University of London
Imperial College London
Department of Electrical and Electronic Engineering

Sequential Gaussian Mixture Techniques for Target Tracking Applications

Stefano Andrea Robbiati

November 2006

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and the
Diploma of the Imperial College in the Faculty of Engineering

Declaration

This dissertation is submitted for the degree of Doctor of Philosophy of the University of London and the Diploma of the Imperial College in the Faculty of Engineering. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

Abstract

Target tracking is a well known state estimation problem which has a simple solution in the form of the Kalman filter under Gaussian and linear model assumptions. The challenge arises in problems where these assumptions are no longer valid. To tackle these cases, a variety of suboptimal moment-matching techniques, such as the extended Kalman filter, have been developed over the years. More recently, randomised sequential Monte Carlo techniques, generally referred to as particle filters, have shown how large numbers of sample simulations can be used to produce very accurate estimates (at the cost of significant computational requirements).

This thesis focuses on the use of Gaussian mixture techniques for the solution of target tracking problems in which model nonlinearity arises in consequence of switching model parameters. Problems of this nature, encountered for example in manoeuvring target tracking or target classification, involve model structures well suited to Gaussian mixture approximations. These structures have been earlier exploited in deterministic algorithms such as pseudo-Bayesian and interacting multiple model techniques. Rao-Blackwellised particle filters, in which random selection is carried out only with respect to the system modes, also employ Gaussian mixture approximations.

In this thesis, efficient mixture filters with features associated with both particle filters and deterministic algorithms will be discussed. These techniques provide a more accurate representation of the posterior distribution of the target, by permitting the complexity of the conditional distribution to expand and contract. This is carried out by means of efficient mixture reduction techniques, a discussion of which will tie in with clustering methods, normally used for data association. Furthermore, it will be shown how these algorithms can be directly extended to take into account of extra nonlinearities, such as multiple and/or nonlinear measurements, while still retaining their simplicity and their computational efficiency.

Acknowledgements

I offer the sincerest of thanks to my project supervisor, Richard Vinter, for his continuous guidance and for his tireless support and encouragement. I would also like to thank co-researcher Martin Clark for his technical advice and for the many hours he dedicated to our work. My research has also greatly benefited from the many fruitful discussions I have had with Neil Cade of BAE Systems: his many ideas and suggestions are gratefully acknowledged. My thanks also go to Moeen Yaqoob for his cooperation and his advice with bearings-only target tracking problems.

I am extremely grateful for the financial support which has made this work possible: during the course of this project I was funded by an EPSRC CASE award bursary with BAE Systems. I also wish to thank Imperial College London, and in particular all my colleagues in the Control & Power research group, for providing such a great and stimulating environment to work in.

Special thanks go to my parents for all their love, their constant support and for always treating my education with the highest importance. I am also particularly grateful to Iria, who has always been there for me and who has brought happiness to my life. Many thanks also go to all my friends and to my brother, Fabio, for providing me with necessary distraction. Above all, I wish to thank God, for giving me the strength to get this far and for bringing me comfort in times of need.

Stefano A. Robbiati

Contents

1 Introduction **13**

2 Target Tracking Background **19**

2.1 Traditional filtering approaches 19

2.1.1 Discrete-time dynamic models and the Bayesian approach 19

2.1.2 The Kalman filter 21

2.1.3 The extended Kalman filter 28

2.1.4 The unscented Kalman filter 32

2.2 Sequential Monte Carlo methods 35

2.2.1 Sequential importance sampling 36

2.2.2 Resampling techniques 37

2.2.3 The bootstrap filter 40

2.3 A multiple-model problem: manoeuvring target tracking 41

2.3.1 Jump Markov linear systems 41

2.3.2 The optimal multiple-model Bayesian solution 46

2.3.3 Generalised pseudo-Bayes approximation techniques 47

2.3.4 Particle filters for JMLS models 50

2.3.5 Preliminary comparisons 52

3 Sequential Gaussian Mixture Filters for Target Tracking **56**

3.1 Introduction & review of mixture filters 56

3.1.1 The basic structure of mixture filters 57

3.1.2 Mixture filters & Rao-Blackwellised particle filters 59

3.2 Mixture Reduction 61

3.2.1 Weight-based Mixture Reduction 62

3.2.2 Distribution-based mixture reduction 78
3.2.3 Applying distribution-based algorithms to multiple-model problems 88
3.2.4 Hybrid mixture reduction 93
3.2.5 Simulation comparisons 95
3.3 Look-ahead mixture filters 101
3.3.1 Derivation of the look-ahead mixture filter (LAMF) 102
3.3.2 The LAMF algorithm 104
3.3.3 Simulation comparisons 106

4 Tracking a Manoeuvring Target in Clutter **111**
4.1 The problem 112
4.2 The optimal Bayesian solution 113
4.3 The multiple measurement mixture filter 117
4.3.1 The MMMF algorithm 118
4.4 Simulation results 120

5 Bearings-only Manoeuvring Target Tracking **124**
5.1 Traditional methods for bearings-only tracking 124
5.2 The shifted Rayleigh filter 127
5.2.1 Measurement treatment 128
5.2.2 Derivation of the shifted Rayleigh filter 129
5.2.3 Summary of the shifted Rayleigh filter equations 137
5.3 Extension to bearings-only manoeuvring tracking 139
5.3.1 Evaluation of the bearing density 140
5.3.2 The shifted Rayleigh mixture filter 143
5.4 Assessment of the SRMF in a multi-sensor target tracking problem 144
5.4.1 Problem definition and state model 145
5.4.2 Multisensor measurement model 146
5.4.3 The SRMF for multisensor tracking with vacuous measurement . . 148
5.4.4 Simulation results 152

6 Conclusion **158**

A The data association problem: tracking in clutter **162**

A.1 The target in clutter problem 162

A.2 Measurement gating 163

A.3 Data association techniques 165

A.3.1 The optimal Bayesian solution 165

A.3.2 Probabilistic data association filter (PDAF) 169

A.3.3 Mixture filters based on mixture reduction 169

B Algorithms implementation **170**

B.1 Multiple-model target tracking algorithms 170

B.1.1 The first order GPB algorithm 170

B.1.2 The second order GPB algorithm 171

B.1.3 The IMM algorithm 172

B.1.4 The multiple model bootstrap filter (MMBF) 173

B.1.5 The Rao-Blackwellised particle filter (RBPF) 174

B.2 Multiple measurement target tracking algorithms 175

B.2.1 The probabilistic data association filter (PDAF) 175

B.2.2 The data association mixture filter 176

B.3 Multiple-model/multiple-measurements target tracking algorithms 178

B.3.1 The IMMPDA algorithm 178

B.3.2 A multiple-measurement/multiple-model particle filter 181

C Conduct of simulations **183**

List of Figures

1.1 Improving the 'efficiency frontier' using mixture filters 16

2.1 The structure of the Kalman filter 27
2.2 The σ-points used by the unscented Kalman filter 33
2.3 An example of sampling importance resampling 39
2.4 An example of stratified sampling. 40
2.5 Sample mode transition diagram 42
2.6 Set of mode histories for *t=0:2* 46
2.7 The second order generalised pseudo-Bayes algorithm (GPB2) 49
2.8 The interacting multiple model algorithm (IMM) 50
2.9 Sample realisation of Track 1 52
2.10 Preliminary simulation comparisons: RMS error vs. computational cost for different numbers of samples (for Track 1) 54

3.1 A comparison of the structure of the mixture filter with that of the Rao-Blackwellised particle filter. 59
3.2 A sample realisation of the sampling and resampling stages of the RBPF . 60
3.3 An example of mixture reduction 62
3.4 An example of sample selection using the Fearnhead and Clifford algorithm 66
3.5 Sample realisation of Track 2 72
3.6 RMS error comparison of weight-based mixture reduction techniques for Track 2 72
3.7 Mode probabilities comparison of weight-based mixture reduction techniques for Track 2 73
3.8 Mean RMS error produced by the α-algorithm for varying values of α for Track 1 76

3.9 RMS error comparison of weight-based mixture reduction techniques for Track 1 76
3.10 Mode probabilities comparison of weight-based mixture reduction techniques for Track 1 77
3.11 Weight-based mixture reduction algorithms comparisons: RMS error vs. computational cost for different numbers of samples (for Track 1) 78
3.12 The structure of the joining algorithm 81
3.13 The structure of the clustering algorithm 83
3.14 Decomposing the ISD cost function 86
3.15 Clustering strategies for multiple-model problems 88
3.16 Calculation of γ_t^n: an illustrative example 90
3.17 Target in clutter: average track-life vs. number of mixture components . . 97
3.18 Target in clutter: average track-life vs. computational cost 99
3.19 Mixture reduction algorithms comparisons: RMS error vs. computational cost for different numbers of samples (for Track 1) 101
3.20 Structure modifications introduced by the look-ahead mixture filter 102
3.21 Look-ahead mixture filter comparisons: RMS error vs. computational cost for different numbers of samples (for Track 1) 107
3.22 Sample realisation of Track 3 108
3.23 Look-ahead mixture filter comparisons: RMS error vs. computational cost for different numbers of samples (for Track 3) 109
3.24 RMS error comparison of the LAMF with the MF for Track 3 109
3.25 Mode probabilities comparison of the LAMF with the MF for Track 3 . . 110

4.1 Multiple-model/multiple-measurement problem: exponential growth in association hypotheses 114
4.2 Manoeuvring target in clutter: average track-life vs. number of mixture components 121
4.3 Manoeuvring target in clutter: average track-life vs. computational cost . 122

5.1 The bearings-only measurement 125
5.2 The bearing probability density 129
5.3 Normalisation of the virtual measurement 130

5.4 The angle of interest when calculating the bearing density 142
5.5 Sample realisation of Track 4 153
5.6 Shifted Rayleigh mixture filter comparisons: RMS error vs. computational cost for different numbers of samples (for Track 4) 154
5.7 Position RMS error comparison of the SRMF with other techniques . . . 156
5.8 Velocity RMS error comparison of the SRMF with other techniques . . . 157
5.9 Mean RMS position error vs. probability of clutter 157

A.1 Multiple-measurement problem: exponential growth in association hypotheses 166

List of Abbreviations

DEA	detection-estimation algorithm
EKF	extended Kalman filter
GPB1	first-order generalised pseudo-Bayes algorithm
GPB2	second-order generalised pseudo-Bayes algorithm
IEKF	iterated extended Kalman filter
IMM	interacting multiple model algorithm
IMMPDA	interacting multiple model probabilistic data association filter
ISD	integral square difference distance
ISDI	integral square difference initialisation algorithm
ISDO	integral square difference optimisation algorithm
JA	joining algorithm
JMLS	jump Markov linear system
LAMF	look-ahead mixture filter
KF	Kalman filter
MCMC	Markov chain Monte Carlo methods
MF	mixture filter
MMBF	multiple model bootstrap filter
MMMF	multiple measurement mixture filter
MMSE	minimum mean square error
MPEKF	modified polar coordinates extended Kalman filter
PDAF	probabilistic data association filter
PF	particle filter
RBPF	Rao-Blackwellised particle filter
RPEKF	range-parameterised extended Kalman filter
RMS	root mean square

RR	residual resampling
SIS	sampling importance sampling
SIR	sampling importance resampling
SRF	shifted Rayleigh filter
SRIMM	shifted Rayleigh interacting multiple model algorithm
SRMF	shifted Rayleigh mixture filter
SS	stratified sampling
UKF	unscented Kalman filter
UIMM	unscented interacting multiple model algorithm (or IMM-UKF)
UMF	unscented mixture filter

Chapter 1

Introduction

The field of target tracking and interception of incoming missiles is currently of great interest within the defence industry. The development of reliable and accurate systems to serve this task is strategically important to the military and great efforts are being made to improve current technology. Within this field, a major area of interest is the tracking phase, the problem of estimating a target's state given noisy radar measurements, which is generally approached as a Bayesian problem.

