exhibits significant difficulty in tracking closely-spaced targets. Hong, et al. [16, 19, 22] suggest that the difficulty in tracking closely-spaced targets exhibited by JPDA is due to a bias inherent in the algorithm, arising from the Kalman filter measurement update equation:

$$\hat{\mathbf{x}}_i(k|k) = \hat{\mathbf{x}}_i(k|k-1) - \mathbf{K}_i(k)[\bar{\mathbf{z}}_i(k) - \alpha_i(k)\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] \quad (3.1)$$

where $\bar{\mathbf{z}}_i(k)$ is the combined measurement for target i , $\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)$ is the predicted measurement for target i , and $\alpha_i(k)$ is the scaling factor which accounts for events under which the target is not detected and no update is performed.

Despite the suggestion above, if the estimate prior to incorporation of the measurement is unbiased, and the residual in Eq. (3.1) is zero-mean, then the updated estimate is guaranteed to be unbiased. To determine whether a bias will be introduced, we need to take the expected value of the residual. Denoting the expected value of the residual for target i as $\mathbf{b}_i(k)$, and expanding using the definitions of Section 2.5.7:

$$\begin{aligned} \mathbf{b}_i(k) &= E\{\bar{\mathbf{z}}_i(k) - \alpha_i(k)\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)\} \\ &= E\left\{\sum_l P\{\Theta_l(k)|\mathbf{Z}^k\}[\mathbf{z}_{m(\Theta_l(k),i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)]\right\} \end{aligned} \quad (3.2)$$

where the summation is taken over all joint events in which target i is hypothesized to have been detected, and $m(\Theta_l(k), i)$ is the measurement associated with target i under the event $\Theta_l(k)$.

The expectation of Eq. (3.2) could be taken over a number of different variables. Since we are concentrating on the bias arising from only a single processing cycle, the measurements from previous cycles are assumed known, and the expectation is taken only over the measurements from the processing cycle under consideration. Cong [16, 22] also takes the expectation over the *number of measurements* ($N_m(k)$) in the cycle under consideration. Observing that the number of measurements will be known *exactly* at run-time when this processing is performed, this would seem unnecessary, and furthermore since:

$$E_{x,y}\{g(x,y)\} = E_y\{E_x\{g(x,y)|y\}\} \quad (3.3)$$

as is derived by:

$$\begin{aligned}
E_y \{E_x \{g(x, y) | y\}\} &= \int \left[\int g(x, y) f\{x|y\} dx \right] f\{y\} dy \\
&= \iint g(x, y) f\{x|y\} f\{y\} dx dy \\
&= \iint g(x, y) f\{x, y\} dx dy \\
&= E_{x,y} \{g(x, y)\}
\end{aligned}$$

Thus, if the expression obtained by assuming conditioning on the number of measurements is shown to be *unbiased* (such that $\mathbf{b}_i(k) = \mathbf{0}$), then the equivalent result also taking expectation over the number of measurements will also be unbiased.

Evaluating Eq. (3.2) assuming conditioning on the previous measurement history and the number of measurements in the current cycle, and expanding $P\{\Theta_l(k) | \mathbf{Z}^k\}$ using Eq. (2.55), we obtain:

$$\begin{aligned}
\mathbf{b}_i(k) &= \int_{-\infty}^{\infty} \left\{ \sum_l P\{\Theta_l(k) | \mathbf{Z}^k\} [\mathbf{z}_{m(\Theta_l(k), i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] \right\} \cdot \\
&\quad \cdot f\{\mathbf{Z}_k | N_m(k), \mathbf{Z}^{k-1}\} d\mathbf{Z}_k \\
&= \int_{-\infty}^{\infty} \left\{ \sum_l \frac{f\{\mathbf{Z}_k | \Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} P\{\Theta_l(k) | N_m(k)\}}{f\{\mathbf{Z}_k | N_m(k), \mathbf{Z}^{k-1}\}} \cdot \right. \\
&\quad \left. \cdot [\mathbf{z}_{m(\Theta_l(k), i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] \right\} f\{\mathbf{Z}_k | N_m(k), \mathbf{Z}^{k-1}\} d\mathbf{Z}_k \\
&= \int_{-\infty}^{\infty} \left\{ \sum_l f\{\mathbf{Z}_k | \Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} P\{\Theta_l(k) | N_m(k)\} \cdot \right. \\
&\quad \left. \cdot [\mathbf{z}_{m(\Theta_l(k), i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] \right\} \cdot \\
&\quad \cdot \frac{1}{f\{\mathbf{Z}_k | N_m(k), \mathbf{Z}^{k-1}\}} f\{\mathbf{Z}_k | N_m(k), \mathbf{Z}^{k-1}\} d\mathbf{Z}_k \tag{3.4}
\end{aligned}$$

where, as in Section 2.2.3, the vector limits $(-\infty, \infty)$ remind us that the integration is to be performed over every element of the vector $\mathbf{Z}_k$. Cancelling the

$f\{\mathbf{Z}_k|N_m(k), \mathbf{Z}^{k-1}\}$ terms in the numerator and denominator, and exchanging the order of integration and summation:

$$\begin{aligned}
\mathbf{b}_i(k) &= \int_{-\infty}^{\infty} \left\{ \sum_l f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} P\{\Theta_l(k)|N_m(k)\} \cdot \right. \\
&\quad \left. \cdot [\mathbf{z}_{m(\Theta_l(k),i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] \right\} d\mathbf{Z}_k \\
&= \sum_l \left\{ P\{\Theta_l(k)|N_m(k)\} \cdot \right. \\
&\quad \left. \cdot \int_{-\infty}^{\infty} f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} [\mathbf{z}_{m(\Theta_l(k),i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] d\mathbf{Z}_k \right\}
\end{aligned} \tag{3.5}$$

The integral in Eq. (3.5) can then be broken up into the following difference:

$$\begin{aligned}
&\int_{-\infty}^{\infty} f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} [\mathbf{z}_{m(\Theta_l(k),i)}(k) - \mathbf{H}\hat{\mathbf{x}}_i(k|k-1)] d\mathbf{Z}_k \\
&= \int_{-\infty}^{\infty} \mathbf{z}_{m(\Theta_l(k),i)}(k) f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} d\mathbf{Z}_k \\
&\quad - \int_{-\infty}^{\infty} \mathbf{H}\hat{\mathbf{x}}_i(k|k-1) f\{\mathbf{Z}_k|\Theta_l(k), N_m(k), \mathbf{Z}^{k-1}\} d\mathbf{Z}_k
\end{aligned} \tag{3.6}$$

The first term in Eq. (3.6) amounts to the predicted mean of the measurement associated with target i under association event $\Theta_l(k)$, which is merely the measurement prediction $\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)$. The second term has no variables in $\mathbf{Z}_k$ other than the density itself, hence the integral of the density evaluates to unity, again leaving $\mathbf{H}\hat{\mathbf{x}}_i(k|k-1)$. These two terms obviously cancel, such that the bias of JPDA for target i is:

$$\begin{aligned}
\mathbf{b}_i(k) &= \sum_l P\{\Theta_l(k)|N_m(k)\} \cdot \mathbf{0} \\
&= \mathbf{0}
\end{aligned} \tag{3.7}$$

Thus it has been shown that, according to the measurement model presented in Section 2.5.2, JPDA is in fact *unbiased*. Following identical steps for the CPDA algorithm, allowing correlation between targets, will arrive at a similar result. The scarce details provided in [16, 22] for the evaluation of the JPDA bias make it very difficult to compare the result derived above with those previously published. One mistake which is easily made¹ is to treat the denominator of Eq. (2.55) as a constant (with respect to the measurement values), thereby evaluating the integral of Eq. (3.5) by the approximation:

$$\int \frac{f_1(x)}{f_2(x)} dx \approx \frac{\int f_1(x) dx}{\int f_2(x) dx} \quad (3.8)$$

In this case, the measurement PDF due to the expectation operation is not cancelled with the measurement PDF in the denominator of the event probability. Instead, the denominator of the event probability is treated as a constant and neglected as per the development of Section 2.5.2,² leaving the product of the two PDFs, which can be evaluated with some difficulty. The cancellation of the measurement PDFs would not have been possible if the expectation operation had been extended initially across the number of measurements — this further suggests that this error may have been made in [16, 22]. However, the expression of Eq. (3.8) is without mathematical basis, hence one would expect that the robustness of any apparent performance gain induced by introducing the approximation would be highly questionable. The results obtained in this study using this approximation appeared promising in some areas, but any overall improvement in any realistic scenario was not evident.

The puzzling aspect of this result is that the JPDA algorithm *does* exhibit a form of “bias” when targets are closely spaced: to such an extent that the two estimates can essentially converge to the mid-point between the targets in scenarios in which targets are closely spaced for extended periods of time. This phenomenon,

¹Indeed a substantial amount of time was lost during this study due to this very error.

²In Section 2.5.2, it was argued that the denominator is constant across all *association events*: it is *not* constant across measurements values, which are the variables of integration in this expression.

referred to as coalescence, was examined in [9, 11, 12], in which it was concluded that [12:256]:

...the conditional density of the targets' joint state has a particular multimodality: in addition to the local optimum for the nonswapped tracks, often other local optima exist for track swap possibilities. The approach of centering a Gaussian optimally (in the MMSE sense) between these local optima implies a preference to track coalescence over track swap.

Hence this multi-modality explains how the JPDA can be *unbiased* yet still have difficulties with track coalescence. As further suggested by Blom and Bloem [12], the major source of uncertainty in this particular scenario is the *identity* of the target. In this case, there will be two primary joint hypotheses: one will be correct, and the other will be identical, but with the two targets exchanged (which is equally valid as far as the tracker is concerned since measurements are not labelled). Thus for all the system knows (from the set of measurements it has been given), either of these two primary joint hypotheses *could be* the correct one, hence the best *unbiased* answer in a minimum mean square error sense is to “hedge your bets” either way. In other words, the system no longer knows which target is which, so it simply takes the average of the two possibilities, tracking the mid-point between them. As highlighted by Blom and Bloem, in virtually any practical situation it is more desirable to track the targets with the incorrect identity than to track the mid-point between the targets, hence it is desirable to *force* the system to choose one joint hypothesis or the other — irrespective of whether the correct hypothesis is selected, the result will be preferable over allowing the tracks to coalesce. This is effectively what is done by the JPDA* and CPDA* algorithms³ presented in [9, 11, 12]: for each proposed set of detected targets and target-originated measurements, only the best association event is maintained, hence avoiding the situation described above.

The problem of track coalescence can be illustrated by considering the position of two targets in joint state space, as per the example discussed in Section 2.5.8. The

³As proposed in [12:248], the ‘*’ notation is short for “track-coalescence-avoiding”.

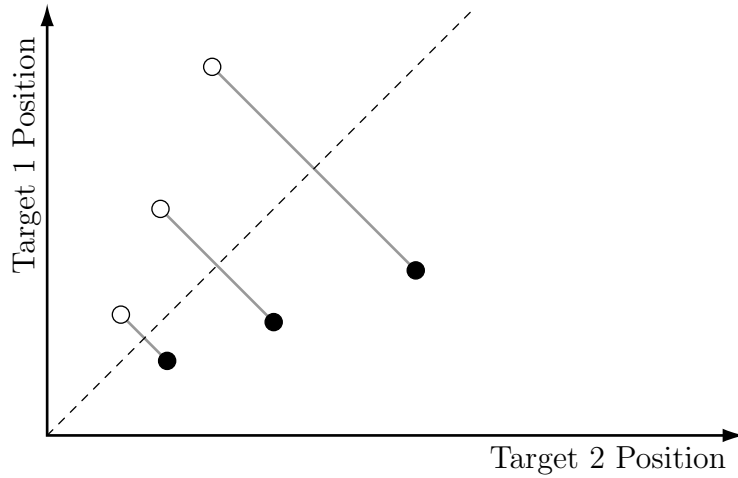


Figure 3.1. Pairs of equally valid tracking solutions in joint target state space.

difficulty of the tracking problem is that the measurements are unlabelled. Accordingly, as far as the radar system is concerned, there are two equally valid tracking solutions: the correct solution, and the same situation with the identity of the two targets exchanged. These solutions are illustrated in the depiction of joint target state space shown in Figure 3.1: for each point in joint target state space (such as the sample points shown by ‘•’), the reflection in the 45° line (mapping the sample points to the locations shown by ‘o’) represents an equally valid tracking solution, with the identity of the two targets exchanged.

When the estimated target position is far from the 45° line, the presence of this alternative tracking solution does not affect the performance of the algorithm, because the alternative solution will be weighted with a very low probability.⁴ If the targets move close together, then the joint state moves close to the 45° line, which brings the two tracking solutions close together, increasing the probability of the alternative solution. As this probability increases, the weighted mean estimate is drawn increasingly towards the centroid of the two tracking solutions, resulting

⁴The weight applied to the alternative solution will be zero if the solution does not satisfy the gating equations.

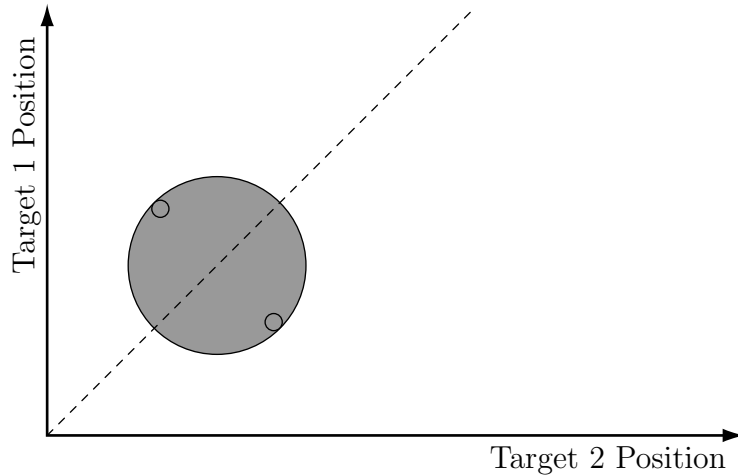


Figure 3.2. Coalesced joint target state estimate and covariance using JPDA algorithm.

in a single coalesced track between the two possibilities. When the targets begin to separate after being close together (as in the case of two crossing tracks), the tracking algorithm will attempt, as far as the PDF representation is able, to fit a single Gaussian component to the two hypotheses. If the two hypotheses are equally likely, then the resulting estimate will be between the two tracking solutions, with a covariance that attempts to encompass both identity possibilities. Such a case is illustrated in Figure 3.2: this is a typical example of coalescence using the JPDA algorithm.

As discussed in Section 2.5.8, CPDA introduces correlation between targets in an effort to improve performance when tracking closely spaced targets, but the inclusion of correlation actually causes track coalescence to worsen. This phenomenon is very difficult to explain. Blom and Bloem [12:254] suggest that it is caused by the strong correlation which develops between targets, making the algorithm prefer to keep estimates between competing measurements. However, as discussed in Section 2.5.8, the correlation which develops is inevitably *negative* correlation, which would tend to prefer to separate the targets (as compared to JPDA), rather than keeping them together.

Considering the discussion above, the conclusion of this study is similar to that of Blom and Bloem: the reduced coalescence performance of the CPDA algorithm is caused by correlation arising from the two tracking hypotheses with exchanged target identity. However, it is *not* that the correlation makes the tracker attempt to keep the targets together that causes the difficulty, but rather that correlation *allows the system to keep both tracking hypotheses within its field of view*. The correlation between the target state vectors operates like blinders on a horse, concentrating the field of view of the algorithm on the area containing the two primary association possibilities, and excluding the distraction caused by other less likely association hypotheses. It is this very distraction that rescues JPDA from coalescence: a comparatively lower weighting will be applied to the two major modalities, hence other association hypotheses will tend to enter, and the estimate will be pulled away from the 45° line, resolving the coalescence.

The two PDFs in Figure 3.3 illustrate an example in which allowing correlation between targets results in a high amount of correlation, with the correlation coefficient at -0.9 , as shown in Figure 3.3(b). Figure 3.3(a) shows the same region of the same PDF, where the correlation between targets has been discarded. Both PDFs are clipped to a maximum value of 0.005 to illustrate the relative size and shape of equally likely contours of the joint target state. As can be seen, the skewness produced by the correlation prolongs the shape of the function towards the two valid tracking possibilities, hence ensuring that both hypotheses remain relatively likely as compared to other possible association events.

Figure 3.4 is a snapshot of the joint target position in a Monte Carlo simulation. The scenario is a two-dimensional tracking problem with two slowly-crossing targets, crossing in the y axis. The dynamics noise was set close to zero in the x axis, essentially making the problem one-dimensional. The diagram shows the joint position of the two targets after coalescence has occurred (both position and velocity are estimated). The ‘ $\times$ ’ marks indicate the association hypotheses, which are

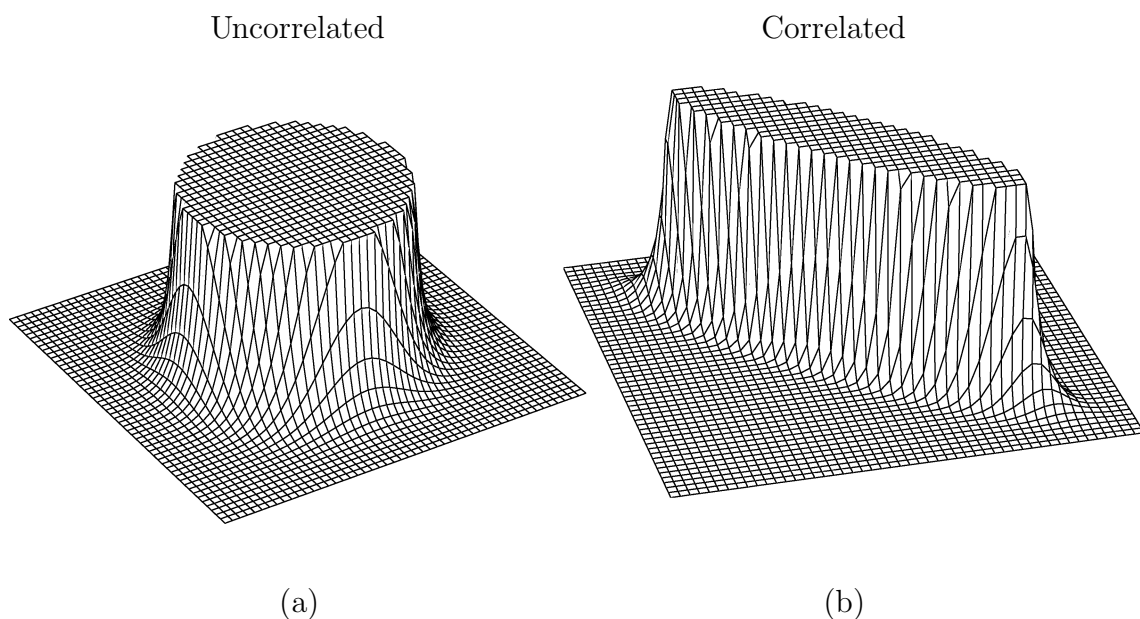


Figure 3.3. Joint target state PDF, (a) disallowing correlation between targets, and (b) allowing correlation between targets (correlation coefficient = -0.9).

combined into the single estimate marked by ‘o’. The covariance of the combined estimate with correlation is illustrated through the error ellipse, demonstrating the high degree of negative correlation between targets which results from combining the hypotheses. If correlation between targets was not admitted, then this ellipse would be roughly *circular*, and the probability of the association hypotheses would be drawn towards the central estimates, before being dragged off by one primary hypothesis or the other. The correlation allows the two primary hypotheses, which are identical other than a switch in tracks, to remain probable, drawing probability away from incorrect hypotheses, which would resolve the deadlock.

3.3 Gaussian Mixture Reduction

As discussed in Section 2.5, the common methods utilized in modern target tracking techniques apply different simplifications to the PDF of target state given the set of received measurements. Techniques such as JPDA [4:310–319] and

GNN [7:338–342] perform a vast simplification, reducing the entire Gaussian mixture to a single Gaussian component, maintaining independence between targets. Techniques including JPDAC [4:328–329], CPDA [12] and Maximum Likelihood methods [25] also reduce the Gaussian mixture PDF to a single component, but by allowing correlation between the target states, information about target relationships is maintained.

While Salmond’s mixture reduction algorithms [44–47] and Pao’s multiple target extension [38] are able to retain any number of Gaussian mixture components, as discussed in Section 2.5.11.2, the marginalization of target PDFs results in loss of all information concerning the relationships between targets, forcing independence between targets. The common MHT implementations permit this dependence by maintaining hypothesis compatibility listings, but the *ad hoc* simplification methods employed to merge and prune hypotheses (such as those described in [6:292–294]) potentially limit the usefulness of the retained mixture components.

Given the extreme rate of growth of hypotheses that results from the MHT algorithm, some form of hypothesis control is unquestionably necessary. The ideal implementation would be to maintain the set of hypotheses which is small enough to be readily computable by the system in question, yet carries the information about the original target PDF to the highest fidelity possible.

To proceed, we first define the original joint PDF of target state, containing $N_h(k)$ joint hypotheses as to the possible locations of the targets, as:

$$f\{\mathbf{X}(k)|\mathbf{\Omega}_{N_h}(k)\}$$

where $\mathbf{\Omega}_{N_h}(k)$ represents the parameters of the $N_h(k)$ hypotheses derived from the measurements up to the current sample period. Our goal is thus to reduce these $N_h(k)$ hypotheses to a simplified representation, containing $N_r(k)$ hypotheses,⁵ re-

⁵The subscript ‘*r*’ denoting ‘reduced’.

sulting in the simplified PDF:

$$f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}$$

where $\bar{\boldsymbol{\Omega}}_{N_r}(k)$ represents the reduced set of parameters, containing, as closely as possible, the same information as the original set $\boldsymbol{\Omega}_{N_h}(k)$.

3.3.1 Cost Measures. In order to simplify the PDF while making the smallest possible overall change, the first step is to select a scalar cost function which measures the difference between two PDFs in order to evaluate whether one PDF approximation is “better” than another. Once such a function has been defined, a wide variety of well-understood optimization methods can be applied to determine the parameters of the reduced PDF which minimize the cost (and hence loss of fidelity) caused by the reduction.

One can conceive of any number of constructs which would serve as an effective cost function. Two which have been previously proposed were discussed in Section 2.5.11.1, the Kolmogorov variational distance and the Bhattacharyya coefficient.

3.3.1.1 Bhattacharyya Distance. The Bhattacharyya coefficient and the closely related Bhattacharyya distance are a measure of similarity of two PDFs, made popular by [23]. The definition of the Bhattacharyya coefficient is as given by Eq. (2.75):

$$J_B = \int \sqrt{f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}}d\mathbf{X}(k)$$

while the Bhattacharyya distance is given by $B_D = -\ln J_B$.

The application in [23] is the maximization of the distance between the two distributions, such as that in the communications problem of determining whether the transmitted bit was a ‘0’ or a ‘1’. Comparing the expected distribution of the received signal when a ‘0’ is transmitted to that when a ‘1’ is transmitted results in information closely related to the probability of error for the system. If this

comparison is performed utilizing a good distance measure, then optimization of the system to maximize the distance between the two distributions will effectively minimize the probability of error.

Computation of the Bhattacharyya distance is an easy matter for the case of two single Gaussian PDFs as the measure takes the product of the two PDFs, which results directly in another Gaussian (with scaled volume) after completing the square (similarly to the development in Appendix A.1). As all terms are multiplicative, the square root can be taken of each term individually, thus the result is indeed a Gaussian form.

In the case of Gaussian mixtures, the product of the two PDFs results in a sum with a term for each pairing of mixture components from the two PDFs, as per the cost function component J_{hr} , expanded in Eq. (3.22). The square root of this expression will not be able to be simplified in general, and intractable numerical integration methods will be necessary.

3.3.1.2 Kolmogorov Variational Distance. The Kolmogorov variational distance, defined in Eq. (2.74), is the integral of the absolute difference between the two probability density functions:

$$J_K = \int |f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} - f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}| d\mathbf{X}(k)$$

The measure provides an indication of the amount of probability mass by which the two PDFs differ. If the two functions are identical throughout the probability space, then the cost will be zero. Conversely, if the two functions are entirely disjoint, then the difference will merely be the sum of the integral of each PDF individually (each evaluating to unity).

Like the Bhattacharyya distance, the Kolmogorov variational distance is not easily evaluated. The absolute value function requires piecewise definition, thus the

integral must be divided into the two portions: where the original PDF is larger in value than the approximation, and where the approximation is larger in value than the original PDF. While the integral over an entire Gaussian function is easily evaluated (indeed it is unity by definition), the integral over an arbitrary portion of a Gaussian function is extremely difficult to evaluate, and resorting to numerical techniques or even Monte Carlo methods will be inevitable.

3.3.1.3 Integral Square Difference Measure. The use of the absolute function in the Kolmogorov variational distance provides an even nonlinearity, to force positive cost and negative cost to be handled identically. Another nonlinearity which could be used in place of the absolute value is a square function, resulting in the following modified cost:

$$J_S = \int (f\{\mathbf{X}(k)|\Omega_{N_h}(k)\} - f\{\mathbf{X}(k)|\bar{\Omega}_{N_r}(k)\})^2 d\mathbf{X}(k) \quad (3.9)$$

where the subscript ‘ S ’ is used to denote the square nonlinearity. This measure will be referred to as the Integral Square Difference (ISD) cost function. The nonlinearity could be replaced with any even integer power, where higher powers will tend to treat areas of larger error with increasingly higher weight than those of lower error. In the limit, as the power approaches infinity (in an even sense), the cost function will apply all priority to the largest error point, tending to *minimize* the *maximum error* committed by the approximation. This is illustrated in Figure 3.5, which compares the absolute value function to other even nonlinear functions, x^2 , x^4 and x^6 . To interpret the diagram, consider the case in which the maximum error of the approximation is unity, which each function maps to the same value. The difference between the various nonlinearities is then the amount of weight applied to points with comparatively lower error: the absolute value function applies weight which reduces linearly to lower errors, while x^2 , x^4 and x^6 apply weights which decrease at a faster rate as the order of the nonlinearity increases.

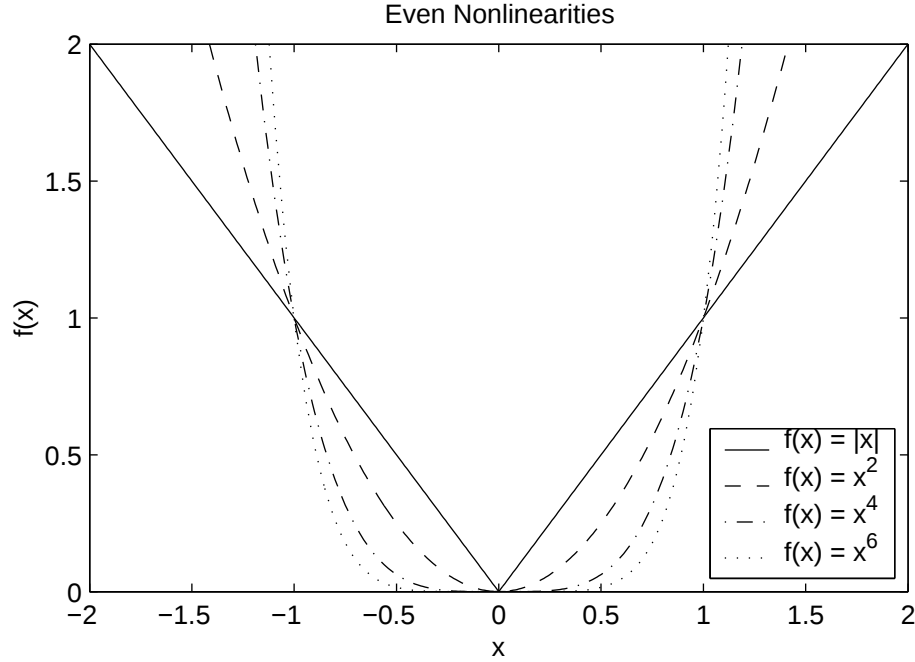


Figure 3.5. Comparison of various even nonlinearities.

The implication of Figure 3.5 to the ISD cost function is that the ISD measure will behave very similarly to the Kolmogorov variational distance, but comparatively higher weight will be applied to areas of larger error, while comparatively lower weight will be applied to areas with lower error. If the original PDF contains components with very small variances producing large peaks alongside components of similar probability weight with larger variances producing smaller and flatter peaks, then one can expect that the square nonlinearity will give higher cost to the lower variance components (with higher peaks) rather than the higher variance components (with lower peaks), where the Kolmogorov variational distance would treat the two identically.

3.3.1.4 Maximum Likelihood Measure. The field of Maximum Likelihood estimation is based upon finding the parameters of a known distribution form that maximize the likelihood of receiving a set of data, assuming that it was drawn from the given form of distribution. If a single datum vector $\mathbf{z}$ is received, then the

most likely value of the parameter $\boldsymbol{\theta}$ can be found from [48:210]:

$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} f\{\mathbf{z}|\boldsymbol{\theta}\} \quad (3.10)$$

It is often more convenient to perform this optimization in terms of the natural logarithm of the density rather than the density itself. Since $\log x$ is a monotonically increasing function for $x > 0$, the peak of the logarithm of the density occurs at the same location as the peak of the density itself. The logarithm of the density is commonly referred to as the log-likelihood function:

$$L(\boldsymbol{\theta}, \mathbf{z}) = \log f\{\mathbf{z}|\boldsymbol{\theta}\} \quad (3.11)$$

Assuming that a set of data vectors $\mathbf{Z} = \{\mathbf{z}_1 \dots \mathbf{z}_n\}$ were drawn from the density of interest such that they are independent and identically distributed, the joint density of the data may be written as:

$$f\{\mathbf{Z}|\boldsymbol{\theta}\} = \prod_{i=1}^n f\{\mathbf{z}_i|\boldsymbol{\theta}\} \quad (3.12)$$

such that the most likely value for the parameter vector $\boldsymbol{\theta}$ may be found by:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg \max_{\boldsymbol{\theta}} f\{\mathbf{Z}|\boldsymbol{\theta}\} \\ &= \arg \max_{\boldsymbol{\theta}} \prod_{i=1}^n f\{\mathbf{z}_i|\boldsymbol{\theta}\} \end{aligned} \quad (3.13)$$

This expression is conveniently simplified using the logarithm operation as discussed above:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg \max_{\boldsymbol{\theta}} \log f\{\mathbf{Z}|\boldsymbol{\theta}\} \\
&= \arg \max_{\boldsymbol{\theta}} \log \prod_{i=1}^n f\{\mathbf{z}_i|\boldsymbol{\theta}\} \\
&= \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log f\{\mathbf{z}_i|\boldsymbol{\theta}\} \\
&= \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n L(\boldsymbol{\theta}, \mathbf{z}_i)
\end{aligned} \tag{3.14}$$

The expression of Eq. (3.14) is used to derive the Expectation Maximization (EM) algorithm for determining the parameters of the Gaussian mixture which best match a given set of data [41]. As the data sample increases in size, it provides a more detailed representation of the true data PDF, resulting in a set of parameters which better represents the distribution. In the limit as the sample size approaches infinity, the data sample converges to represent the true PDF:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg \max_{\boldsymbol{\theta}} \lim_{n \rightarrow \infty} \sum_{i=1}^n \log f\{\mathbf{z}_i|\boldsymbol{\theta}\} \\
&= \arg \max_{\boldsymbol{\theta}} \int f\{\mathbf{z}\} \log f\{\mathbf{z}|\boldsymbol{\theta}\} d\mathbf{z}
\end{aligned} \tag{3.15}$$

The above derivation, resulting in Eq. (3.15), provides a means for finding the parameters $\boldsymbol{\theta}$ of the density form $f\{\mathbf{z}|\boldsymbol{\theta}\}$ which best match the true data density $f\{\mathbf{z}\}$. This function is synonymous to that required for this study: we want to solve for the parameters of a Gaussian mixture which provide the best fit to a mixture of higher complexity. Hence a natural measure of the fit of the reduced density using parameters $\bar{\boldsymbol{\Omega}}_{N_r}(k)$ to the higher-order density represented by parameters $\boldsymbol{\Omega}_{N_h}(k)$ is

provided by the cost function defined by:

$$J_{ML} = \int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} \log f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\} d\mathbf{X}(k) \quad (3.16)$$

Following from this interpretation, the author considers this expression to be the ideal cost function for the application. However, the logarithm of a Gaussian mixture is not able to be simplified, hence the cost function is unable to be evaluated without numerical integration or approximation.

The expression of Eq. (3.16) somewhat resembles the divergence as defined in [29:6]:

$$\begin{aligned} J(1,2) &= \int (f_1\{x\} - f_2\{x\}) \log \frac{f_1\{x\}}{f_2\{x\}} dx \\ &= \int (f_1\{x\} \log f_1\{x\} - f_1\{x\} \log f_2\{x\} \\ &\quad - f_2\{x\} \log f_1\{x\} + f_2\{x\} \log f_2\{x\}) dx \end{aligned} \quad (3.17)$$

The divergence is a measure of the difficulty of discriminating from which distribution ($f_1\{x\}$ or $f_2\{x\}$) a sample vector was drawn, and hence a measure of the similarity between them. Comparing Eq. (3.17) to Eq. (3.16), the divergence consists of four terms: the information content (entropy) [32:166] of each distribution, and the “cross-entropy” terms, one of which can be seen to be the Maximum Likelihood cost function, as defined in Eq. (3.16).

3.3.2 Analysis of Integral Square Difference Measure. After analyzing the various cost function options, the ISD distance measure is the only option which leads to a cost function that can be evaluated in closed form, without requiring expensive numerical integration. Furthermore, as will be seen in Section 3.3.3, the *derivatives* of the cost function with respect to each of the parameters can *also* be evaluated in closed form, allowing iterative optimization techniques to be employed efficiently.

On an intuitive level, the cost function does not have the appealing probability mass interpretation of the Kolmogorov variational distance, or the optimal reduced parameter fit interpretation of the Maximum Likelihood measure; however, it is a reasonable measure of the distance between the two PDFs. The function reaches its lowest possible value of zero when the two PDFs are identical throughout the space,⁶ and its maximum possible value when the two PDFs are completely disjoint (i.e., when the product of the two PDFs is zero at every point in the space).

The ISD distance can be seen to be very similar in form to the Kolmogorov variational distance, with the only disparity being the squaring of the integrand in the former. Considering the impact of this squaring, it will tend to treat regions in which the difference between the PDFs is smaller with lower weight than the absolute difference method, and regions in which the difference is larger with higher weight. Accordingly, we can expect that the result obtained using the ISD method will be more averse to committing larger errors, and less averse to committing smaller value errors, even if the volume contained in the errors is the same. Considering this in the context of a Gaussian mixture PDF, we expect that the ISD measure will give more consideration to mixture components with lower variance (and thus higher value) over mixture components with higher variance, even if the probability weights are identical.

Expanding the ISD distance measure equation yields the following terms:

$$\begin{aligned}
J_S = & \int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}^2 + \\
& - 2f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\} + \\
& + f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}^2 d\mathbf{X}(k)
\end{aligned} \tag{3.18}$$

The three terms of Eq. (3.18) each have their own interpretation. The first represents the *self-likeness* of the original PDF — this term will be larger if the PDF is more

⁶Except at points of zero probability mass.

concentrated in space, and smaller if the PDF is more spread out. The second represents the *cross-likeness* of the original PDF to the new PDF. This term is critical to the function as it directly measures the volume of probability mass⁷ that the two functions have in common. The final term is the *self-likeness* of the reduced PDF, possessing similar characteristics to the other self-likeness term. The cross-likeness term serves to balance the two self-likeness terms, cancelling the overall cost function value to zero if the two functions are identical, and increasing the overall cost function value as the difference between the functions increases.