In its simplest form the target tracking problem can be solved optimally using the *Kalman filter*. For scenarios in which the target dynamics can be modelled using linear state-space models and for which the state and measurement noises are additive Gaussian processes, evaluation of the integrals constituting the exact Bayesian solutions is in fact possible. Unfortunately in most practical problems these assumptions are not valid, and suboptimal approximations need to be used.

Tracking problems involving non-linear/non-Gaussian dynamics have traditionally been approached using *moment-matching filters*, techniques which are based on local or statistical linearisation of the system's dynamics, and which describe the conditional distributions in terms of their first and second moments. The most commonly used of these techniques, the *extended Kalman filter* (EKF) [12], produces a local linearisation of the system equations by means of a Taylor series expansion around the prior estimate. In order to keep the complexity of the algorithm manageable, generally only the first or second order terms of the expansions are used. In view of the linearisation involved, this

algorithm is successful mainly when the nonlinearities in the problem are mild.

Another popular moment-matching technique is the *unscented Kalman filter* (UKF) [42], a method which involves taking a number of deterministically chosen sample points from the prior distribution and propagating them through the nonlinearities of the system. First and second moments are then matched to these transformed points to fit a Gaussian distribution to the posterior. This approach is generally accepted to be more accurate than the EKF, as the moments are matched up to 3rd order Taylor series expansion terms [70]. A number of other moment-matching approaches have been developed for specific types of problems, such as the *pseudomeasurement filter* [1] or the more recent *shifted Rayleigh filter* [24] for bearings-only tracking.

More recently, a new class of *density approximation filters*, generally referred to as *sequential Monte Carlo algorithms* (or *particle filters*) ([28],[37]), has been introduced to attempt to approximate the posterior conditional density of the state by drawing sufficiently numerous samples from it (a simulation-based approach known as Monte Carlo integration method). As it is not generally possible to sample directly from the posterior distribution, sequential Monte Carlo methods make use of an *importance sampling distribution*, from which particles may be easily drawn recursively. This approach is very simple to implement for nearly any type of system and is generally extremely effective as long as enough samples are used. These benefits however, come at a high computational cost, which is proportional to the number of particles being used. This nonetheless grants particle filters a degree of flexibility as the performance-to-cost ratio may be customised according to the user's requirements or hardware constraints.

Multiple-model problems

This thesis follows these general approaches, but focuses on a particular type of problem in which the nonlinearity is introduced by a switching parameter in the system's equations. For these problems a model of target motion generally referred to as *jump Markov linear system* is adopted that allows for the parameters in a set of state-space equations to change according to a hidden Markov process. This model can be used

for a variety of problems, such as manoeuvring target tracking or target classification problems, in which the main uncertainty concerns the behaviour or the nature of the target (respectively).

The Bayesian solution to this type of *multiple-model* target tracking problem is computationally infeasible as it involves a summation over all possible 'mode-histories' of the target (a tree-like branching process), so suboptimal techniques are generally used. The two most popular approaches are the *generalised pseudo-Bayes* (GPB) algorithms [10] and sequential Monte Carlo techniques. The former attempt to approximate the optimal Bayesian solution by combining those branches of the mode-history tree that share a common path for a given number of time steps in the past. The most widely used of these techniques is the *interacting multiple model* (IMM) algorithm [17], a second order pseudo-Bayes filter with the computational complexity of a first order one. For problems where the underlying dynamics are linear and Gaussian, and where each mode-history may be evaluated using the Kalman filter, the IMM can produce high quality estimates at a modest computational cost.

The most popular alternatives to the pseudo-Bayes filters are sequential Monte Carlo methods. Given their high flexibility, these techniques can be applied to just about any type of problem, including the multiple-model one. For this type of application, their computational efficiency can be further increased by means of a technique known as *Rao-Blackwellisation*. Since the main uncertainty concerns the mode of the target, if the underlying dynamics are linear and Gaussian, sampling can in fact be limited to the conditional distribution of the mode process. Kalman filters are then be used to evaluate the state distribution of each particle. Filters of this type (also known as *mixture Kalman filters* [22]) effectively approximate the optimal Bayesian solution using a scalable number of Gaussian components, and can be more computationally efficient than traditional particle filters.

Sequential Gaussian mixture techniques

An analysis of the performance of these approaches shows that there is room left for

improvement. GPB techniques can produce very good estimates, but are not computationally scalable. High-order GPB filters (i.e. 3^{rd} order or more) tend to offer little improvement over efficient 2^{nd} order filters such as the IMM. On the other hand, particle filters (and their Rao-Blackwellised variants) allow a trade-off between estimate accuracy and computational costs. While with enough particles their estimates match the accuracy of least squares estimators, the curve they produce on a computational cost vs. estimation error chart, generally shows that they incur a considerable cost before they can outperform the IMM.

The first plot in Figure 1.1 shows a schematic version of such chart (based on the results found in this thesis), where a dotted orange line, which we refer to as 'efficiency frontier', shows the best possible estimates which can be achieved for a given computational input. As illustrated, for quite a large range of computational values (which we label as 'gap from improvement'), the most accurate estimates are still provided by the IMM. This clearly leaves room for improved scalable techniques which can outperform pseudo-Bayes methods at a lower computational cost than particle filters, thus reducing (or ideally eliminating) the 'gap from improvement'. This is illustrated in the second plot of Figure 1.1.

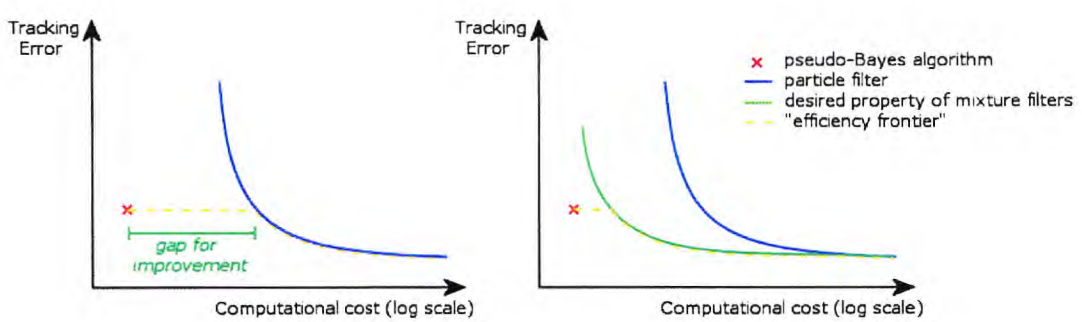


Figure 1.1: Improving the 'efficiency frontier' using mixture filters.

This thesis aims to investigate and expand on *sequential Gaussian mixture techniques* (or *mixture filters*) that can help improve computational efficiency precisely in this manner. These methods – like pseudo-Bayes filters – employ a Gaussian mixture approximation to the conditional distribution, but – like sequential Monte Carlo methods – allow full

flexibility in the choice of its complexity. Unlike Rao-Blackwellised particle filters, mixture filters evaluate the posterior distribution exactly (for a given prior) and control the number of mixture components via efficient mixture reduction methods. This recursive expansion and contraction in the complexity of the conditional distribution allows for high quality estimates at a lower cost than particle filters.

The key to achieving this performance relies on the use of efficient mixture reduction techniques. While mixture filters that evaluate the exact posterior distribution at each iteration have been proposed before ([69],[33]), the mixture reduction techniques used have, for the most part, been limited to a simple component selection on the basis of the distribution of the weights of the constituent densities. These 'weight-based' algorithms are computationally efficient and can be designed to satisfy local optimality conditions [33], but fail to take account of the spatial configuration of the individual mixture components.

One of the main contributions of this thesis is the adaptation of these mixture filters for use with a class of 'distribution-based' mixture reduction techniques ([62]-[73]), normally used for data association. These techniques are based on an analysis of the spatial configuration of the mixture components of the posterior distribution and promise greater accuracy in their approximations. A joint analysis of both weight-based and distribution-based reduction schemes is carried out to test both types of algorithm in both a multiple-model and a data association context. A new 'hybrid' method is also proposed to combine the advantages of both techniques. It is shown how these algorithms can help mixture filters to significantly reduce the 'gap' in performance between generalised pseudo-Bayes algorithms and particle filters.

Furthermore, we also modify the iteration of the mixture filter to allow for the use of the very latest measurement in the mixture reduction process. This leads to the development of a class of *look-ahead mixture filters* which, combined with the new reduction methods, can further increase computational efficiency. Improvements are most in evidence for scenarios involving intensive manoeuvring.

Extensions

The final chapters of this thesis show how the simple structure of these sequential Gaussian mixture techniques allows for their adaptation to more advanced types of problems featuring additional non-linearities.

A natural extension is the development of a *multiple measurement mixture filter* to deal with problems involving a manoeuvring target in clutter. This is effectively the combination of the multiple-model and data association problems and is an ideal testing ground for the various mixture reduction techniques. We also show how this method can outperform other popular approaches to this type of problem.

Finally, mixture filters are applied to the problem of tracking a manoeuvring target given bearings-only measurements. Here, the underlying system dynamics are nonlinear/non-Gaussian and moment-matching techniques need to be used instead of the Kalman filter to generate the Gaussian components of the mixture. The shifted Rayleigh filter is adapted to this task and a *shifted Rayleigh mixture filter* [23], which outperforms particle filters (and a related algorithm based on the unscented Kalman filter) is developed.

Chapter 2

Target Tracking Background

2.1 Traditional filtering approaches

This section provides a review of target tracking methods and outlines a number of optimal and suboptimal Bayesian filters. In Section 2.1.1 the general problem is formulated and the Bayesian solution outlined. Section 2.1.2 shows how the *Kalman filter* provides this exact solution for linear systems with additive Gaussian noise, while Sections 2.1.3 and 2.1.4 show how suboptimal approximations, namely the *extended* and the *unscented Kalman filters*, may be obtained for problems where these assumptions do not hold.

This section intends to cover the most well established filtering techniques and those most relevant to the work presented in this thesis. For a more general review of the target tracking field, including some of the latest developments, the reader can refer to [15] or [39].

2.1.1 Discrete-time dynamic models and the Bayesian approach

Target tracking involves the estimation of the dynamic state of a target given discrete-time noisy observations. In the Bayesian approach, the system is usually modelled using discrete-time state-space equations of the form:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t^s, \boldsymbol{\nu}_t^s), \tag{2.1.1}$$

$$\boldsymbol{y}_t = h(\boldsymbol{x}_t, \boldsymbol{u}_t^m, \boldsymbol{\nu}_t^m), \tag{2.1.2}$$

where, $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are the state and measurement vectors at time t, $\boldsymbol{u}_t^s$ and $\boldsymbol{u}_t^m$ are exogenous state and measurement input vectors, and $\boldsymbol{\nu}_t^s$ and $\boldsymbol{\nu}_t^m$ are independently distributed system and measurement noise processes. The presence of noise implies that the state and the observations sequences are themselves discrete-time stochastic processes, and that a probabilistic analysis is required to estimate the state.

The aim of Bayesian state estimation is the recursive evaluation of $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$, the posterior conditional distribution (or probability density function (pdf)) of the state given the set of all observations available up to time t assuming knowledge of the initial pdf of the state $p(\boldsymbol{x}_0)$. Consider the predicted pdf of the state up until time t, based on measurement information up until $t-1$. This can be written as:

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1})p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1}). \tag{2.1.3}$$

As the state is generally assumed to be a first order Markov process, $\boldsymbol{x}_{t-1}$ incorporates all the relevant statistics and information of the state from time 0 to time $t-1$. Eqn. (2.1.3) therefore simplifies to:

$$p(\boldsymbol{x}_{0:t}|, \boldsymbol{y}_{1:t-1}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1}). \tag{2.1.4}$$

This is a state prediction which needs to be updated using measurement $\boldsymbol{y}_t$ in order to produce the posterior conditional distribution. This is done by applying Bayes rule to $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$:

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) = p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_t, \boldsymbol{y}_{1:t-1}) = \frac{p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} \tag{2.1.5}$$

$$= p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1})\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})}. \tag{2.1.6}$$

This can be simplified once again by using the Markov properties of the state:

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) = p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1})\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} \tag{2.1.7}$$

$$= p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1})\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} \tag{2.1.8}$$

$$= p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1})\frac{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_t)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})}. \tag{2.1.9}$$

This formula can be used to evaluate the posterior pdf recursively given each new measurement. It is generally of more interest to evaluate the marginal or (*filtering*) posterior

conditional distribution of the state, $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, which can be similarly evaluated by using the prediction and update steps just discussed. In this case:

Prediction:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1})d\boldsymbol{x}_{0:t-1} \tag{2.1.10}$$

$$= \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})d\boldsymbol{x}_{t-1}\,. \tag{2.1.11}$$

Update:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \frac{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_t)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})}\,. \tag{2.1.12}$$

It should be noted that the prediction component $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ can be calculated using the state equation (2.1.1) and the state noise process $\boldsymbol{\nu}_t^s$ (assumed to be of known statistics). Furthermore, the denominator in (2.1.12) is a normalising term given by $\int p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_t)d\boldsymbol{x}_t$. Here, $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$ is known as the *likelihood* of the measurement and can be evaluated by means of the observation equation (2.1.2) and the measurement noise process of known statistics, $\boldsymbol{\nu}_t^m$.

Once this posterior pdf is constructed, it is then possible to obtain a minimum mean-squared error (MMSE) estimate of the state. This is $E[\boldsymbol{x}_t|\boldsymbol{y}_{1:t}]$, the conditional expectation of $\boldsymbol{x}_t$ given measurements $\boldsymbol{y}_{1:t}$, which is given by:

$$E[\boldsymbol{x}_t|\boldsymbol{y}_{1:t}] = \int \boldsymbol{x}_t p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})d\boldsymbol{x}\,. \tag{2.1.13}$$

Unfortunately, this exact evaluation is not usually possible due to the intractability of the integrals involved. The main exception is the special case in which the state-space equations are linear and the state and measurement processes Gaussian. In this case the Kalman filter is the optimal Bayesian estimator.

2.1.2 The Kalman filter

The Kalman filter, first introduced by R. E. Kalman [44] in 1960, is a Bayesian estimator which provides an optimal solution to the state estimation problem under strict conditions on the state and measurement processes.

The fundamental requirement for the system state process, $\boldsymbol{x}_t$, is that its dynamics

should be governed by known linear system equations driven by a known exogenous input, $\boldsymbol{u}_t^s$, and an additive zero-mean Gaussian noise process, $\boldsymbol{\nu}_t^s$, with known covariance, $\boldsymbol{Q}_t^s$. The initial state is assumed to be unknown and is modelled as a Gaussian random variable with a known mean, $\hat{\boldsymbol{x}}_0$, and a covariance, $\boldsymbol{P}_0$.

The observable measurement process, $\boldsymbol{y}_t$, should also be a known linear function of the state, which should be driven by a known exogenous measurement input $\boldsymbol{u}_t^m$ and by an additive zero-mean Gaussian noise process, $\boldsymbol{\nu}_t^m$, with known covariance, $\boldsymbol{Q}_t^m$. The noise processes and the initial estimates are all assumed to be mutually uncorrelated.

The system just described is a special case of the set of nonlinear equations describing the state and observation processes introduced in (2.1.1)-(2.1.2). It can be summarised as:

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_t^s)^{1/2} \boldsymbol{\nu}_t^s , \tag{2.1.14}$$

$$\boldsymbol{y}_t = \boldsymbol{H}_t \boldsymbol{x}_t + \boldsymbol{u}_t^m + (\boldsymbol{Q}_t^m)^{1/2} \boldsymbol{\nu}_t^m , \tag{2.1.15}$$

where:

$\boldsymbol{x}_t \sim R$-dimensional state process vector,

$\boldsymbol{y}_t \sim L$-dimensional measurement process vector,

$\boldsymbol{\nu}_t^s \sim \mathcal{N}(0, \boldsymbol{I}_{R\times R})$, R-dimensional state noise process vector,

$\boldsymbol{\nu}_t^m \sim \mathcal{N}(0, \boldsymbol{I}_{L\times L})$, L-dimensional measurement noise process vector,

and where:

$\boldsymbol{F}_t \sim R \times R$ system matrix,

$\boldsymbol{Q}_t^s \sim R \times R$ state noise covariance matrix,

$\boldsymbol{H}_t \sim L \times R$ measurement matrix,

$\boldsymbol{Q}_t^m \sim L \times L$ measurement covariance matrix,

$\boldsymbol{u}_t^s \sim R$ dimensional exogenous state input vector,

$\boldsymbol{u}_t^m \sim L$ dimensional exogenous measurement input vector.

In the following subsections, we derive and explain the Kalman filter equations following the presentation of Bar-Shalom and Li [10], and we then prove how the filter minimises

the estimation variance. More detailed reviews of the filter can be found in Sorenson [66], Jazwinski [40], Maybeck [52] and Anderson and Moore [5].

a) The structure of the Kalman filter

Being a recursive Bayesian estimator of the form (2.1.10)-(2.1.12), the Kalman filter involves a prediction and an update stage. These steps can be summarised in a few key equations that, for a given prior pdf, evaluate the first and second moments of the posterior conditional distribution. We now derive these equations following the approach of [10].

The prediction stage

The first step of the Kalman filter is the evaluation of $\hat{\boldsymbol{x}}_{t|t-1}$, the *predicted state estimate* at time t based on $\hat{\boldsymbol{x}}_{t-1|t-1} = E[\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}]$, the prior updated state estimate at time $t-1$. Using (2.1.14), and noting that the system noise is white and zero-mean, this prediction can be written as:

$$\begin{aligned}\hat{\boldsymbol{x}}_{t|t-1} &= E[\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}] \\ &= \boldsymbol{F}_t\hat{\boldsymbol{x}}_{t-1|t-1} + \boldsymbol{u}_t^s\,. \end{aligned} \tag{2.1.16}$$

The *state prediction error* (which is required to calculate the state prediction covariance) is then:

$$\begin{aligned}\tilde{\boldsymbol{x}}_{t|t-1} &= \boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1} && (2.1.17)\\ &= (\boldsymbol{F}_t\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_t^s)^{1/2}\boldsymbol{\nu}_t^s) - (\boldsymbol{F}_t\hat{\boldsymbol{x}}_{t-1|t-1} + \boldsymbol{u}_t^s) \\ &= \boldsymbol{F}_t(\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1|t-1}) + \boldsymbol{u}_t^s + (\boldsymbol{Q}_t^s)^{1/2}\boldsymbol{\nu}_t^s - \boldsymbol{u}_t^s \\ &= \boldsymbol{F}_t\tilde{\boldsymbol{x}}_{t-1|t-1} + (\boldsymbol{Q}_t^s)^{1/2}\boldsymbol{\nu}_t^s\,, && (2.1.18)\end{aligned}$$

where $\tilde{\boldsymbol{x}}_{t-1|t-1} = \boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1|t-1}$ is the *estimation error* associated with the prior estimate. Bearing in mind that the prior estimate at $t-1$ is uncorrelated with the system noise at t, and that $\boldsymbol{\nu}_t^s \sim \mathcal{N}(0, \boldsymbol{I}_{R\times R})$, it is now possible to evaluate the *state prediction covariance*, $\boldsymbol{P}_{t|t-1}$, as follows:

$$\begin{aligned}\boldsymbol{P}_{t|t-1} &= E[(\tilde{\boldsymbol{x}}_{t|t-1})(\tilde{\boldsymbol{x}}_{t|t-1})^{\mathrm{T}}|\boldsymbol{y}_{1:t-1}] \\ &= \boldsymbol{F}_t\,E[(\tilde{\boldsymbol{x}}_{t-1|t-1})(\tilde{\boldsymbol{x}}_{t-1|t-1})^{\mathrm{T}}|\boldsymbol{y}_{1:t-1}]\boldsymbol{F}_t^{\mathrm{T}} + E[((\boldsymbol{Q}_t^s)^{1/2}\boldsymbol{\nu}_t^s)((\boldsymbol{Q}_t^s)^{1/2}\boldsymbol{\nu}_t^s)^{\mathrm{T}}] \\ &= \boldsymbol{F}_t\boldsymbol{P}_{t-1|t-1}\boldsymbol{F}_t^T + \boldsymbol{Q}_t^s\,, \end{aligned} \tag{2.1.19}$$

where $\boldsymbol{P}_{t-1|t-1}$ is the covariance of the prior estimate. The Kalman filter next uses the measurement equation to generate the *measurement prediction*, $\hat{\boldsymbol{y}}_{t|t-1}$, and the *measurement prediction covariance*, $\boldsymbol{S}_t$, (also known as the *innovation covariance*) by means of the *measurement prediction error* (or *innovation*), $\tilde{\boldsymbol{y}}_t$:

$$\hat{\boldsymbol{y}}_{t|t-1} = E[\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}] = \boldsymbol{H}_t\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{u}_t^m\,, \tag{2.1.20}$$

$$\begin{aligned}\tilde{\boldsymbol{y}}_t &= \boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} &(2.1.21)\\ &= \boldsymbol{H}_t(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}) + \boldsymbol{u}_t^m + (\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^o - \boldsymbol{u}_t^m\\ &= \boldsymbol{H}_t\tilde{\boldsymbol{x}}_{t|t-1} + (\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^o\,, &(2.1.22)\end{aligned}$$

$$\begin{aligned}\boldsymbol{S}_t &= E[(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1})(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1})^T|\boldsymbol{y}_{1:t-1}]\\ &= \boldsymbol{H}_t\,E[(\tilde{\boldsymbol{y}}_t)(\tilde{\boldsymbol{y}}_t)^{\mathrm{T}}|\boldsymbol{y}_{1:t-1}]\boldsymbol{H}_t^{\mathrm{T}} + E[((\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^m)((\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^m)^{\mathrm{T}}]\\ &= \boldsymbol{H}_t\boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T + \boldsymbol{Q}_t^m\,. &(2.1.23)\end{aligned}$$

Up to this stage, we have only used the statistics of the measurement $\boldsymbol{y}_t$, not its actual value.