Defining these three components as:

$$\begin{aligned}
J_{hr} &= \int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}d\mathbf{X}(k) \\
J_{rr} &= \int f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}^2d\mathbf{X}(k) \\
J_{hh} &= \int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}^2d\mathbf{X}(k)
\end{aligned} \tag{3.19}$$

we can then write Eq. (3.18) as:

$$J_S = J_{hh} - 2J_{hr} + J_{rr} \tag{3.20}$$

In the problem of interest, the two PDFs are both Gaussian mixtures, which can be expanded as:

$$\begin{aligned}
f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} &= \sum_{i=1}^{N_h(k)} p_i \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_i, \mathbf{P}_i\} \\
f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\} &= \sum_{i=1}^{N_r(k)} \bar{p}_i \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_i, \bar{\mathbf{P}}_i\}
\end{aligned} \tag{3.21}$$

⁷Although it is in square units rather than the units typically associated with probability mass measure.

where $\{p_i, \boldsymbol{\mu}_i, \mathbf{P}_i\}$ are the weights, means and covariances of the Gaussian functions composing the mixture for original PDF, and $\{\bar{p}_i, \bar{\boldsymbol{\mu}}_i, \bar{\mathbf{P}}_i\}$ are the same parameters of the reduced PDF. Substituting these expressions into Eq. (3.19):

$$\begin{aligned}
J_{hr} &= \int \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_i, \mathbf{P}_i\} \bar{p}_j \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_j\} d\mathbf{X}(k) \\
J_{rr} &= \int \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_i, \bar{\mathbf{P}}_i\} \bar{p}_j \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_j\} d\mathbf{X}(k) \\
J_{hh} &= \int \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_i, \mathbf{P}_i\} p_j \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_j, \mathbf{P}_j\} d\mathbf{X}(k) \quad (3.22)
\end{aligned}$$

By linearity of the integration operation, the summation and integration of each of the expressions in Eq. (3.22) can be reversed, resulting in:

$$\begin{aligned}
J_{hr} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \bar{p}_j \int \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_i, \mathbf{P}_i\} \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_j\} d\mathbf{X}(k) \\
J_{rr} &= \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \bar{p}_j \int \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_i, \bar{\mathbf{P}}_i\} \mathcal{N}\{\mathbf{X}; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_j\} d\mathbf{X}(k) \\
J_{hh} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \int \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_i, \mathbf{P}_i\} \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_j, \mathbf{P}_j\} d\mathbf{X}(k) \quad (3.23)
\end{aligned}$$

Following from the derivation of Appendix A.1, the product of two Gaussian PDFs, which forms the basic building block of Eq. (3.23), can be simplified to the following form:

$$\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_1, \mathbf{P}_1\} \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_2, \mathbf{P}_2\} = \alpha \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_3, \mathbf{P}_3\} \quad (3.24)$$

where α , $\boldsymbol{\mu}_3$ and $\mathbf{P}_3$ are as given by Eq. (A.13):

$$\begin{aligned}\alpha &= \mathcal{N}\{\boldsymbol{\mu}_1; \boldsymbol{\mu}_2, \mathbf{P}_1 + \mathbf{P}_2\} \\ \mathbf{P}_3 &= (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1} \\ \boldsymbol{\mu}_3 &= \mathbf{P}_3(\mathbf{P}_1^{-1}\boldsymbol{\mu}_1 + \mathbf{P}_2^{-1}\boldsymbol{\mu}_2)\end{aligned}$$

Substituting this simplification into the expressions of Eq. (3.23) and noticing that the integral over a Gaussian PDF evaluates to unity, we find:

$$\begin{aligned}J_{hr} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \bar{p}_j \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \bar{\mathbf{P}}_j\} \int \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_a, \mathbf{P}_a\} d\mathbf{X}(k) \\ &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i \bar{p}_j \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \bar{\mathbf{P}}_j\} \\ J_{rr} &= \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \bar{p}_j \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_i + \bar{\mathbf{P}}_j\} \int \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_b, \mathbf{P}_b\} d\mathbf{X}(k) \\ &= \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} \bar{p}_i \bar{p}_j \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \bar{\mathbf{P}}_i + \bar{\mathbf{P}}_j\} \\ J_{hh} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \mathcal{N}\{\boldsymbol{\mu}_i; \boldsymbol{\mu}_j, \mathbf{P}_i + \mathbf{P}_j\} \int \mathcal{N}\{\mathbf{X}; \boldsymbol{\mu}_c, \mathbf{P}_c\} d\mathbf{X}(k) \\ &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \mathcal{N}\{\boldsymbol{\mu}_i; \boldsymbol{\mu}_j, \mathbf{P}_i + \mathbf{P}_j\} \tag{3.25}\end{aligned}$$

where $\boldsymbol{\mu}_a$, $\boldsymbol{\mu}_b$, $\boldsymbol{\mu}_c$, $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$, are the combined means and covariances of the respective Gaussian component pairs from Eq. (3.23), calculated as according to Eq. (3.24). Interpreting Eqs. (3.20) and (3.25), the cost function consists of the sum of similarity measures of all pairs of two components from the original mixture, plus similarity measures of all pairs of two components from the reduced mixture, balanced by the sum of similarity measures of all pairs of one component from the original mixture and one component from the reduced mixture.

3.3.2.1 Normalization. While the ISD cost function will have a minimum value of zero (corresponding to the case in which the PDFs are identical), the peak value of the cost (corresponding to the case in which the PDFs are essentially disjoint) will vary depending on the PDFs under consideration. If it is desirable for the cost function to be bounded, it could be normalized using an expression such as:

$$\begin{aligned}
J_{S'} &= \frac{\int (f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} - f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\})^2 d\mathbf{X}(k)}{\int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}^2 + f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}^2 d\mathbf{X}(k)} \\
&= 1 - 2 \frac{\int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\} f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\} d\mathbf{X}(k)}{\int f\{\mathbf{X}(k)|\boldsymbol{\Omega}_{N_h}(k)\}^2 + f\{\mathbf{X}(k)|\bar{\boldsymbol{\Omega}}_{N_r}(k)\}^2 d\mathbf{X}(k)} \quad (3.26)
\end{aligned}$$

which will result in a function which is bounded between zero and one, a desirable characteristic if a fixed threshold is to be utilized to limit the maximum allowable cost incurred by the PDF reduction. In this study, the reduction was performed until the desired number of components was achieved, hence bounding was not utilized.

3.3.3 Iterative Optimization. The cost function described in Section 3.3.2 provides a measure of the dissimilarity between two Gaussian mixtures, and has the following desirable characteristics:

1. The equation for J_S can be evaluated completely in closed form, resulting in a sum of multivariate Gaussian functions with one term for each pairing of components in the original and reduced mixtures.
2. The resulting closed form evaluation is continuously differentiable, hence standard gradient-based optimization techniques can be employed.
3. As we will see in the following pages, the expressions for the first gradient of the cost function can also be written in closed form using standard vector-matrix notation, again simplifying the employment of gradient-based iterative optimization techniques.

Thus, by careful selection of cost function, we are able to apply the optimization methods described in Section 2.6 to find the parameters of the reduced order Gaussian mixture that provide the best fit to the higher order function. The parameters to be optimized are (recalling Eq. (3.21)):

1. The probability weights of the reduced mixture, $\{\bar{p}_i\}$.
2. The mean vectors of the reduced mixture, $\{\bar{\boldsymbol{\mu}}_i\}$.
3. The covariance matrices of the reduced mixture, $\{\bar{\mathbf{P}}_i\}$.

In order to produce a valid Gaussian mixture as an output, there are three constraints for the optimization:

1. The probability weights must be non-negative: $\bar{p}_i \geq 0 \forall i$.
2. The probability weights must sum to unity: $\sum_i \bar{p}_i = 1$.
3. The covariance matrices must be positive-definite: $\mathbf{x}^T \bar{\mathbf{P}}_i \mathbf{x} > 0 \forall i, \mathbf{x}$.

Such constraints complicate the optimization greatly, requiring the addition of parameters such as Lagrange multipliers [36:152–153]. If the optimization is re-posed in terms of transformed parameters such that these constraints are guaranteed to be satisfied, this complexity is completely avoided, and much simpler *unconstrained* optimization methods can be employed. One set of transformations which produces this result is:

$$\begin{aligned}\bar{p}_i &= \frac{q_i^2}{\sum_j q_j^2} \\ \bar{\mathbf{P}}_i &= \mathbf{L}_i \mathbf{L}_i^T\end{aligned}\tag{3.27}$$

The first line in Eq. (3.27) replaces the probability weights $\{\bar{p}_i\}$ with a the set of functions of $\{q_i\}$, which, for the optimization performed in terms of these transformed variables, guarantees that the resultant $\{\bar{p}_i\}$ set will be non-negative

and sum to unity. The second line of Eq. (3.27) replaces each covariance matrix with a matrix square, a form which is guaranteed to produce a positive *semi*-definite matrix [52:333]. To ensure positive definiteness (i.e., no zero eigenvalues), we rely on the premise that, if the original mixture to which we are fitting the reduced-order density has no singular covariance components, then the optimization is unlikely to pull towards such a solution. In other words, the positive definiteness of the solution is not strictly guaranteed, but this is unlikely to cause difficulties in any physically-motivated problem.

To commence the iteration, the transformed parameters can be determined from the initial values of the reduced parameters by the following relationships:

$$\begin{aligned} q_i &= \sqrt{p_i} \\ \mathbf{L}_i &= \sqrt[{}^c]{\mathbf{P}_i} \end{aligned} \tag{3.28}$$

where $\sqrt[{}^c]{\mathbf{P}_i}$ denotes the Cholesky square root, as defined in [34:370].

After some experimentation, it was decided that the constraint for the probability weights to sum to unity was unnecessary. To understand this, consider the problem in which a density that has several peaks, all of which are separated from each other, is to be reduced to such an extent that not all of these peaks can be modelled. If the probability weights are constrained such that they must sum to unity, then the weights of the remainder of the components must be increased to take up the weight of the components which are no longer being modelled, and this increase in the remaining weights will create additional error in the area of the remaining components. Thus, if we want the only cost associated with the deletion of a component to be the error induced in the region of that component, then the other components weights *should not be forced to be adjusted*. Accordingly, during the optimization process, we allow the weights be de-normalized, drifting to whatever total value (virtually always *close* to unity) provides the best fit to the original

function. Normalization is then applied as a final step after the optimization process has been completed.

Following this discussion, the transformation of the probability weights of Eq. (3.27) was changed to:

$$\bar{p}_i = q_i^2 \quad (3.29)$$

and the initial value of the $\{q_i\}$ parameters remains as per Eq. (3.28). The cost function must then be re-written in terms of the transformed parameters, resulting in the following expression:

$$J_S = J_{hh} - 2J_{hr} + J_{rr}$$

where:

$$\begin{aligned} J_{hr} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_r(k)} p_i q_j^2 \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} \\ J_{rr} &= \sum_{i=1}^{N_r(k)} \sum_{j=1}^{N_r(k)} q_i^2 q_j^2 \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} \\ J_{hh} &= \sum_{i=1}^{N_h(k)} \sum_{j=1}^{N_h(k)} p_i p_j \mathcal{N}\{\boldsymbol{\mu}_i; \boldsymbol{\mu}_j, \mathbf{P}_i + \mathbf{P}_j\} \end{aligned} \quad (3.30)$$

In the following sections, the derivatives of these components, J_{hr} , J_{rr} and J_{hh} , are calculated separately with respect to each parameter. The J_{hh} equation contains only components from the original mixture, hence the derivative of it with respect to parameters of the reduced mixture evaluates to zero.

3.3.3.1 Derivatives with respect to weights.

The expressions in Eq. (3.31) below show the partial derivatives of the cost function components J_{hr}

and J_{rr} with respect to the probability weights $\{q_j\}$:

$$\begin{aligned}
\frac{\partial J_{hr}}{\partial q_j} &= \sum_{i=1}^{N_h(k)} p_i \cdot 2q_j \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} \\
&= 2q_j \sum_{i=1}^{N_h(k)} p_i \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} \\
\frac{\partial J_{rr}}{\partial q_j} &= 2 \sum_{\substack{i=1 \\ i \neq j}}^{N_r(k)} q_i^2 \cdot 2q_j \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} \\
&\quad + 4q_j^3 \mathcal{N}\{\bar{\boldsymbol{\mu}}_j; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_j \mathbf{L}_j^T + \mathbf{L}_j \mathbf{L}_j^T\} \\
&= 4q_j \sum_{i=1}^{N_r(k)} q_i^2 \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} \tag{3.31}
\end{aligned}$$

3.3.3.2 Derivatives with respect to means. The derivative of a scalar with respect to a vector is a vector in which each component is equal to the derivative of the scalar with respect to the corresponding component of the vector:

$$\left\{ \frac{\partial J}{\partial \boldsymbol{\mu}} \right\}_i = \frac{\partial J}{\partial \{\boldsymbol{\mu}\}_i} \tag{3.32}$$

While the usual convention [34:23] is that the derivative of a scalar with respect to a vector produces a row vector, the following development chooses for convenience to define it as the column vector (the transpose of the conventional result). By applying Eq. (3.32), one can derive relationships which allow calculation of the derivative of common scalar-vector functions in terms of standard vector notation. One of the most common examples of this is the vector quadratic product:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \{ \boldsymbol{\mu}^T \mathbf{A} \boldsymbol{\mu} \} = 2 \mathbf{A} \boldsymbol{\mu} \tag{3.33}$$

where $\mathbf{A}$ is assumed symmetric. Using Eq. (3.33), we can arrive at the expressions:

$$\begin{aligned}
\frac{\partial J_{hr}}{\partial \bar{\boldsymbol{\mu}}_j} &= \sum_{i=1}^{N_h(k)} p_i q_j^2 \cdot -(\mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_i) \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} \\
&= -q_j^2 \sum_{i=1}^{N_h(k)} p_i (\mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_i) \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} \\
\frac{\partial J_{rr}}{\partial \bar{\boldsymbol{\mu}}_j} &= 2 \sum_{\substack{i=1 \\ i \neq j}}^{N_r(k)} q_i^2 q_j^2 \cdot -(\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_j - \bar{\boldsymbol{\mu}}_i) \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} \\
&= -2q_j^2 \sum_{i=1}^{N_r(k)} q_i^2 (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_j - \bar{\boldsymbol{\mu}}_i) \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\}
\end{aligned} \tag{3.34}$$

where the $i = j$ term is included in the final equality for convenience, noting that it will evaluate to zero anyway.

3.3.3.3 Derivatives with respect to covariances. In a manner similar to the derivative of a scalar with respect to a vector, the derivative of a scalar with respect to a matrix is defined as the matrix whose (i, j) element is the derivative of the scalar with respect to the (i, j) element of the matrix:

$$\left\{ \frac{\partial J}{\partial \mathbf{A}} \right\}_{ij} = \frac{\partial J}{\partial \{\mathbf{A}\}_{ij}} \tag{3.35}$$

Expanding the Gaussian component of the form of J_{rr} terms:

$$|2\pi(\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j) \right\} \tag{3.36}$$

we find that we need to calculate the derivative of the following two expressions:

$$\begin{aligned}
\text{Leading coefficient:} & \quad \frac{\partial}{\partial \mathbf{L}_i} |\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T|^{-\frac{1}{2}} \\
\text{Exponent:} & \quad \frac{\partial}{\partial \mathbf{L}_i} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)
\end{aligned} \tag{3.37}$$

While the results of the above derivatives are not obvious, the following results from [20:614] and [13] give the general form of the solution:

$$\begin{aligned}\frac{\partial \log |\mathbf{X}^T \mathbf{A} \mathbf{X}|}{\partial \mathbf{X}} &= 2\mathbf{A} \mathbf{X} (\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1} \\ \frac{\partial \text{tr}[(\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1} \mathbf{C}]}{\partial \mathbf{X}} &= -\mathbf{A} \mathbf{X} (\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1} (\mathbf{C} + \mathbf{C}^T) (\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1}\end{aligned}\quad (3.38)$$

Defining $\mathbf{X} = \begin{bmatrix} \mathbf{L}_i & \mathbf{L}_j \end{bmatrix}^T$, $\mathbf{A} = \mathbf{I}$ and $\mathbf{C} = (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T$ such that:

$$\begin{aligned}\mathbf{X}^T \mathbf{A} \mathbf{X} &= \begin{bmatrix} \mathbf{L}_i & \mathbf{L}_j \end{bmatrix} \mathbf{I} \begin{bmatrix} \mathbf{L}_i^T \\ \mathbf{L}_j^T \end{bmatrix} \\ &= \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\end{aligned}\quad (3.39)$$

and

$$\begin{aligned}\text{tr}[(\mathbf{X}^T \mathbf{A} \mathbf{X})^{-1} \mathbf{C}] &= \text{tr}[(\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T] \\ &= (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)\end{aligned}\quad (3.40)$$

we find forms that closely match our desired solution. Using the chain rule, we can find the common expression:

$$\begin{aligned}\frac{d e^{f(x)}}{d x} &= \frac{d e^f}{d f} \cdot \frac{d f(x)}{d x} \\ &= \frac{d f(x)}{d x} \cdot e^{f(x)}\end{aligned}\quad (3.41)$$

In order to utilize this expression, we define:

$$\begin{aligned}f(\mathbf{X}) &= \log \left[|2\pi(\mathbf{X}^T \mathbf{X})|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{X}^T \mathbf{X})^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j) \right\} \right] \\ &= -\frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{X}^T \mathbf{X}| - \frac{1}{2} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{X}^T \mathbf{X})^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)\end{aligned}\quad (3.42)$$

where N is the dimensionality of the mixture, such that the derivative of Eq. (3.36) is evaluated as:

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{X}} |2\pi(\mathbf{X}^T \mathbf{X})|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{X}^T \mathbf{X})^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j) \right\} \\
&= -\frac{1}{2} \cdot \frac{\partial}{\partial \mathbf{X}} \left[\log |\mathbf{X}^T \mathbf{X}| + (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{X}^T \mathbf{X})^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j) \right] \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{X}^T \mathbf{X}\} \\
&= -\left[\mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} - \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T (\mathbf{X}^T \mathbf{X})^{-1} \right] \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{X}^T \mathbf{X}\} \\
&= -\mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \left[\mathbf{X}^T \mathbf{X} - (\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T \right] (\mathbf{X}^T \mathbf{X})^{-1} \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{X}^T \mathbf{X}\}
\end{aligned} \tag{3.43}$$

Substituting in the partitioned matrix $\mathbf{X}$, taking the transpose and keeping only the partition in which we are interested, we find:

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{L}_i} \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} \\
&= \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \cdot \\
&\quad \cdot \left[(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T - (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T) \right] (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \mathbf{L}_i
\end{aligned} \tag{3.44}$$

The derivatives of Eq. (3.30) then become:

$$\begin{aligned}
\frac{\partial J_{hr}}{\partial \mathbf{L}_j} &= \sum_{i=1}^{N_h(k)} p_i q_j^2 \mathcal{N}\{\boldsymbol{\mu}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T\} (\mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \cdot \\
&\quad \cdot \left[(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}_j)(\boldsymbol{\mu}_i - \bar{\boldsymbol{\mu}}_j)^T - (\mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T) \right] (\mathbf{P}_i + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \mathbf{L}_j \\
\frac{\partial J_{rr}}{\partial \mathbf{L}_j} &= 2 \sum_{i=1}^{N_r(k)} q_i^2 q_j^2 \mathcal{N}\{\bar{\boldsymbol{\mu}}_i; \bar{\boldsymbol{\mu}}_j, \mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T\} (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \cdot \\
&\quad \cdot \left[(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)(\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j)^T - (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T) \right] (\mathbf{L}_i \mathbf{L}_i^T + \mathbf{L}_j \mathbf{L}_j^T)^{-1} \mathbf{L}_j
\end{aligned} \tag{3.45}$$

3.3.3.4 Verification. The above results were calculated by hand and coded manually using MATLAB[®]. To verify the expressions, the cost function ex-

pressions of Eq. (3.30) were entered symbolically and the derivative was calculated with respect to each parameter using the MATLAB[®] Symbolic Toolbox. The resultant expressions were then subtracted from the hand-coded expressions with symbolic variables, and it was verified that the results of each block cancelled to zero, indicating the algebraic equivalence of the manual calculation to the computer-aided solution.

3.3.3.5 Newton-Raphson Algorithm. As discussed in Section 2.6, the Newton-Raphson algorithm operates similarly to the gradient algorithm, but converges to the solution at a much faster rate (through from within a smaller ball of convergence). Intuitively, utilizing full second derivative information is an extremely desirable step, as it incorporates a great amount of information about the *interaction* of the parameters into the optimization process. It was the hope of the author that, using a carefully chosen starting point, the cost function would be well approximated by a parabola, and the Newton-Raphson algorithm would reach the solution in one or two iterations. These hopes were not realized, however, and the conclusion was reached that the algorithm is inappropriate for this application.

The full Hessian matrix was calculated for the case of a *scalar* Gaussian mixture, utilizing the constraint transformations described in Eq. (3.27),⁸ coded in MATLAB[®] and verified using the Symbolic Toolbox as described in Section 3.3.3.4. Simulation results revealed major difficulties with the technique. In very few steps of the algorithm, mixture components converged *toward each other*, at which point the Hessian matrix became singular and the algorithm could not continue. This reveals the inappropriateness of the technique to this application: *there are simply too many actions which produce an equivalent result in the overall function*, and the method is attracted to points producing singularities in the Hessian. This is a commonly understood problem of the Newton-Raphson algorithm: the method converges to

⁸i.e., the probability weights were constrained to sum to unity.

a critical point [18:101] — whether that critical point is a maximum, minimum or saddle point (producing a singular Hessian) depends purely on the structure of the cost function in the local region.

Apart from the difficulties discussed above, implementation of a full Newton-Raphson technique in any practical situation is computationally intractable. If the Gaussian mixture contains 10 components, each of which is a six-dimensional Gaussian PDF, then the optimization parameters will include 10 weights, 10 six-dimensional means (60 parameters), and 10 covariance matrices which, if represented in factored triangular form will each contain 21 parameters. The total number of parameters is thus 280, even for this small problem, and each step will require a 280×280 matrix inversion.

While the Newton-Raphson algorithm was demonstrated to be inappropriate for this application, the Newton-Raphson approximations (i.e., weighted gradient algorithms) which utilize a diagonal or block-diagonal Hessian matrix (as discussed Section 2.6) could potentially provide a significant improvement in the rate of convergence of the search. For the purposes of this study, however, the gradient technique provided an adequate rate of convergence, and these modifications were not attempted.

3.3.4 Initialization Algorithm. The cost function describing the fit of a reduced complexity Gaussian mixture to a Gaussian mixture of higher order is an extremely complicated multi-modal function with many peaks and troughs, of which all but one represent local minima rather than the true global minimum. Considering the fact that virtually all gradient-based iterative optimization methods will converge on a local minimum, this reveals that selecting the initialization point for the optimization is in fact the most critical function for the algorithm, more so than the iterative optimization itself.

One could conceive of any number of algorithms that could be used for this function. Simple component pruning (keeping the hypotheses with the N_r highest weights), as used in the standard MHT algorithm, would perhaps be the easiest solution; any number of merging algorithms such as n -scan merging [49] or Salmond’s mixture reduction [44] would supplement the pruning well. However, such algorithms are purely *ad hoc*, and there is no guarantee that the result will lead the iterative optimization to the *global* minimum. In fact, there is quite probably a local minimum close to every initialization point we could propose using such methods, as can be expected intuitively when one considers the case in which the Gaussian mixtures in the original model are well-spaced. In such an environment, the result generated by the iterative optimization is only as good as the initialization point provided to the algorithm.

The cost function proposed in Section 3.3.2 provides a systematic means of evaluating the relative merit of two possible solutions to the optimization. This function is utilized extensively in the iterative algorithm described in Section 3.3.3, and, considering the importance of the initialization, it is desirable to utilize the cost function also in the algorithm that selects the starting point. It was observed experimentally that the optimal solution generally has most of its mixture components similar to the respective components of the original mixture, and that the major changes produced by the reduction are that similar components are merged, and smaller probability weight or larger variance components are deleted.⁹ These observations support the merging and pruning methods commonly employed in MHT implementations, but further guidance is needed towards selecting which components should be merged, which should be deleted and which should remain unmodified.

The algorithm developed provides a systematic methodology of selecting components for merging and deletion using the cost function described in Section 3.3.2.

⁹As discussed in Section 3.3.1.3, the ISD measure tends to favor keeping lower-variance components and deleting higher-variance components.

The basis of the algorithm at each stage is to evaluate the cost of each possible action, and then to select the lowest cost action. At each step, the possible actions are to delete one of the remaining components or to merge a pair of remaining components. This is illustrated in the block diagram of Figure 3.6. When two components are merged, the parameters of the merged component are calculated according to Eq. (2.24), such that the mean and covariance of the overall mixture remains unchanged.

There is no claim that the algorithm will produce the optimal starting point, nor that the starting point will lead the iterative algorithm to the global minimum. Although the action taken at each step leads to the minimum cost increase for that single step, there is no guarantee that the result over multiple steps is optimal, or that there is not a better multiple-step solution which does not take the optimal single step action at each step. However, the starting point obtained is the result of a sensible algorithm which at each step selects the best action, taking into account the *full PDF* rather than considering individual pairs of components in isolation, hence the result is likely to produce significantly better results than algorithms which consider only individual component pairs.

3.3.4.1 Implementation. The algorithm was implemented using MATLAB[®] version 6.5. As described in Section 3.3.2, the cost function consists of three components: the *self-likeness* of the original mixture with itself, the *cross-likeness* of the original mixture and the simplified mixture, and the *self-likeness* of the simplified mixture with itself. Each of these components is the sum over a matrix, the entries of which represent the likeness of pairs of individual Gaussian components.

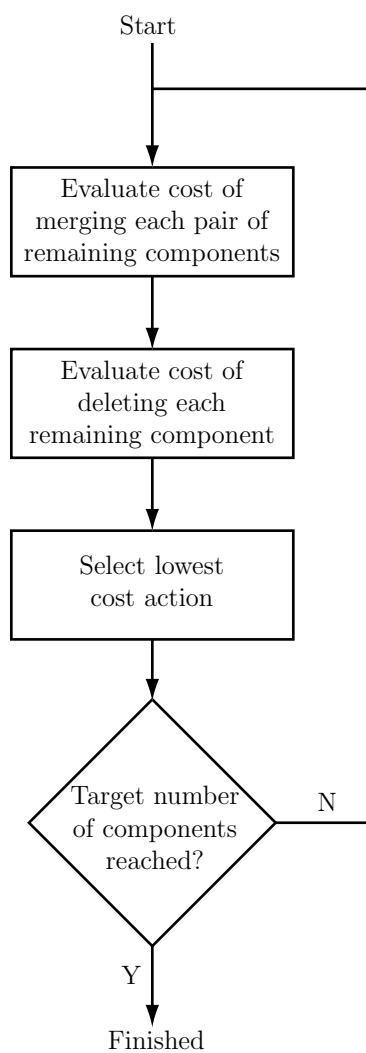


Figure 3.6. Block diagram of proposed Gaussian mixture reduction initialization algorithm.

The individual entries of the matrices are of the form of a multivariate Gaussian function evaluation:

$$\begin{aligned}
& p_1 p_2 \mathcal{N}\{\boldsymbol{\mu}_1 \boldsymbol{\mu}_2, \mathbf{P}_1 + \mathbf{P}_2\} \\
& = p_1 p_2 |2\pi(\mathbf{P}_1 + \mathbf{P}_2)|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\}
\end{aligned}
\tag{3.46}$$

The original implementation evaluated the cost of each of the possible merging and deletion possibilities at each processing cycle, with no regard for the calculations which do not change between cycles. Consequently, the speed of the original implementation was very poor: even using MATLAB[®] version 6.5 (using compiled rather than interpreted code) the time required to simplify a 60-component, four-dimensional Gaussian mixture down to 10 components was on the order of 349 seconds.¹⁰ Considering that, even in the simplest tracking environment with a single target in clutter, this level of processing would need to be performed during every measurement interval, this implementation is clearly unacceptable for practical use.

The elements of the cost function are illustrated in Figure 3.7: each square represents an evaluation of the similarity measure between two single multivariate Gaussian components from the respective mixtures, as according to Eq. (3.46). Consideration of the components of the calculations that are able to be stored and not repeated at every processing cycle leads to the following observations:

1. The original mixture self-likeness matrix does not change when simplification steps are performed, hence the sum of this matrix can be calculated once and never re-evaluated.
2. When a component is deleted, a column of the cross-likeness matrix will be deleted. Similarly, when two components are merged, two columns will be re-

¹⁰All benchmarks discussed in this section were performed on a 1.4GHz AMD Athlon system.

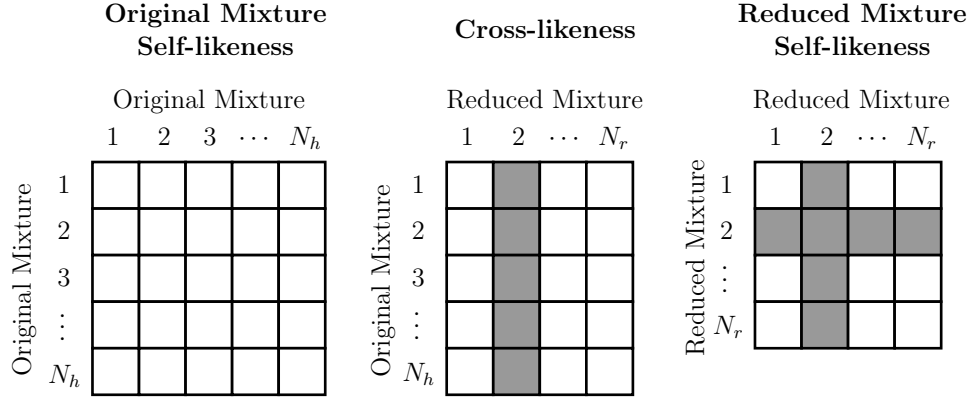


Figure 3.7. Elements of ISD cost function. Each square represents a multivariate Gaussian evaluation to measure the similarity of the respective components of the two mixtures. Shaded squares represent the components that need to be re-evaluated if the second component in the reduced mixture is modified.

placed by a single new column, representing the cross-likeness of the original Gaussian mixture to the newly merged component. To evaluate the cost of every possible merge, the new columns will need to be calculated for every possible merge. However, when a merge action is taken, only the new columns for merge possibilities involving the modified component will need to be recalculated, rather than the new columns for every possible merge. There are $\frac{1}{2} N(N - 1)$ total merge possibilities, where N is the number of components in the reduced mixture at the current processing cycle, starting from N_h at the commencement of the algorithm and successively reducing to N_r as components are merged and deleted. Only $(N - 1)$ of these merge possibilities involve a given component.

3. When a component is deleted, one column and one row of the reduced self-likeness matrix will be deleted. When two components are merged, two rows and columns will be replaced with a single row and column representing the

newly merged component.¹¹ To evaluate the cost for every possible merge, the new column will need to be calculated for every pair of components. When a merge action is taken, the full column will need to be recalculated for each merge possibility involving the modified component, and the single entry corresponding to the modified component will need to be recalculated for all other possibilities.

The implementation was modified to store the cost components for every possible merge and reuse wherever possible, reducing the complexity of the algorithm to a stage at which the time to simplify a 60-component four-dimensional Gaussian mixture down to 10 components was 30.3 seconds. Although this is a significant improvement over the 349 second time of the original implementation, the algorithm is still to be applied to problems significantly more complicated than this at every measurement interval, hence it remains unacceptable for real-time application.

Analysis of the optimized implementation using the MATLAB[®] Profiler revealed that 78% the 30.3 second processing time was spent evaluating the multivariate Gaussian function of Eq. (3.46) above. The MATLAB[®] implementation used for the equation was (the variable `mud` stores the difference between the two mean vectors, μ_d , as seen in Eq. (3.47)):

```
mud = mu(:,i) - mu(:,j);
Pc = P(:, :, i) + P(:, :, j);
dist = ps(i)*ps(j)*exp(-0.5*mud'*inv(Pc)*mud)/...
      real(sqrt(det(2*pi*Pc)));
```

The `real()` function call around the `sqrt()` is necessary to allow the expression to be compiled using MATLAB[®] version 6.5 due to the possibility of a complex result. On the surface the code appears to allow little room for optimization; however, in

¹¹Note that, since the matrix is symmetric, calculating the new column also gives the entries required for the corresponding row.

the particular example discussed, the equation above was evaluated 291,726 times, hence further consideration is warranted.

Considering the quadratic expression `mud'*inv(Pc)*mud`, the result is a single scalar value, yet the full matrix inverse is calculated. Perhaps the most obvious simplification possible is to replace the full matrix inverse with the MATLAB® left matrix divide command `mud'*Pc\mud`, which implements Gaussian elimination to reach a triangular form, followed by back-substitution with the mean difference vector `mud` without calculating the full matrix inverse [33]. Furthermore, the determinant function `det()` is commonly implemented as the product of the pivots, again found using Gaussian elimination [33]. These same calculations have necessarily been performed to find the inverse (or to perform back-substitution), hence implementation causes the same calculations to be performed twice.

Neither of these implementation options exploit the positive-definite symmetry of the `Pc` matrix. This could be exploited using the Cholesky square-root function [34:370] to factor the matrix into a triangular square root such that:

$$\mathbf{P}_c = \sqrt[3]{\mathbf{P}_c} \sqrt[3]{\mathbf{P}_c}^T$$

Using the MATLAB® Cholesky square-root function,¹² the expression could be replaced by:

```
PcChol = chol(Pc)';
dist = ps(i)*ps(j)*exp(-0.5*sum((PcChol\mud).^2)/...
    prod(sqrt(2*pi)*diag(PcChol)));
```

where `PcChol` is the Cholesky square root of the matrix `Pc`. (Note that the determinant has been simplified to the product of the diagonal terms of the triangular Cholesky square root matrix.) This implementation leaves further room for opti-

¹²The MATLAB® implementation of the Cholesky square root returns the transpose of the conventional factorization, hence the transpose is taken of the result.

mization in that the left matrix divide command may not immediately recognize the triangular form of the Cholesky square root matrix, and that the Cholesky factorization routine performs several expensive square root evaluations [34:403–405].

For the above reasons, a U-D factorization routine was implemented [34:392], followed by a custom back-substitution procedure. The U-D factorization routine factors the covariance sum P_c into an upper triangular matrix and a diagonal matrix, such that the quadratic can be simplified as:

$$\begin{aligned}\mu_d^T P_c^{-1} \mu_d &= \mu_d^T (U D U^T)^{-1} \mu_d \\ &= \mu_d^T U^{-T} D^{-1} U^{-1} \mu_d \\ &= (U^{-1} \mu_d)^T D^{-1} (U^{-1} \mu_d)\end{aligned}\tag{3.47}$$

where the vector $(U^{-1} \mu_d)$ is evaluated using the custom back-substitution algorithm, with the answer stored in a vector named `Uimud`. The expression is then evaluated as:

```
dist = ps(i)*ps(j)*exp(-0.5*sum(Uimud.^2 ./ diag(D))/...
    prod(2*pi*diag(D)));
```

A simple test scenario was created, evaluating the expression for all possible pairings of 500 randomly generated four-dimensional Gaussian mixture components. The original implementation required 31.6 seconds to execute, the left matrix divide method required 29.5 seconds to execute, the Cholesky square root method (avoiding the redundant determinant calculation) required 21.5 seconds to execute, and the custom U-D factorization method required 11.9 seconds to execute. The results confirm the efficiency of the U-D factorization method, especially considering that the time was far less than other methods even though a pure MATLAB[®] script implementation was used, incorporating three levels of nested `for` loops.¹³

¹³Analysis using the MATLAB[®] version 6.5 profiler confirmed that the routine and surrounding loops were indeed compiled completely.

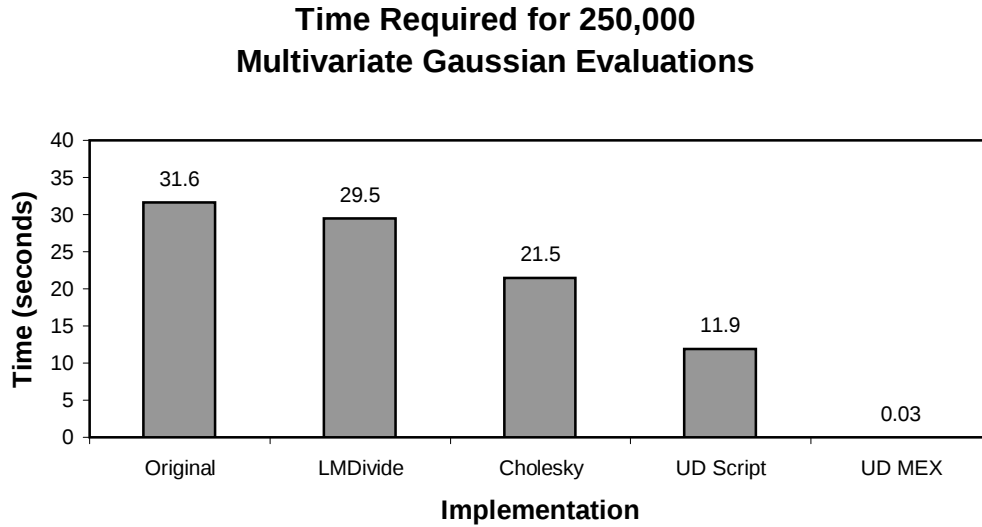


Figure 3.8. Execution times for various implementations for evaluating the match between all pairings of 500 randomly generated four-dimensional Gaussian multivariate PDFs.