The update stage

Having completed the prediction stage, the Kalman filter is now ready to update its predictions in the light of $\boldsymbol{y}_t$, which is now assumed to be available. The first step of the update stage is the evaluation of the innovation vector $\tilde{\boldsymbol{y}}_t$, given by (2.1.21). Secondly, the Kalman filter calculates a matrix known as *Kalman gain*, $\boldsymbol{K}_t$, which is used to scale the innovation when the state prediction is later adjusted. $\boldsymbol{K}_t$ is calculated as:

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\,\boldsymbol{H}_t^{\mathrm{T}}\boldsymbol{S}_t^{-1}\,. \tag{2.1.24}$$

As shown in Theorem 2.1.1 below, this choice of $\boldsymbol{K}_t$ minimises the error variance of the updated estimates.

Finally, having calculated the innovation and the Kalman gain, the filter updates the state prediction and its covariance. The resulting *updated state estimate*, $\hat{\boldsymbol{x}}_{t|t}$, and *up-*

dated state covariance, $\boldsymbol{P}_{t|t}$, are calculated as follows:

$$\hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t \,, \tag{2.1.25}$$

$$\boldsymbol{P}_{t|t} = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{H}_t \boldsymbol{P}_{t|t-1} \,. \tag{2.1.26}$$

b) The Kalman Filter as the linear MMSE estimator

The theorem presented in this section (largely based on [40] and [67]) asserts that the Kalman Filter is the linear minimum mean square error (MMSE) estimator under the Gaussian assumption for the prior distribution and the additive noise processes. Note that, for simplicity of notation, variables which are conditioned on the same time frame no longer show their conditioning (ex. $\hat{\boldsymbol{x}}_{t|t}$ is now simply $\hat{\boldsymbol{x}}_t$). Also the system transition matrices and noise covariance matrices are assumed to be time-invariant (except where otherwise stated) and the time index is dropped (for example, $\boldsymbol{F}_t = \boldsymbol{F}$, and $\boldsymbol{Q}_t^s = \boldsymbol{Q}_s$).

Theorem 2.1.1 *For the linear Gaussian state-space model of (2.1.14)-(2.1.15):*

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s \,, \tag{2.1.27}$$

$$\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m \,, \tag{2.1.28}$$

the MMSE estimate, $\hat{\boldsymbol{x}}_t$, and the error covariance matrix, $\boldsymbol{P}_t$, under the assumption that $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}) \sim \mathcal{N}(\boldsymbol{x}_{t-1}; \hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1})$ are given by:

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t \left(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} \right) \,, \tag{2.1.29}$$

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}) \boldsymbol{P}_{t|t-1} \,. \tag{2.1.30}$$

Here $\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\boldsymbol{H}^T \boldsymbol{S}_t^{-1}$ is the Kalman gain, and $\hat{\boldsymbol{x}}_{t|t-1}$, $\boldsymbol{P}_{t|t-1}$, $\hat{\boldsymbol{y}}_{t|t-1}$ and $\boldsymbol{S}_t$ are calculated using Eqns. (2.1.16), (2.1.19), (2.1.20) and (2.1.23) respectively.

Proof. Let the estimation error at time t be given by

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \,.$$

Then, if $\mathrm{tr}[\cdot]$ defines the trace of a matrix, the mean square error, ε_t, can be written as:

$$\begin{aligned} \varepsilon_t &= \mathrm{tr}\left[E[\tilde{\boldsymbol{x}}_t \tilde{\boldsymbol{x}}_t^T | \boldsymbol{y}_{1:t}] \right] \\ &= \mathrm{tr}\left[E[(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)^T | \boldsymbol{y}_{1:t}] \right] = \mathrm{tr}[\boldsymbol{P}_t] \,. \end{aligned} \tag{2.1.31}$$

From (2.1.29), we assume that $\hat{\boldsymbol{x}}_t$ is given by:

$$\begin{aligned}\hat{\boldsymbol{x}}_t &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\left(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1}\right)\\ &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m - \boldsymbol{H}(\boldsymbol{F}\hat{\boldsymbol{x}}_{t-1} + \boldsymbol{u}_t^s) - \boldsymbol{u}_t^m)\\ &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m - \boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1})\,. \qquad (2.1.32)\end{aligned}$$

Therefore the error covariance matrix can be re-written as:

$$\begin{aligned}\boldsymbol{P}_t &= E[\tilde{\boldsymbol{x}}_t\tilde{\boldsymbol{x}}_t^T|\boldsymbol{y}_{1:t}]\\ &= E[(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t(\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m - \boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1}))\\ &\qquad (\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t(\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m - \boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1}))^T|\boldsymbol{y}_{1:t}] \qquad (2.1.33)\\ &= E[((\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}) - \boldsymbol{K}_t\boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m)\\ &\qquad ((\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}) - \boldsymbol{K}_t\boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m)^T|\boldsymbol{y}_{1:t}]\,. \qquad (2.1.34)\end{aligned}$$

Since the prediction error $(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1})$ is uncorrelated with the measurement noise $\boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m$, the covariance simplifies to:

$$\begin{aligned}\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})&E[(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1})^T|\boldsymbol{y}_{1:t}](\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})^T\\ &+ \boldsymbol{K}_t E[(\boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m)(\boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m)^T]\boldsymbol{K}_t^T\,. \qquad (2.1.35)\end{aligned}$$

Finally, as $\boldsymbol{\nu}_t^m \sim \mathcal{N}(0, \boldsymbol{I})$:

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})\boldsymbol{P}_{t|t-1}(\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H})^T + \boldsymbol{K}_t\boldsymbol{Q}_m\boldsymbol{K}_t^T\,. \qquad (2.1.36)$$

The desired value of $\boldsymbol{K}_t$ is the one which minimises the mean square error, and for which:

$$\frac{d\varepsilon_t}{d\boldsymbol{K}_t} = \frac{d\mathrm{tr}[\boldsymbol{P}_t]}{d\boldsymbol{K}_t} = 0\,.$$

As the trace of a matrix is equal to the trace of its transpose, this can be written as:

$$\begin{aligned}\frac{d\mathrm{tr}[\boldsymbol{P}_t]}{d\boldsymbol{K}_t} &= \frac{d(\mathrm{tr}[\boldsymbol{P}_{t|t-1}] - 2\mathrm{tr}[\boldsymbol{K}_t\boldsymbol{H}\boldsymbol{P}_{t|t-1}] + \mathrm{tr}[\boldsymbol{K}_t\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T\boldsymbol{K}_t^T] + \mathrm{tr}[\boldsymbol{K}_t\boldsymbol{Q}_m\boldsymbol{K}_t^T])}{d\boldsymbol{K}_t}\\ &= -2(\boldsymbol{H}\boldsymbol{P}_{t|t-1}) + 2\boldsymbol{K}_t(\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T + \boldsymbol{Q}_m) = 0\,. \qquad (2.1.37)\end{aligned}$$

Noting that $(\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T + \boldsymbol{Q}_m)$ is the measurement prediction covariance $\boldsymbol{S}_t$, the value of $\boldsymbol{K}_t$ which minimises the error variance is exactly the Kalman gain:

$$\begin{aligned}\boldsymbol{K}_t &= \boldsymbol{P}_{t|t-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T + \boldsymbol{Q}_m)^{-1} \qquad (2.1.38)\\ &= \boldsymbol{P}_{t|t-1}\boldsymbol{H}^T\boldsymbol{S}_t^{-1}\,. \qquad (2.1.39)\end{aligned}$$

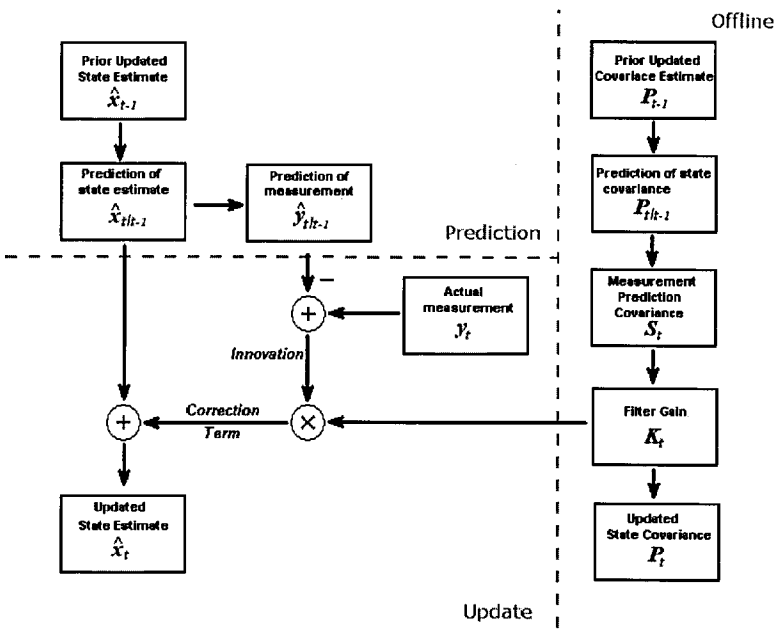


Figure 2.1: The structure of the Kalman filter.

c) Summary of the Kalman filter equations

The Kalman Filter is now concisely summarised for future reference (its structure is also illustrated in Figure 2.1).

PREDICTION STAGE:

$$\hat{\boldsymbol{x}}_{t|t-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{t-1} + \boldsymbol{u}_t^s \tag{2.1.40}$$

$$\boldsymbol{P}_{t|t-1} = \boldsymbol{F}\boldsymbol{P}_{t-1}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{Q}_s \quad (*) \tag{2.1.41}$$

$$\hat{\boldsymbol{y}}_{t|t-1} = \boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{u}_t^m \tag{2.1.42}$$

$$\boldsymbol{S}_t = \boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{Q}_m \quad (*) \tag{2.1.43}$$

UPDATE STAGE:

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} \tag{2.1.44}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{S}_t^{-1} \quad (*) \tag{2.1.45}$$

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\tilde{\boldsymbol{y}}_t \tag{2.1.46}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{H}\boldsymbol{P}_{t|t-1} \quad (*) \tag{2.1.47}$$

Equations marked by an asterisk can be pre-computed offline.

2.1.3 The extended Kalman filter

The *extended Kalman filter* (EKF) is a popular extension of the Kalman filter which enables it to cope with general non-linear state-space models of the form:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t^s) + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s, \tag{2.1.48}$$

$$\boldsymbol{y}_t = h(\boldsymbol{x}_t) + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^o. \tag{2.1.49}$$

The key principle of the EKF, as outlined by Bar-Shalom *et al.* [12], consists in calculating state and measurement predictions (and their covariances) using locally linearised versions of these equations. The approach makes use of a Taylor series expansion around the latest estimate, $\hat{\boldsymbol{x}}_{t-1}$, in which only the first or the first and second order terms are used (this leads to first- or second-order EKF respectively). This linearisation involves the calculation of the Jacobians (and of the Hessians for the second-order EKF) of the state and measurement equations. Besides increasing the complexity of the algorithm, this has also the effect that the covariance computations are now coupled with state estimation and must be performed online.

More serious limitations of the EKF can however arise because of the use of the Taylor series expansions as an approximation to the nonlinear equation. According to Bar-Shalom *et al.* [12], the fact that higher-order terms are neglected can give rise to unmodelled errors which violate the filter's assumptions and, accumulating over time, cause the filter's actual mean square errors to diverge from the filter's computed variances. The EKF is, generally speaking, an effective algorithm only in relatively benign scenarios without the presence of severe nonlinearities and high-intensity noise, and in which the initial estimates are reasonably accurate.

A wide variety of approaches have been proposed to compensate for these limitations. The use of second order terms is generally preferable to the more popular first order filter, but even this extra complexity may not guarantee the suppression of the unmodelled errors. In [34], Fitzgerald also suggests the use of an artificial increase in the state and measurement noise covariances $\boldsymbol{Q}_s$ and $\boldsymbol{Q}_m$.

A more refined (if more computationally demanding) approach is known as the *iter-*

ated extended Kalman filter (see Blackman [15], Jazwinski [40] or Kerr [47]), in which the Taylor approximations are adjusted iteratively.

For specific nonlinear problems such as bearings-only tracking, algorithms have also been proposed in which banks of EKFs are run in parallel with different initial configurations, in order to reduce the estimation error. These algorithms, referred to as the *range-parameterised EKFs* [57], are further discussed in Section 5.1.

a) The structure of the extended Kalman filter

A derivation of the first order EKF and an outline of the algorithm are now presented following Bar-Shalom *et al.* [12]. More detail on the EKF (including derivations for the second order filter) can be found in [12]; alternatively the reader may also refer to filtering books such as Jazwinski [40] or Anderson & Moore [5].

Prediction Stage

The EKF generates state predictions by locally linearising the system equations about the prior estimate $\hat{\boldsymbol{x}}_{t-1} = E[\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}]$ and covariance $P_{t-1} = E[(\tilde{\boldsymbol{x}}_{t-1})(\tilde{\boldsymbol{x}}_{t-1})^T|\boldsymbol{y}_{1:t-1}]$ (where $\tilde{\boldsymbol{x}} = \boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1}$ is the prior estimation error). This is done by calculating the Taylor series expansion of the state equation of (2.1.48) up to the first-order term (for simplicity we here omit $\boldsymbol{u}_t^s$):

$$\boldsymbol{x}_t = f(\hat{\boldsymbol{x}}_{t-1}) + \boldsymbol{f}_{\boldsymbol{x}} \cdot (\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1}) + \text{H.O.T.} + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s. \tag{2.1.50}$$

Here H.O.T represents higher-order terms of the expansion (which are neglected in the first-order EKF and will henceforth be ignored) and where

$$\boldsymbol{f}_{\boldsymbol{x}} = \left.\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{t-1}} \tag{2.1.51}$$

is the Jacobian of $f(\boldsymbol{x})$ evaluated at $\boldsymbol{x} = \hat{\boldsymbol{x}}_{t-1}$ (the prior state estimate). Taking conditional expectations with respect to $\boldsymbol{y}_{1:t-1}$ across (2.1.50) and ignoring higher order terms, we arrive at:

$$\begin{aligned}\hat{\boldsymbol{x}}_{t|t-1} &= E[\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}] \\ &= f(\hat{\boldsymbol{x}}_{t-1}).\end{aligned} \tag{2.1.52}$$

The predicted state covariance can be similarly derived. Let the prediction error be $\tilde{\boldsymbol{x}}_{t|t-1} = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}$. Using (2.1.50) and (2.1.52) this can be rewritten as:

$$\begin{aligned}\tilde{\boldsymbol{x}}_{t|t-1} &= f(\hat{\boldsymbol{x}}_{t-1}) + \boldsymbol{f}_{\boldsymbol{x}} \cdot (\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1}) + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s - f(\hat{\boldsymbol{x}}_{t-1}) \\ &= \boldsymbol{f}_{\boldsymbol{x}} \cdot (\hat{\boldsymbol{x}}_{t-1} - \boldsymbol{x}_{t-1}) + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s = \boldsymbol{f}_{\boldsymbol{x}}\tilde{\boldsymbol{x}}_{t-1} + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s. \end{aligned} \tag{2.1.53}$$

As the prediction error and the system noise are not correlated, the predicted state covariance is given by:

$$\begin{aligned}\boldsymbol{P}_{t|t-1} &= E[(\tilde{\boldsymbol{x}}_{t|t-1})(\tilde{\boldsymbol{x}}_{t|t-1})^T|\boldsymbol{y}_{1:t-1}] \\ &= E[(\boldsymbol{f}_{\boldsymbol{x}}\tilde{\boldsymbol{x}}_{t-1} + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s)(\boldsymbol{f}_{\boldsymbol{x}}\tilde{\boldsymbol{x}}_{t-1} + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s)^T|\boldsymbol{y}_{1:t-1}] \\ &= \boldsymbol{f}_{\boldsymbol{x}}\boldsymbol{P}_{t-1}\boldsymbol{f}_{\boldsymbol{x}}^T + \boldsymbol{Q}_s\,. \end{aligned} \tag{2.1.54}$$

The same procedure is then applied to generate the measurement prediction and its covariance. These can therefore be written as:

$$\begin{aligned}\hat{\boldsymbol{y}}_{t|t-1} &= E[\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}] \\ &= h(\hat{\boldsymbol{x}}_{t|t-1})\,, \end{aligned} \tag{2.1.55}$$

and

$$\begin{aligned}\boldsymbol{S}_t &= E[(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1})(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1})^T|\boldsymbol{y}_{1:t-1}] \\ &= \boldsymbol{h}_{\boldsymbol{x}}\boldsymbol{P}_{t|t-1}\boldsymbol{h}_{\boldsymbol{x}}^T + \boldsymbol{Q}_m\,, \end{aligned} \tag{2.1.56}$$

where

$$\boldsymbol{h}_{\boldsymbol{x}} = \left.\frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{t|t-1}}. \tag{2.1.57}$$

Update Stage

The update stage of the extended Kalman filter is identical to that of the linear Kalman Filter presented in Section 2.1.2, where linear matrices $\boldsymbol{H}$ and $\boldsymbol{F}$ are now replaced by the Jacobians, $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{h}_{\boldsymbol{x}}$, of the nonlinear state and measurement functions.

b) Summary of the extended Kalman filter equations

The extended Kalman filter equations are now concisely summarised for future reference.

PREDICTION STAGE:

$$\hat{\boldsymbol{x}}_{t|t-1} = f(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t^s) \tag{2.1.58}$$

$$\boldsymbol{P}_{t|t-1} = \boldsymbol{f}_x \boldsymbol{P}_{t-1} \boldsymbol{f}_x^{\mathrm{T}} + \boldsymbol{Q}_s \tag{2.1.59}$$

$$\hat{\boldsymbol{y}}_{t|t-1} = h(\hat{\boldsymbol{x}}_{t|t-1}, \boldsymbol{u}_t^m) \tag{2.1.60}$$

$$\boldsymbol{S}_t = \boldsymbol{h}_x \boldsymbol{P}_{t|t-1} \boldsymbol{h}_x^{\mathrm{T}} + \boldsymbol{Q}_m \tag{2.1.61}$$

UPDATE STAGE:

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} \tag{2.1.62}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\, \boldsymbol{h}_x^{\mathrm{T}} \boldsymbol{S}_t^{-1} \tag{2.1.63}$$

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t \tilde{\boldsymbol{y}}_t \tag{2.1.64}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{h}_x \boldsymbol{P}_{t|t-1} \tag{2.1.65}$$

where:

$$\boldsymbol{f}_x = \left. \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{t-1}}, \tag{2.1.66}$$

$$\boldsymbol{h}_x = \left. \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{t|t-1}}. \tag{2.1.67}$$

c) The iterated extended Kalman filter

The *iterated extended Kalman filter* (IEKF) is a refinement to the standard first-order EKF algorithm just presented, which is better suited to dealing with difficult problems involving highly nonlinear measurement equations for which the EKF might diverge. In the IEKF, the EKF update equations (2.1.64) and (2.1.65) are replaced by an iteration of the form:

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^l, \tag{2.1.68}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t^l \boldsymbol{h}_x(l) \boldsymbol{P}_{t|t-1}, \tag{2.1.69}$$

where, for $i = 0, 1, \ldots, l-1$:

$$\hat{\boldsymbol{x}}_t^{i+1} = \hat{\boldsymbol{x}}_{t|t-1}^0 + \boldsymbol{K}_t^i \tilde{\boldsymbol{y}}_t^i, \tag{2.1.70}$$

$$\tilde{\boldsymbol{y}}_t^i = \boldsymbol{y}_t - h(\hat{\boldsymbol{x}}_{t|t-1}^i, \boldsymbol{u}_t^m) - \boldsymbol{h}_{\boldsymbol{x}}(i)(\hat{\boldsymbol{x}}_{t|t-1}^0 - \hat{\boldsymbol{x}}_{t|t-1}^i), \tag{2.1.71}$$

$$\boldsymbol{K}_t^i = \boldsymbol{P}_{t|t-1}\boldsymbol{h}_{\boldsymbol{x}}(i)(\boldsymbol{h}_{\boldsymbol{x}}(i)\boldsymbol{P}_{t|t-1}(\boldsymbol{h}_{\boldsymbol{x}}(i))^T + \boldsymbol{Q}_m), \tag{2.1.72}$$

$$\boldsymbol{h}_{\boldsymbol{x}}(i) = \left.\frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{t|t-1}^i}, \tag{2.1.73}$$

$$\hat{\boldsymbol{x}}_{t|t-1}^0 = f(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t^s). \tag{2.1.74}$$

This iteration is normally repeated until the state estimates no longer change, or alternatively l can be chosen arbitrarily. According to Wishner *et al.* [74], a single-stage iteration is often sufficient to significantly improve the error and to outperform both first-order and second-order EKFs. Note that, for linear Gaussian problems, the iterations in each step of this algorithm generate the original estimate and error covariance. However the iterations can improve the estimate in other cases.