Following the above results, the U-D covariance factorization algorithm was translated into the C language, using the MATLAB[®] MEX interface. For the simple test above, the highly optimized C implementation reduced the time to 0.03 seconds, a 1000 $\times$ saving over the original implementation (which required 31.6 seconds). The MEX implementation was extended to implement the entire initialization algorithm, reducing the time for reducing the same 60-component Gaussian mixture described above from 30.3 seconds for the previous optimized version to 0.42 seconds, a further 72 $\times$ improvement over the previous optimization, and an 831 $\times$ improvement over the original implementation (which required 349 seconds). The time reductions achieved using the various optimization methods on these two problems are illustrated in Figures 3.8 and 3.9.

3.4 Summary

This chapter has developed a structured, cost function-based technique which reduces the number of components in a Gaussian mixture while modifying the overall

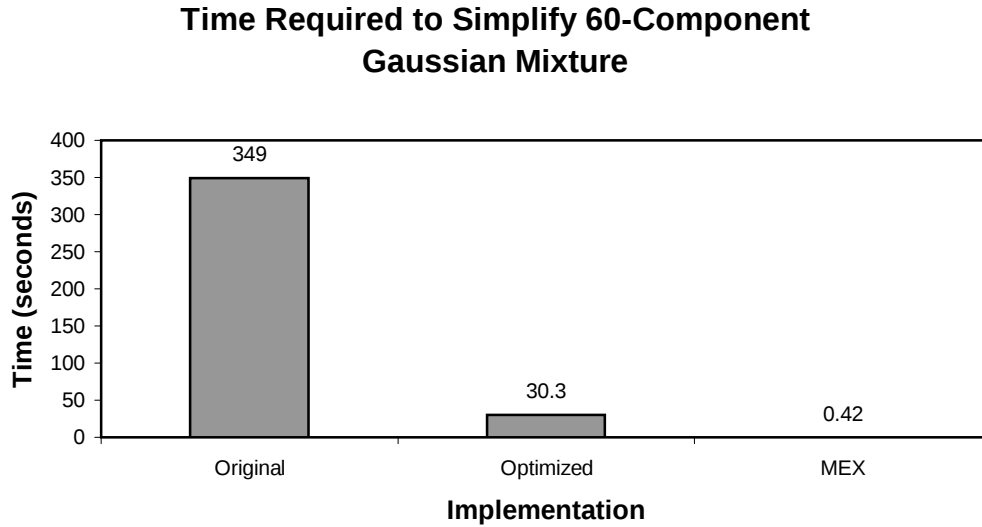


Figure 3.9. Execution times for various implementations of cost function Gaussian mixture reduction initialization algorithm to simplify 60-component four-dimensional Gaussian mixture to 10 components.

PDF structure less than any of the previously developed *ad hoc* methods. The presentation in Section 3.2 examined some of the problems commonly experienced with the techniques utilizing the most compact target state representations, JPDA and CPDA, providing new insight into the problem of the bias of the JPDA algorithm, and the reason why CPDA exhibits track coalescence to a greater extent than JPDA. Section 3.3 then developed the cost function-based optimization method. Selections for the cost function were discussed in Section 3.3.1; our choice was the ISD measure, which was found to be both physically meaningful and computationally tractable. A gradient-based iterative optimization algorithm was then developed using this cost function, as well as an initialization algorithm to find a near-optimal starting point.

IV. Simulation Results

4.1 Introduction

The following sections present the results of simulations performed to examine the implementation of the Gaussian mixture reduction algorithm described in Sections 3.3.3 and 3.3.4. Section 4.2 presents the results of applying the initialization algorithm to a simple one-dimensional problem with parameters chosen to demonstrate several characteristics of the technique. Section 4.3 then illustrates the refinement which may be gained through iterative optimization. Sections 4.4–4.6 present the results of simulations which examine the performance of the algorithms in practical applications: first tracking a single target in clutter, then multiple targets in clutter, and finally tracking a maneuvering target.

4.2 Initialization Algorithm

The initialization algorithm described in Section 3.3.4 provides a systematic methodology of selecting mixture components for merging and deletion using the Integral Square Difference (ISD) distance measure. The algorithm can be used to provide a starting point for subsequent refinement using the iterative optimization techniques described in Section 3.3.3. The following sections illustrate the application of the initialization algorithm on a one-dimensional, five-component Gaussian mixture. The parameters of the mixture are shown in Table 4.1. The main peak of

Component #	Weight	Mean	Variance
1	0.083	1	0.1
2	0.167	2	20
3	0.25	3	2
4	0.333	4	2
5	0.167	10	2

Table 4.1. Parameters of the one-dimensional Gaussian mixture used to test the initialization algorithm.

the mixture is produced by components 3 and 4, which have similar means and probability weights and the same variance. The peak at $x = 10$ has the same variance as the two central components, and a weight half that of the larger central component. The wide component at $x = 2$ with large variance has the same weight as the peak on the right hand at $x = 10$, but its variance is 10 times that of the right-hand peak, hence the size of the peak is much smaller. The tall, narrow peak at $x = 1$ has half the probability weight of the previous two, and a variance which is one twentieth of the components at $x = 3$, $x = 4$ and $x = 10$, and one two-hundredth of the larger variance component at $x = 2$.

Figure 4.1 shows the original five-component Gaussian mixture (in the top-left corner), and the approximations produced by the ISD initialization algorithm using four, three and two components. The results are shown de-normalized, such that the weights of the remaining components are not increased when a component is deleted. The component weights are normalized at the end of each processing cycle in the testing performed in Sections 4.4–4.6.

Table 4.2 shows the steps which are made to produce the approximations of Figure 4.1, and the cost (using the ISD measure) of these steps. The first step is to merge the two components (components 3 and 4 in Table 4.1) which combine to produce the central peak. Visually, the approximation produced by this step appears to be excellent, as illustrated in the top left plot in Figure 4.1. This subjective assessment is supported by the small cost, 9.9×10^{-7} , which is incurred by the approximation. The parameters of the merged pair are placed in the lower component

Step	Action	Cost
1	Merge components 3 and 4 (of 5)	9.9×10^{-7}
2	Delete component 2 (of 4)	1.8×10^{-3}
3	Merge components 1 and 2 (of 3)	5.7×10^{-3}

Table 4.2. Reduction steps for Gaussian mixture example.

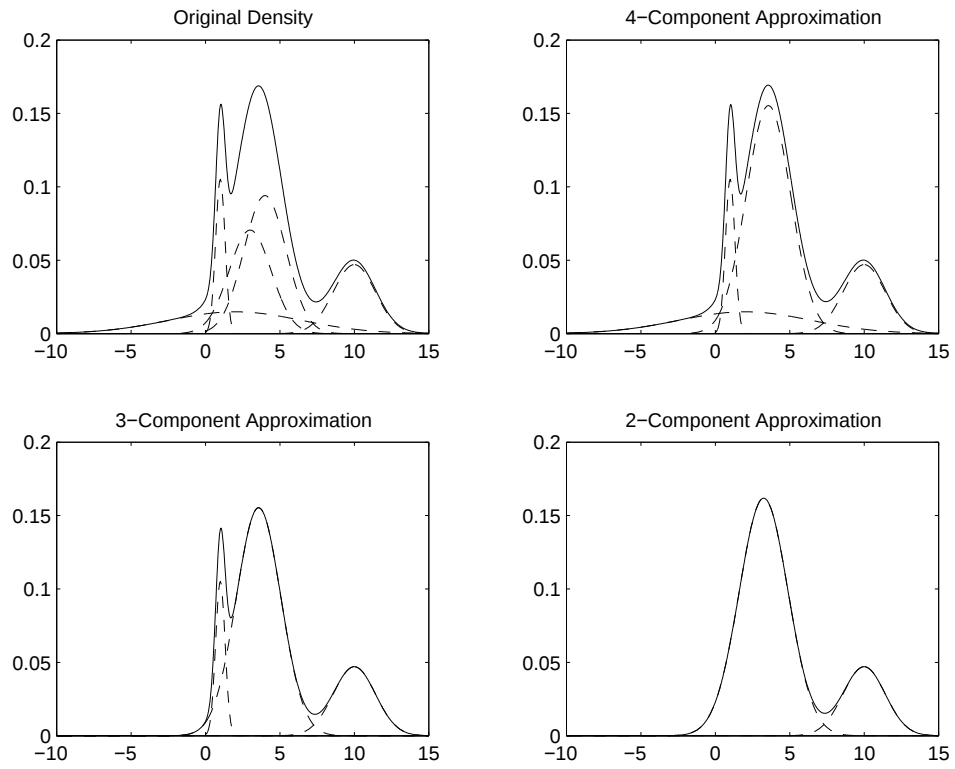


Figure 4.1. Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using the ISD initialization algorithm.

index, and the higher index component is deleted and indexing adjusted accordingly, such that the newly merged component becomes number 3 of 4.

The second step in Table 4.2 is to delete the second mixture component. As discussed in Section 3.3.1.3, the ISD cost measure applies more cost to smaller variance components (which produce large, narrow peaks) than to larger variance components (with flatter, broader peaks) with the same probability weight. This reduction step demonstrates this predisposition: even though the narrow peak produced by component 1 carries half the probability mass of the much broader component 2 (which has a variance 200 times that of component 1), the cost function prefers to discard component 2. The cost of this step (1.8×10^{-3}) is three orders of magnitude greater than the first approximation step, as reflected visually in the modification in the overall function produced by the step.

The final step in the reduction is to merge components 1 and 2, which correspond to the narrow peak discussed above, and the large peak produced by the first merging step. The cost of this step (5.7×10^{-3}) is on the same order of magnitude as the previous approximation, as illustrated by the significant change produced in the overall function. Interestingly, the cost of deleting component 1 (the narrow peak) would have been 1.0×10^{-2} , which is even larger than the cost of deleting component 3 (the smaller, wider peak on the right), 8.4×10^{-3} . This again demonstrates the predisposition of the ISD measure towards giving more consideration to smaller variance components than to larger variance components.

Figures 4.2 and 4.3 show the results of the same approximations using Salmond's joining and clustering algorithms, discussed in Section 2.5.11.2. Visually, the representations provided by these approximations are poor compared to the corresponding plots in Figure 4.1. This suggests that, on a visual level at least, the approximations produced by the ISD initialization algorithm are superior to those produced by Salmond's joining and clustering algorithms.

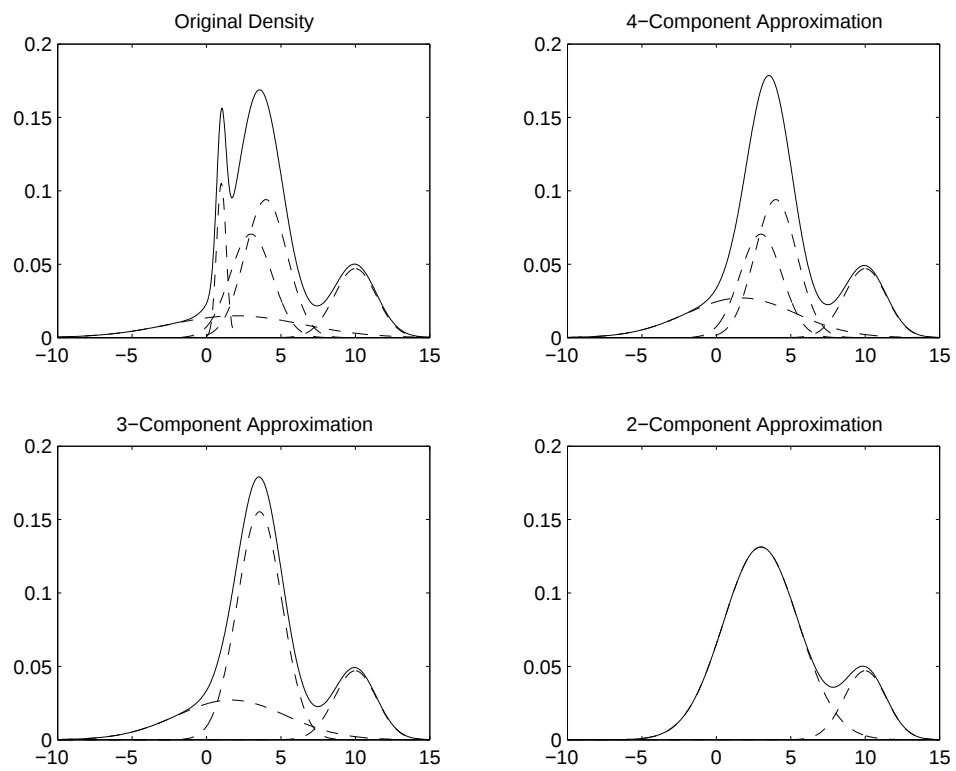


Figure 4.2. Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using Salmond's joining algorithm.

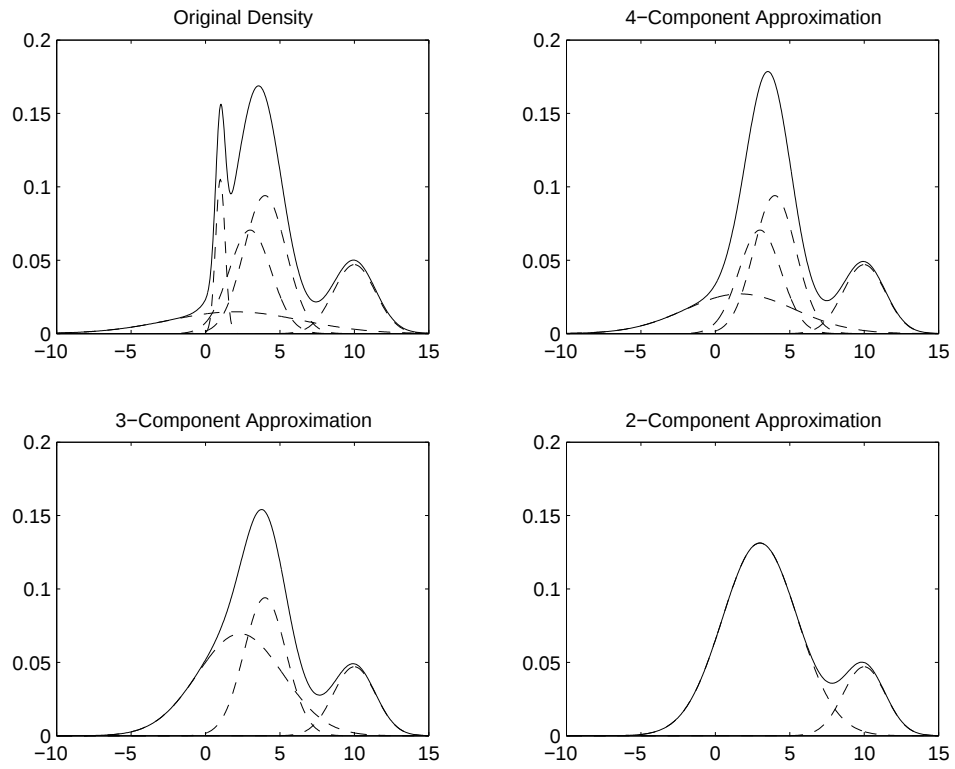


Figure 4.3. Reduction of a five-component Gaussian mixture to four-, three- and two-component approximations using Salmond's clustering algorithm.

4.3 Iterative Optimization

The iterative optimization techniques presented in Section 3.3.3 provide a mechanism for converging to a local cost minimum which is close to the starting point produced by the initialization algorithm. This iterative convergence acts as a successive refinement of the PDF approximation, tuning the parameters in order to provide a better representation of the original function. The operation of the optimization algorithm is illustrated in Figure 4.4, using the same example discussed in Section 4.2. The top left figure shows the starting point for the optimization, calculated using the ISD initialization algorithm. The result of the iterative optimization is shown after 1, 2 and 12 iterations, providing successively closer approximations to the original PDF (the approximation is shown using a solid line, while the original PDF is shown using a dashed line). It is clear from Figure 4.4 that the *overall structure* of the PDF is remaining unchanged: the changes made by the iterative optimization represent more of a *fine tuning* than a large modification.

The cost function reduction as the optimization proceeds is shown in the upper plot of Figure 4.5. The initial reduction is very significant, reducing the cost to just 48% of its original value in two iterations, and 37% of its original value in four iterations; the reduction after this initial period is less significant. The break in the line at the ninth iteration indicates that the gradient step caused an increase in the cost function value, hence the step was discarded, and repeated using a smaller step size. The adaptive step size control algorithm described in Section 2.6 was implemented; its operation is shown in the lower plot of Figure 4.5. The algorithm was set to terminate after fifty optimization steps, or when the improvement produced by an optimization step was less than 0.001 of the cost at the starting point. Figures 4.5 and 4.7 suggest that a more aggressive stopping criterion would be to terminate the optimization when a step that increases the cost is taken (noting that most of the benefit of the optimization has been gained by this stage).

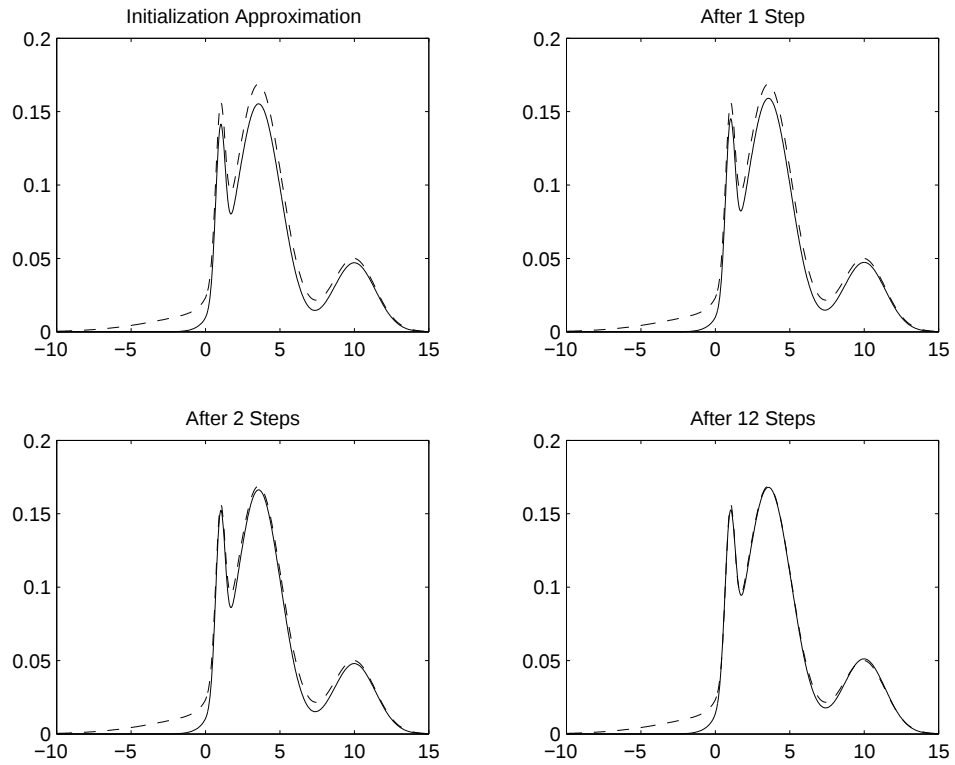


Figure 4.4. Iterative optimization of a 3-component approximation (shown in solid line) to a 5-component Gaussian mixture (shown in dashed line). The top left figure shows the starting point for the optimization, calculated using the ISD initialization algorithm. Remaining figures show the refined solution after 1, 2, and 12 gradient iterations.

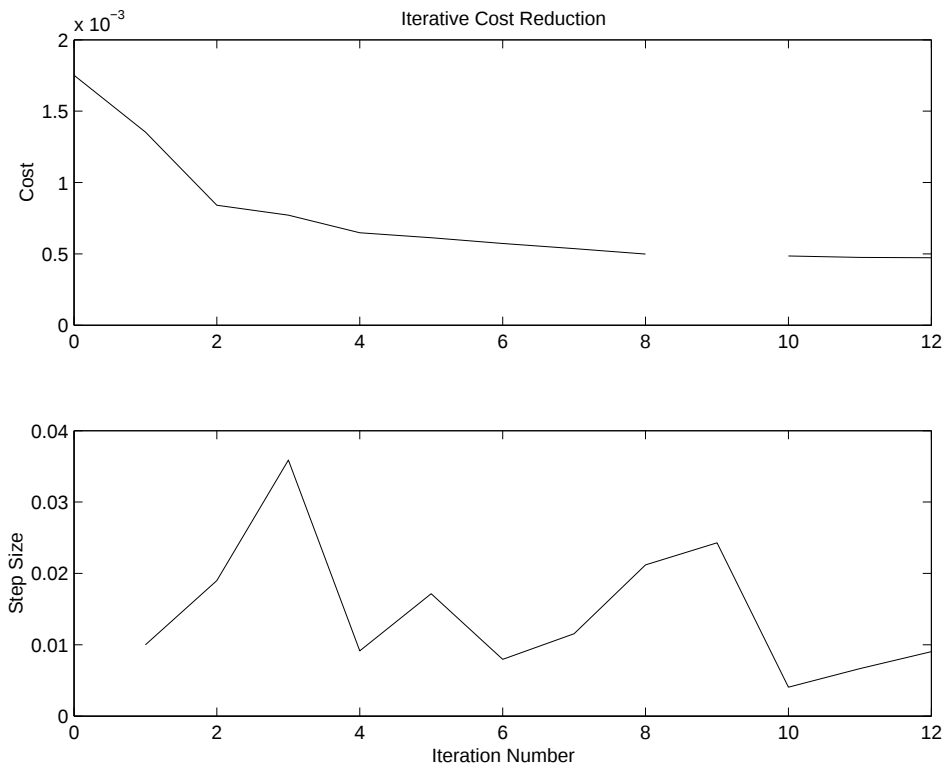


Figure 4.5. Cost function trajectory and step size adjustment. The top figure shows the cost reduction as the PDF approximation is optimized iteratively, while the bottom figure shows the gradient step size adaptation.

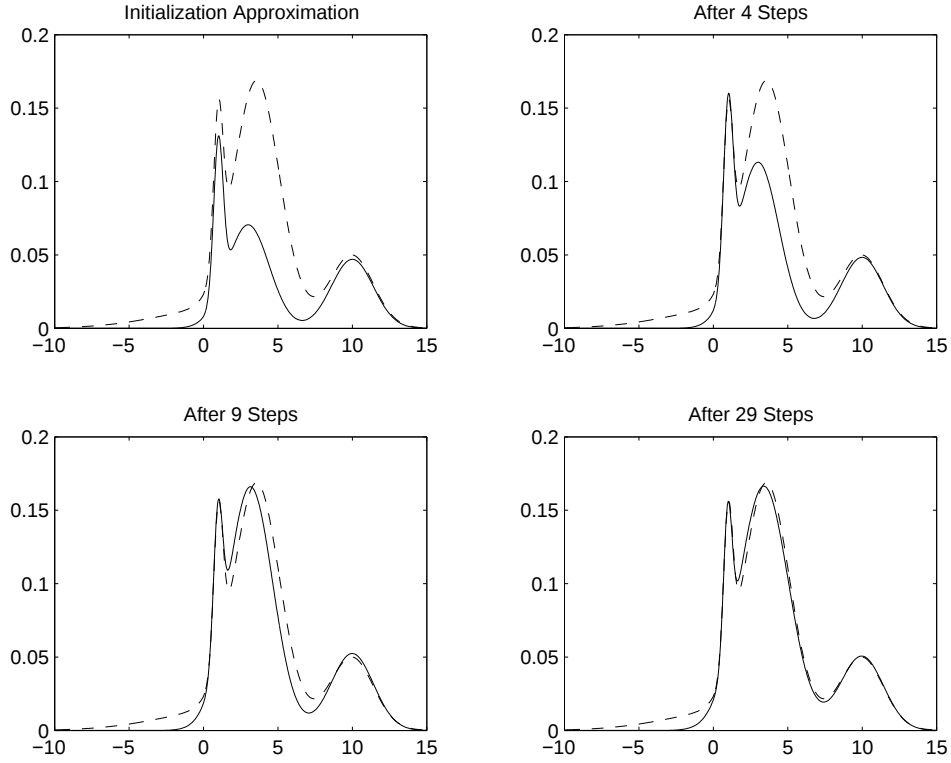


Figure 4.6. Iterative optimization of a 3-component approximation (shown in solid line) to a 5-component Gaussian mixture (shown in dashed line). The top left figure shows the starting point for the optimization, calculated using the ISD initialization algorithm. The remaining figures show the refined solution after 4, 9, and 29 gradient iterations.

The operation of the iterative optimization technique is more obvious when the starting point provided to the algorithm is further from a minimum. Figures 4.6 and 4.7 illustrate such a situation, in which the starting point is far from a local minimum. The example reveals that the same minimum is reached eventually; many other starting points will not produce this result.

4.4 Single Target in Clutter

The single target scenario presented in [44] was reproduced to test the performance of the ISD initialization and optimization algorithms in a realistic

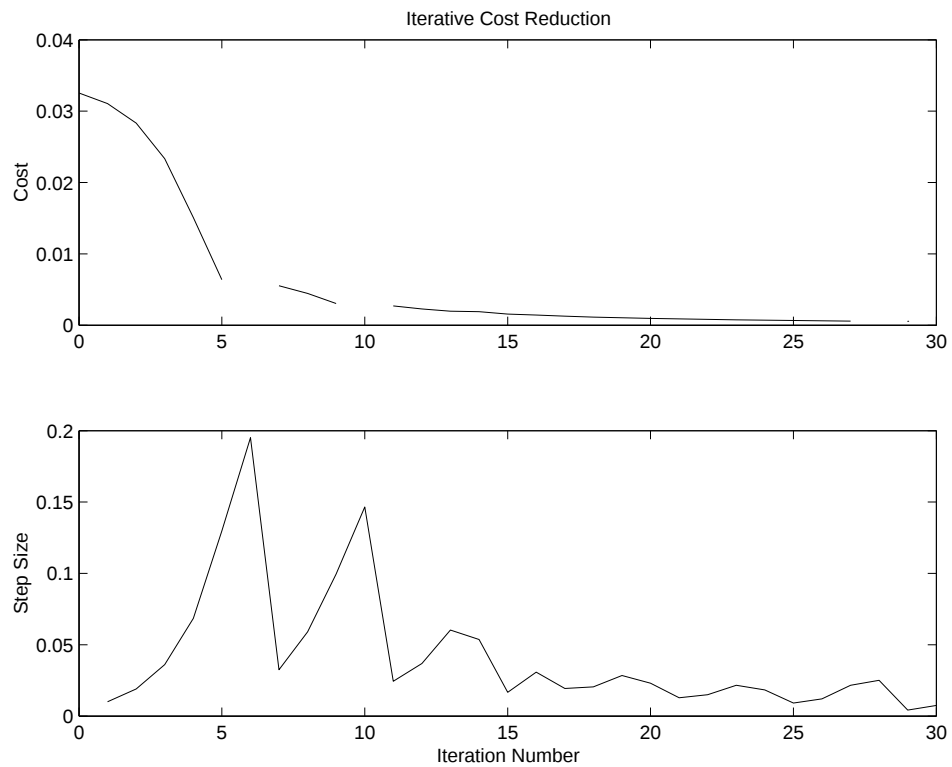


Figure 4.7. Cost function trajectory and step size adjustment. The top figure shows the cost reduction as the PDF approximation is optimized iteratively, while the bottom figure shows the gradient step size adaptation.

tracking environment. The scenario simulates a radar tracking a target flying through dense clutter. The target state evolves according to the following constant velocity state model:

$$\begin{aligned}
\hat{\mathbf{x}}(k) &= \begin{bmatrix} \hat{p}_x(k) \\ \hat{v}_x(k) \\ \hat{p}_y(k) \\ \hat{v}_y(k) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{x}}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \mathbf{w}(k-1) \\
\mathbf{z}(k) &= \begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \hat{\mathbf{x}}(k) + \mathbf{v}(k)
\end{aligned} \tag{4.1}$$

where T is the time between measurement intervals $(k-1)$ and k , and $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are two independent zero-mean white noise processes such that:

$$\begin{aligned}
E\{\mathbf{w}(k)\mathbf{w}(k)^T\} &= \mathbf{Q} = q\mathbf{I} \\
E\{\mathbf{v}(k)\mathbf{v}(k)^T\} &= \mathbf{R} = r\mathbf{I}
\end{aligned}$$

The system is provided with noise-corrupted measurements of the target position (x and y coordinates) through a linear measurement model; the system could be extended to polar measurements (i.e., range and angle) using an extended Kalman filter as described in Section 2.2.4.

To match the parameters presented by Salmond [44:16], T , q and r were all normalized to unity, the clutter density λ was set to 0.012, and the probability of detection (P_d) was set to unity. The gate size was reduced such that $P_g = 0.99$, reducing computational complexity significantly over the value used in [44:43], $P_g = 0.999$. The target was initially located at the origin with a velocity of 10 units/sec in each coordinate.

The measurement space was populated with false targets according to a Poisson distribution with density $\lambda = 0.012$ measurements per unit area. The region populated was a square, centered on the actual target location, with side $200\sqrt{r}$. This value was chosen to be large such that hypotheses could be deceived by clutter measurements for several processing cycles without leaving the populated region. The expected number of false targets in each processing cycle for this configuration is 480.

The criterion for loss of track suggested in [44:14–15] is that the target-originated measurement has not been incorporated into the measurement gate of any hypothesis for the last five consecutive time steps, or that the combined estimate is more than 10σ from the true target location for five consecutive time steps (the comparison being performed independently for each state element, ignoring off-diagonal covariance elements), where σ is the standard deviation of the state estimate from the Kalman filter without measurement origin ambiguity. The problem with the latter criterion is that the combined estimate *can and will* venture far from the correct target location without the system losing track: as long as at least one hypothesis remains within the vicinity of the actual target location, the combined estimate will probably (or, at least, potentially) move back to the correct location when the uncertainty is resolved using information from later measurement sets. To resolve this difficulty, the latter criterion was modified such that loss of track is declared if *all hypotheses* are more than 10σ from the correct target location for more than five consecutive time steps, hence taking into account the deferred decision making capability of the multiple hypothesis techniques.

It is worth noting that, using the original criteria proposed in [44], even the optimal Bayesian solution (with unbounded computational and memory requirements) will potentially have a very limited track life. While the optimal solution is guaranteed¹ to maintain a mixture component corresponding to the correct hypothesis,

¹Although the use of measurement gating would weaken this guarantee.

# Comp.	1	2	3–8	9	10	15	20	25	30	35	40
ISD Init.	200	200	200	200	200	200	200	200	200	106	-
Pruning	200	200	200	200	200	200	200	200	200	200	200
Joining	200	200	200	200	200	200	200	169	110	109	-
Clustering	200	200	200	200	200	200	200	200	110	109	-
Lainiotis	-	-	50	-	-	-	-	-	-	-	-
Iter. Opt.	50	50	50	50	50	-	-	-	-	-	-

Table 4.3. Number of Monte Carlo simulations run for each algorithm and number of mixture components.

there is *no* guarantee that the combined estimate will be close to the correct location at any point in time. Hence, to evaluate how effectively a particular hypothesis reduction technique approximates the full Bayesian solution, the modified criterion is preferable.

The initial set of results consisted of 200 Monte Carlo simulations, each of which was allowed to run until loss of track was declared.² The simulations were run for the ISD initialization algorithm, the standard MHT pruning algorithm (discussed in Section 2.5.11), and Salmond’s joining and clustering algorithms (discussed in Section 2.5.11.2), using various numbers of hypotheses for each. Some simulations were also run using the mixture reduction algorithm described in Lainiotis [31] (as outlined in Section 2.5.11.1), and using the iterative optimization technique described in Section 3.3.3. Some of the simulations using large numbers of mixture components required a large amount of time to process (mainly due to the extremely long track life achieved using the algorithms), hence they were terminated before the full 200 runs had completed.³ The number of simulations run for each algorithm and each number of mixture components is summarized in Table 4.3.

The average track life for the various algorithms tested is compared in Figure 4.8; this was one of the major metrics used to compare algorithm performance

²The number of time steps was actually capped at 10,000 for each simulation, but using the various algorithms, this limit was never reached.

³In fact the simulations presented were computed using 23 Intel® Pentium® IV-based systems over a period of approximately two weeks.

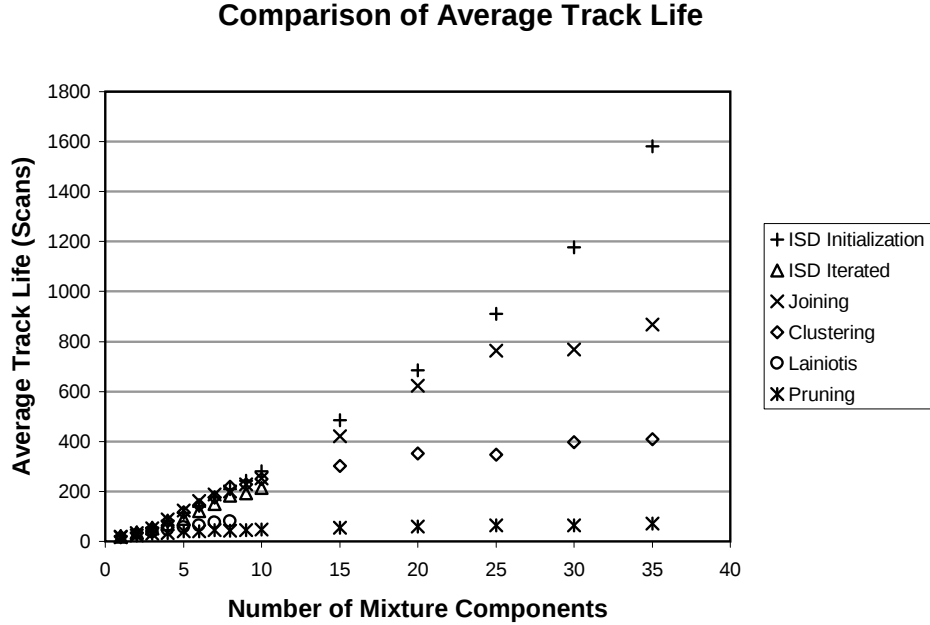


Figure 4.8. Average track life for various merging and pruning algorithms.

in [44]. The diagram clearly reveals the remarkable performance of the ISD initialization technique using a large number of mixture components: the average track life is significantly greater than that of the algorithms which previously have been considered to provide the best performance in this scenario. The exponential increase of track life exhibited by the ISD initialization algorithm indicates that the tracking performance is limited only by the availability of computational resources. This is in sharp contrast to the algorithms which were previously considered to provide the best performance, which demonstrate an average track life that levels out as the number of mixture components is increased, indicating that additional computer resources would provide little performance benefit. The following sections examine the performance of this algorithm in comparison with the existing systems shown in Figure 4.8: standard MHT pruning, Salmond's joining and clustering algorithms, and a modified Lainiotis algorithm, as well as the ISD-based iterative optimization technique.