2.1.4 The unscented Kalman filter

The *unscented Kalman filter (UKF)*, first proposed by Julier and Uhlmann [42]-[43], is an alternative filter for the state estimation of nonlinear/non-Gaussian systems of the form (2.1.48)-(2.1.49), which matches the first and second order moments of the estimate to those of the true distribution. This is done by a deterministic sampling approach in which a number of sample points (normally referred to as σ-points) are carefully selected and weighted so as to provide a discrete approximation to $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$. As these σ-points are then passed through the nonlinear dynamics of the systems, they are transformed so as to reflect the posterior mean and covariance up to the 3rd order Taylor series expansion [70]. A Gaussian distribution is then fitted to these points, to match the first and second order moments of the posterior distribution.

Julier and Uhlmann [41],[42],[43] and Wan and Van der Merwe [70] have shown how the UKF generally outperforms the EKF while retaining similar computational complexity.

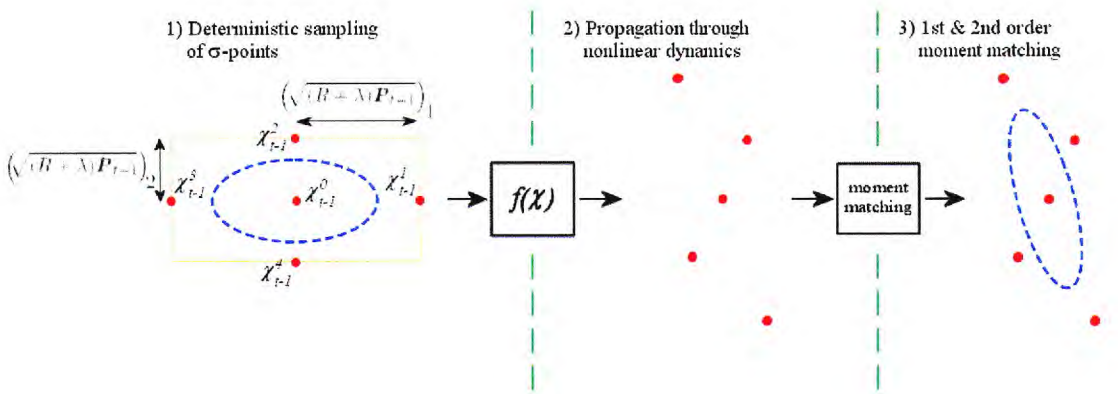


Figure 2.2: The σ-points used by the unscented Kalman filter.

a) The structure of the unscented Kalman filter

In this section, the unscented Kalman filter algorithm is presented and explained for nonlinear equations of the form (2.1.48)-(2.1.49). For variations of the algorithm applying to different combinations of linear/nonlinear state and measurement equations we refer to [51].

The first step in the UKF recursion is the deterministic generation of the σ-points ($\boldsymbol{\chi}_{t-1}$) which are to be propagated through the nonlinear dynamics. As shown in Figure 2.2 (for $R = 2$), these are normally placed in the middle and at the centre of the faces of an R-dimensional hypercube. Here R, is the dimension of the state space. Hence the number of σ-points used is $2R + 1$. Alternatives have been proposed in which these points are placed at the corners of a hypercube or at the corners of a simplex [51].
For the more traditional form of the algorithm, the initial position of the σ-points is obtained using the following equations:

$$\boldsymbol{\chi}^0_{t-1} = \hat{\boldsymbol{x}}_{t-1}\,, \tag{2.1.75}$$

$$\boldsymbol{\chi}^i_{t-1} = \hat{\boldsymbol{x}}_{t-1} + \left(\sqrt{(R+\lambda)\boldsymbol{P}_{t-1}}\right)_{i^{\text{th}}\text{ col.}} \tag{2.1.76}$$

$$\text{for } i = 1, \ldots, R\,,$$

$$\boldsymbol{\chi}^i_{t-1} = \hat{\boldsymbol{x}}_{t-1} - \left(\sqrt{(R+\lambda)\boldsymbol{P}_{t-1}}\right)_{(i-R)^{\text{th}}\text{ col.}} \tag{2.1.77}$$

$$\text{for } i = R+1, \ldots, 2R\,,$$

where $\left(\sqrt{(R+\lambda)\boldsymbol{P}_{t-1}}\right)_{i^{\text{th}}\text{ col.}}$ refers to the i^{th}-column of the (matrix) square root of $(R+\lambda)\boldsymbol{P}_{t-1}$. λ is a design parameter which allows the filter to be adjusted heuristically in order to reflect higher order moments. For Gaussian distributions, Julier and Uhlmann [43] recommend a choice of $\lambda + R = 3$. The weighting of the σ-points is also determined by λ, and is normally given by:

$$\begin{aligned} w_0 &= \lambda/(R+\lambda)\,, \\ w_i &= 1/(R+\lambda) \qquad\qquad \text{for } i = 1,\ldots,2R\,. \end{aligned}$$

Wan and Van der Merwe [70] suggest a slightly different weighting for the covariance calculations of the central component ($\boldsymbol{\chi}^0_{t-1}$) in order to further refine the accuracy of third and higher moments.

Once the points are generated, they are propagated and updated using techniques reminiscent of the Kalman filter. Firstly the state prediction $\hat{\boldsymbol{x}}_{t|t-1}$ and covariance $\boldsymbol{P}_{t|t-1}$ are calculated about the predictions generated by the σ-points:

$$\boldsymbol{\chi}^i_{t|t-1} = f(\boldsymbol{\chi}^i_{t-1}, u^s_t) \qquad \text{for } i = 0,\ldots,2R, \tag{2.1.78}$$

$$\hat{\boldsymbol{x}}_{t|t-1} = \sum_{i=0}^{2R} w_i \boldsymbol{\chi}^i_{t|t-1}\,, \tag{2.1.79}$$

$$\boldsymbol{P}_{t|t-1} = \left[\sum_{i=0}^{2R} w_i \left(\boldsymbol{\chi}^i_{t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)\left(\boldsymbol{\chi}^i_{t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)^T\right] + \boldsymbol{Q}_s\,. \tag{2.1.80}$$

Similar equations are then used to calculate the measurement prediction, $\hat{\boldsymbol{x}}_{t|t-1}$, and the innovation covariance, $\boldsymbol{S}_t$:

$$\boldsymbol{\zeta}^i_{t|t-1} = h(\boldsymbol{\chi}^i_{t|t-1}, u^m_t) \qquad \text{for } i = 0,\ldots,2R, \tag{2.1.81}$$

$$\hat{\boldsymbol{y}}_{t|t-1} = \sum_{i=0}^{2R} w_i \boldsymbol{\zeta}^i_{t|t-1}\,, \tag{2.1.82}$$

$$\boldsymbol{S}_t = \left[\sum_{i=0}^{2R} w_i \left(\boldsymbol{\zeta}^i_{t|t-1} - \hat{\boldsymbol{y}}_{t|t-1}\right)\left(\boldsymbol{\zeta}^i_{t|t-1} - \hat{\boldsymbol{y}}_{t|t-1}\right)^T\right] + \boldsymbol{Q}_m\,. \tag{2.1.83}$$

Next the UKF calculates the cross-correlation matrix $\boldsymbol{P}^{xy}_{t|t-1}$ and the filter gain:

$$\boldsymbol{P}^{xy}_{t|t-1} = \sum_{i=0}^{2R} w_i \left(\boldsymbol{\chi}^i_{t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)\left(\boldsymbol{\zeta}^i_{t|t-1} - \hat{\boldsymbol{y}}_{t|t-1}\right)^T, \tag{2.1.84}$$

$$\boldsymbol{K}_t = \boldsymbol{P}^{xy}_{t|t-1}\left(\boldsymbol{S}_t\right)^{-1}. \tag{2.1.85}$$

In the final update stage, the mean and covariance, $\hat{\boldsymbol{x}}_t$ and $\boldsymbol{P}_t$, of a Gaussian approximation to the posterior distribution are calculated according to the following equations:

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t \left(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} \right) , \tag{2.1.86}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{S}_t \boldsymbol{K}_t^T . \tag{2.1.87}$$

2.2 Sequential Monte Carlo methods

The algorithms just described attempt to directly evaluate (or approximate) the moments of the posterior conditional distribution of $\boldsymbol{x}_t$ using analytical expressions. The techniques described in this section, generally referred to as *sequential Monte Carlo (MC) methods* (or *particle filters*), take a different approach, and attempt to approximate the posterior distribution of the target state by means of the simulation-based Monte Carlo integration method.

The key idea is to approximate the posterior conditional density of the state by a discrete distribution, obtained by sampling from this density ('constructing an empirical distribution'). As this is not generally possible to do directly, these methods typically make use of an *importance sampling distribution*, from which samples may be easily drawn. By choosing this distribution to be the prior distribution, the algorithm can be made recursive. Clearly the quality of the approximations is dependent on the number of samples used, N. It can be shown that as $N \rightarrow \infty$, the moments calculated from the empirical distribution converge to the moments of the true conditional distribution.

In practice, the simplest form of sequential Monte Carlo technique, known as the *bootstrap filter* ([28],[37]), consists of two steps: a *sequential importance sampling* step and a *resampling* step. The former is the key feature of the bootstrap filter. It involves drawing N samples from the prior distribution, which takes the role of importance distribution, and then evaluating their importance weights. To avoid degeneration of the empirical density (a situation in which only one particle has non-zero weight after a few iterations), a resampling stage is introduced to propagate high likelihood samples and discard low likelihood ones.

In Section 2.2.1, we discuss sequential importance sampling and in Section 2.2.2 we discuss the resampling techniques necessary to avoid particle degeneracy. A concise outline of the bootstrap filter follows in Section 2.2.3. The exposition in these sections follows [28], where additional details and an extensive review of further developments and applications of particle filters may be found. For information on more advanced particle filter developments, such as the *auxiliary particle filter*, the reader is referred to [58].

2.2.1 Sequential importance sampling

Particle filters are based on the Monte Carlo integration concept according to which, given N independently and identically distributed (i.i.d.) samples (or particles) from the posterior conditional distribution $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$, it is possible to generate an empirical approximation of this distribution given by:

$$\hat{p}(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\hat{\boldsymbol{x}}_{0:t}^{i})\,.$$

Here $\delta(\hat{\boldsymbol{x}}_{0:t}^{i})$ denotes the Dirac delta mass located at $\hat{\boldsymbol{x}}_{0:t}^{i}$. Define $E_{\hat{p}(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[\boldsymbol{x}_{0:t}]$ to be the estimate of $\boldsymbol{x}_{0:t}$ based on this distribution. Then this is simply:

$$\begin{aligned} E_{\hat{p}(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}] &= \int \boldsymbol{x}_{0:t}\hat{p}(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) \\ &= \frac{1}{N}\sum_{i=1}^{N}\hat{\boldsymbol{x}}_{0:t}^{i}\,. \end{aligned} \tag{2.2.1}$$

This estimate is unbiased and, in consequence of the strong law of large numbers, converges to the exact estimate $E_{p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[\boldsymbol{x}_{0:t}]$ as the number of samples N tends to infinity.

As it is generally not possible to sample directly from $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$ without making use of iterative algorithms such as *Markov chain Monte Carlo* (MCMC) methods [36] not well suited for inclusion in a recursive algorithm, an alternative technique known as *importance sampling* ([14],[35]) can be used. This involves drawing samples from an *importance sampling distribution*, $\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$ with the property that, if $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) > 0$, then $\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) > 0$. In this case:

$$E_{p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[\boldsymbol{x}_{0:t}] = \frac{E_{\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[\boldsymbol{x}_{0:t}w_{0:t}]}{E_{\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}[w_{0:t}]}\,,$$

where $w(\boldsymbol{x}_{0:t})$, named *importance weight*, is the ratio:

$$w(\boldsymbol{x}_{0:t}) = \frac{p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}{\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})}.$$

Given N i.i.d. samples from $\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$, then $\hat{\boldsymbol{x}}_{0:t}$, the estimate of $\boldsymbol{x}_{0:t}$ provided by this approach is:

$$\begin{aligned}\hat{\boldsymbol{x}}_{0:t} &= \frac{\sum_{i=1}^{N} w_{0:t}^{i} \hat{\boldsymbol{x}}_{0:t}^{i}}{\sum_{i=1}^{N} w_{0:t}^{i}} \\ &= \sum_{i=1}^{N} \tilde{w}_{t}^{i} \hat{\boldsymbol{x}}_{0:t}^{i}. \end{aligned} \tag{2.2.2}$$

Here $\tilde{w}_t^i = \frac{w_{0:t}^i}{\sum_{i=1}^N w_{0:t}^i}$ are the *normalised importance weights.*

In *sequential importance sampling* (SIS), $\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t})$ is chosen such that $\hat{\boldsymbol{x}}_{0:t}$ may be computed recursively. This requires an importance distribution which satisfies a relationship of the form:

$$\begin{aligned}\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) &= \pi(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1})\pi(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t}) \\ &= \pi(\boldsymbol{x}_0) \prod_{k=1}^{t} \pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{1:k}). \end{aligned} \tag{2.2.3}$$

A natural choice for the importance distribution is the prior distribution:

$$\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}) = p(\boldsymbol{x}_{0:t}) = p(\boldsymbol{x}_0) \prod_{k=1}^{t} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}). \tag{2.2.4}$$

As the normalised importance weights satisfy the relationship:

$$\tilde{w}_t^i \propto \tilde{w}_{t-1}^i \frac{p(\boldsymbol{y}_t|\hat{\boldsymbol{x}}_t^i)p(\hat{\boldsymbol{x}}_t^i|\hat{\boldsymbol{x}}_{t-1}^i)}{\pi(\hat{\boldsymbol{x}}_t^i|\hat{\boldsymbol{x}}_{0:t-1}^i, \boldsymbol{y}_{1:t})}, \tag{2.2.5}$$

if the prior distribution is used as the importance distribution, this relationship simplifies to:

$$\tilde{w}_t^i \propto \tilde{w}_{t-1}^i p(\boldsymbol{y}_t|\hat{\boldsymbol{x}}_t^i), \tag{2.2.6}$$

which allows for a recursive implementation.

2.2.2 Resampling techniques

The resampling stage in particle filters is necessary to avoid sample degeneracy, a phenomenon caused by repeated randomisation which results in most samples being negligibly weighted after only a few iterations. For this reason, a randomised sample selection

process is introduced at the end of each iteration such that high-weight particles are retained and propagated (equally reweighted), while low-weight particles are discarded. This ensures that the distribution of the weights does not become excessively skewed. A number of resampling algorithms are now described.

a) Sampling importance resampling (SIR)

Sampling importance resampling is a popular resampling technique which was included in the original bootstrap filter [37], and it involves the use of a randomised lottery to select particles proportionally to their importance weights. As shown in Figure 2.3, this technique involves drawing samples u_i from the uniform distribution $\mathcal{U}(0,1)$. The particles whose cumulative weights $\mu_t^n = \sum_{j=1}^{n} \tilde{w}_t^j$ (for $n \in \{1,\ldots,N\}$) match the random samples are selected. For each sample u_i, the particle M_i (taking values in $\{1,\ldots,N\}$) is selected to satisfy:

$$\mu_t^{M_i-1} < u_i \leq \mu^{M_i}.$$

This operation is repeated for $i = 1,\ldots,N$: particles which are never selected by this process are discarded, while the surving particles are effectively cloned a number of times N^i equal to the number of times they were selected. The total number of particles thus generated remains equal to N (i.e. $\sum_{i=1}^{N} N^i = N$), and the weight of each new sample is set to $1/N$.

b) Residual resampling (RR)

In [49], Liu & Chen show how it is possible to reduce the variance of the resampling process by using *residual resampling* (RR), a technique which introduces a deterministic component to the selection. RR deterministically makes $\tilde{N}_t^i = \lfloor N\tilde{w}_t^i \rfloor$[1] copies of particle i and then selects the remaining $\hat{N}_t = N - \sum_{i=1}^{N} \tilde{N}_t^i$ particles by performing SIR on the residual weights,

$$\hat{w}_t^i = \frac{N\tilde{w}_t^i - \tilde{N}_t^i}{\hat{N}_t}.$$

[1] $\lfloor A \rfloor$ indicates the 'floor' function of A, the largest non-negative integer less than or equal to A.

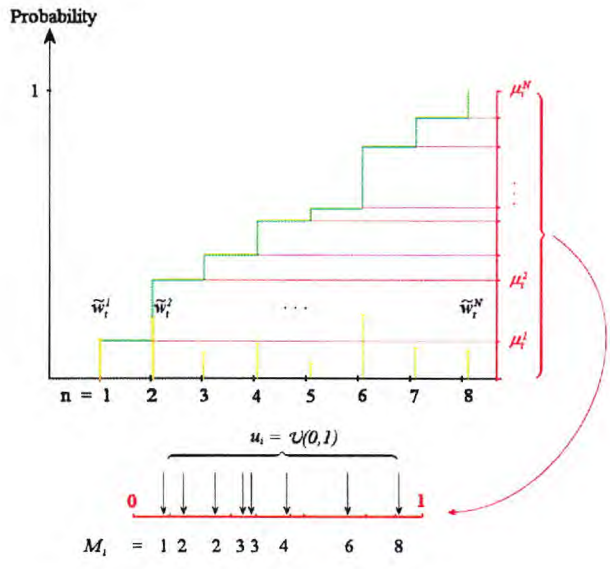


Figure 2.3: Sampling importance resampling, an example for $N = 8$. The top figure shows how, for $i \in \{1, \ldots, N\}$, the cumulative weights μ_t^i are calculated from the original weights, $\tilde{w}_t^i$ (marked in yellow). The bottom part of the figure shows how the sampling process matches $N = 8$ random numbers u_i to the cumulative weight distribution, to select eight indices: [1, 2, 2, 3, 3, 4, 6, 8]. The particles indexed in this manner are propagated to the next iteration.

c) Stratified sampling (SS)

Stratified sampling ([21]-[32]) takes this variance reduction a step further by using a well known survey sampling theorem [26] (p.97) which states that the minimum variance Monte Carlo estimator based on a sample population of size N, involves the generation of $\tilde{N}_i$ samples for each original sample $i \in \{1, \ldots, N\}$ such that $\tilde{N}_i \propto \tilde{w}_t^i$. As $\tilde{N}_i = N\tilde{w}_t^i$ is not generally going to be an integer, stratified sampling involves a sampling method that models $\tilde{N}_i$ as a random variable whose expectation $E[\tilde{N}_i] = N\tilde{w}_t^i$ and which takes values in either $\lfloor N\tilde{w}_t^i \rfloor$ or $\lfloor N\tilde{w}_t^i \rfloor + 1$. The constraint that $\sum_i \tilde{N}_i = N$ is also satisfied.

As with SIR, the first step consists in the calculation of the cumulative weights $\mu_t^n = \sum_{j=1}^{n} \tilde{w}_t^j$ of the particles. Instead of sampling randomly over the entire range from 0 to 1, in stratified sampling the probability space is split into N segments of probability $1/N$ (shown in green in Figure 2.4) and a sample is drawn from each of these bands. While the first sample is drawn randomly as $u_1 \sim \mathcal{U}(0, 1/N)$, the remaining $N - 1$ samples are

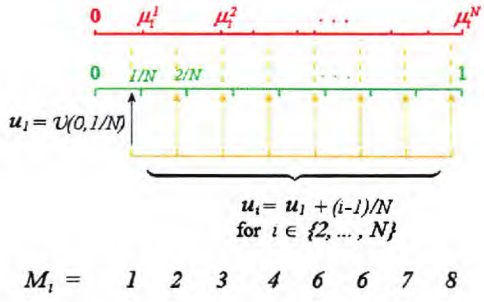


Figure 2.4: Stratified sampling, an example for $N = 8$. This figure shows how stratified sampling selects $N = 8$ indices from the cumulative weights μ_t^i (calculated as shown in Figure 2.3). The first sample is selected randomly from $\mathcal{U}(0, 1/N)$, the others are selected deterministically.

calculated deterministically as:

$$u_i = u_1 + \frac{i-1}{N} \qquad \text{for } i \in \{2, \ldots, N\}\,. \tag{2.2.7}$$

As for the SIR, the particle M_i, satisfying

$$\mu_t^{M_i - 1} < u_i \leq \mu^{M_i}\,,$$

is selected. This choice of samples ensures that the above conditions are met and that the variance of the selection procedure is minimised (see [32] for further detail). For the rest of this thesis, all particle filters are assumed to make use of this resampling technique unless otherwise stated.

2.2.3 The bootstrap filter

The bootstrap filter is now outlined in detail. The following steps assume nonlinear system equations of the form of (2.1.1)-(2.1.2) and the availability of N i.i.d. samples, $\hat{\boldsymbol{x}}_{t-1}^i$, from the prior distribution. It should be noted that, in the following equations, the prior weights, $\tilde{w}_{t-1}^i$, are omitted as the resampling step at $t-1$ produces uniform weights equal to $1/N$. The inclusion of these terms in the equations is thus redundant.