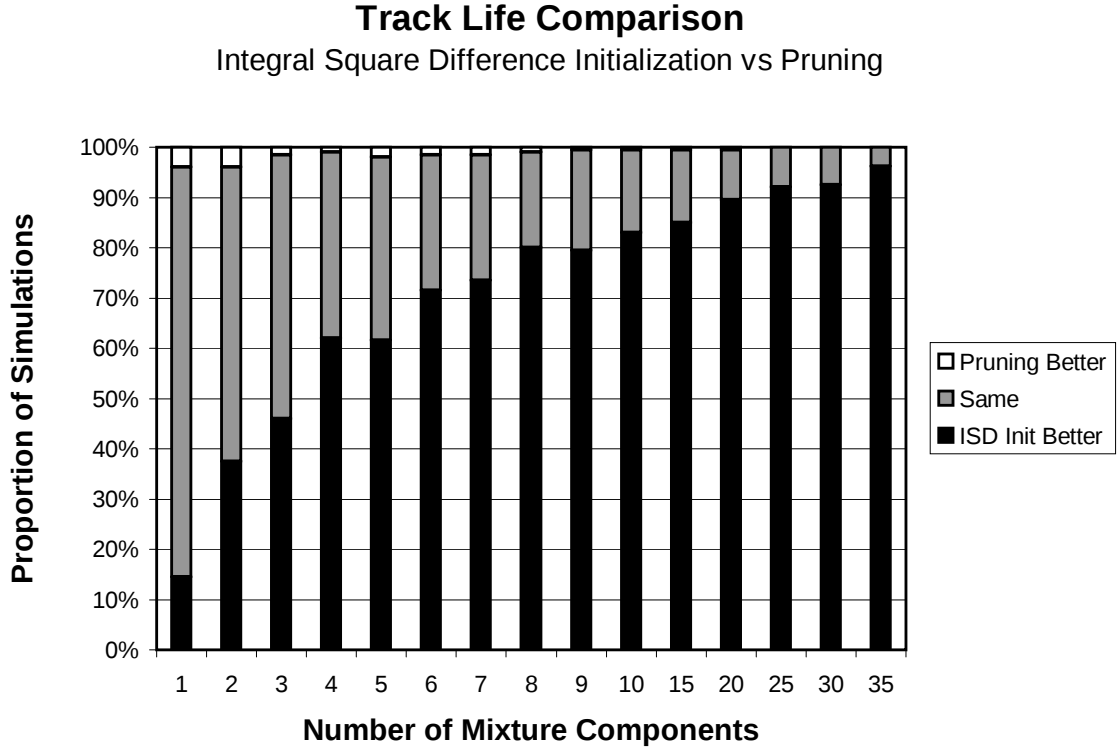


Figure 4.9. Performance of ISD initialization algorithm compared to the standard MHT pruning algorithm.

4.4.1 Comparison with Pruning Algorithm. Figure 4.9 compares the performance of the ISD initialization algorithm to a pruning algorithm which keeps the most likely hypotheses, up to the desired number. The bars in the graph of Figure 4.9 indicate the proportion of simulations in which each algorithm outperformed the other: the black region denotes the proportion of simulations in which the track life of the ISD initialization algorithm was longer than that of the pruning algorithm; the white region denotes the proportion of simulations in which the track life of the pruning algorithm was longer than that of the ISD initialization algorithm; and the gray region denotes the proportion of simulations in which the track life of the two algorithms was essentially the same (i.e., within 10 scans of each other). It is the belief of the author that this method of presentation provides the most reliable comparison of the performance of two algorithms. If the two algorithms lose track at

approximately the same time, then the same sequence of measurements caused loss of track on each. If the sequence of measurements causes loss of track on one algorithm but not the other, then the surviving algorithm is demonstrated to be superior in that circumstance. The amount of time that the surviving algorithm maintains track after the point where loss of track occurred for the other algorithm is largely irrelevant, as the performance of the algorithm losing track *has not been tested* past the point where loss of track occurred.⁴ Accordingly, a performance metric considering each run equally according to *which algorithm maintained track longer*, and not how much longer the surviving algorithm maintained track, provides the fairest comparison.

The scatter plot in Figure 4.10 compares the track life of the two algorithms for each of the 200 Monte Carlo simulations, using 25 mixture components for each algorithm. Each cross on the diagram represents a single Monte Carlo simulation: the x -coordinate is the number of scans for loss of track to occur using the ISD initialization algorithm, while the y -coordinate is the number of scans for loss of track to occur using the pruning algorithm on exactly the same simulation. The dashed 45° line is the contour for which the performance of the two algorithms is identical. A cross above this dashed line represents a Monte Carlo simulation for which the life of the pruning algorithm was longer than that of the ISD initialization algorithm; conversely, a cross below the dashed line represents a Monte Carlo simulation for which the life of the ISD initialization algorithm was longer than that of the pruning algorithm. The concentration of crosses significantly below the 45° line indicates that the ISD initialization algorithm performs significantly better than the pruning algorithm.

In many Monte Carlo simulations it was obvious that the hypothesis utilization of the simplistic pruning method was poor. The performance of the pruning

⁴i.e., there is nothing to suggest that the algorithm losing track would not have been able to maintain track as well as or better than the surviving algorithm if the algorithm losing track had been able to maintain track through the sequence of measurements causing the loss.

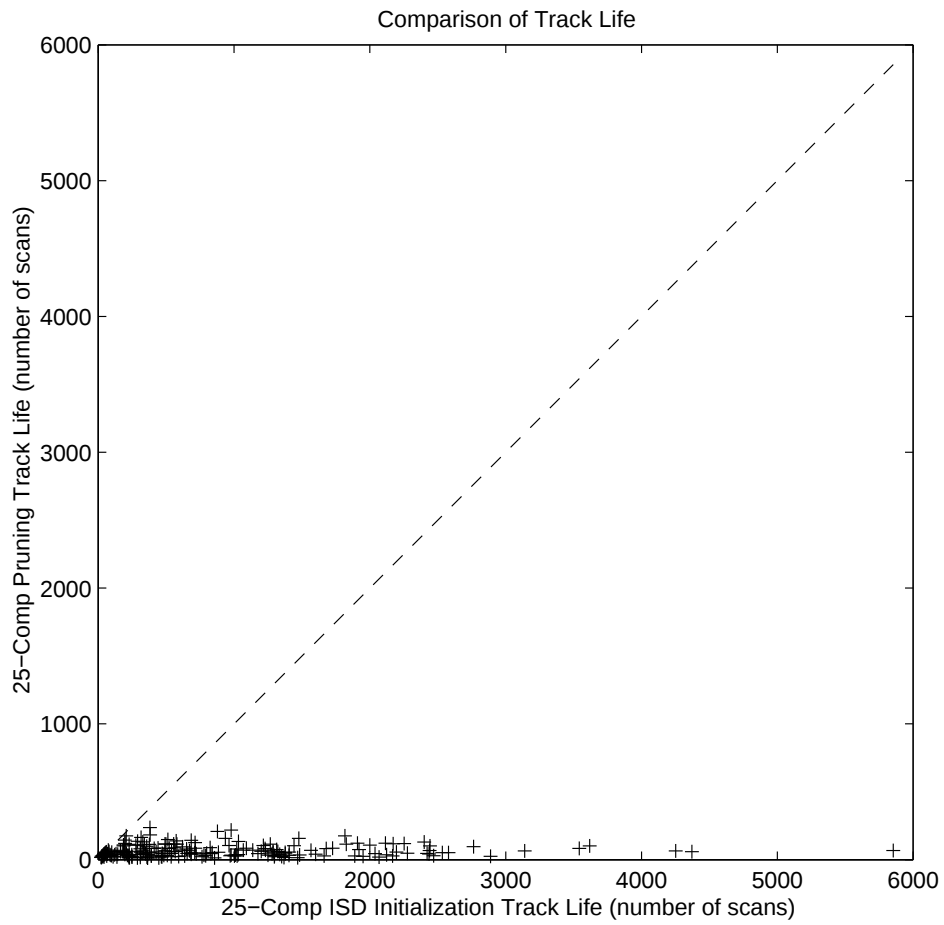


Figure 4.10. Performance of 25-component ISD initialization algorithm compared to 25-component pruning algorithm.

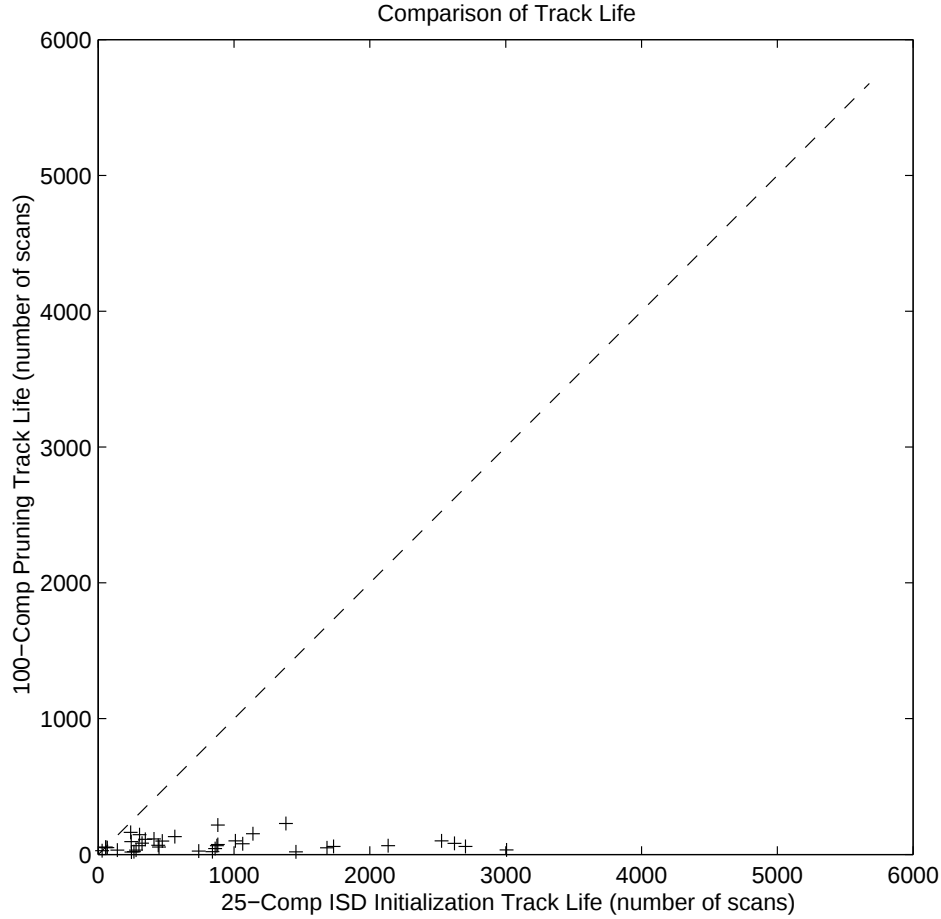


Figure 4.11. Performance of 25-component ISD initialization algorithm compared to 100-component pruning algorithm.

algorithm utilizing 100 mixture components is compared to the 25-component ISD initialization algorithm in Figure 4.11.⁵ The diagram demonstrates that, even using four times the number of mixture components, the performance of the pruning algorithm is consistently inferior to that of the ISD initialization algorithm. Even in these simulations, utilizing 100 mixture components, it was observed that the hypotheses remained clustered in a bunch for most of the simulation, rather than being utilized effectively to follow significantly different branches. The Gaussian mixture

⁵The simulation utilized the extended clutter population discussed in Section 4.4.2; 37 Monte Carlo runs were calculated.

PDF has the capability to model complex multi-modal distributions, yet if a simple pruning mechanism is used to keep only the most likely hypotheses, the hypotheses retained by the algorithms will be anything but multi-modal. Only with an efficient merging algorithm is the true power of the MHT realized.

4.4.2 Comparison with Salmond's Joining and Clustering Algorithms. On the surface, the computational complexity of the initialization method discussed in Section 3.3.4 appears to be an order of magnitude higher than the complexity of the joining and clustering algorithms presented in [44–47]. However, in many simulations in this study, instability was encountered in the merged covariance matrices, increasing the computational complexity of the joining and clustering algorithms to an order of magnitude above the ISD initialization method.

The joining and clustering filter covariance appeared stable in simulations in which the probability of detection was set close to unity. However, when the probability of detection was reduced to 0.95 (which is still higher than experienced in many practical applications), the covariance of hypotheses farther from the true target tended to increase to such an extent that the measurements for the entire surveillance region were associated with the hypothesis. In such a situation, the computational complexity increases dramatically with the hypothesis covariance (more measurements are within the association gate, resulting in more hypotheses), an increase which is bounded only by the size of the surveillance region. This problem was addressed by limiting the maximum number of measurements in the association gate for any single hypothesis to 50,⁶ which should not be exceeded in any practical situation with a stable filter covariance. This approximation was not applied in the initial set of simulations; the performance of the algorithm using this limit is shown in Figure 4.17.

⁶i.e., the 50 measurements closest to the predicted value for that hypothesis.

As discussed in Section 3.3.1.3, the ISD cost function applies lower cost weighting to components with larger covariance than to those with smaller covariance. Because of this de-prioritization, large covariance components tend to be discarded, thus avoiding this explosive increase in computational complexity. In this regard, the ISD initialization algorithm appeared stable throughout the simulations, far more so than algorithms which concentrate purely on merging.

Another problem with the joining and clustering algorithms is the delicate relationship which exists between the threshold used to discard components, and the probability that the target is detected and is within the association gate. This was observed in simulations in which the probability of detection was set to 0.999, and the probability that target is within the gate was set to 0.98 and omitted from the hypothesis probability calculation (effectively setting $P_{dg} = 0.999$ in Eq. (2.60) rather than P_d). In the resulting set of hypotheses, events in which the target is not detected are de-weighted by 999 times in comparison with events in which the target is detected. Following the advice of Salmond [44], the threshold for discarding components was set such that the least likely 1% of hypotheses are discarded and the remainder maintained and merged until the desired level of reduction has been achieved. However, with the probability of detection set to 0.999, this guarantees that almost all events hypothesizing missed detection will be discarded without further consideration. In such situations, if the target is not detected (or if the target-originated measurement falls outside of the association gate), the correct hypothesis (missed detection) will be discarded, and the system will quite possibly lose track. Even with this incorrect hypothesis probability calculation, the ISD initialization method performed well, demonstrating the robustness which is incorporated into the algorithm through the trade-off between the cost of merging components and the cost of deleting components.

The performance of Salmond's joining algorithm is compared to the ISD initialization algorithm in Figure 4.12. The diagram demonstrates that each algorithm

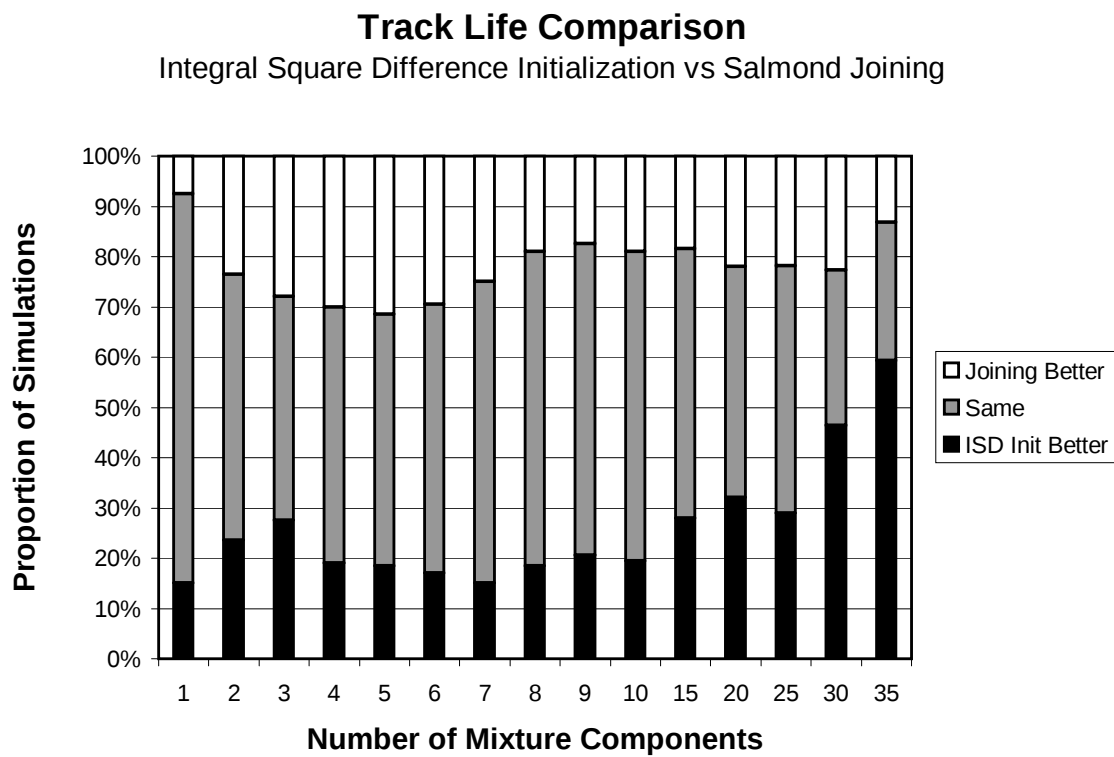


Figure 4.12. Performance of ISD initialization algorithm compared to Salmond joining algorithm.

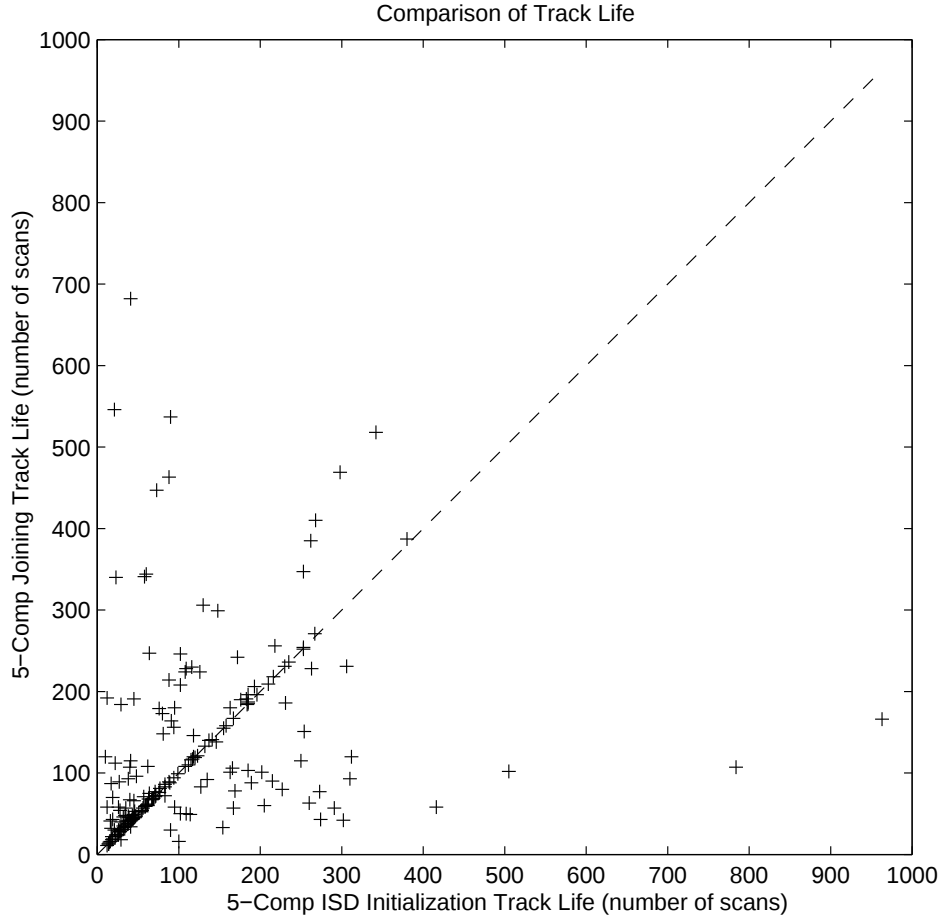


Figure 4.13. Performance of 5-component ISD initialization algorithm compared to 5-component Salmond joining algorithm.

exhibits better performance at different ends of the spectrum. For a small number of mixture components, the performance of the two algorithms is roughly equivalent. The Salmond joining algorithm exhibits noticeably better performance than the ISD initialization algorithm when the number of mixture components is between four and seven. The scatter plot comparing the ISD initialization and joining algorithms using five components is shown in Figure 4.13. The diagram demonstrates that there is a major concentration of simulations in which the joining algorithm performs *slightly* better than the ISD initialization algorithm, and that apart from this cluster (in the

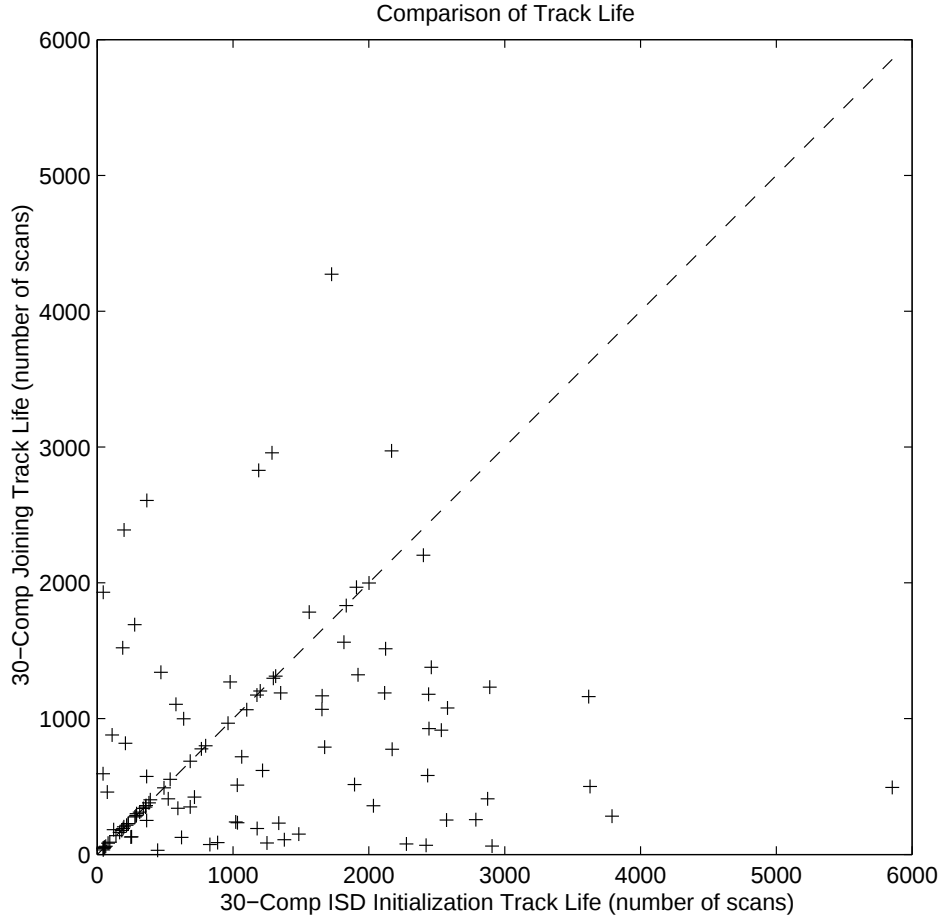


Figure 4.14. Performance of 30-component ISD initialization algorithm compared to 30-component Salmond joining algorithm.

bottom left corner, slightly above the 45° line), there is little difference between the two algorithms.

At the upper end of the scale (for 15 mixture components and greater), it is apparent that the ISD initialization algorithm significantly outperforms the joining algorithm, which previously provided the best performance in this scenario. The scatter plots for the 30- and 35-component cases are shown in Figures 4.14 and 4.15 respectively. The diagrams illustrate that the ISD initialization algorithm outperforms the joining algorithm significantly for a large proportion of the simulations.

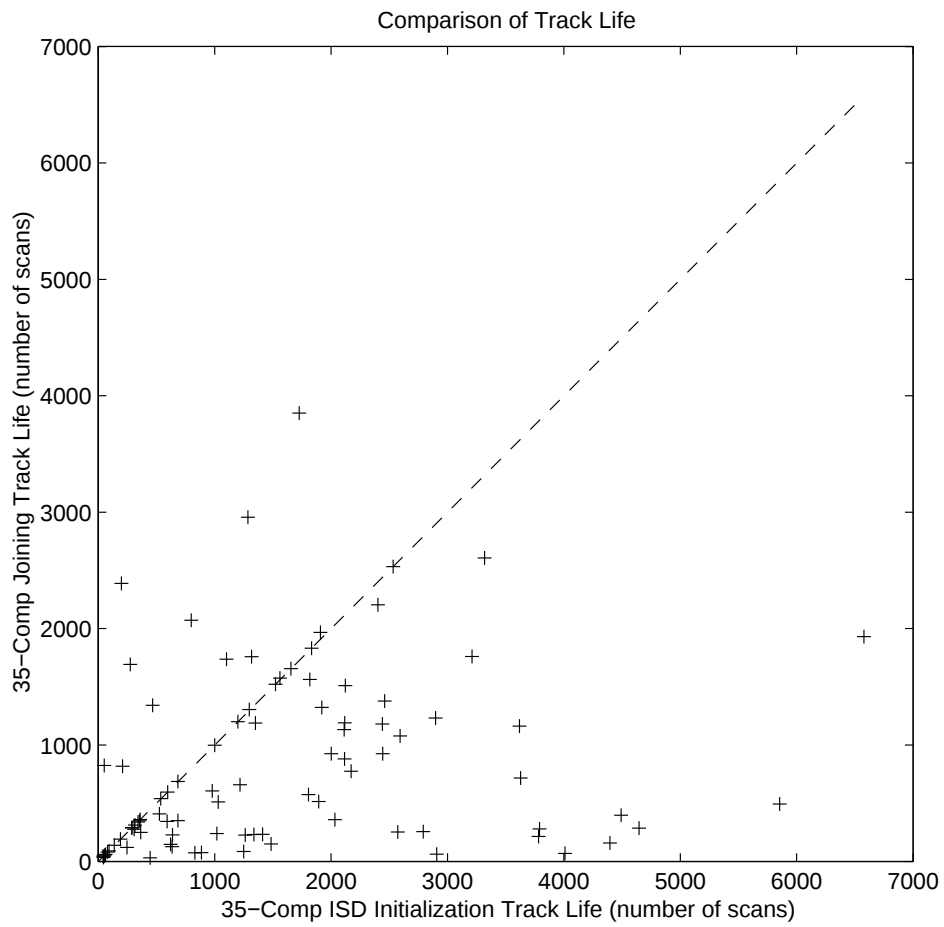


Figure 4.15. Performance of 35-component ISD initialization algorithm compared to 35-component Salmond joining algorithm.

The excellent performance of the ISD initialization algorithm is indicative of the efficiency gained by taking *all* mixture components into consideration when selecting merging and deletion actions, rather than considering only individual pairs, as done in the Salmond algorithm. Another explanation of the failing performance of the joining algorithm is the deletion threshold. As the total number of hypotheses increases in number, the probability of each individual hypothesis decreases, and there is a large number of *incorrect* hypotheses with which the *correct* hypothesis must compete to gain probability. In such a situation it is possible that the *correct* hypothesis could fall within the least likely 1% of hypotheses, and mistakenly be deleted. This is another example of the *lack of robustness* which is unavoidable in algorithms that are unable to evaluate the relative cost between merging components and deleting components.

The performance of Salmond’s clustering algorithm is compared to the ISD initialization algorithm in Figure 4.16. The diagram exhibits the same general trends as Figure 4.12, although the overall performance of the clustering algorithm is inferior to that of the joining algorithm.⁷ The ISD initialization algorithm’s performance increases for large numbers of components far more in comparison with the clustering algorithm than in the comparison with the joining algorithm, which indicates that the clustering algorithm is less efficient when a large number of components is being used.

The joining and clustering implementations previously discussed did not limit the maximum number of measurements associated with a single hypothesis, hence difficulties with unstable covariance growth were experienced, as previously described. The comparison is repeated in Figure 4.17 for the 30- and 35-component simulations (utilizing 200 Monte Carlo simulations for the 30-component case and 106 for the 35-component case) with the maximum number of hypotheses spawned

⁷As described in [44–47], the clustering algorithm was designed as a further approximation to the joining algorithm in an attempt to reduce the computational complexity.

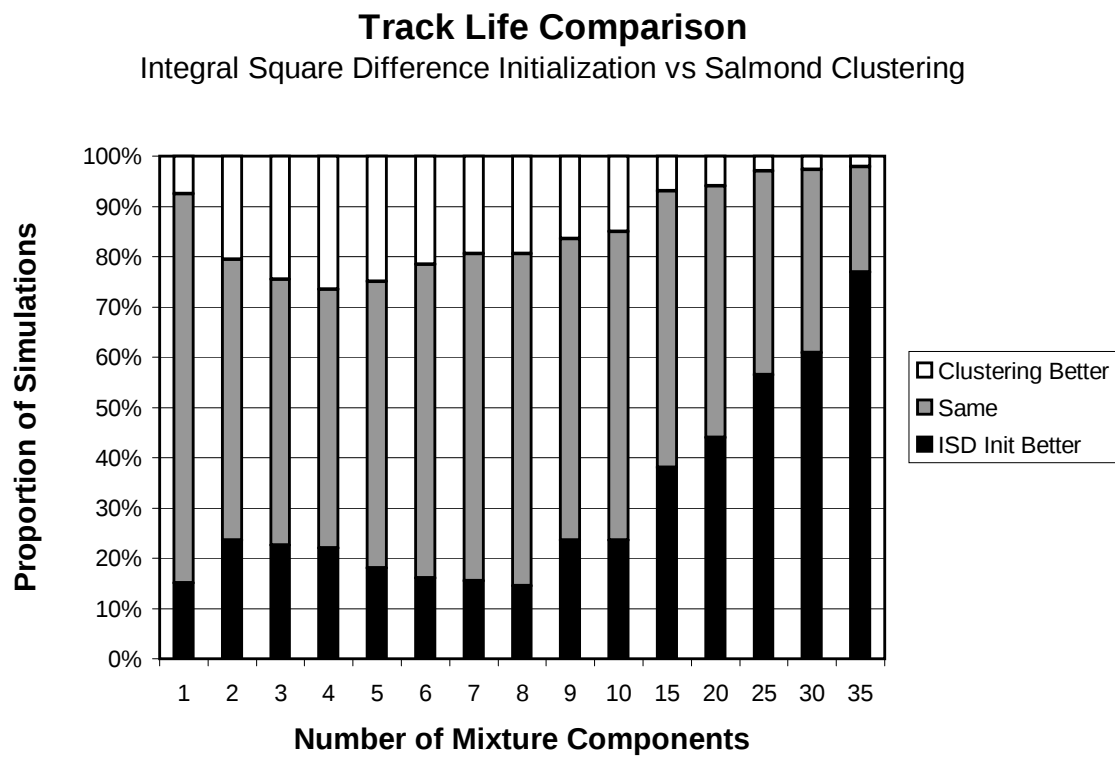


Figure 4.16. Performance of ISD initialization algorithm compared to Salmond clustering algorithm.

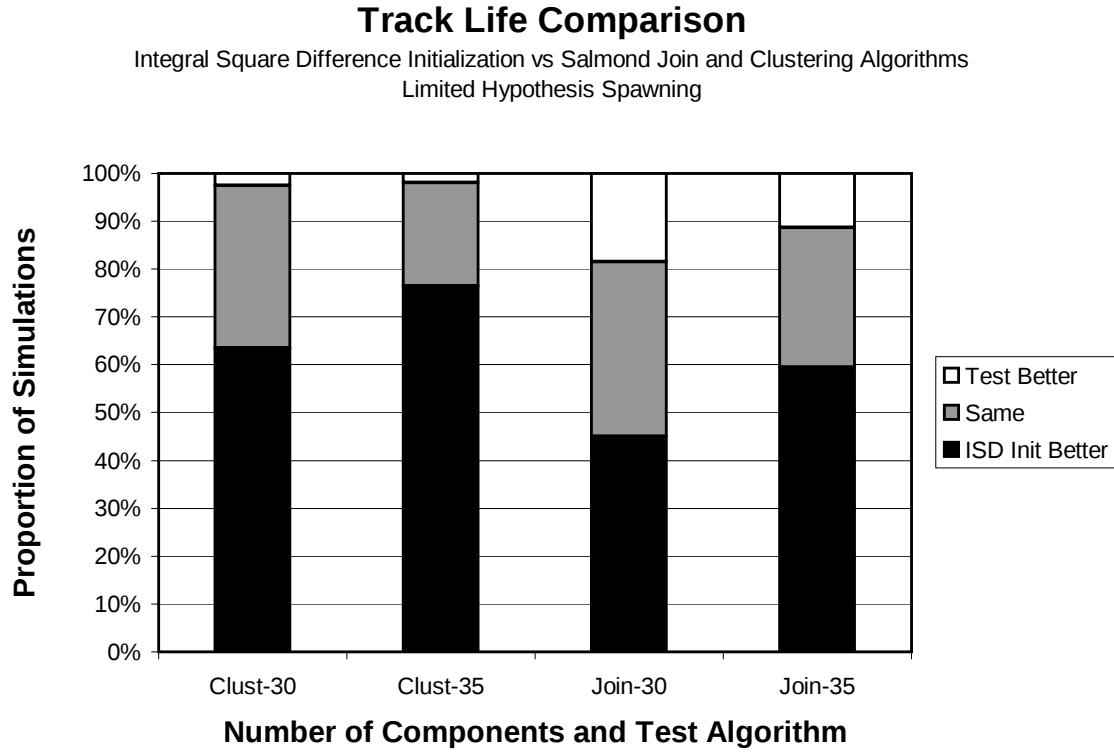


Figure 4.17. Performance of ISD initialization algorithm compared to Salmond clustering and joining algorithms with the maximum number of hypotheses spawned by any parent hypothesis limited to 50.

by any parent hypothesis limited to 50. Comparing the diagram with Figures 4.12 and 4.16 indicates that the hypothesis limiting makes very little difference to the performance of Salmond's algorithms. Any practical implementations of the algorithms would need to limit the number of hypotheses to ensure computational tractability.

Following from the earlier discussion of the instability of component covariances using Salmond's joining algorithm, the 25-, 30- and 35-component simulations were repeated, with the region populated by clutter increased in size by $10\times$, from a square of side $200\sqrt{r}$ to a square of side $2,000\sqrt{r}$. This change in the clutter population region increases the expected number of clutter-originated measurements for each scan interval from 480 to 48,000. To process this number of false measure-

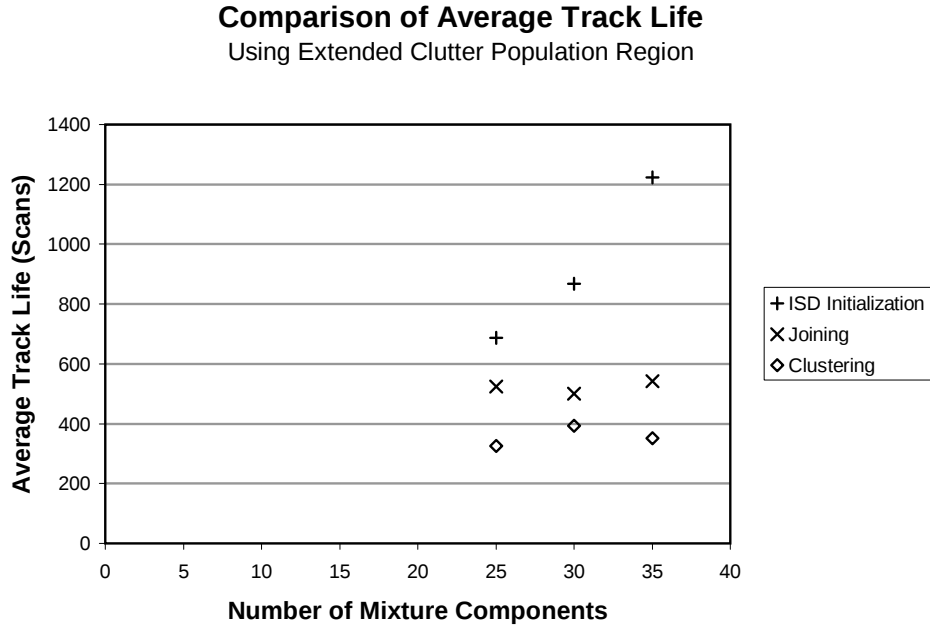


Figure 4.18. Average track life for scenario using extended clutter population region.

ments, a more efficient gating routine was necessary; the algorithm described in Appendix A.2, which is algebraically equivalent to the original gating methodology, was employed. Using these updated parameters, 73 Monte Carlo simulations were calculated. As the clutter *density* was not modified, one would expect that the performance would be unchanged (on average), unless the algorithms were being aided by the limited clutter population region. The average track life of each algorithm for the modified scenario is shown in Figure 4.18 (all algorithms had the number of hypotheses spawned by any parent hypothesis limited to 50). The diagram confirms the trend that the performance of the ISD initialization algorithm increases exponentially as the number of components is increased, whereas the performance of the joining and clustering algorithms levels out. This again indicates that the performance of the ISD initialization algorithm is limited only by the availability of computing resources, whereas little performance benefit is gained using the joining and clustering algorithms by increasing the size of the mixture beyond 25 compo-

nents. Comparing Figure 4.18 to Figure 4.8 reveals that the average track life of *all* algorithms has been reduced somewhat, indicating that all of the algorithms may have been assisted by the limited clutter population region somewhat. The histogram of the data used for these points is shown in Figure 4.19. From this diagram, it is difficult to judge whether the increased clutter population region caused a deterioration in the performance of the various algorithms. There is a visible difference between the original simulation histogram and the extended clutter population region histogram for the 35-component joining algorithm; in most other cases it is difficult to declare such a difference. This reduction in performance for the case in which the clutter population region is extended in size indicates that the algorithm is benefitting from the limited clutter population region. This phenomenon is probably caused by incorrect hypotheses drifting beyond the area populated by false measurements and being deleted when they otherwise may have survived longer.

Even if the various algorithms were aided by the limited clutter population region, this is not entirely unrealistic. Practical radar systems have limited detection ranges, and individual detections are limited by the maximum unambiguous range (and Doppler) imposed by the waveform in use [50]. Hence the realistic radar measurement space will *not* extend infinitely, but rather it will exist within a bounded region.

The performance of the ISD initialization algorithm is compared to the joining and clustering algorithms for the updated scenario in Figure 4.20. The diagram again confirms the superior performance of the ISD initialization algorithm for large numbers of mixture components; the performance difference is significantly greater than that shown in Figure 4.12, indicating that Salmond's algorithm was being assisted by the finite clutter population region to a much greater extent than the ISD initialization algorithm. The ISD initialization and Salmond joining algorithms using 25, 30 and 35 mixture components are compared in the scatter plots of Figures 4.21, 4.22 and 4.23 respectively. The diagrams demonstrate that the ISD initializa-

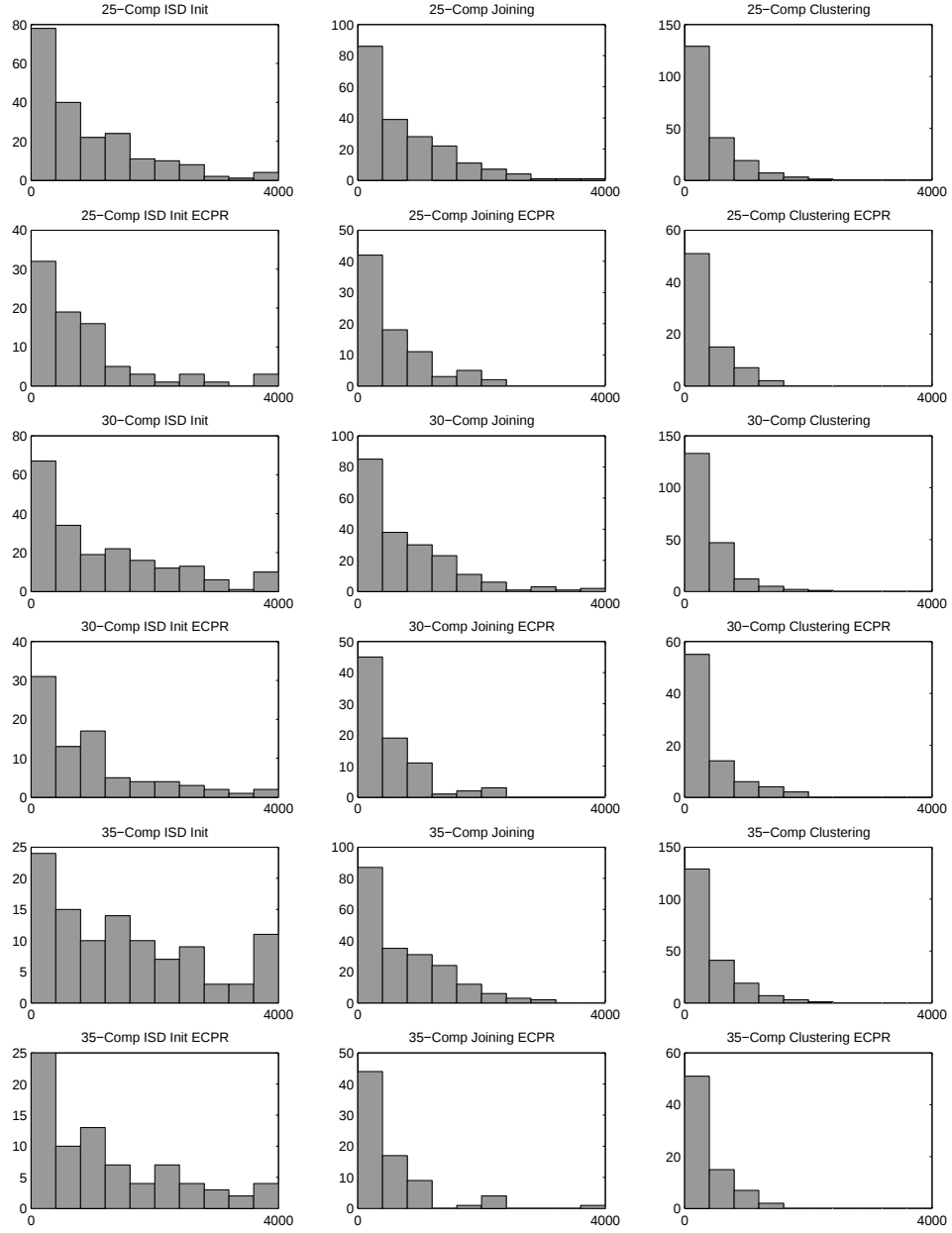


Figure 4.19. Histogram of track life for ISD initialization and Salmond's joining and clustering algorithms, utilizing 25, 30 and 35 mixture components. Plots labelled "ECPR" describe the Monte Carlo simulations utilizing the extended clutter population region; the remaining plots describe the original scenario.

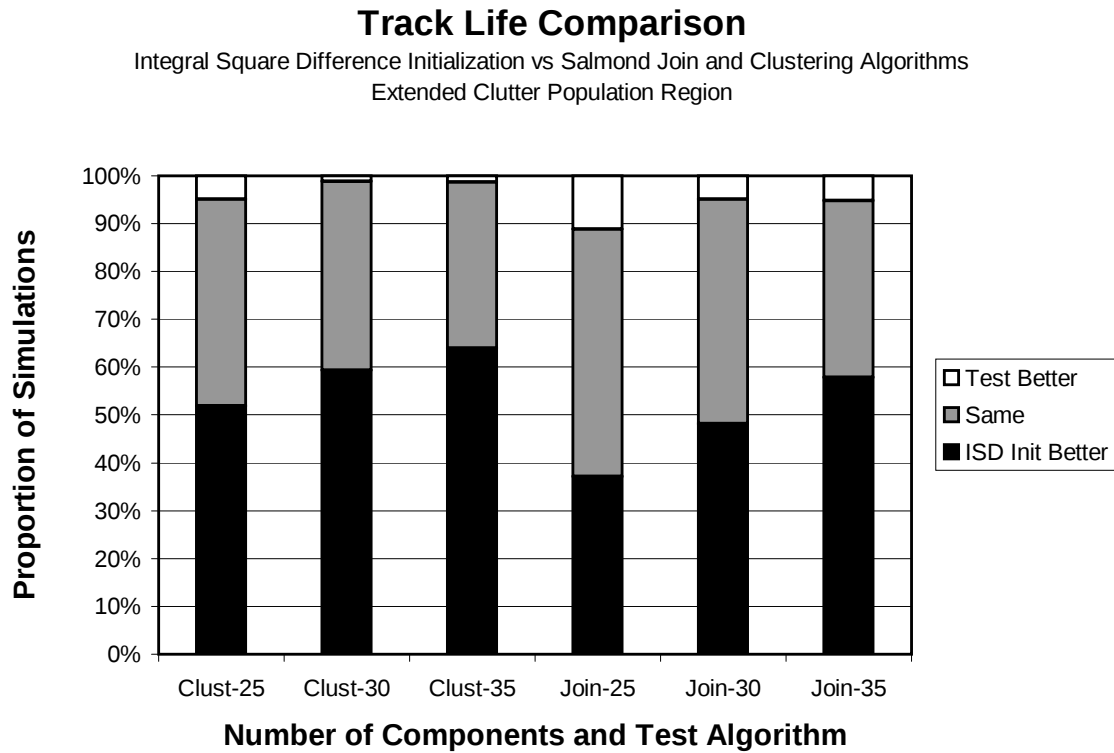


Figure 4.20. Performance of ISD initialization algorithm compared to Salmond joining and clustering algorithms, with clutter population region increased in size by ten times in both x and y axis directions.

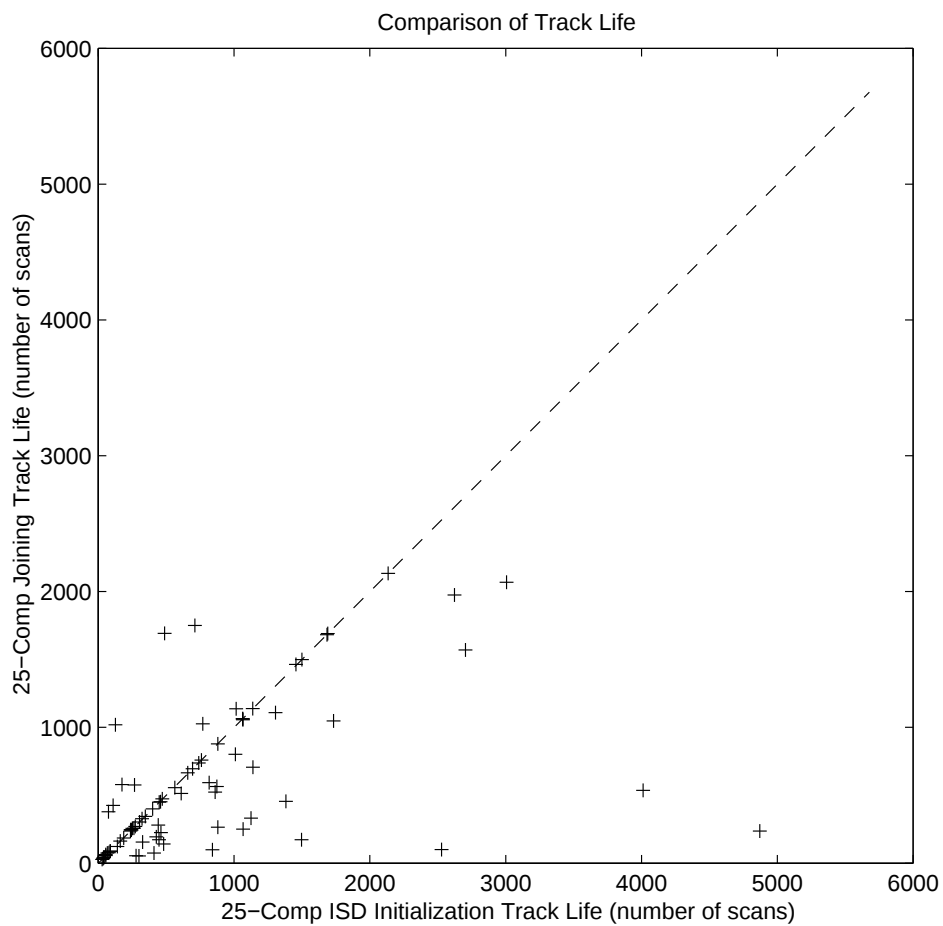


Figure 4.21. Performance of 25-component ISD initialization algorithm compared to 25-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.

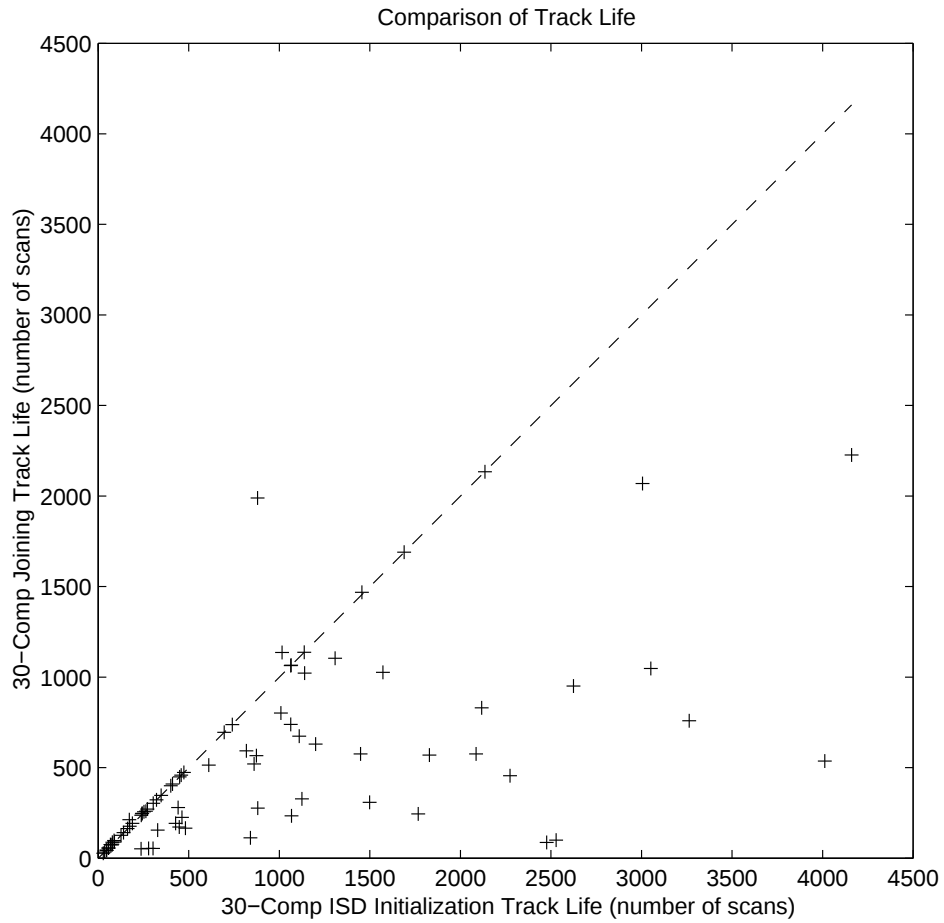


Figure 4.22. Performance of 30-component ISD initialization algorithm compared to 30-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.

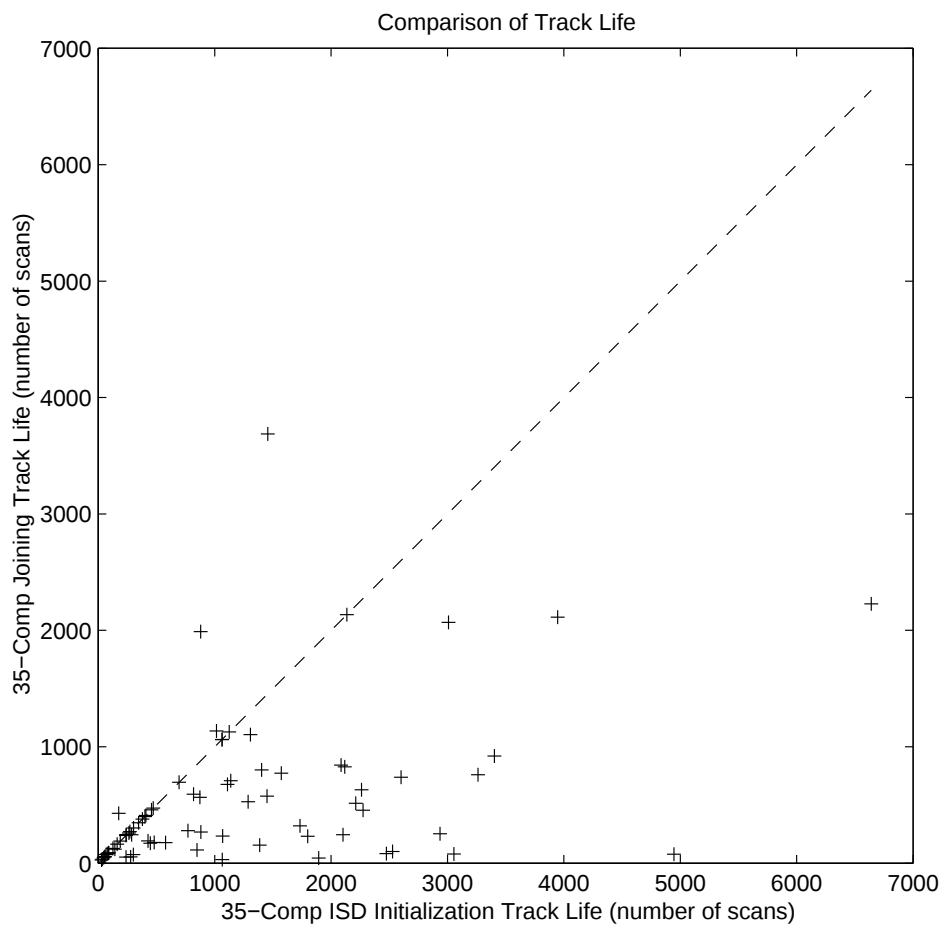


Figure 4.23. Performance of 35-component ISD initialization algorithm compared to 35-component Salmond joining algorithm, with clutter population region increased in size by ten times in both x and y axis directions.

tion algorithm increasingly outperforms the Salmond joining filter as the number of mixture components grows. The scatter plots have very few points above the 45° line, and a large number of points significantly below the 45° line, indicating that the ISD algorithm is outperforming Salmond's joining algorithm, which has previously provided the best performance in this scenario, by a considerable margin. The difference between the algorithms appears to be greater than that shown in Figures 4.14 and 4.15, which again indicates that the joining algorithm was being aided by the limited clutter population region. Once again it should be noted that Salmond's joining algorithm previously was considered to provide the best performance for this type of scenario.

4.4.3 Comparison with Lainiotis Algorithm. The algorithm of Lainiotis [31] was considered as another reference to provide further comparison. The implementation described in [31:627] uses two separate thresholds for merging and deleting, using Eqs. (2.76) and (2.77) to evaluate the cost of each action. In order to provide a better comparison to the initialization algorithm presented in Section 3.3.4, the algorithm was implemented to compare the cost of all possible actions (both merging and deleting), and the action producing the smallest cost was taken. When this implementation was executed, the algorithm was found to *delete* mixture components at almost every step, rarely choosing to merge at all.⁸ This trend was observed also when the algorithm was applied to the simple one-dimensional reduction discussed in Section 4.2, leading to extremely poor reduced PDF approximations. This indicates that the costs calculated for deleting and merging components are not suitable for comparison: they provide a reasonable mechanism for evaluating the relative cost of deleting different components, and likewise a reasonable mechanism for evaluating the relative cost of merging pairs of components, but they *do not* provide a trade-off

⁸The increased performance of the various merging algorithms presented previously as compared to the standard MHT pruning algorithm demonstrates that *merging* components is almost always a better choice than *deleting* components.

between deleting and merging. This also reveals why the author chose to select a different threshold for deleting and merging.

After observing from the ISD initialization algorithm that the lowest cost action is usually *merging* components, and that the system rarely chooses to *delete* components, the Lainiotis algorithm was implemented to merge sequentially the pair of components which leads to the smallest reduction in the Bhattacharyya coefficient between the original PDF and the approximation (denoted ρ_a), as according to the approximation of Eq. (2.77):

$$\rho_a \geq 1 - (p_i + p_j) \sqrt{1 - \rho_{i,j}^2}$$

where p_i and p_j are the weights of the two components to be merged, and $\rho_{i,j}$ is the Bhattacharyya coefficient between the two individual components being considered for merging. The merging continues until the number of mixture components has been reduced to the desired level. The algorithm was run for the same test case described above, for 50 Monte Carlo runs, using 3, 4, 5, 6, 7 and 8 mixture components. The results of the algorithm are compared to the ISD initialization algorithm in Figure 4.24. The diagram indicates that the performance of the technique is not as good as that provided by either joining or clustering, and thus the technique is further inferior to ISD initialization than either of Salmond's algorithms. Observing the form of the cost function of Eq. (2.77), the difference is probably caused by the difference in the way in which the probability weights are incorporated into the equation. The Salmond expressions (Eqs. (2.78) and (2.79)) incorporate the probability weights through the factor $p_i p_j / (p_i + p_j)$. This expression is a smooth interpolation of the minimum of the two probability weights, similar to the combined resistance of two resistors in parallel. Accordingly, Salmond's expressions will tend to allow very small components to be merged into larger components, effectively providing a mechanism for deleting small components without changing the overall structure of the PDF. Lainiotis' expression incorporates the probability weights through the

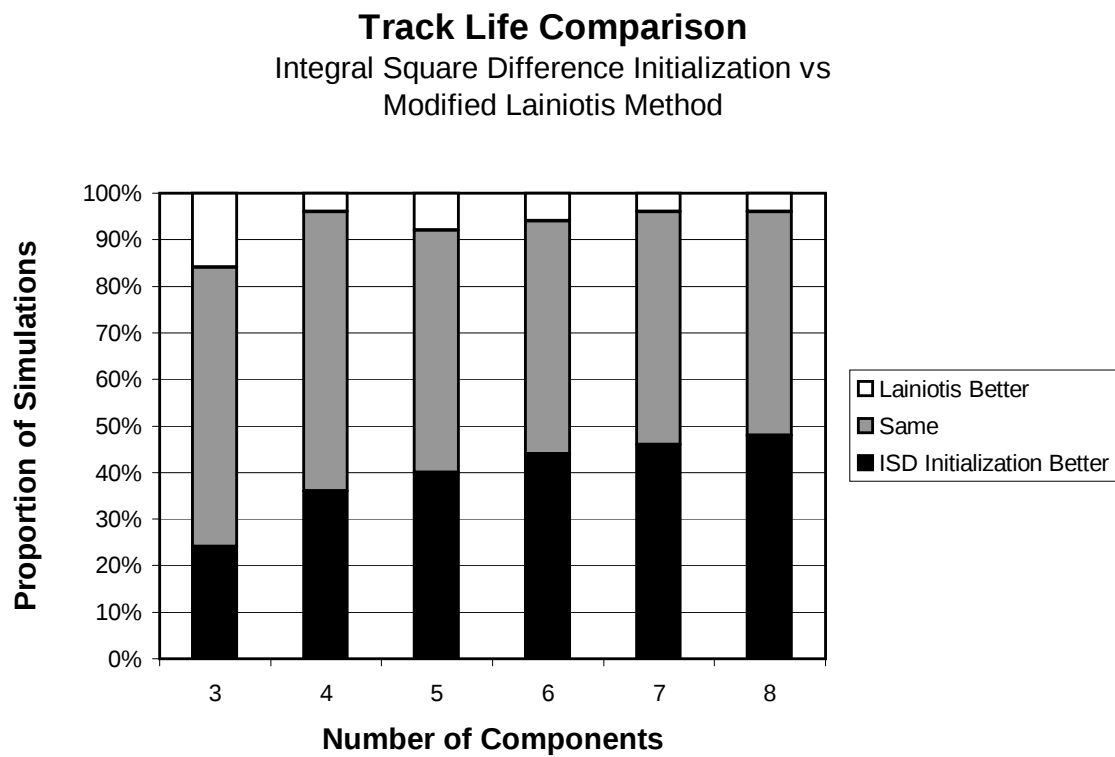


Figure 4.24. Performance of ISD initialization algorithm compared to modified Lainiotis algorithm.

factor $(p_i + p_j)$, which will be large if *either* weight is large, not providing such a deletion mechanism.

4.4.4 Comparison with Iterative Optimization Algorithm. The results presented previously were obtained using the ISD *initialization* algorithm, without utilizing the *iterative optimization* method described in Section 3.3.3. The initialization algorithm was utilized without the iterative optimization algorithm mainly due to the computational complexity of the iterative technique.

The iterative optimization method was applied over 50 Monte Carlo simulations in order to evaluate the performance enhancement produced. Considering the discussion of Section 4.3, it would not be surprising if the iterative optimization technique did not produce a substantial increase in performance, as the overall structure of the PDF approximation remains largely unmodified. However, the results of these simulations appear to indicate that the performance using the iterative optimization method is actually slightly *worse* than the initialization algorithm alone, which is rather surprising. The performance of the two algorithms is compared in Figure 4.25, using 1 to 10 mixture components. The scatter plot for the 10-component case is shown in Figure 4.26, demonstrating that the iterative optimization substantially reduces the track life in a number of simulations.

This result is even more surprising when one considers that the iterative optimization technique is *guaranteed* not to increase the cost of the reduced PDF representation, and that in almost any practical situation the cost will indeed be reduced. Thus the outcome of this is that a PDF representation producing a lower ISD cost *does not necessarily result in better tracking performance*. An interesting interpretation of this result is found when one considers the equations used to calculate the parameters of the merged component when two mixture components are combined in the ISD initialization algorithm, as shown in Eq. (2.24). The parameters in this equation are such that the mean and covariance of the overall mixture remain

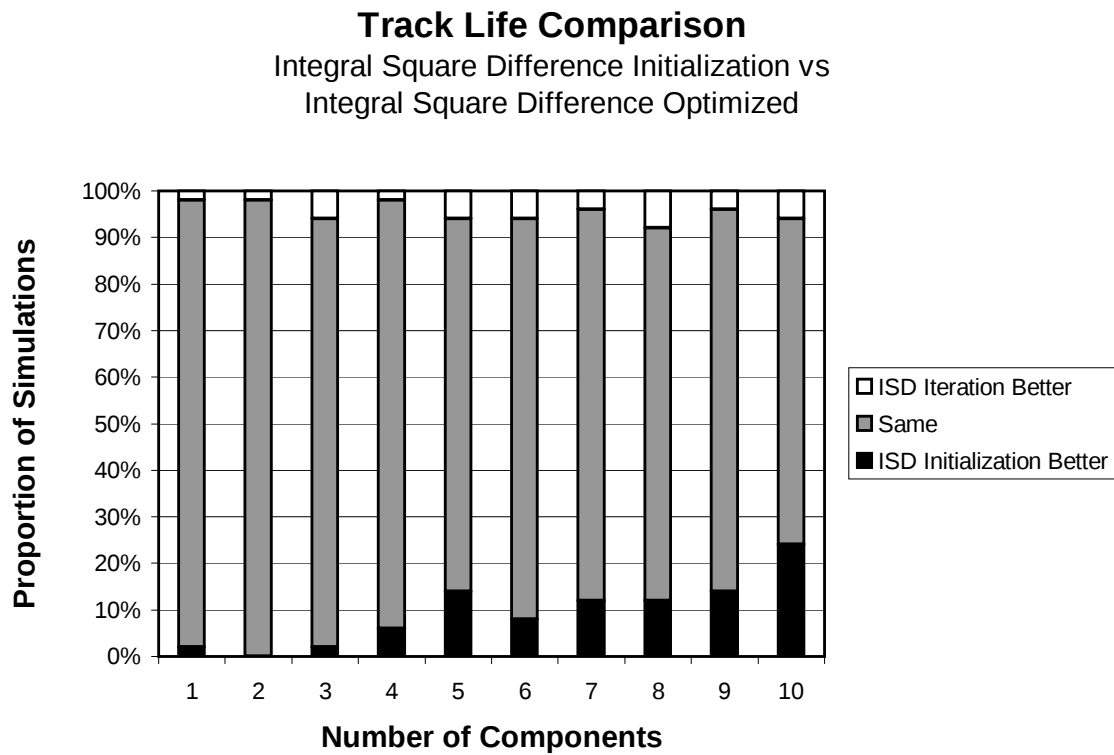


Figure 4.25. Performance of ISD initialization algorithm compared to same algorithm utilizing iterative optimization to refine the approximation.

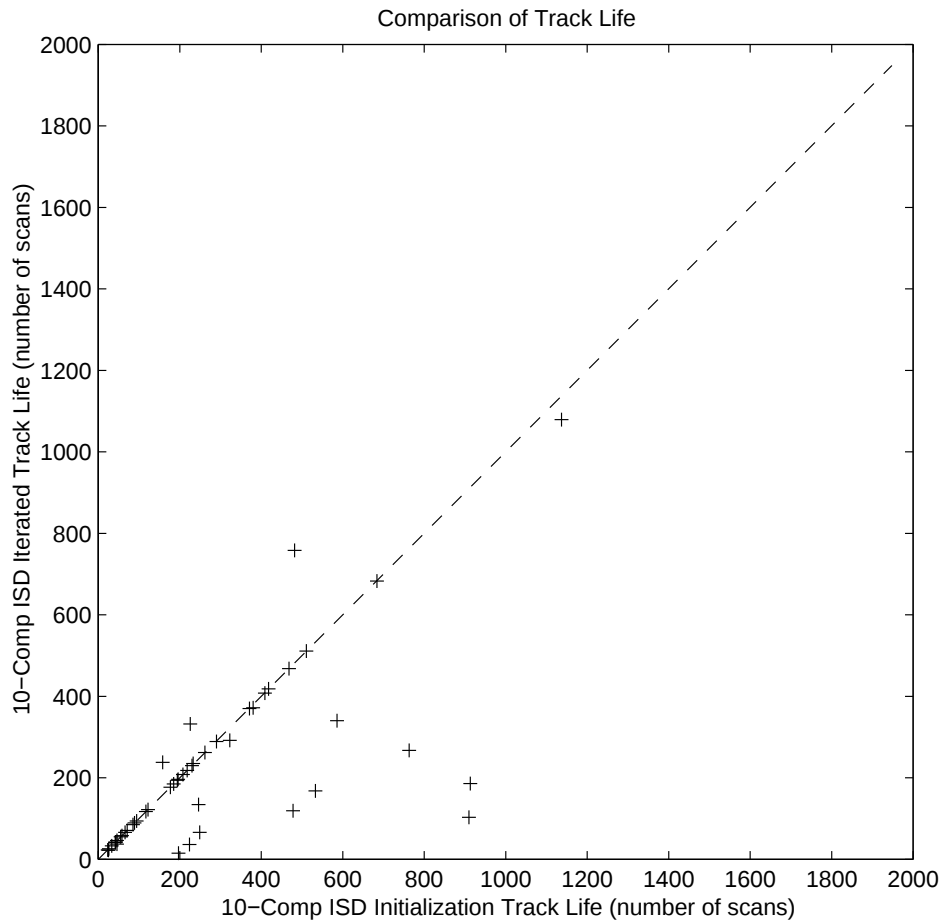


Figure 4.26. Comparison of track life for simulations of 10-component ISD initialization algorithm, and the same algorithm utilizing iterative optimization to refine the approximation.

unchanged through the merging operation. These parameters do not necessarily represent the optimal fit of a single component to the original two components according to the ISD cost function.⁹ Rather, these parameters can be shown to be the optimum parameters according to the Maximum Likelihood measure, as presented in Section 3.3.1.4. Accordingly, the ISD initialization algorithm is *not* based on the ISD cost function alone: it uses the ISD cost function to select which components to merge or delete, and then uses the Maximum Likelihood measure to calculate the parameters of the merged components. The result of this section therefore indicates that the performance of this “hybrid” approach, incorporating the Maximum Likelihood measure to select the parameters of the merged components, is better than that of a “pure” ISD implementation. This is not surprising when one considers that the Maximum Likelihood measure was the preferred cost function in terms of physical meaningfulness — except that it was unable to be evaluated.¹⁰

4.4.5 Comparison with PDA Algorithm. In order to provide a comparison with the performance of the various algorithms using a single Gaussian mixture, the simulations were run for the Probabilistic Data Association (PDA) algorithm, described in Section 2.5.7. In this scenario the clutter density was high enough that the covariance of the PDA algorithm exhibited unbounded growth until loss of track occurred. The results of the simulations are compared in Figure 4.27. It is not surprising that the performance of the Salmond merging and clustering algorithms is almost identical to that of the PDA algorithm. Other than the deletion logic

⁹As far as the author is aware, there is no closed-form solution for the optimal parameters, according to the ISD cost function, of a single component representing a pair of components.

¹⁰The parameters for the merged component can be found in closed form using the Maximum Likelihood measure because the logarithm operation in Eq. (3.16) is of the *reduced* mixture. Thus, if the reduced mixture contains only a single component (as is the case when we are solving for the parameters of the single component which provide the best fit to a pair of components), then the logarithm will be of a single Gaussian function, which results in an expression which can be evaluated in closed form, and a parameter optimization that can be solved also in closed form. If the reduced mixture contains multiple components, then the logarithm operation cannot be simplified, and the cost function must be evaluated through numerical integration or approximation. Hence, purely due to computational tractability, the ISD cost function remains preferable.

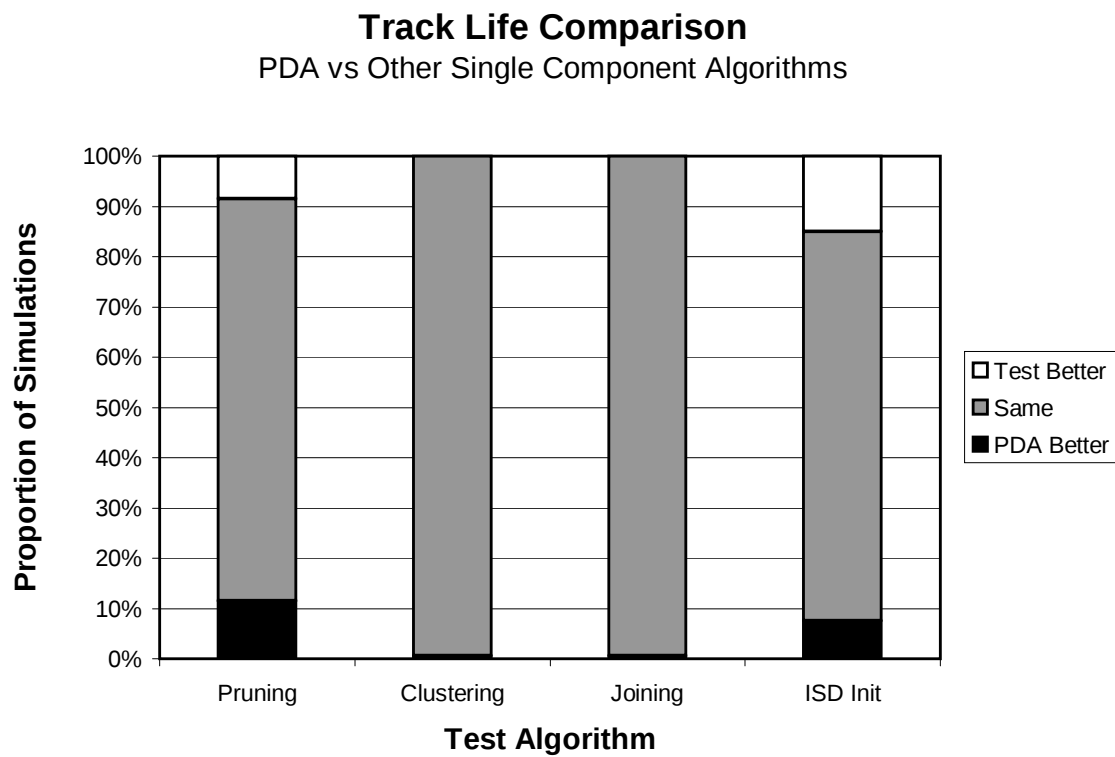


Figure 4.27. Performance of PDA compared to other algorithms using a single Gaussian component.

(which discards the set of least likely hypotheses which contribute 1% of the overall probability mass), these algorithms are algebraically identical to PDA. The slight reduction in performance indicates that the deletion logic is harmful to the overall performance. It is also not surprising that the performance of the pruning algorithm is worse than that of PDA. A single-component pruning algorithm *is* the Nearest Neighbor algorithm described in Section 2.5.6, and the poor performance of the Nearest Neighbor algorithm compared to PDA has been well documented [4:139–141, 7:373].

The surprising result of Figure 4.27 is that the single-component ISD initialization algorithm visibly outperforms PDA. In order to verify this result, this comparison was repeated for 1,000 Monte Carlo simulations. The scatter plot for these simulations is shown in Figure 4.28. In this new set of simulations, the ISD initialization algorithm outperformed the PDA algorithm 12.5% of the time, the PDA algorithm outperformed the ISD initialization algorithm 8.8% of the time, and the two were essentially identical (track life within 10 scans of each other) for 78.7% of the time. The diagram shows that there is a large concentration of simulations for which the PDA performs slightly better than the ISD initialization algorithm, but the difference in performance is less than 10 scans, hence they are counted as being identical. The performance of the ISD initialization algorithm is spread much further out towards the larger values in Figure 4.28, indicating that the track life is more likely to be significantly longer than that of the PDA algorithm. The histograms of the track life of the two algorithms are shown in Figure 4.29. The diagram shows that, while the means of the two track life distributions is roughly identical, the ISD initialization is further skewed such that there are more points in the tail of the distribution, representing significantly longer track life. Overall, however, one would suppose that the mean performance is not significantly improved.

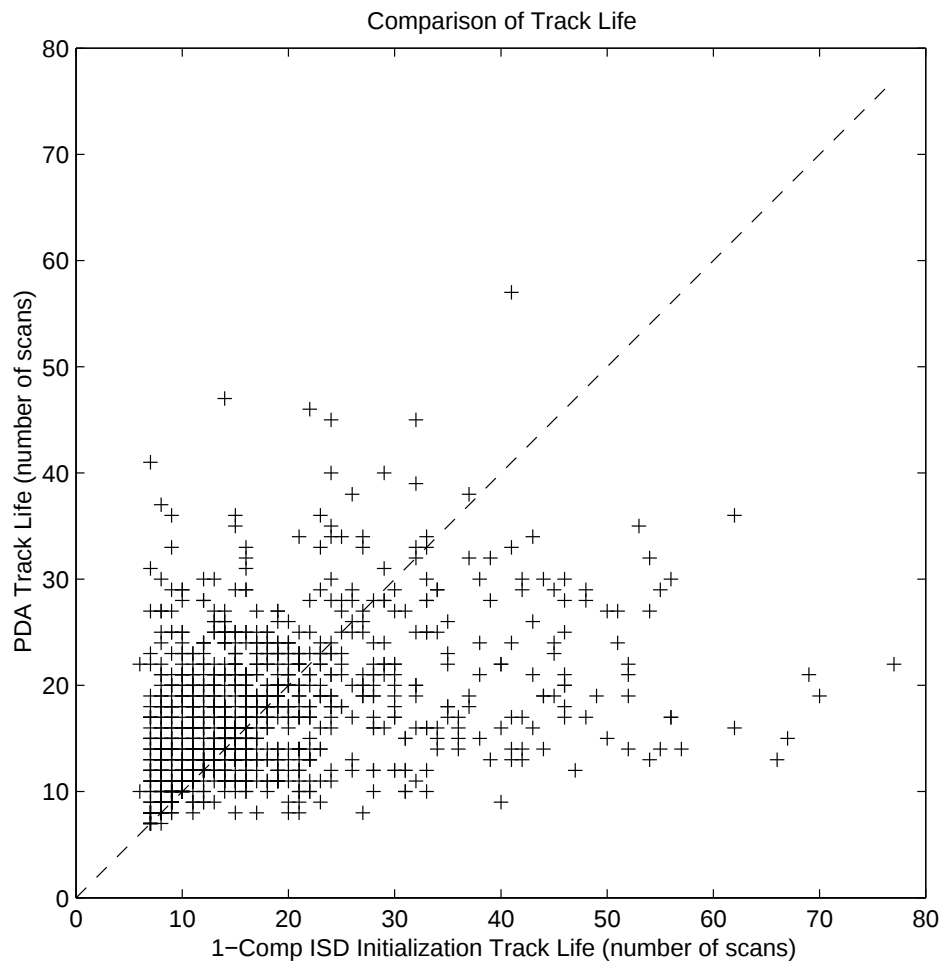


Figure 4.28. Performance of PDA compared to other algorithms using a single Gaussian component.

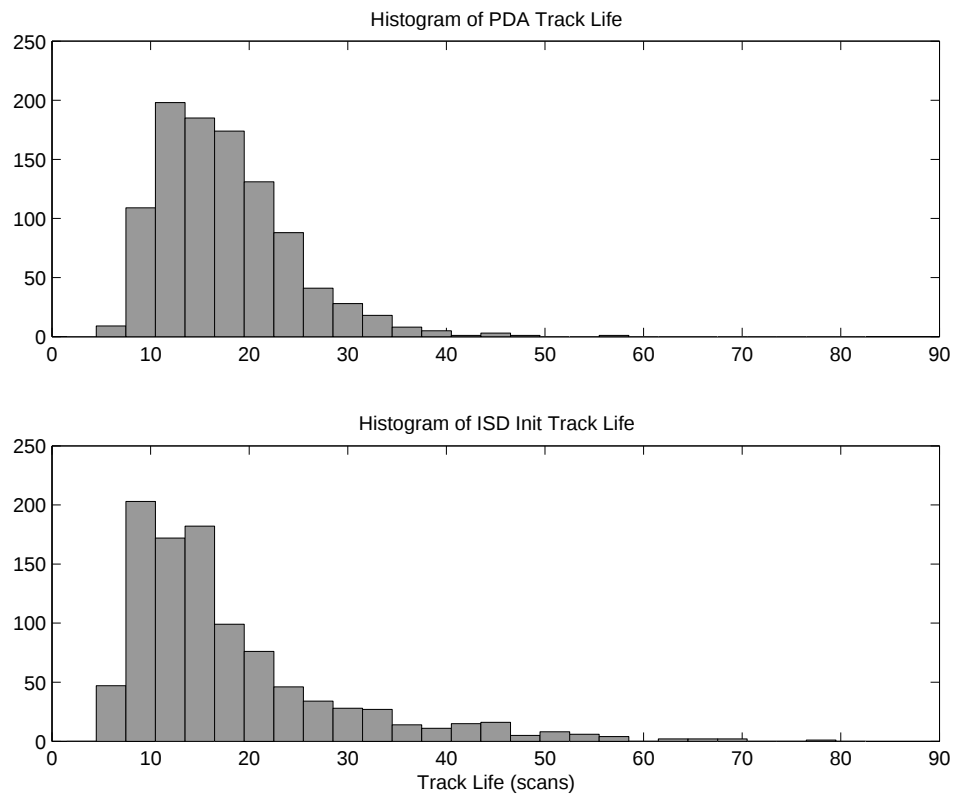


Figure 4.29. Histograms of track life for PDA algorithm and single-component ISD initialization algorithm.

4.5 *Multiple Targets in Clutter*

In order to test the performance of the system tracking multiple targets in clutter, the system was extended to the multiple target model as presented in Section 2.5, and tested using a two-target scenario. The system was programmed to generate joint association hypotheses for every joint association event, each with the estimated joint state of the targets, joint state covariance and hypothesis weight, as described in Section 2.5.5. While the implementation performed well (as expected) in low-clutter tracking conditions, the computational complexity prevented any testing from being conducted in high clutter conditions (where the more efficient merging algorithm is beneficial). This prevented any meaningful comparison with previously published merging and pruning algorithms.

The SB-MHT algorithm described in Section 2.5.10 maintains separate sets of single target hypotheses for each individual target, alongside listings of compatible hypotheses which can be used to form joint hypotheses (each including one single target hypothesis from each target's list). Such a structure retains much of the benefit of the direct joint target state representation, but for a fraction of the memory and computational cost. A multiple target extension of the ISD initialization algorithm using such an architecture could be performed; however, due to time limitations this was not attempted.

4.6 *Single Maneuvering Target*

As discussed in Section 2.4.2, the PDF of target state for a single target which switches between different dynamics models at unknown time instants is also a Gaussian mixture in which the number of components grows exponentially with time. Accordingly, this is another problem to which the ISD initialization and optimization algorithms could be applied.

In order to test the performance of such an implementation, a full-order Bayesian filter was developed using a Markov transition model. The development

of the algorithm follows Section 2.4.2; the structure of the algorithm is shown in Figure 2.3. At the end of each processing cycle the hypotheses are combined using the ISD initialization algorithm such that the maximum number of hypotheses is not exceeded. When hypotheses are merged, a different form of transition probability will be required, to represent the probability that the system transitions from one of a merged set of models to a new model. This calculation is described in Appendix A.3.

The algorithm was tested using a scenario adapted from [21], which simulates a target flying on segments of constant velocity, in between segments of constant acceleration turn. The state space representation of the target truth model is:

$$\begin{aligned}
\mathbf{x}(k) &= \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \mathbf{u}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \mathbf{w}(k-1) \\
\mathbf{z}(k) &= \begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(k) + \mathbf{v}(k)
\end{aligned} \tag{4.2}$$

where T is the time between measurement intervals $(k-1)$ and k , and $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are two independent zero-mean white noise processes such that:

$$\begin{aligned}
E\{\mathbf{w}(k)\mathbf{w}(k)^T\} &= \mathbf{Q} = q\mathbf{I} \\
E\{\mathbf{v}(k)\mathbf{v}(k)^T\} &= \mathbf{R} = r\mathbf{I}
\end{aligned}$$

where the measurement noise covariance $r = 2000$, the dynamics noise covariance $q = 10^{-4}$, and the acceleration input $\mathbf{u}(k)$ is as shown in Table 4.4. The initial velocity of the target is 15 units/sec in the $-y$ direction.

Sample	Model	Acceleration
1–35	Constant velocity	$\mathbf{0}$
36–40	Constant velocity plus input	$[0.5 \ 0.5]^T$
41–55	Constant velocity	$\mathbf{0}$
56–65	Constant velocity plus input	$[-0.2 \ -0.2]^T$
66–100	Constant velocity	$\mathbf{0}$

Table 4.4. Parameters for maneuvering target scenario.

The filter bank consists of three filters, two of which utilize constant acceleration models and the other utilizes a constant velocity model. The constant velocity filter uses the model described in Eq. (4.1), with $q = 10^{-4}$, as per the truth model. The constant acceleration models utilized the standard extension of Eq. (4.1):

$$\begin{aligned}
\hat{\mathbf{x}}(k) &= \begin{bmatrix} \hat{p}_x(k) \\ \hat{v}_x(k) \\ \hat{a}_x(k) \\ \hat{p}_y(k) \\ \hat{v}_y(k) \\ \hat{a}_y(k) \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{x}}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 1 & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \\ 0 & 1 \end{bmatrix} \mathbf{w}(k-1) \\
\mathbf{z}(k) &= \begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \hat{\mathbf{x}}(k) + \mathbf{v}(k)
\end{aligned} \tag{4.3}$$

where, as previously, T is the time between measurement intervals $(k-1)$ and k , and $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are two independent zero-mean white noise processes such that:

$$\begin{aligned}
E\{\mathbf{w}(k)\mathbf{w}(k)^T\} &= \mathbf{Q} = q\mathbf{I} \\
E\{\mathbf{v}(k)\mathbf{v}(k)^T\} &= \mathbf{R} = r\mathbf{I}
\end{aligned}$$

One of the constant acceleration filters uses $q = 10^{-4}$ to handle the constant acceleration input once the model has stabilized after the initial maneuver onset; the other

has $q = 0.25$ to aid convergence at onset. The Markov transition model was set such that the probability that the system will stay in a given model is 0.98, and the probability that the system will switch from a given model to either of the remaining models is 0.01.

The simulations were computed for 50 Monte Carlo runs using the Bayesian switching model approximation, and the IMM. The Bayesian switching model approximation used the ISD initialization algorithm to combine the outputs of the filters down to three estimates at the end of the processing cycle. Each of these three estimates was then processed using each dynamics model in the following processing cycle, similarly to the GPB-2 structure. The Root-Mean-Square (RMS) error of the system using the Bayesian switching model approximation is shown in Figure 4.30, and the RMS error of system using the IMM is shown in Figure 4.31. The results demonstrate that the performance of the Bayesian switching model approximation is worse than that of the IMM. The reason for this is that, as discussed in Section 3.3.1.3 and as illustrated in Section 4.2, the ISD cost function applies more cost to components with smaller variance than to those with larger variance. In the problem of data association, as examined in previous sections, most of the mixture components have covariances of similar magnitude, hence this weighting is not harmful, and at times it can even be beneficial. In the problem of switching models, however, the covariance matrices proposed by the various models are of vastly different orders of magnitude — some proposing that the target is travelling on a regular, predictable path, and others proposing that the target is exhibiting a high-jerk maneuver. In this situation, the ISD initialization algorithm will tend to merge or discard the more agile maneuver hypotheses, even if they are more probable than the lower covariance non-maneuvering hypotheses. This explains why the error of the ISD initialization system is lower than the error of the IMM in non-maneuvering portions of the simulations, and higher than the IMM at the harsh maneuver onset, as seen in Figures 4.30 and 4.31.

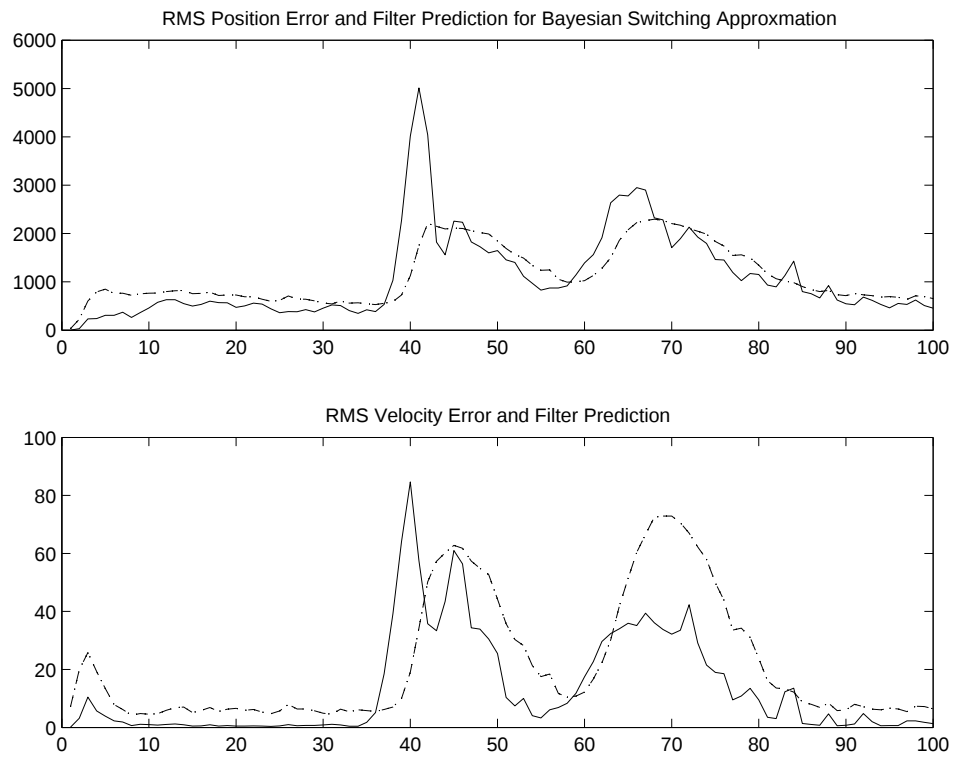


Figure 4.30. RMS position and velocity error of system utilizing Bayesian switching model approximation. Filter-predicted RMS error shown in dashed line.

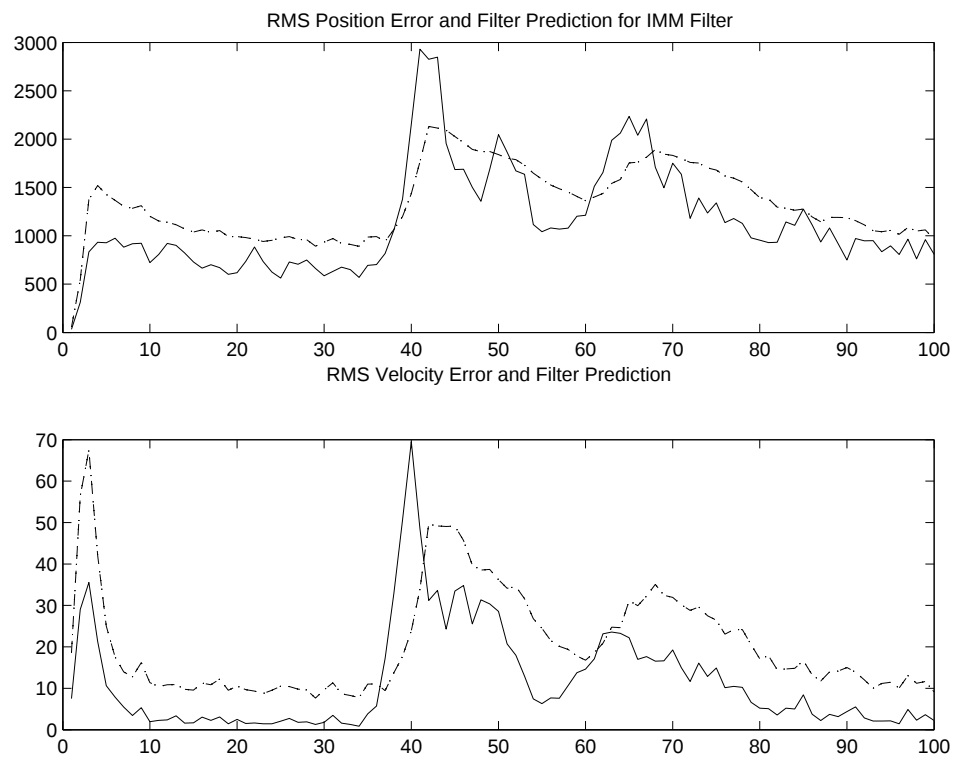


Figure 4.31. RMS position and velocity error of IMM system. Filter-predicted RMS error shown in dashed line.

The result of Figure 4.30 was largely unmodified when nine hypotheses were retained rather than three, effectively lengthening the memory of the system by a further sample period. This indicates that the limiting factor in the performance of the IMM is more the Markov transition model, rather than the approximation applied to the target state PDF. This suggests that altering the transition model, perhaps to a time-varying Markov model, could be of great benefit. A suggestion for such a model is discussed in Section 5.4.

4.7 Summary

The major outcome of this chapter is that the performance of the ISD initialization algorithm for tracking a single target in clutter is significantly better than that of any of the previously published methods tested in the comparison. Furthermore, it was demonstrated that the performance of the ISD initialization algorithm increases *exponentially* as the number of mixture components increases, whereas existing methods are unable to provide any significant improvement using more than 25 mixture components. Although no results were obtained for the multiple target tracking problem, a result similar to the single target case is likely if a computationally feasible extension is developed.

V. Conclusions and Recommendations

5.1 Restatement of Research Goal

As stated in Section 1.2, the goal of this study was to develop a technique of maintaining a high fidelity representation of the target state Probability Density Function (PDF) while limiting the number of Gaussian mixture components to retain computational tractability. The procedure defined a physically meaningful cost function to measure the deviation from the true target state PDF, and proceeded by sequentially selecting the simplification steps to minimize the cost of the reduction. The performance of this cost function-based approximation was demonstrated in a realistic single target tracking problem, as presented by Salmond [44]. These simulations indicate that the track life (the standard metric for comparison of such algorithms) achievable utilizing the new approximation raises tracking performance to a previously unattainable level.

5.2 Summary of Results

The results presented in Chapter IV demonstrate the performance of the Gaussian mixture reduction algorithm based on the Integral Square Difference (ISD) cost function. Section 4.2 applied the initialization algorithm to a one-dimensional problem, illustrating the competency of the reduction steps chosen. Section 4.3 then demonstrated the refinement offered using the iterative optimization technique.

5.2.1 Single Target Tracking Performance. The results presented in Section 4.4 reveal the significant improvement in performance offered by the ISD initialization algorithm. It was demonstrated that, while the performance of the algorithm is no better than previous techniques when fewer than 10 components are utilized, when 25 or more components are used, the track life performance is considerably better than that achievable using any of the existing methods compared. Further-

more, the trend of the average track life shown in Figures 4.8 and 4.18 suggests that the increase in performance will continue as the number of components grows: there is no indication that the performance will level out as seen with the other algorithms. Accordingly, the ISD initialization algorithm not only provides a level of performance which was previously unattainable, but the level of performance achievable using the algorithm appears to be limited only by the computational resources available. As computational power increases, the algorithm has the potential to extend the track life possible in a high clutter environment far beyond that provided by any previous algorithm.

5.2.2 Multiple Target Tracking Performance. The application of the ISD initialization algorithm to the multiple target tracking problem revealed the excessive computational complexity of the methodology used in the multiple target extension. As discussed below in Section 5.4, an MHT-like extension of the hypothesis creation algorithm which maintains separate lists of hypotheses for each target, alongside a list of joint hypotheses linking the single-target hypotheses together, would allow the ISD initialization technique to be applied to the multiple target tracking problem, providing a similar performance benefit to the single target case, but with a more modest computational load.

5.2.3 Maneuvering Target Tracking Performance. The results presented in Section 4.6 demonstrate that the ISD cost function is not appropriate for use with the problem of switching dynamics models due to the large variability of the covariance in each model. The results also appeared to indicate that the performance in the scenario is limited more by the Markov transition model than the PDF representation. An extension to a time-varying Markov transition model is suggested in Section 5.4.

5.3 Significant Contributions of Research

The Multiple Hypothesis Tracking (MHT) technique represents the state-of-the-art tracking algorithm in modern civilian and military radar systems. However, the common implementation relies on simplistic *ad hoc* pruning and merging techniques to perform the most vital function of the algorithm: hypothesis control. This thesis directly addresses the problem of hypothesis control, making several important contributions, including those listed in the following pages.

1. The Integral Square Difference (ISD) cost function defined in Section 3.3.1.3 is both *physically meaningful* and *computationally tractable*; this latter attribute was seen to be rare among common cost function selections. By developing a cost function which can be evaluated in closed form, the resultant reduction algorithm is able to consider the impact of a merging or pruning operation on the *entire mixture*, rather than individual components or component pairs, leading to a remarkable improvement in tracking performance.
2. Apart from being able to be evaluated in closed form, the ISD cost function is also continuously differentiable, and its *first derivatives* are also able to be evaluated in closed form using standard vector-matrix notation. This leads to an easy application of iterative optimization methods as described in Section 3.3.3, which have not previously been applied to the Gaussian mixture reduction problem. Although the simulation results presented in Section 4.4.4 indicate that the improvement gained over the initialization algorithm is negligible for the target tracking problem, it remains a valuable concept which may be beneficial in other applications.
3. The tracking performance of the ISD initialization algorithm presented in Section 4.4 demonstrates the benefit of the cost function-based technique. For larger numbers of mixture components, the performance of the ISD initialization algorithm is significantly greater than that of previously published tech-

niques. The trend illustrated in Figure 4.8 indicates that, in the problem under consideration, the performance achieved using the ISD initialization algorithm with 30 mixture components is greater than attainable with existing algorithms using *any* feasible number of components. Furthermore, as the computational power available increases, the algorithm is capable of providing a level of performance that increases *exponentially* with the number of mixture components, whereas the previously proposed algorithms are *unable* to improve performance beyond that achieved using 25 components.

4. The significance of the Maximum Likelihood cost function proposed in Section 3.3.1.4 should not be overlooked. Although this function does not lead to a tractable implementation, its physical interpretation as the “goodness of fit” of the reduced-complexity PDF to the full PDF (as derived in Section 3.3.1.4) distinguishes it as possibly the most physically meaningful cost function of which one could conceive. Approximations to this cost function may be able to yield a significant alternative to the more mathematically tractable ISD technique developed herein.
5. The tutorial on existing data association algorithms presented in Section 2.5 differs significantly from previous presentations (such as those in [2, 4]), and provides a clear understanding of the approximations inherent to the algorithms, and the resultant strengths and weaknesses.
6. The examination of the bias and coalescence problems of the JPDA and CPDA algorithms presented in Section 3.2 reveals new insight into the cause of the difficulties commonly experienced with these techniques. In Eqs. (3.2)–(3.7) it is proven that JPDA is in fact unbiased, which is in direct contradiction to the analyses presented in [16, 19, 22]. The thorough explanation of the poor performance of CPDA as compared with JPDA expands and corrects the previous theory, as published in [12].

7. Finally, the efficient method of evaluating a multivariate Gaussian PDF outlined in Section 3.3.4.1 and the two-stage gating procedure described in Appendix A.2 both provide major computational savings, and are applicable to a wide range of scientific computation applications. To the knowledge of the author, neither of these developments has been previously published.

5.4 *Recommendations for Future Investigations*

While the ISD initialization algorithm proposed in Section 3.3.4 provides a substantial increase in performance over existing methods, the computational complexity of the technique will be of significant concern for any practical implementation. An important area for future investigation is to examine computational enhancements of the algorithm. For example, the current implementation considers the cost for *every possible* action at each step, selecting only a single action. From the beginning of the reduction process, it will commonly be clear that many possible actions are *not worth* considering, and thus the computational load of the algorithm could be reduced considerably by neglecting such options.

As discussed in the previous section, the Maximum Likelihood measure derived in Section 3.3.1.4 is probably the most physically meaningful cost function for this application. Although the ISD cost function was chosen for its tractability, its predisposition toward neglecting higher variance components was clear, and this characteristic was demonstrated to make the function inappropriate for some applications. There is potential for significant improvement in performance through development of techniques based on the Maximum Likelihood measure or approximations thereof.

The results presented in Section 4.4 clearly demonstrate the performance of the ISD initialization algorithm in a single target tracking problem. The method adopted for a multiple target implementation, directly forming and merging joint hypotheses, led to a structure which was computationally untenable, preventing generation of any

meaningful results (discussed in Section 4.5). The extension of the Gaussian mixture reduction algorithm to a multiple target scenario while maintaining links between compatible single target hypotheses (as opposed to the target PDF marginalization inherent to the extension proposed by Pao [38], discussed in Section 2.5.11.2) remains a significant area of research.

The application of the ISD initialization algorithm to the problem of switching target dynamics models demonstrated that the ISD cost function was inappropriate for this application, and that the time invariant Markov transition model was quite potentially the more important limitation on the performance of the system. As mentioned briefly in Section 2.4.2, the use of the Markov model assumes that transition probabilities depend *only* on the previous model index, and *not* on prior model histories *or* prior measurements. These assumptions are applied in the development of Eq. (2.39), in which the model switching probabilities, which naturally depend on the entire model history *and* measurement history, are assumed to depend only on the previous model index. A simple variant of these assumptions would be to *allow* dependence of the Markov transition probabilities on *recent measurements*, hence creating a *time varying* Markov model which adapts itself as observations are received. One idea for such a structure would be to adjust the transition probabilities according to the properties of the residuals for each of the filters in recent processing cycles. If one filter is clearly dominant, then the transition probabilities can be adjusted accordingly to use this filter almost exclusively in the estimator output, and to reinitialize other filters continually using this estimate. If the residual properties of all filters are similar, or if the model with the smallest residual changes, then the transition probabilities corresponding to a change of model could be increased to respond to this uncertainty. Using a structure based on the Sequential Probability Ratio Test (SPRT) or the extensions discussed in [54], the transition probabilities could be increased whenever the most recent residuals indicate that, to within a certain confidence level, the model in force is changing. Such a development could

enhance both the steady state performance of the system, and the speed with which the system responds to the onset of a maneuver.

Appendix A. Derivations

A.1 Product of Two Gaussians of Same Dimension

This section develops a simplification of the result of the product of two multi-variate Gaussian PDFs of the same dimension. The result presented herein is utilized throughout Chapter III.

The variable of both PDFs is denoted $\mathbf{x}$; the first Gaussian has mean $\boldsymbol{\mu}_1$ and covariance $\mathbf{P}_1$, while the second has mean $\boldsymbol{\mu}_2$ and covariance $\mathbf{P}_2$. Writing the product in full:

$$\begin{aligned}
 & \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_1, \mathbf{P}_1\} \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_2, \mathbf{P}_2\} \\
 &= |2\pi\mathbf{P}_1|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \mathbf{P}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) \right\} \cdot \\
 & \quad |2\pi\mathbf{P}_2|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T \mathbf{P}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right\} \\
 &= |2\pi\mathbf{P}_1|^{-\frac{1}{2}} |2\pi\mathbf{P}_2|^{-\frac{1}{2}} \cdot \\
 & \quad \exp \left\{ -\frac{1}{2} \left[(\mathbf{x} - \boldsymbol{\mu}_1)^T \mathbf{P}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + (\mathbf{x} - \boldsymbol{\mu}_2)^T \mathbf{P}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right] \right\}
 \end{aligned} \tag{A.1}$$

Manipulating the exponents:

$$\begin{aligned}
 & (\mathbf{x} - \boldsymbol{\mu}_1)^T \mathbf{P}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + (\mathbf{x} - \boldsymbol{\mu}_2)^T \mathbf{P}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \\
 &= \mathbf{x}^T \mathbf{P}_1^{-1} \mathbf{x} - 2\mathbf{x}^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 \\
 & \quad + \mathbf{x}^T \mathbf{P}_2^{-1} \mathbf{x} - 2\mathbf{x}^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\
 &= \mathbf{x}^T (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1}) \mathbf{x} - 2\mathbf{x}^T (\mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \mathbf{P}_2^{-1} \boldsymbol{\mu}_2) + \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2
 \end{aligned} \tag{A.2}$$

Examining the form of Eq. (A.2), we see that the resulting function will be a Gaussian PDF with a scaled volume. Denoting $\boldsymbol{\mu}_3$ and $\mathbf{P}_3$ as the mean and covariance

of the resultant Gaussian, and α as the volume scaling factor, we seek to fit Eq. (A.1) into the form:

$$\alpha \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_3, \mathbf{P}_3\} = \alpha |2\pi \mathbf{P}_3|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_3)^T \mathbf{P}_3^{-1} (\mathbf{x} - \boldsymbol{\mu}_3) \right\} \quad (\text{A.3})$$

where the exponent expands to:

$$(\mathbf{x} - \boldsymbol{\mu}_3)^T \mathbf{P}_3^{-1} (\mathbf{x} - \boldsymbol{\mu}_3) = \mathbf{x}^T \mathbf{P}_3^{-1} \mathbf{x} - 2\mathbf{x}^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 + \boldsymbol{\mu}_3^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 \quad (\text{A.4})$$

Matching the coefficients of the terms in Eq. (A.2) to those in Eq. (A.4), we find:

$$\begin{aligned} \mathbf{x}^T \mathbf{P}_3^{-1} \mathbf{x} &= \mathbf{x}^T (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1}) \mathbf{x} \quad \forall \mathbf{x} \\ \therefore \mathbf{P}_3^{-1} &= \mathbf{P}_1^{-1} + \mathbf{P}_2^{-1} \\ \therefore \mathbf{P}_3 &= (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1} \\ &= \mathbf{P}_1 - \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1 = \mathbf{P}_2 - \mathbf{P}_2 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_2 \end{aligned} \quad (\text{A.5})$$

where the final equality is due to the matrix inversion lemma [34:213]. Similarly, matching the $\mathbf{x}$ coefficients and using the result of Eq. (A.5):

$$\begin{aligned} 2\mathbf{x}^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 &= 2\mathbf{x}^T (\mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \mathbf{P}_2^{-1} \boldsymbol{\mu}_2) \quad \forall \mathbf{x} \\ \therefore \boldsymbol{\mu}_3 &= \mathbf{P}_3 (\mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \mathbf{P}_2^{-1} \boldsymbol{\mu}_2) \\ &= \mathbf{P}_3 \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\ &= [\mathbf{P}_1 - \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1] \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + [\mathbf{P}_2 - \mathbf{P}_2 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_2] \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\ &= \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 - \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \boldsymbol{\mu}_1 - \mathbf{P}_2 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \boldsymbol{\mu}_2 \end{aligned} \quad (\text{A.6})$$

From Eqs. (A.5) and (A.6) we can expand the final term in Eq. (A.4):

$$\begin{aligned}
\boldsymbol{\mu}_3^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 &= (\boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1}) \mathbf{P}_3 \mathbf{P}_3^{-1} \mathbf{P}_3 (\mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \mathbf{P}_2^{-1} \boldsymbol{\mu}_2) \\
&= \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + 2 \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\
&= \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} [\mathbf{P}_1 - \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1] \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + 2 \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\
&\quad + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} [\mathbf{P}_2 - \mathbf{P}_2 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_2] \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\
&= \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_1^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \boldsymbol{\mu}_2 \\
&\quad + 2 \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \\
&= \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&\quad + 2 \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 - 2 \boldsymbol{\mu}_1^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \boldsymbol{\mu}_2 \\
&= \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&\quad + 2 \boldsymbol{\mu}_1^T [\mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1}] \boldsymbol{\mu}_2 \tag{A.7}
\end{aligned}$$

Manipulating the weighting matrix on the cross-term:

$$\begin{aligned}
&\mathbf{P}_1^{-1} \mathbf{P}_3 \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \\
&= \mathbf{P}_1^{-1} [\mathbf{P}_1 - \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1] \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \\
&= \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1 \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \\
&= (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\mathbf{P}_1 + \mathbf{P}_2) \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \mathbf{P}_1 \mathbf{P}_2^{-1} - (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \\
&= (\mathbf{P}_1 + \mathbf{P}_2)^{-1} [(\mathbf{P}_1 + \mathbf{P}_2) \mathbf{P}_2^{-1} - \mathbf{P}_1 \mathbf{P}_2^{-1} - \mathbf{I}] \\
&= (\mathbf{P}_1 + \mathbf{P}_2)^{-1} [\mathbf{P}_1 \mathbf{P}_2^{-1} + \mathbf{I} - \mathbf{P}_1 \mathbf{P}_2^{-1} - \mathbf{I}] \\
&= (\mathbf{P}_1 + \mathbf{P}_2)^{-1} [\mathbf{0}] \\
&= \mathbf{0} \tag{A.8}
\end{aligned}$$

Hence substituting Eq. (A.8) into Eq. (A.7):

$$\boldsymbol{\mu}_3^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 = \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{A.9}$$

Finally equating the expressions in Eqs. (A.1) and (A.3):

$$\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_1, \mathbf{P}_1\} \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_2, \mathbf{P}_2\} = \alpha \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_3, \mathbf{P}_3\} \quad (\text{A.10})$$

Expanding each side of the expression:

$$\begin{aligned} \text{LHS} &= |2\pi\mathbf{P}_1|^{-\frac{1}{2}} |2\pi\mathbf{P}_2|^{-\frac{1}{2}} \cdot \\ &\quad \exp \left\{ -\frac{1}{2} \left[\mathbf{x}^T \mathbf{P}_3^{-1} \mathbf{x} - 2\mathbf{x}^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 + \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \right] \right\} \\ \text{RHS} &= \alpha |2\pi\mathbf{P}_3|^{-\frac{1}{2}} \cdot \\ &\quad \exp \left\{ -\frac{1}{2} \left[\mathbf{x}^T \mathbf{P}_3^{-1} \mathbf{x} - 2\mathbf{x}^T \mathbf{P}_3^{-1} \boldsymbol{\mu}_3 + \boldsymbol{\mu}_1^T \mathbf{P}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \mathbf{P}_2^{-1} \boldsymbol{\mu}_2 \right. \right. \\ &\quad \left. \left. - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right] \right\} \end{aligned} \quad (\text{A.11})$$

Using the last remaining variable α to satisfy the equality of Eqs. (A.10) and (A.11):

$$\alpha = \sqrt{\frac{|2\pi\mathbf{P}_3|}{|2\pi\mathbf{P}_1||2\pi\mathbf{P}_2|}} \exp \left\{ -\frac{1}{2} \left[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\mathbf{P}_1 + \mathbf{P}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right] \right\} \quad (\text{A.12})$$

where:

$$\begin{aligned} \sqrt{\frac{|2\pi\mathbf{P}_3|}{|2\pi\mathbf{P}_1||2\pi\mathbf{P}_2|}} &= \sqrt{|2\pi\mathbf{P}_1|^{-1} |2\pi\mathbf{P}_3|^{-1} |2\pi\mathbf{P}_2|^{-1}}^{-1} \\ &= \sqrt{|2\pi\mathbf{P}_1 \mathbf{P}_3^{-1} \mathbf{P}_2|}^{-1} \\ &= \sqrt{|2\pi\mathbf{P}_1 (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1}) \mathbf{P}_2|}^{-1} \\ &= \sqrt{|2\pi(\mathbf{P}_1 \mathbf{P}_1^{-1} \mathbf{P}_2 + \mathbf{P}_1 \mathbf{P}_2^{-1} \mathbf{P}_2)|}^{-1} \\ &= \sqrt{|2\pi(\mathbf{P}_1 + \mathbf{P}_2)|}^{-1} \end{aligned}$$

Hence:

$$\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_1, \mathbf{P}_1\} \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_2, \mathbf{P}_2\} = \alpha \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}_3, \mathbf{P}_3\} \quad (\text{A.13})$$

where:

$$\begin{aligned}
\mathbf{P}_3 &= (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1} \\
&= \mathbf{P}_1 - \mathbf{P}_1(\mathbf{P}_1 + \mathbf{P}_2)^{-1}\mathbf{P}_1 \\
&= \mathbf{P}_2 - \mathbf{P}_2(\mathbf{P}_1 + \mathbf{P}_2)^{-1}\mathbf{P}_2 \\
\boldsymbol{\mu}_3 &= \mathbf{P}_3(\mathbf{P}_1^{-1}\boldsymbol{\mu}_1 + \mathbf{P}_2^{-1}\boldsymbol{\mu}_2) \\
&= \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 - \mathbf{P}_1(\mathbf{P}_1 + \mathbf{P}_2)^{-1}\boldsymbol{\mu}_1 - \mathbf{P}_2(\mathbf{P}_1 + \mathbf{P}_2)^{-1}\boldsymbol{\mu}_2 \\
\alpha &= |2\pi(\mathbf{P}_1 + \mathbf{P}_2)|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T(\mathbf{P}_1 + \mathbf{P}_2)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right\} \\
&= \mathcal{N}\{\boldsymbol{\mu}_1; \boldsymbol{\mu}_2, \mathbf{P}_1 + \mathbf{P}_2\}
\end{aligned}$$

Considering the special case where $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu}$ and $\mathbf{P}_1 = \mathbf{P}_2 = \mathbf{P}$:

$$\begin{aligned}
\mathbf{P}_3 &= (\mathbf{P}^{-1} + \mathbf{P}^{-1})^{-1} = (2\mathbf{P}^{-1})^{-1} = \frac{1}{2}\mathbf{P} \\
\boldsymbol{\mu}_3 &= \mathbf{P}_3(\mathbf{P}^{-1}\boldsymbol{\mu} + \mathbf{P}^{-1}\boldsymbol{\mu}) = \frac{1}{2}\mathbf{P}(2\mathbf{P}^{-1}\boldsymbol{\mu}) = \boldsymbol{\mu} \\
\alpha &= |2\pi(\mathbf{P} + \mathbf{P})|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} \left[(\boldsymbol{\mu} - \boldsymbol{\mu})^T (2\mathbf{P})^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}) \right] \right\} \\
&= |4\pi\mathbf{P}|^{-\frac{1}{2}}
\end{aligned} \tag{A.14}$$

Hence:

$$[\mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}\}]^2 = |4\pi\mathbf{P}|^{-\frac{1}{2}} \mathcal{N}\{\mathbf{x}; \boldsymbol{\mu}, \frac{1}{2}\mathbf{P}\} \tag{A.15}$$

A.2 Modified Gating Algorithm

The measurement gating algorithm described in Section 2.5.1 centers on the following calculation:

$$[\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)]^T \mathbf{S}_i(k)^{-1} [\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)] \leq \gamma \tag{A.16}$$

where $\mathbf{z}_j(k)$ is the j -th measurement in the k -th scan, $\hat{\mathbf{z}}_i(k|k-1)$ is the predicted measurement for the i -th hypothesis, and $\mathbf{S}_i(k)$ is the covariance of the residual

for hypothesis i formed with the target-originated measurement. This expression requires the calculation of the difference of two vectors, followed by the multiplication of a matrix by a vector, and finally the inner product of two vectors. For an N -dimensional measurement space, the first operation will require N additions, the second operation will require N^2 multiplications and $N(N - 1)$ additions, and the final operation will require N multiplications and $(N - 1)$ additions. This totals $(N^2 + N)$ multiplications and $(N^2 + N - 1)$ additions.¹ While this may seem a small number, these calculations must be repeated for every pairing of hypothesis and measurement. The matrix inversion is not included in the calculation as this needs to be performed only once for each hypothesis; it does not need to be repeated for each measurement considered. As described in Section 4.4.2, the region populated by clutter measurements can contain on the order of 48,000 measurements for the latter simulations, and the algorithms being tested maintain up to 35 hypotheses between processing intervals, hence these calculations must be performed 1,680,000 times (on average) in each processing cycle.

As illustrated in Figure A.1(a), the gating procedure described by Eq. (A.16) determines whether or not a given measurement is within an ellipse that is centered on the measurement prediction $\hat{\mathbf{z}}_i(k|k - 1)$, and with major and minor axis and orientation that are determined by the residual covariance $\mathbf{S}_i(k)$. The idea of the following development is to form a square which is aligned with the coordinate axes and completely encloses the ellipse such that if a measurement is outside of the square it can be discarded without performing the calculation in Eq. (A.16). To determine whether or not a measurement lies within a square requires only $2N$ logical comparisons, hence avoiding the complex calculations described previously. The calculation of Eq. (A.16) can then be performed for the relatively small number of

¹This may be reduced somewhat by exploiting the symmetry of the covariance matrix, as utilized in Section 3.3.4.

measurements which are found to be within the enclosing square. This is illustrated in Figure A.1(c).

Dividing both sides of Eq. (A.16) by γ , we obtain the following equation:

$$[\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)]^T [\gamma \mathbf{S}_i(k)]^{-1} [\mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k|k-1)] \leq 1 \quad (\text{A.17})$$

Following from [52:335–336], the major and minor axes of this ellipse will be the square roots of the eigenvalues of $\gamma \mathbf{S}_i(k)$, and the orientation of the axes will be in the directions of the corresponding eigenvectors. If a circle is drawn centered on the measurement prediction ($\hat{\mathbf{z}}_i(k|k-1)$) with a radius of the square root maximum eigenvalue² of $\gamma \mathbf{S}_i(k)$ (denoted $\sqrt{\lambda_1}$), then this will be the smallest circle which encloses the gating ellipse. This is illustrated in Figure A.1(b). It is then an easy matter to form the square which encloses the circle, and is aligned to the coordinate axes, as illustrated in Figure A.1(c). The square will be centered on the measurement prediction (as was the circle), and will have a side of $2\sqrt{\lambda_1}$. The gating operation can thus be performed first using this square, avoiding the calculation of Eq. (A.16) for the vast majority of the measurements, providing a major computational saving.

A.3 Switching Bayesian Transition Probability

The switching model estimator approximation discussed in Section 4.6 implements the structure of the full switching Bayesian algorithm shown in Figure 2.3, employing the ISD initialization algorithm to combine hypotheses at the end of each processing cycle. It is quite possible that estimates arising from different models in the latest processing interval will be merged in the hypothesis reduction process. In order to propagate these estimates to the following processing interval, a different form of transition probability will be necessary. For example, if the estimates from

²While calculation of the eigenvalues of a matrix is itself a computationally demanding operation, this will only need to be performed once for each hypothesis, not for each measurement, hence the computational burden associated with it is not of concern.

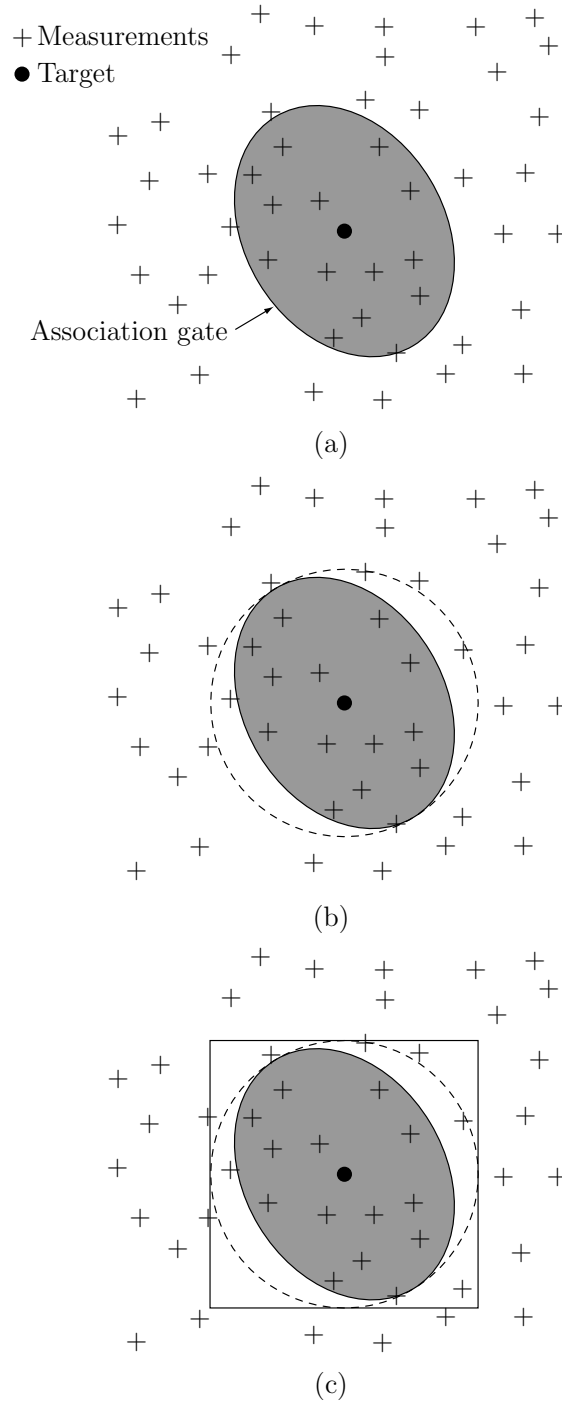


Figure A.1. Measurement gating: the gating equation describes an ellipse as shown in (a); the smallest circle enclosing the ellipse is shown in (b); the square aligned with coordinate axes enclosing the circle is shown in (c).

models 1 and 2 in the $(k - 1)$ -th processing cycle are combined in the hypothesis reduction process, then the transition probability required to weight the model estimates in the k -th processing cycle will be $P\{M_{k,j}|M_{k-1,1} \cup M_{k-1,2}\}$, rather than the standard Markov transition probability $P\{M_{k,j}|M_{k-1,i}\}$. To see the source of this modified form, consider the expression in which the transition probably first arose, Eq. (2.38):

$$\begin{aligned}
P\{M^{k,l}|\mathbf{Z}^k\} &= P\{M^{k,l}|\mathbf{z}(k), \mathbf{Z}^{k-1}\} \\
&= \frac{f\{M^{k,l}, \mathbf{z}(k)|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\
&= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\}P\{M^{k,l}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\
&= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\}P\{M_{k,j}, M^{k-1,l'}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}} \\
&= \frac{f\{\mathbf{z}(k)|M^{k,l}, \mathbf{Z}^{k-1}\}P\{M_{k,j}|M^{k-1,l'}, \mathbf{Z}^{k-1}\}P\{M^{k-1,l'}|\mathbf{Z}^{k-1}\}}{f\{\mathbf{z}(k)|\mathbf{Z}^{k-1}\}}
\end{aligned} \tag{A.18}$$

The modification due to merging of models commences from the second-last line of Eq. (A.18). If hypotheses are merged, then the latter term in the numerator will become $P\{M_{k,j}, M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\}$, which can be expanded as:³

$$\begin{aligned}
&P\{M_{k,j}, M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\} \\
&= P\{M_{k,j}|M^{k-1,l'_1} \cup M^{k-1,l'_2}, \mathbf{Z}^{k-1}\}P\{M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\} \\
&= P\{M_{k,j}|M_{k-1,1}, M^{k-2,l''_1} \cup M_{k-1,2}, M^{k-2,l''_2}, \mathbf{Z}^{k-1}\}P\{M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\} \\
&= P\{M_{k,j}|M_{k-1,1} \cup M_{k-1,2}\}P\{M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\}
\end{aligned} \tag{A.19}$$

³For this example, the two model history events which are to be merged, M^{k-1,l'_1} and M^{k-1,l'_2} , are assumed to contain consist of models 1 and 2 respectively in the most recent entry (events $M_{k-1,1}$ and $M_{k-1,2}$), alongside the previous model history events M^{k-2,l''_1} and M^{k-2,l''_2} .

where the final step is due to the Markov assumption, as previously applied in Eq. (2.39), and $P\{M^{k-1,l'_1} \cup M^{k-1,l'_2} | \mathbf{Z}^{k-1}\}$ is the combined probability weight of the merged models:

$$P\{M^{k-1,l'_1} \cup M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} = P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \quad (\text{A.20})$$

This latter step is possible because all model history events are disjoint.

To evaluate the modified transition probability, consider the alternative expansion of Eq. (A.19):

$$\begin{aligned} & P\{M_{k,j}, M^{k-1,l'_1} \cup M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j}, M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + P\{M_{k,j}, M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j} | M^{k-1,l'_1}, \mathbf{Z}^{k-1}\} P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + \\ &\quad + P\{M_{k,j} | M^{k-1,l'_2}, \mathbf{Z}^{k-1}\} P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j} | M_{k-1,1}, M^{k-2,l''_1}, \mathbf{Z}^{k-1}\} P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + \\ &\quad + P\{M_{k,j} | M_{k-1,2}, M^{k-2,l''_2}, \mathbf{Z}^{k-1}\} P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j} | M_{k-1,1}\} P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + P\{M_{k,j} | M_{k-1,2}\} P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \end{aligned} \quad (\text{A.21})$$

where $P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\}$ and $P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\}$ are the probabilities of the two hypotheses to be merged *before* merging. Equating the expressions of Eqs. (A.19) and (A.21), we obtain:

$$\begin{aligned} & P\{M_{k,j}, M^{k-1,l'_1} \cup M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j} | M_{k-1,1} \cup M_{k-1,2}\} P\{M^{k-1,l'_1} \cup M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \\ &= P\{M_{k,j} | M_{k-1,1}\} P\{M^{k-1,l'_1} | \mathbf{Z}^{k-1}\} + P\{M_{k,j} | M_{k-1,2}\} P\{M^{k-1,l'_2} | \mathbf{Z}^{k-1}\} \end{aligned} \quad (\text{A.22})$$

thus:

$$\begin{aligned}
P\{M_{k,j}|M_{k-1,1} \cup M_{k-1,2}\} &= \frac{P\{M_{k,j}|M_{k-1,1}\}P\{M^{k-1,l'_1}|\mathbf{Z}^{k-1}\}}{P\{M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\}} + \\
&+ \frac{P\{M_{k,j}|M_{k-1,2}\}P\{M^{k-1,l'_2}|\mathbf{Z}^{k-1}\}}{P\{M^{k-1,l'_1} \cup M^{k-1,l'_2}|\mathbf{Z}^{k-1}\}}
\end{aligned} \tag{A.23}$$

Considering the definition of the denominator of Eq. (A.23) in Eq. (A.20), the result in Eq. (A.23) can be seen to be a weighted sum of the transition probabilities from the models corresponding to the merged hypotheses to the new model under consideration. The weights for this sum are simply the probabilities of the original hypotheses that were merged together.

Appendix B. Matrix Reference Manual

The following pages contain a reproduction of the world-wide web page entitled “Matrix Reference Manual: Matrix Calculus”, maintained by Mr Mike Brooks of Imperial College, University of London. The Uniform Resource Locator (URL) for the page is <http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/calculus.html>. Many thanks go to Mr Mike Brookes for giving permission for the document to be reproduced in this thesis.

Matrix Reference Manual

Matrix Calculus

Go to: [Introduction](#), [Notation](#), [Index](#)

Contents of Calculus Section

- [Notation](#)
- Derivatives of [Linear](#), [Quadratic](#) and [Cubic](#) Products
- Derivatives of [Inverses](#), [Trace](#) and [Determinant](#)
- [Jacobians](#) and [Hessian](#) matrices

Notation

- d/dx ($\mathbf{y}$) is a vector whose (i) element is $dy(i)/dx$
- $d/d\mathbf{x}$ ($\mathbf{y}$) is a vector whose (i) element is $dy/dx(i)$
- $d/d\mathbf{x}$ ($\mathbf{y}^T$) is a matrix whose (i,j) element is $dy(j)/dx(i)$
- d/dx ($\mathbf{Y}$) is a matrix whose (i,j) element is $dy(i,j)/dx$
- $d/d\mathbf{X}$ ($\mathbf{y}$) is a matrix whose (i,j) element is $dy/dx(i,j)$
- x_R and x_I are the real and imaginary parts of x
- x^* is the complex conjugate of x
- j is the square root of -1

An expression, y , can only be differentiated with respect to a complex x if it satisfies the Cauchy-Riemann equations: $dy/dx_R = j dy/dx_I$. Expressions involving the complex conjugate or Hermitian transpose do not normally satisfy this requirement, so separate expressions for dy/dx_R and dy/dx_I are given in these cases.

In the expressions below matrices and vectors $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ do not depend on $\mathbf{X}$.

Derivatives of Linear Products

- $d/dx (\mathbf{A}\mathbf{Y}\mathbf{B}) = \mathbf{A} * d/dx (\mathbf{Y}) * \mathbf{B}$
 - $d/dx (\mathbf{A}\mathbf{y}) = \mathbf{A} * d/dx (\mathbf{y})$
- $d/d\mathbf{x} (\mathbf{x}^T \mathbf{A}) = \mathbf{A}$
 - $d/d\mathbf{x} (\mathbf{x}^T) = \mathbf{I}$
 - $d/d\mathbf{x} (\mathbf{x}^T \mathbf{a}) = d/d\mathbf{x} (\mathbf{a}^T \mathbf{x}) = \mathbf{a}$
- $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X} \mathbf{b}) = \mathbf{a} \mathbf{b}^T$
 - $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X} \mathbf{a}) = d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{a}) = \mathbf{a} \mathbf{a}^T$

- $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{b}) = \mathbf{b} \mathbf{a}^T$
- $d/d\mathbf{x} (\mathbf{Y} \mathbf{Z}) = \mathbf{Y} * d/d\mathbf{x} (\mathbf{Z}) + d/d\mathbf{x} (\mathbf{Y}) * \mathbf{Z}$
- $dy/d\mathbf{x}_R (\mathbf{Y}^H) = (dy/d\mathbf{x}_R (\mathbf{Y}))^H$
- $dy/d\mathbf{x}_I (\mathbf{Y}^H) = (dy/d\mathbf{x}_I (\mathbf{Y}))^H$
- $dy/d\mathbf{x}_R (\mathbf{x}^H \mathbf{A}) = \mathbf{A}$
 - $dy/d\mathbf{x}_R (\mathbf{x}^H) = \mathbf{I}$
- $dy/d\mathbf{x}_I (\mathbf{x}^H \mathbf{A}) = -j\mathbf{A}$
 - $dy/d\mathbf{x}_I (\mathbf{x}^H) = -j\mathbf{I}$

Derivatives of Quadratic Products

- $d/d\mathbf{x} (\mathbf{Ax}+\mathbf{b})^T \mathbf{C} (\mathbf{Dx}+\mathbf{e}) = \mathbf{A}^T \mathbf{C} (\mathbf{Dx}+\mathbf{e}) + \mathbf{D}^T \mathbf{C}^T (\mathbf{Ax}+\mathbf{b})$
 - $d/d\mathbf{x} (\mathbf{x}^T \mathbf{Cx}) = (\mathbf{C}+\mathbf{C}^T)\mathbf{x}$
 - $[\mathbf{C}=\mathbf{C}^T]: d/d\mathbf{x} (\mathbf{x}^T \mathbf{Cx}) = 2\mathbf{Cx}$
 - $d/d\mathbf{x} (\mathbf{x}^T \mathbf{x}) = 2\mathbf{x}$
 - $d/d\mathbf{x} (\mathbf{Ax}+\mathbf{b})^T (\mathbf{Dx}+\mathbf{e}) = \mathbf{A}^T (\mathbf{Dx}+\mathbf{e}) + \mathbf{D}^T (\mathbf{Ax}+\mathbf{b})$
 - $d/d\mathbf{x} (\mathbf{Ax}+\mathbf{b})^T (\mathbf{Ax}+\mathbf{b}) = 2\mathbf{A}^T (\mathbf{Ax}+\mathbf{b})$
 - $[\mathbf{C}=\mathbf{C}^T]: d/d\mathbf{x} (\mathbf{Ax}+\mathbf{b})^T \mathbf{C} (\mathbf{Ax}+\mathbf{b}) = 2\mathbf{A}^T \mathbf{C} (\mathbf{Ax}+\mathbf{b})$
- $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{X} \mathbf{b}) = \mathbf{X} (\mathbf{a} \mathbf{b}^T + \mathbf{b} \mathbf{a}^T)$
 - $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{X} \mathbf{a}) = 2\mathbf{X} \mathbf{a} \mathbf{a}^T$
- $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{C} \mathbf{X} \mathbf{b}) = \mathbf{C}^T \mathbf{X} \mathbf{a} \mathbf{b}^T + \mathbf{C} \mathbf{X} \mathbf{b} \mathbf{a}^T$
 - $d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{C} \mathbf{X} \mathbf{a}) = (\mathbf{C} + \mathbf{C}^T) \mathbf{X} \mathbf{a} \mathbf{a}^T$
 - $[\mathbf{C}=\mathbf{C}^T]: d/d\mathbf{X} (\mathbf{a}^T \mathbf{X}^T \mathbf{C} \mathbf{X} \mathbf{a}) = 2\mathbf{C} \mathbf{X} \mathbf{a} \mathbf{a}^T$
- $d/d\mathbf{X} ((\mathbf{Xa}+\mathbf{b})^T \mathbf{C} (\mathbf{Xa}+\mathbf{b})) = (\mathbf{C}+\mathbf{C}^T)(\mathbf{Xa}+\mathbf{b})\mathbf{a}^T$
- $d/d\mathbf{x}_R (\mathbf{Ax}+\mathbf{b})^H \mathbf{C} (\mathbf{Dx}+\mathbf{e}) = \mathbf{A}^H \mathbf{C} (\mathbf{Dx}+\mathbf{e}) + \mathbf{D}^T \mathbf{C}^T (\mathbf{Ax}+\mathbf{b})^*$
 - $d/d\mathbf{x}_R (\mathbf{x}^H \mathbf{Cx}) = \mathbf{Cx} + \mathbf{C}^T \mathbf{x}^* = \mathbf{Cx} + (\mathbf{x}^H \mathbf{C})^T$
 - $[\mathbf{C}=\mathbf{C}^T]: d/d\mathbf{x}_R (\mathbf{x}^H \mathbf{Cx}) = 2\mathbf{Cx}_R$
 - $[\mathbf{C}=\mathbf{C}^H]: d/d\mathbf{x}_R (\mathbf{x}^H \mathbf{Cx}) = 2(\mathbf{Cx})_R$
 - $d/d\mathbf{x}_R (\mathbf{x}^H \mathbf{x}) = 2\mathbf{x}_R$
- $d/d\mathbf{x}_I (\mathbf{Ax}+\mathbf{b})^H \mathbf{C} (\mathbf{Dx}+\mathbf{e}) = j(\mathbf{D}^T \mathbf{C}^T (\mathbf{Ax}+\mathbf{b})^* - \mathbf{A}^H \mathbf{C} (\mathbf{Dx}+\mathbf{e}))$
 - $d/d\mathbf{x}_I (\mathbf{x}^H \mathbf{Cx}) = j(\mathbf{C}^T \mathbf{x}^* - \mathbf{Cx}) = j((\mathbf{x}^H \mathbf{C})^T - \mathbf{Cx})$
 - $[\mathbf{C}=\mathbf{C}^T]: d/d\mathbf{x}_I (\mathbf{x}^H \mathbf{Cx}) = 2\mathbf{Cx}_I$
 - $[\mathbf{C}=\mathbf{C}^H]: d/d\mathbf{x}_I (\mathbf{x}^H \mathbf{Cx}) = 2(\mathbf{Cx})_I$
 - $d/d\mathbf{x}_I (\mathbf{x}^H \mathbf{x}) = 2\mathbf{x}_I$

Derivatives of Cubic Products

- $d/d\mathbf{x} (\mathbf{x}^T \mathbf{A} \mathbf{x} \mathbf{x}^T) = (\mathbf{A} + \mathbf{A}^T) \mathbf{x} \mathbf{x}^T + \mathbf{x}^T \mathbf{A} \mathbf{x} \mathbf{I}$

Derivatives of Inverses

- $d/dx (Y^{-1}) = -Y^{-1} d/dx (Y) Y^{-1}$ [2.1]
- $d/dX (a^T X^{-1} b) = -X^{-T} a b^T X^{-T}$ [2.6]

Derivative of Trace

Note: matrix dimensions must result in an $n*n$ argument for $\text{tr}()$.

- $d/dX (\text{tr}(X)) = d/dX (\text{tr}(X^T)) = I$ [2.4]
- $d/dX (\text{tr}(X^k)) = k(X^{k-1})^T$
- $d/dX (\text{tr}(AX^k)) = \text{SUM}_{r=0:k-1} (X^r A X^{k-r-1})^T$
- $d/dX (\text{tr}(AX^{-1}B)) = -(X^{-1}BAX^{-1})^T = -(X^{-T}ABX^{-T})$ [2.5]
 - $d/dX (\text{tr}(AX^{-1})) = d/dX (\text{tr}(X^{-1}A)) = -X^{-T}A^T X^{-T}$
- $d/dX (\text{tr}(A^T X B^T)) = d/dX (\text{tr}(B X^T A)) = AB$ [2.4]
 - $d/dX (\text{tr}(X A^T)) = d/dX (\text{tr}(A^T X)) = d/dX (\text{tr}(X^T A)) = d/dX (\text{tr}(A X^T)) = A$
- $d/dX (\text{tr}(A X B X^T C)) = A^T C^T X B^T + C A X B$
 - $d/dX (\text{tr}(X A X^T)) = d/dX (\text{tr}(A X^T X)) = d/dX (\text{tr}(X^T X A)) = X(A + A^T)$
 - $d/dX (\text{tr}(X^T A X)) = d/dX (\text{tr}(A X X^T)) = d/dX (\text{tr}(X X^T A)) = (A + A^T)X$
- $d/dX (\text{tr}(A X B X)) = A^T X^T B^T + B^T X^T A^T$
-
- [C:symmetric] $d/dX (\text{tr}((X^T C X)^{-1} A)) = d/dX (\text{tr}(A (X^T C X)^{-1})) = -(C X (X^T C X)^{-1})(A + A^T)(X^T C X)^{-1}$
- [B,C:symmetric] $d/dX (\text{tr}((X^T C X)^{-1} (X^T B X))) = d/dX (\text{tr}((X^T B X)(X^T C X)^{-1})) = -2(C X (X^T C X)^{-1}) X^T B X (X^T C X)^{-1} + 2B X (X^T C X)^{-1}$
-

Derivative of Determinant

Note: matrix dimensions must result in an $n*n$ argument for $\text{det}()$. Some of the expressions below involve inverses: these forms apply only if the quantity being inverted is square and non-singular.

- $d/dX (\text{det}(X)) = d/dX (\text{det}(X^T)) = \underline{\text{ADJ}}(A)^T = \text{det}(X) * X^{-T}$
 - $d/dX (\text{det}(A X B)) = A^T \underline{\text{ADJ}}(A X B) B^T = \text{det}(A X B) * A^T (A X B)^{-T} B^T = \text{det}(A X B) * X^{-T}$
 - $d/dX (\ln(\text{det}(A X B))) = A^T (A X B)^{-T} B^T = X^{-T}$
- $d/dX (\text{det}(X^k)) = k * \text{det}(X^k) * X^{-T}$
 - $d/dX (\ln(\text{det}(X^k))) = k X^{-T}$
- [Real] $d/dX (\text{det}(X^T C X)) = \text{det}(X^T C X) * (C + C^T) X (X^T C X)^{-1}$
 - [C: Real, Symmetric] $d/dX (\text{det}(X^T C X)) = 2 \text{det}(X^T C X) * C X (X^T C X)^{-1}$
- [C: Real, Symmetric] $d/dX (\ln(\text{det}(X^T C X))) = 2 C X (X^T C X)^{-1}$

Jacobian

If y is a function of x , then dy^T/dx is the Jacobian matrix of y with respect to x .

Its determinant, $|d\mathbf{y}^T/d\mathbf{x}|$, is the *Jacobian* of $\mathbf{y}$ with respect to $\mathbf{x}$ and represents the ratio of the hyper-volumes $d\mathbf{y}$ and $d\mathbf{x}$. The Jacobian occurs when changing variables in an integration: $\text{Integral}(f(\mathbf{y})d\mathbf{y}) = \text{Integral}(f(\mathbf{y}(\mathbf{x})) |d\mathbf{y}^T/d\mathbf{x}| d\mathbf{x})$.

Hessian matrix

If f is a function of $\mathbf{x}$ then the symmetric matrix $d^2f/d\mathbf{x}^2 = d/d\mathbf{x}^T(df/d\mathbf{x})$ is the *Hessian* matrix of $f(\mathbf{x})$. A value of $\mathbf{x}$ for which $df/d\mathbf{x} = \mathbf{0}$ corresponds to a minimum, maximum or saddle point according to whether the Hessian is positive definite, negative definite or indefinite.

- $d^2/d\mathbf{x}^2 (\mathbf{a}^T \mathbf{x}) = 0$
- $d^2/d\mathbf{x}^2 (\mathbf{Ax} + \mathbf{b})^T \mathbf{C} (\mathbf{Dx} + \mathbf{e}) = \mathbf{A}^T \mathbf{CD} + \mathbf{D}^T \mathbf{C}^T \mathbf{A}$
 - $d^2/d\mathbf{x}^2 (\mathbf{x}^T \mathbf{Cx}) = \mathbf{C} + \mathbf{C}^T$
 - $d^2/d\mathbf{x}^2 (\mathbf{x}^T \mathbf{x}) = 2\mathbf{I}$
 - $d^2/d\mathbf{x}^2 (\mathbf{Ax} + \mathbf{b})^T (\mathbf{Dx} + \mathbf{e}) = \mathbf{A}^T \mathbf{D} + \mathbf{D}^T \mathbf{A}$
 - $d^2/d\mathbf{x}^2 (\mathbf{Ax} + \mathbf{b})^T (\mathbf{Ax} + \mathbf{b}) = 2\mathbf{A}^T \mathbf{A}$
 - [$\mathbf{C}$: symmetric]: $d^2/d\mathbf{x}^2 (\mathbf{Ax} + \mathbf{b})^T \mathbf{C} (\mathbf{Ax} + \mathbf{b}) = 2\mathbf{A}^T \mathbf{CA}$

The Matrix Reference Manual is written by [Mike Brookes](#), Imperial College, London, UK. Please send any comments or suggestions to mike.brookes@ic.ac.uk

Appendix C. Source Code

The following listing contains the source code for the MEX implementation of the ISD initialization algorithm, developed in Section 3.3.4. The source code was compiled using `lcc-win32`, which is included with the student version of MATLAB® Release 12.

The function requires four input arguments. The first is a vector of length N_h , which contains the probability weights of the N_h hypotheses. The second is an $N \times N_h$ matrix, the columns of which contain the mean vectors for each of the hypotheses. The third argument is a three-dimensional matrix of dimension $N \times N \times N_h$, which contains the covariance matrices for each of the N_h hypotheses. The final input is a scalar, which specifies the number of hypotheses to which the N_h should be reduced.

There are three output arguments returned by the function, containing the probability weights, mean vectors and covariance matrices of the reduced set of hypotheses in the same format as the input. The probability weights are returned in de-normalized form, such that they will not necessarily sum to unity; normalization should be applied as a later step.

C.1 ISDInit.c

```
/* ISDInit.c
Integral Square Difference Initialization Algorithm

This MEX function performs the cost function-based mixture reduction
described in Section 3.3.4 of the thesis. The implementation is
highly optimized to avoid re-calculation of portions of the cost
function which do not change when the reduction steps are taken, and
it utilizes the efficient multivariate Gaussian evaluation described
in the thesis.

The Matlab function takes four inputs. The first is the vector which
contains the probability weights for the numMix hypotheses. The
second is a numVar x numMix matrix whose columns contain the mean
vectors for the hypotheses. The third is a three-dimensional matrix
of dimensions numVar x numVar x numMix, which contains the
covariance matrices for the hypotheses. The final input is a scalar
number (numNewMix) which indicates the number of mixture components
to which the input mixture is to be simplified.

The function provides three outputs, which contain the probability
weights, mean vectors and covariance matrices for the reduced set of
hypotheses.

(c) 07Jan03 Flight Lieutenant Jason L. Williams, RAAF
AFIT GE-03M */

#include "mex.h"
#include "matrix.h"
#include <math.h>
#include <float.h>

/* If debug is set to '1', debugging information will be written to
Matlab stdout device during execution. */
#define DEBUG 0

/* mergePossNum converts two component numbers to the one-dimensional
index corresponding to that merge possibility */
#define mergePossNum(m1,m2) ((m2)-1 + ((m1)*(2*numMix - ((m1)+3))>>1))

/* Assert function definition which works when not compiled in debug
mode -- if the specified condition is not true then the function
is terminated and the specified error message is written to the
screen */
#define jlwAssert(cond,message) {if (!(cond)) {mexErrMsgTxt(message);}}

/* Function prototypes */
void calcCurCost(void);
void calcCostOptions(void);
void copyMixtureParameters(void);
void deleteMixture(int mix);
void mergeMixtures(int mix1, int mix2);
void calcOrigCosts(void);
void calcOrigMergePoss(void);
void calcMergeParam(int m1, int m2);
double calcDist(double p1, double *mean1, double *cov1,
double p2, double *mean2, double *cov2);
```

```

/* Global Variables
Implementation makes extensive use of global variables to speed
execution overhead associated with passing large data structures.

Definitions of variables are as follows:

Inv2PI: the constant (1/(2*pi))
numMix: the number of mixture components in the original mixture
numNewMix: the number of mixture components to which the mixture
is to be simplified
numVar: the number of variables -- i.e. the dimensionality of
the space in which the multivariate Gaussian mixture
components reside
numMergePoss: the number of possible merge actions which can be
taken to simplify the original mixture -- i.e. the number of
unique pairs of two components selected from the original mixture
numCurMix: the counter which tracks the number of mixture components
as it is reduced from numMix to numNewMix
mixMask: an array of flags indicating which components are still in
the reduced mixture. When components are deleted, this flag is set
to zero to indicate that the respective component should no longer
be counted in the mixture.
probs: the hypothesis probabilities of the original mixture
means: the mean vectors of the original mixture
covs: the covariance matrices of the original mixture
newProbs: the hypothesis probabilities of the reduced mixture
newMeans: the mean vectors of the reduced mixture
newCovs: the covariance matrices of the reduced mixture
muD: a temporary variable used to store the difference between two
mean vectors
P: a temporary variable used to store the sum of two covariance
matrices
Di: the inverse of the diagonal portion of the U-D factored
covariance matrix
mergeP: temporary variable used to store the probability of the
component resulting from the merging of two hypotheses
mergeMu: temporary variable used to store the mean vector of the
component resulting from the merging of two hypotheses
mergeP: temporary variable used to store the covariance matrix of
the component resulting from merging two hypotheses
self: the matrix whose (i,j) component represents the similarity
between components i and j of the reduced mixture
cross: the matrix whose (i,j) component represents the similarity
between component i of the original mixture and component j of the
reduced mixture
sumSelf: each entry of the sumSelf array contains the sum of the
entries of the reduced mixture self-likeness matrix (self) which
are due to the respective mixture -- i.e., the sum of the row and
column highlighted in the right-hand diagram of Figure 3.7 in the
thesis
sumCross: the sum of each of the columns of the cross-likeness
matrix
newSelf: matrix containing the new column/row for the self matrix
for each merge possibility
newCross: matrix containing the new column for the cross matrix for
each merge possibility
newSumSelf: array containing the sum of each of the newSelf columns.
Entries of this array are actually the sum of the new row/column
which would replace the previous row/column, as per the
description of sumSelf above.
newSumCross: array containing the sum of each of the newCross cols
actMix1, actMix2: variables used to store the best action found so
far. If the best action is a deletion, then actMix1 contains the
component number to be deleted and actMix2 is 0; otherwise actMix1
and actMix2 are the numbers of the components to be merged.
actMergePoss: contains the merge possibility index corresponding
to merging actMix1 and actMix2

```

```

    curCost: contains the current cost -- i.e., the cost of the
        reduction steps performed already
    sumDist: contains the sum of the original mixture self likeness
        matrix. The contents of this matrix do not change, hence this term
        can be used throughout the reduction to calculate the actual cost. */

const double Inv2PI = 1.591549430918954e-001;
int numMix, numNewMix, numVar, numMergePoss, numCurMix;
char *mixMask;
double *probs, *means, *covs, *newProbs, *newMeans, *newCovs;
double *muD, *P, *Di, mergep, *mergeMu, *mergeP;
double *self, *cross, *sumSelf, *sumCross;
double *newSelf, *newCross, *newSumSelf, *newSumCross;
int actMix1, actMix2, actMergePoss;
double curCost, sumDist;

/* mexFunction
   This is the root function which is called by Matlab
   nlhs contains the number of output arguments and plhs is the pointer
   to the output argument array; nrhs contains the number of input
   arguments and prhs is the pointer to the input argument array */
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
                  const mxArray *prhs[])
{
    const mxArray *mxProbs, *mxMeans, *mxCovs, *mxNumNewMix; /* inputs */
    mxArray *mxNewProbs, *mxNewMeans, *mxNewCovs; /* outputs */
    double doubNumNewMix, *outProbs, *outMeans, *outCovs;
    int newDims[3], numDims;
    const int *dims;

    /* Get inputs and verify input types */
    jlwAssert(nrhs == 4, "Four inputs required");
    mxProbs = prhs[0]; mxMeans = prhs[1]; mxCovs = prhs[2];
    mxNumNewMix = prhs[3];
    jlwAssert(mxGetClassID(mxProbs) == mxDOUBLE_CLASS &&
        !mxIsComplex(mxProbs), "Inputs must be real doubles");
    jlwAssert(mxGetClassID(mxMeans) == mxDOUBLE_CLASS &&
        !mxIsComplex(mxMeans), "Inputs must be real doubles");
    jlwAssert(mxGetClassID(mxCovs) == mxDOUBLE_CLASS &&
        !mxIsComplex(mxCovs), "Inputs must be real doubles");
    jlwAssert(mxGetClassID(mxNumNewMix) == mxDOUBLE_CLASS &&
        !mxIsComplex(mxNumNewMix), "Inputs must be real doubles");

    /* Check that dimensionality of inputs is consistent */
    numDims = mxGetNumberOfDimensions(mxProbs);
    if (numDims == 1) {
        dims = mxGetDimensions(mxProbs);
        numMix = dims[0];
    } else if (numDims == 2) {
        dims = mxGetDimensions(mxProbs);
        if (dims[0] == 1)
            numMix = dims[1];
        else
            numMix = dims[0];
    } else {
        mexErrMsgTxt("Invalid probability array.");
    }
}

```

```

/* Check dimensionality of means */
jlwAssert(mxGetNumberOfDimensions(mxMeans) == 2,
    "means should be 2-dimensional");
dims = mxGetDimensions(mxMeans);
numVar = dims[0];
jlwAssert(dims[1] == numMix,
    "Size of means inconsistent with size of probs");

/* Check dimensionality of covariances */
jlwAssert(mxGetNumberOfDimensions(mxCovs) == 3,
    "covs should be 3-dimensional");
dims = mxGetDimensions(mxCovs);
jlwAssert(dims[0] == numVar,
    "Size of covs inconsistent with size of probs and means");
jlwAssert(dims[1] == numVar,
    "Size of covs inconsistent with size of probs and means");
jlwAssert(dims[2] == numMix,
    "Size of covs inconsistent with size of probs and means");

/* Check for valid number of mixture components */
jlwAssert(mxGetNumberOfElements(mxNumNewMix) == 1,
    "Number of components should be scalar");
doubNumNewMix = mxGetScalar(mxNumNewMix);
numNewMix = (int) doubNumNewMix;
jlwAssert(((double) numNewMix) == doubNumNewMix,
    "Number of components should be integer");
jlwAssert(numNewMix > 0,
    "Number of components should be positive");
jlwAssert(numNewMix < numMix,
    "Number of output components should be less than input number");

/* Get pointers to the real data arrays of the inputs */
probs = mxGetPr(mxProbs);
means = mxGetPr(mxMeans);
covs = mxGetPr(mxCovs);

/* Create output data structures */
jlwAssert(nlhs == 3, "Three outputs required.");
plhs[0] = mxNewProbs = mxCreateDoubleMatrix(1, numNewMix, mxREAL);
plhs[1] = mxNewMeans = mxCreateDoubleMatrix(numVar, numNewMix, mxREAL);
newDims[0] = numVar; newDims[1] = numVar; newDims[2] = numNewMix;
plhs[2] = mxNewCovs = mxCreateNumericArray(3, newDims, mxDOUBLE_CLASS,
    mxREAL);
jlwAssert(mxNewProbs != NULL && mxNewMeans != NULL &&
    mxNewCovs != NULL, "Memory allocation failure");
outProbs = mxGetPr(mxNewProbs);
outMeans = mxGetPr(mxNewMeans);
outCovs = mxGetPr(mxNewCovs);

/* Allocate memory for temporary variables */
muD = (double *) mxMalloc(sizeof(double)*numVar);
P = (double *) mxMalloc(sizeof(double)*numVar*numVar);
Di = (double *) mxMalloc(sizeof(double)*numVar);
jlwAssert(muD != NULL && P != NULL && Di != NULL,
    "Memory allocation failure");

/* Allocate memory for temporary variables for merging components */
mergeMu = (double *) mxMalloc(sizeof(double)*numVar);
mergeP = (double *) mxMalloc(sizeof(double)*numVar*numVar);
jlwAssert(mergeMu != NULL && mergeP != NULL,
    "Memory allocation failure");

```

```

/* Allocate memory for new mixture parameters */
newProbs = (double *) mxMalloc(sizeof(double)*numMix);
newMeans = (double *) mxMalloc(sizeof(double)*numMix*numVar);
newCovs = (double *) mxMalloc(sizeof(double)*numMix*numVar*numVar);
mixMask = (char *) mxMalloc(sizeof(char)*numMix);
jlwAssert(newProbs != NULL && newMeans != NULL &&
    newCovs != NULL && mixMask != NULL, "Memory allocation failure");

/* Allocate memory for distance matrices */
self = (double *) mxMalloc(sizeof(double)*numMix*numMix);
cross = (double *) mxMalloc(sizeof(double)*numMix*numMix);
sumSelf = (double *) mxMalloc(sizeof(double)*numMix);
sumCross = (double *) mxMalloc(sizeof(double)*numMix);
jlwAssert(self != NULL && cross != NULL &&
    sumSelf != NULL && sumCross != NULL, "Memory allocation failure");

/* Allocate memory for merge possibilities */
numMergePoss = (numMix*(numMix-1)) >> 1;
newSelf = (double *) mxMalloc(sizeof(double)*numMergePoss*numMix);
newCross = (double *) mxMalloc(sizeof(double)*numMergePoss*numMix);
newSumSelf = (double *) mxMalloc(sizeof(double)*numMergePoss);
newSumCross = (double *) mxMalloc(sizeof(double)*numMergePoss);
jlwAssert(newSelf != NULL && newCross != NULL &&
    newSumSelf != NULL && newSumCross != NULL,
    "Memory allocation failure");

/* Set up structures */
copyMixtureParameters();
calcOrigCosts();
calcOrigMergePoss();

/* Reduce mixtures -- this is the main loop for the reduction */
for (numCurMix = numMix; numCurMix > numNewMix; numCurMix--) {
    /* calculate the current cost -- the cost of the reduction steps
       already taken */
    calcCurCost();

    /* calculate the cost of each of the merge and deletion options */
    calcCostOptions();

    /* take the lowest cost option */
    if (actMix2 == 0) {
        /* Lowest cost option was to delete a component actMix1 */
        deleteMixture(actMix1);
    } else {
        /* Lowest cost option was to merge components actMix1 and actMix2
           Hence we remove actMix2 from the mixture and replace actMix1
           with the parameters for the merged components */
        deleteMixture(actMix2);
        mergeMixtures(actMix1, actMix2);
    }
}

```

```

/* Store results in Matlab output structure */
{
    int mi, mo, i, j, k;
    mo = 0;

    for (mi = 0; mi < numMix; mi++) {
        if (mixMask[mi]) {

            for (i = 0; i < numVar; i++) {
                outMeans[mo*numVar + i] = newMeans[mi*numVar + i];

                for (j = i; j < numVar; j++) {
                    outCovs[mo*numVar*numVar + i + j*numVar] =
                        outCovs[mo*numVar*numVar + j + i*numVar] =
                            newCovs[mi*numVar*numVar + i + j*numVar];
                }
            }

            outProbs[mo] = newProbs[mi];
            mo++;
        }
    }
}

/* Deallocate memory */
mxFree(newSumCross);mxFree(newSumSelf);mxFree(newCross);
mxFree(newSelf);mxFree(sumCross);mxFree(sumSelf);mxFree(cross);
mxFree(self);mxFree(mixMask);mxFree(newCovs);mxFree(newMeans);
mxFree(newProbs);mxFree(mergeP);mxFree(mergeMu);mxFree(Di);mxFree(P);
mxFree(muD);
}

/* calcCurCost -- Calculates the current cost -- i.e. the cost of the
reduction steps already chosen.
Precondition: sumDist, mixMask, sumSelf and sumCross structures
populated and up to date
Postcondition: curCost will contain the cost of the current reduced
PDF representation. */
void calcCurCost(void)
{
    register int i;

    /* Commence with the cost due to the original mixture
self-likeness */
    curCost = sumDist;

    /* Add the cost components due to each mixture component in the
cross-likeness and reduced self-likeness matrices */
    for (i = 0; i < numMix; i++) {
        if (mixMask[i]) {
            curCost += 0.5*(sumSelf[i] + self[i*(numMix+1)]) - 2*sumCross[i];
        }
    }
}

```

```

/* calcCostOptions -- Calculates the cost of all options for deleting
or merging mixture components
Precondition: mixMask, curCost, sumCross, sumSelf, self,
newSumCross, numSumSelf populated and up to date
Postcondition: actMix1, actMix2 and actMergePoss contain values
indicating the lowest cost action. If actMix2 is zero
then the lowest cost action was to delete component
actMix1. Otherwise, the lowest cost action was to
merge actMix1 and actMix2, which corresponds to merge
possibility number actMergePoss. */
void calcCostOptions(void)
{
    register int i, j, mergePoss;
    register double minCost = DBL_MAX, costOpt;

    for (i = 0; i < numMix; i++) {
        if (mixMask[i]) {

            /* Calculate cost for deleting mixture */
            costOpt = curCost + 2*sumCross[i] - sumSelf[i];

            if (costOpt < minCost) {
                minCost = costOpt;
                actMix1 = i; actMix2 = 0;
            }

            for (j = i+1; j < numMix; j++) {
                if (mixMask[j]) {
                    mergePoss = mergePossNum(i,j);

                    /* Calculate cost for merging mixtures */
                    costOpt = curCost + 2*sumCross[i] + 2*sumCross[j] +
                        -sumSelf[i] - sumSelf[j] + 2*self[i+j*numMix] +
                        -2*newSumCross[mergePoss] + newSumSelf[mergePoss];

                    if (costOpt < minCost) {
                        minCost = costOpt;
                        actMix1 = i; actMix2 = j;
                        actMergePoss = mergePoss;
                    }
                }
            }
        }
    }

    /* Print debugging information to screen if flag is true */
    if (DEBUG) {
        if (actMix2 == 0) {
            mexPrintf("Current cost %g; Deleting mixture %d for cost %g\n",
                curCost, actMix1, minCost);
        } else {
            mexPrintf("Current cost %g; Merging mix %d and %d for cost %g\n",
                curCost, actMix1, actMix2, minCost);
        }
    }
}

```

```

/* copyMixtureParameters -- Copies probabilities, means and covariances
   from original structures into new working structures
   to provide the starting point for the reduction
   process.
   Precondition: probs, means and covs contain the parameters for the
   original mixtures, memory is allocated for newProbs,
   newMeans and newCovs
   Postcondition: Data from probs, means and covs are copied into
   newProbs, newMeans and newCovs. */
void copyMixtureParameters(void)
{
    register int i;
    int numElem;

    /* Copy probabilities */
    numElem = numMix;
    for (i = 0; i < numElem; i++)
        newProbs[i] = probs[i];

    /* Copy means */
    numElem *= numVar;
    for (i = 0; i < numElem; i++)
        newMeans[i] = means[i];

    /* Copy covariances */
    numElem *= numVar;
    for (i = 0; i < numElem; i++)
        newCovs[i] = covs[i];

    /* Initialize the current number of mixture components */
    numCurMix = numMix;
}

/* deleteMixture -- Deletes the specified component, updates all costs
   Precondition: mix contains the index of the mixture to be deleted
   Postcondition: newSumSelf (self-likeness entries for each merge
   possibility) and sumSelf (partial sums of self-
   likeness entries for current reduced mixture) are
   updated to reflect the new cost after the specified
   component has been deleted */
void deleteMixture(int mix)
{
    register int m1, m2, mergePoss;

    /* Clear the flag for the mixture to indicate that it has been
       deleted */
    mixMask[mix] = 0;

    /* Update stored new columns for the cross-likeness and self-likeness
       matrices for all merge possibilities */
    for (m1 = 0; m1 < numMix; m1++) {
        if (mixMask[m1]) {
            for (m2 = m1+1; m2 < numMix; m2++) {
                if (mixMask[m2]) {
                    mergePoss = mergePossNum(m1,m2);
                    newSumSelf[mergePoss] -= 2*newSelf[mergePoss*numMix+mix];
                }
            }
        }
    }
}

```

```

    /* Update partial sums of the self likeness matrix to reflect removal
       of component */
    for (m1 = 0; m1 < numMix; m1++) {
        if (mixMask[m1]) {
            /* Subtract self distances due to deleted component */
            sumSelf[m1] -= 2*self[mix*numMix+m1];
        }
    }
}

/* mergeMixtures -- Updates all merge possibilities with the newly
   merged component, placing the parameters for merged
   components in mix1
   Precondition: mix1 and mix2 contain the indices of the two
   components to be merged. mix2 should have been
   deleted already (using deleteMixture())
   Postcondition: parameters of merged components are calculated and
   stored in place of mix1; cross and self matrix
   entries (and sum vector entries) are updated with new
   costs; merge possibility cost structures are updated
   to reflect the changes due to the merged components.*/
void mergeMixtures(int mix1, int mix2)
{
    int m1, m2, m3, i, j, k, mergePoss;
    double d;

    /* Calculate the parameters (weight, mean, covariance) for the
       merged components */
    calcMergeParam(mix1, mix2);

    /* Store parameters for newly merged component in place of mix1 */
    for (i = 0; i < numVar; i++) {
        for (j = i; j < numVar; j++) {
            k = i + j*numVar;
            newCovs[mix1*numVar*numVar+k] = mergeP[k];
        }

        newMeans[mix1*numVar+i] = mergeMu[i];
    }
    newProbs[mix1] = mergep;

    /* Update distance matrices to reflect merge
       (using the pre-computed parameters from the merge possibility
       structure) */
    mergePoss = mergePossNum(mix1, mix2);
    for (m1 = 0; m1 < numMix; m1++) {
        /* Store cross distances for new component */
        cross[m1+mix1*numMix] = newCross[mergePoss*numMix+m1];

        if (mixMask[m1]) {
            /* Store self distances for new component & update sums */
            sumSelf[m1] -= 2*self[mix1+m1*numMix];
            d = self[mix1+m1*numMix] = self[m1+mix1*numMix] =
                newSelf[mergePoss*numMix+m1];
            sumSelf[m1] += 2*d;
        }
    }
    sumCross[mix1] = newSumCross[mergePoss];
    sumSelf[mix1] = newSumSelf[mergePoss];
}

```



```

/* calcOrigCosts -- Populate the original cost matrix
   Precondition: memory should be allocated for all structures; probs,
                 means and covs should contain parameters for original
                 mixture components
   Postcondition: cross and self matrices are populated, partial sums
                 are calculated, sumDist is calculated, mixture mask
                 flags are initialized */
void calcOrigCosts(void)
{
    int m1, m2, i, j;

    /* Zero out the partial sums */
    for (m1 = 0; m1 < numMix; m1++)
        sumCross[m1] = 0.0;

    /* Calculate similarity measure for every pair of components */
    for (m1 = 0; m1 < numMix; m1++) {
        for (m2 = m1; m2 < numMix; m2++) {
            i = m1*numMix+m2; j = m2*numMix+m1;
            cross[i] = cross[j] = self[i] = self[j] =
                calcDist(probs[m1], &means[m1*numVar], &covs[m1*numVar*numVar],
                        probs[m2], &means[m2*numVar], &covs[m2*numVar*numVar]);

            /* Update the partial sums for the two components */
            sumCross[m1] += cross[i];
            if (m1 != m2)
                sumCross[m2] += cross[i];
        }
    }

    sumDist = 0;
    for (m1 = 0; m1 < numMix; m1++) {
        /* Calculate partial self sum from cross sum (this contains the
           sum of the matrix row and column due to the respective
           component) */
        sumSelf[m1] = 2*sumCross[m1] - cross[m1*(numMix+1)];

        /* Calculate total sum for original mixture */
        sumDist += sumCross[m1];

        /* Initialize mask flags */
        mixMask[m1] = 1;
    }
}

```

```

/* calcOrigMergePoss -- Calculate all merge possibilities for original
mixture
Precondition: memory is allocated for structures, distance matrices
(self and cross) and partial sums are populated;
probs, means and covs contain parameters of original
mixture
Postcondition: newSelf, newCross, newSumSelf and numSumCross are
populated to reflect the new entries for the self and
cross matrices if each pair of components are selected
for merging */
void calcOrigMergePoss(void)
{
    int m1, m2, m3, mergePoss;
    double d;

    for (m1 = 0; m1 < numMix; m1++) {
        for (m2 = m1+1; m2 < numMix; m2++) {
            mergePoss = mergePossNum(m1,m2);
            newSumCross[mergePoss] = 0;

            /* Calculate parameters for merging components m1 & m2 */
            calcMergeParam(m1,m2);

            /* Calculate distance of this merged component to all other
            components */
            for (m3 = 0; m3 < numMix; m3++) {
                d = calcDist(mergep,mergeMu,mergeP,
                    probs[m3],&means[m3*numVar],&covs[m3*numVar*numVar]);
                newSelf[mergePoss*numMix+m3] =
                    newCross[mergePoss*numMix+m3] = d;
                newSumCross[mergePoss] += d;
            }

            /* Calculate self distance for component */
            d = calcDist(mergep,mergeMu,mergeP,mergep,mergeMu,mergeP);
            newSelf[mergePoss*numMix+m1] = d;
            newSelf[mergePoss*numMix+m2] = 0;
            newSumSelf[mergePoss] = 2*(newSumCross[mergePoss] -
                newCross[mergePoss*numMix+m1] -
                newCross[mergePoss*numMix+m2]) + d;
        }
    }
}

```

```

/* calcMergeParam -- Calculates the parameters (mean, cov, prob) for
   merging a pair of components, puts them in the global
   holding area mergep, mergeMu, mergeP
   Precondition: newProbs, newMeans and newCovs contain the current
   parameters of the reduced mixture; m1 and m2 contain
   the indices of the components to be merged
   Postcondition: mergep, mergeMu and mergeP contain the weight, mean
   and covariance for the component fitted to the pair of
   components, with the parameters such that the overall
   mean and covariance remains unchanged. muD is used for
   temporary calculation.
   Note: only lower triangle of matrix is calculated; upper
   triangle is neither calculated nor populated */
void calcMergeParam(int m1, int m2)
{
    register int i, j, k;
    register double p1, p2;
    double di, *mean1 = &newMeans[numVar*m1],
    *mean2 = &newMeans[numVar*m2],
    *cov1 = &newCovs[numVar*numVar*m1],
    *cov2 = &newCovs[numVar*numVar*m2];

    p1 = newProbs[m1]; p2 = newProbs[m2];
    mergep = p1 + p2;
    di = 1.0/mergep;
    p1 *= di; p2 *= di;

    /* Calculate difference of means and combined mean */
    for (i = 0; i < numVar; i++) {
        muD[i] = mean1[i] - mean2[i];
        mergeMu[i] = p1*mean1[i] + p2*mean2[i];
    }

    /* Calculate combined covariance */
    for (i = 0; i < numVar; i++) {
        for (j = i; j < numVar; j++) {
            k = i + j*numVar;
            mergeP[k] = p1*cov1[k] + p2*cov2[k] + p1*p2*muD[i]*muD[j];
        }
    }
}

/* calcDist -- Calculate a single distance entry between the given
   parameters. This is the "engine", containing the
   highly optimized implementation of Eq. (3.46)
   described in Section 3.3.4.1.
   Precondition: p1, mean1 and cov1, and p2, mean2 and cov2 contain the
   parameters of the pair of components to be merged
   Postcondition: the similarity measure between the two components is
   calculated and returned. The temporary structures muD
   and P are used for the calculation. */
double calcDist(double p1, double *mean1, double *cov1,
                double p2, double *mean2, double *cov2)
{
    register int i, j, k;
    register double d, di;
    double diProd, cost;

```

```

/* Calculate the sum of the two covariances and the difference
   of the two means (only calculate lower triangle of the covariance
   sum) */
for (i = 0; i < numVar; i++) {
    for (j = i; j < numVar; j++) {
        k = i + j*numVar;
        P[k] = cov1[k] + cov2[k];
    }

    muD[i] = mean1[i] - mean2[i];
}

/* Divide right-most column by lower-right element */
di = 1.0/P[numVar*numVar - 1];
Di[numVar-1] = di;
diProd = di*Inv2PI;
for (j = 0; j < numVar-1; j++)
    P[j + numVar*(numVar-1)] *= di;

/* Complete U-D factorization in-place */
for (j = numVar-2; j >= 0; j--) {
    /* Calculate diagonal element for column */
    d = P[j*(numVar+1)];
    for (k = j+1; k < numVar; k++) {
        di = P[j+k*numVar];
        d -= P[k*(numVar+1)]*di*di;
    }

    P[j*(numVar+1)] = d;
    di = 1.0/d;
    Di[j] = di;
    diProd *= di*Inv2PI;

    /* Calculate rest of column */
    for (i = j-1; i >= 0; i--) {
        d = P[i+j*numVar];
        for (k = j+1; k < numVar; k++)
            d -= P[k*(numVar+1)]*P[i+k*numVar]*P[j+k*numVar];

        P[i+j*numVar] = d*di;
    }
}

if (mean1 == mean2) {
    /* Calculate self cost if the two components were the same */
    return p1*p2*sqrt(diProd);
} else {
    /* Solve back-substitution with mean */
    di = 0;
    for (j = numVar-1; j >= 0; j--) {
        d = muD[j];
        for (i = j+1; i < numVar; i++)
            d -= muD[i]*P[j+i*numVar];

        muD[j] = d;
        di += d*d*Di[j];
    }

    /* Calculate cost & return */
    return p1*p2*exp(-0.5*di)*sqrt(diProd);
}
}

```

Bibliography

1. Alspach, D.L. "A Gaussian Sum Approach to the Multitarget Identification–Tracking Problem," *Automatica*, 11(3):285–296 (May 1975).
2. Bar-Shalom, Yaakov and Thomas E. Fortmann. *Tracking and Data Association*. Orlando, FL: Academic Press, Inc., 1988.
3. Bar-Shalom, Yaakov and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques and Software*. Norwood, MA: Artech House, 1993.
4. Bar-Shalom, Yaakov and Xiao-Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT: YBS Publishing, 1995.
5. Billetter, Dale R. *Multifunction Array Radar*. Norwood, MA: Artech House, 1989.
6. Blackman, Samuel S. *Multiple-Target Tracking with Radar Applications*. Norwood, MA: Artech House, 1986.
7. Blackman, Samuel S. and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.
8. Blackman, S.S., et al. "Application of Multiple Hypothesis Tracking to Multiradar Air Defense Systems," *Multisensor Multitarget Data Fusion, Tracking and Identification Techniques for Guidance and Control Applications*, NATO AGARD AG-337:96–120 (October 1996).
9. Bloem, Edwin A. and Henk A.P. Blom. "Joint Probabilistic Data Association Methods Avoiding Track Coalescence," *Proceedings of the 34th IEEE Conference on Decision and Control*, 3:2752–2757 (December 1995).
10. Blom, Henk A.P. and Yaakov Bar-Shalom. "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Transactions on Automatic Control*, 33(8):780–783 (August 1988).
11. Blom, Henk A.P. and Edwin A. Bloem. "Joint Probabilistic Data Association Avoiding Track Coalescence," *IEE Colloquium on Algorithms for Target Tracking*, 1/1–1/3 (May 1995).
12. Blom, Henk A.P. and Edwin A. Bloem. "Probabilistic Data Association Avoiding Track Coalescence," *IEEE Transactions on Automatic Control*, 45(2):247–259 (February 2000).
13. Brooks, Mike. "Matrix Reference Manual: Matrix Calculus." Online reference material. n. pag. <http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/calculus.html>. 04 December 2002. Reproduced in Appendix B.

14. Burns, Brendan T., James B. Moody and Jason L. Williams. *WhoAmI? Person Verification System*. BE(Electronics) Undergraduate Project, Queensland University of Technology, Brisbane, Australia, 1998.
15. Busch, M. and S. Blackman. "Evaluation of IMM Filtering for an Air Defense System Application," *SPIE Signal and Data Processing of Small Targets*, 2561:435–447 (July 1995).
16. Cong, Shan. *Statistical Studies in Multiple Target Tracking*. M.S. Eng Thesis, Wright State University, Dayton, OH, 1996.
17. Dempster, R.J., et al. "Combining IMM Filtering and MHT Data Association for Multitarget Tracking." *Proceedings of the 29th Southeastern Symposium on System Theory*. 123–127. Cookeville, TN: IEEE Press, March 1997.
18. Dennis, John E. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
19. Ding, Z. and L. Hong. "Bias Phenomenon and Compensation for PDA/JPDA Algorithms," *Mathematical Computer Modelling*, 27(12):1–16 (June 1998).
20. Dwyer, Paul S. "Some Application of Matrix Derivatives in Multivariate Analysis," *American Statistical Association Journal*, 607–625 (June 1967).
21. Hong, Lang. "EE718 Multitarget Tracking and Data Association." Class Assignment. Wright State University, Dayton, OH, 2002.
22. Hong, Lang and Shan Cong. "Bias Phenomenon and Compensation in Multiple Target Tracking Algorithms," *Mathematical Computer Modelling*, 31(8–9):147–165 (May 2000).
23. Kailath, Thomas. "The Divergence and Bhattacharyya Distance Measures in Signal Selection," *IEEE Transactions on Communication Theory*, COM-15(1):52–60 (February 1967).
24. Kalman, Rudolph E. "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, 82(Series D):35–45 (1960).
25. Kastella, Keith. "A Maximum Likelihood Estimator for Report-to-Track Association," *SPIE Signal and Data Processing of Small Targets*, 1954:386–393 (October 1993).
26. Kastella, Keith. "Comparison of Mean-Field Tracker and Joint Probabilistic Data Association Tracker in High-Clutter Environments," *SPIE Signal and Data Processing of Small Targets*, 2561:489–495 (September 1995).
27. Koch, Wolfgang. "Experimental Results on Bayesian MHT for Maneuvering Closely-Spaced Objects in a Densely Cluttered Environment," *Radar 97 (Conf. Publ. No. 449)*, 729–733 (October 1997).

28. Koch, Wolfgang and Günter van Keuk. "Multiple Hypothesis Track Maintenance with Possibly Unresolved Measurements," *IEEE Transactions on Aerospace and Electronic Systems*, 883–892 (July 1997).
29. Kullback, Solomon. *Information Theory and Statistics* (Second Edition). Mineola, NY: Dover Publications, 1997.
30. Kurien, Thomas. "Issues in the Design of Practical Multitarget Tracking Algorithms." *Multitarget-Multisensor Tracking: Advanced Applications*. 43–83. Norwood, MA: Artech-House, 1990.
31. Lainiotis, D.G. and S.K. Park. "On Joint Detection, Estimation and System Identification: Discrete Data Case," *International Journal of Control*, 17(3):609–633 (March 1973).
32. Leon-Garcia, Alberto. *Probability and Random Processes for Electrical Engineering* (Second Edition). Reading, MA: Addison-Wesley, 1994.
33. *MATLAB® 6.0 Online Function Reference*. Natick, MA: Mathworks, Inc, 2001.
34. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, Volume 1*. Arlington, VA: Navtech, 1994.
35. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, Volume 2*. Arlington, VA: Navtech, 1994.
36. Maybeck, Peter S. "EE844 Computational Aspects of Modern Control." Lecture Notes. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2002.
37. Maybeck, Peter S. "EENG768 Multiple Model Adaptive Estimation." Lecture Notes. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2002.
38. Pao, Lucy Y. "Multisensor Multitarget Mixture Reduction Algorithms for Tracking," *Journal of Guidance, Control, and Dynamics*, 17(6):1205–1211 (November–December 1994).
39. Poore, A.B. and A.J. Robertson. "A New Lagrangian Relaxation-Based Algorithm for a Class of Multidimensional Assignment Problems," *Computational Optimization and Applications*, 8(2):129–150 (September 1997).
40. Reid, Donald B. "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control*, AC-24(6):843–854 (December 1979).
41. Reynolds, D.A. and R.C. Rose. "Robust Text-Independent Speaker Identification using Gaussian Mixture Speaker Models," *IEEE Transactions Speech and Audio Processing*, 3(1):72–83 (January 1995).

42. Ristic, Branko and Sanjeev Arulampalam. "Multitarget Mixture Reduction Algorithm with Incorporated Target Existence Recursions," *SPIE Signal and Data Processing of Small Targets*, 4048:366–377 (July 2000).
43. Roecker, J.A. "Multiple Scan Joint Probabilistic Data Association," *IEEE Transactions Aerospace and Electronic Systems*, 31:1204–1210 (July 1995).
44. Salmond, David J. *Mixture Reduction Algorithms for Uncertain Tracking*. Technical Report 88004, Farnborough, UK: Royal Aerospace Establishment, January 1988. DTIC Number ADA197641.
45. Salmond, David J. "Mixture Reduction Algorithms for Target Tracking." *IEE Colloquium on State Estimation in Aerospace and Tracking Applications*. 7/1–7/4. London, UK: IEE Publishing, December 1989.
46. Salmond, David J. *Tracking in Uncertain Environments*. Technical Memorandum AW 121, Farnborough, UK: Royal Aerospace Establishment, September 1989. DTIC Number ADA215866. Taken from a D Phil thesis of the University of Sussex.
47. Salmond, David J. "Mixture Reduction Algorithms for Target Tracking in Clutter," *SPIE Signal and Data Processing of Small Targets*, 1305:434–445 (April 1990).
48. Scharf, Louis L. *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*. Reading, MA: Addison-Wesley, 1991.
49. Singer, R.A., et al. "Derivation and Evaluation of Improved Tracking Filters for use in Dense Multi-target Environments," *IEEE Transactions on Information Theory*, IT-20(4):423–832 (July 1974).
50. Skolnik, Merrill I. *Introduction to Radar Systems* (Third Edition). New York, NY: McGraw-Hill, 2001.
51. Stimson, George W. *Introduction to Airborne Radar* (Second Edition). Raleigh, NC: Scitech Publishing, 1998.
52. Strang, G. *Linear Algebra and its Applications* (Third Edition). Orlando, FL: Harcourt College Publishers, 1988.
53. Streit, Roy L. and Tod E. Luginbuhl. *Probabilistic Multi-Hypothesis Tracking*. Technical Report, Newport, RI: Naval Undersea Warfare Center Division, 1995.
54. Tantaratana, Sawasd. "Some Recent Results of Sequential Detection." *Advances in Statistical Signal Processing Volume 2*. 265–296. Greenwich, CT: JAI Press, 1993.
55. Wark, Timothy. *Multi-Modal Speech Processing for Automatic Speaker Recognition*. PhD Thesis, Queensland University of Technology, Brisbane, Australia, 2001.

56. Weiss, J.L., et al. "Finite Computable Filters for Linear Systems Subject to Time Varying Model Uncertainty." *Proceedings of NAECON*. 349–355. Dayton, OH: IEEE Press, May 1983.
57. Williams, Jason L. and Craig Larson. "EENG768 Multiple Model Adaptive Estimation Project." Student Project. Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2002.
58. Wilson, R. "Image Analysis and Segmentation using Mixture Models." *IEE Seminar on Time-scale and Time-Frequency Analysis and Applications*. 11/1–11/6. London, UK: IEE Publishing, February 2000.

Vita

Flight Lieutenant Jason L. Williams graduated from Queensland University of Technology in April 1999, receiving a Bachelor of Engineering (Electronics) with First Class Honours, and a Bachelor of Information Technology with Distinction. He joined the Royal Australian Air Force in 1996 through the undergraduate sponsorship program. His first assignment was at the Electronic Warfare Squadron in Adelaide, South Australia, where he received the E-Systems Commander's Trophy for Excellence in Electronic Warfare. In 2001 he was selected to study the Master of Science in Electrical Engineering program at the United States Air Force Institute of Technology, concentrating on Stochastic Estimation and Control and Signal Processing. Upon graduation he will be assigned to the Aircraft Self Protection Systems Program Office in Canberra, Australia.

Flight Lieutenant Williams is a member of Eta Kappa Nu and Tau Beta Pi, as well as the Golden Key National Honor Society. He is a student member of the Institute of Electrical and Electronic Engineers.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 25-03-2003		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) Jul 2002 – Mar 2003	
4. TITLE AND SUBTITLE GAUSSIAN MIXTURE REDUCTION FOR TRACKING MULTIPLE MANEUVERING TARGETS IN CLUTTER				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Williams, Jason L., Flight Lieutenant, RAAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/03-19	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/SNAT Attn: Mr. Stanton H. Musick 2241 Avionics Circle WPAFB OH 45433-7765 DSN: 785-1115, ext 4292 e-mail: Stanton.Musick@wpafb.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The problem of tracking multiple maneuvering targets in clutter naturally leads to a Gaussian mixture representation of the Probability Density Function (PDF) of the target state vector. State-of-the-art Multiple Hypothesis Tracking (MHT) techniques maintain the mean, covariance and probability weight corresponding to each hypothesis, yet they rely on <i>ad hoc</i> merging and pruning rules to control the growth of hypotheses. This thesis investigates the performance benefit achievable by applying a structured cost function-based approach to the hypothesis control problem. A new cost function, the Integral Square Difference (ISD) cost, is proposed for measuring the difference between the full target state PDF and a reduced-order approximation. The ISD cost function is physically meaningful, and, unlike any previously proposed cost function, it is also mathematically tractable, requiring neither numerical integration nor approximation for evaluation. A reduction algorithm is proposed which selects components for merging or pruning to minimize the increase in the ISD cost. This solution is used directly, and also as the starting point for an iterative gradient-based optimization. The performance of the ISD-based algorithm for tracking a single target in heavy clutter is compared to that of Salmond's joining filter, which previously had provided the highest performance in the scenario examined. For a large number of mixture components, it is shown that the ISD algorithm outperforms the joining filter remarkably, yielding an average track life more than double that achievable using the joining filter. The results indicate that the tracking performance of the ISD-based filter in heavy clutter is significantly higher than achievable using any previously published algorithm.					
15. SUBJECT TERMS Radar tracking, Search radar, Automatic tracking, Track while scan, Radar clutter, Kalman filtering, Bayes' theorem, Probability density functions, Maximum likelihood estimation, Optimization, Statistical distributions, Stochastic processes					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)
U	U	U	UU	247	Dr Peter S. Maybeck (937) 255-3636, ext 4581; e-mail: Peter.Maybeck@afit.edu