Step 1: Sequential Importance Sampling

For $i \in \{1, \ldots, N\}$, the samples $\tilde{\boldsymbol{x}}_t^i \in p(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t-1}^i)$ are simulated. This is done by sampling N values of the state noise $\boldsymbol{\nu}_t^s(i) \sim \mathcal{N}(0, \boldsymbol{Q}_m)$, and then using the state equation:

$$\tilde{\boldsymbol{x}}_t^i = f(\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{u}_t^s, \boldsymbol{\nu}_t^s(i)) \tag{2.2.8}$$

For each sample, the relevant importance weight is then calculated as:

$$\tilde{w}_t^i = \frac{1}{c} p(y|\tilde{\boldsymbol{x}}_t^i) = \mathcal{N}(\boldsymbol{y}_t; h(\tilde{\boldsymbol{x}}_t^i, \boldsymbol{u}_t^s, 0), \boldsymbol{Q}_m), \tag{2.2.9}$$

where $c = \sum_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_t; h(\tilde{\boldsymbol{x}}_t^i, \boldsymbol{u}_t^s, 0), \boldsymbol{Q}_m]$ is a normalising constant.

STEP 2: RESAMPLING

For $i \in \{1, \ldots, N\}$, resample with replacement N particles $\hat{\boldsymbol{x}}_t^i$ from the set of samples $\tilde{\boldsymbol{x}}_t^i$ according to importance weights $\tilde{w}_t^i$, using one of the techniques described in Section 2.2.2. The new particles each have weight $1/N$.

2.3 A multiple-model problem: manoeuvring target tracking

In the previous sections, we have discussed widely used solutions to the general target tracking problem (in both its linear and nonlinear case). We now investigate a specific type of target tracking problem in which nonlinearity is introduced by a switching parameter in the system dynamics. Problems of this type arise in *manoeuvring target tracking* and are generally referred to as *multiple-model tracking problems.* The techniques described below apply to other tracking problems involving multiple models (some target classification problems, for example).

2.3.1 Jump Markov linear systems

The most widely used formulation of the multiple model tracking problem involves the *jump Markov linear system* (JMLS) (used, for example, in [10] or [29]), which is sometimes also referred to as *conditional dynamic linear model* [22]. The JMLS assumes the system can be, at any time, in any one of d different 'modes', each of which is associated with a linear system as outlined in Section 2.1.2. A Markov chain process (hidden to the observer) is used to model the transitions between these modes. Figure 2.5 shows a representative mode-transition diagram for a system jumping between three modes. The numbers by the arrows represent the probabilities of these mode transitions

In manoeuvring target tracking the modes generally represent the different manoeuvres that the target can perform, and the transitions probabilities represent the probability of a switch at a particular point in time. In target classification the modes generally

model the target dynamics of a specific target type (eg. aircraft or missile), and while in this context the true mode does not generally change, estimated mode transitions reflect misclassifications. The rest of this thesis focuses on the manoeuvring target tracking problem. Interested readers may find further information on the target classification problem using multiple-models, in Angelova & Mihaylova [6].

The JMLS can be modelled by adjusting (2.1.14) and (2.1.15) to take into account the mode process:

$$\boldsymbol{x}_t = F(r_t)\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s(r_t) + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s, \tag{2.3.1}$$

$$\boldsymbol{y}_t = H(r_t)\boldsymbol{x}_t + \boldsymbol{u}_t^m(r_t) + (\boldsymbol{Q}_m(r_t))^{1/2}\boldsymbol{\nu}_t^m. \tag{2.3.2}$$

Here, r_t is the hidden Markov chain which takes values in the set of all possible modes, $\mathcal{S} = \{1, \ldots, d\}$. The mode transition probabilities are assumed to be known and are given by:

$$\pi_{ij} = \Pr(r_t = j | r_{t-1} = i). \tag{2.3.3}$$

a) The coordinated turn model

The choice of the mode-dependent state equation parameters is strictly related to the nature of the target being tracked. Numerous variations of the JMLS have been proposed in the literature, but generally modes are used to simulate the turns in the target's dynamics. Throughout this thesis a *coordinated turn model* (adapted from [12]) is used,

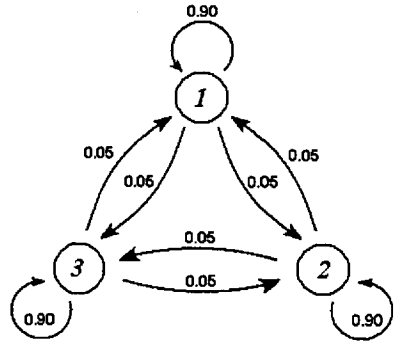


Figure 2.5: Sample mode transition diagram for a three mode system.

in which a target travels at constant speed in the two dimensional x-y plane. The possible target manoeuvres are modelled using three different modes ($d = 3$): a mode models straight-line motion, while the remaining model the rotation of the velocity components of the target in either direction.

Let the continuous component of the state (describing the positions and velocities of the target) be a 4-dimensional vector:

$$\boldsymbol{x}_t = [x(t), \dot{x}(t), y(t), \dot{y}(t)]^T , \tag{2.3.4}$$

where $x(t)$ and $y(t)$ refer to the x- and y-axis position coordinates of the target, and $\dot{x}(t)$ and $\dot{y}(t)$ the respective velocities. The equations governing the motion of the variables $(x(t), y(t))$ are obtained by sampling (with period h) the exact response of the stochastic differential equation:

$$\ddot{x}(t) = -\omega\dot{y}(t) + q_s\nu^x \tag{2.3.5}$$

$$\ddot{y}(t) = \omega\dot{x}(t) + q_s\nu^y \tag{2.3.6}$$

in which ν^x and ν^y are independent Gaussian white noise processes with unit intensity. The coordinated turn model is obtained by sampling the solution to this continuous time stochastic differential equation (reduced to state-space form and modified to include an exogenous input). When $q_s = 0$, with appropriate initial conditions, $(x(t), y(t))$ executes a circular motion with a constant angular speed of $|\omega|$ rad s^{-1}, anticlockwise if $\omega > 0$ and clockwise if $\omega < 0$. This is achieved by introducing the matrices:

$$\boldsymbol{A}_\omega = \begin{bmatrix} 1 & \frac{\sin(\omega h)}{\omega} & 0 & \frac{\cos(\omega h)-1}{\omega} \\ 0 & \cos(\omega h) & 0 & -\sin(\omega h) \\ 0 & \frac{1-\cos(\omega h)}{\omega} & 1 & \frac{\sin(\omega h)}{\omega} \\ 0 & \sin(\omega h) & 0 & \cos(\omega h) \end{bmatrix} \quad \text{and} \quad \boldsymbol{A}_0 = \begin{bmatrix} 1 & h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{2.3.7}$$

Define also:

$$\boldsymbol{R}_\omega^D = \begin{bmatrix} 2(\omega h - \sin(\omega h))/\omega^3 & (1-\cos(\omega h))/\omega^2 \\ (1-\cos(\omega h))/\omega^2 & h \end{bmatrix},$$

$$\boldsymbol{R}_\omega^C = \begin{bmatrix} 0 & (\omega h - \sin(\omega h))/\omega^2 \\ -(\omega h - \sin(\omega h))/\omega^2 & 0 \end{bmatrix},$$

and let:

$$\boldsymbol{R}_\omega = \begin{bmatrix} \boldsymbol{R}_\omega^D & \boldsymbol{R}_\omega^C \\ -\boldsymbol{R}_\omega^C & \boldsymbol{R}_\omega^D \end{bmatrix}, \quad \text{and} \quad \boldsymbol{R}_0 = \begin{bmatrix} h^3/3 & h^2/2 & 0 & 0 \\ h^2/2 & h & 0 & 0 \\ 0 & 0 & h^3/3 & h^2/2 \\ 0 & 0 & h^2/2 & h \end{bmatrix}. \tag{2.3.8}$$

The state and state-noise transition matrices ($\boldsymbol{F}(r_t)$ and $\boldsymbol{Q}_s(r_t)$ respectively) in (2.3.1) can then be specified to be:

$$\begin{aligned} \boldsymbol{F}(r_t = 1) &= \boldsymbol{A}_{-\omega_0}, & \boldsymbol{Q}_s(r_t = 1) &= q_s \boldsymbol{R}_{-\omega_0}, \\ \boldsymbol{F}(r_t = 2) &= \boldsymbol{A}_0, & \boldsymbol{Q}_s(r_t = 2) &= q_s \boldsymbol{R}_0, \\ \boldsymbol{F}(r_t = 3) &= \boldsymbol{A}_{+\omega_0}, & \boldsymbol{Q}_s(r_t = 3) &= q_s \boldsymbol{R}_{+\omega_0}, \end{aligned} \tag{2.3.9}$$

for a given turn rate $\omega_0 > 0$. ω_0 is generally assumed to be known, although this may not be necessarily so. In these cases it is usually good practice to overestimate ω_0 as simulations (in [60]) show that the estimators can more easily cope with slower than expected turn-rates (by adjustment of the estimated mode-probabilities) than higher-than-expected. The use of extra modes to simulate multiple turn rates is an alternative option, but the results were found not to justify the extra computational cost that this involves [60]. Efe and Atherton [30] have also suggested a model in which the turn rate is a variable that is estimated at each recursion by means of the velocity and acceleration estimates of the chosen tracking filter (for this to be possible, acceleration is added as an extra state in the central non-turning mode).

The coordinated turn model just outlined assumes a mode-independent exogenous state input and a mode-independent measurement process. System equations (2.3.1)-(2.3.2) can therefore be simplified to:

$$\boldsymbol{x}_t = \boldsymbol{F}(r_t)\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s, \tag{2.3.10}$$

$$\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m. \tag{2.3.11}$$

b) The mode-matched Kalman filter

Most techniques that deal with multiple-model problems (and whose solutions involve mixtures of Gaussians) adopt banks of Kalman filters to update the track on the basis of

specific mode-transitions. To simplify the description of these techniques in the following sections, a *mode-matched Kalman filter* is here introduced. This is the Kalman filter introduced in Section 2.1.2, in which the state transition matrices are mode dependent as defined in (2.3.10)-(2.3.11).

For a concise formulation of the mode-matched Kalman filter, new notation is introduced. Define, for each $j \in \mathcal{S}$, the mapping:

$$\mathrm{KF}_x(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, \boldsymbol{y}_t, r_t = j, t) \;=\; (\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j)\,, \tag{2.3.12}$$

where, for the system of (2.3.10) and (2.3.11),

$$\hat{\boldsymbol{x}}_{t|t-1}^j = \boldsymbol{F}(j)\hat{\boldsymbol{x}}_{t-1} + \boldsymbol{u}_t^s\,, \tag{2.3.13}$$

$$\boldsymbol{P}_{t|t-1}^j = \boldsymbol{F}(j)\boldsymbol{P}_{t-1}\boldsymbol{F}(j)^{\mathrm{T}} + \boldsymbol{Q}_s(j)\,, \tag{2.3.14}$$

$$\hat{\boldsymbol{y}}_{t|t-1}^j = \boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1}^j + \boldsymbol{u}_t^m\,, \tag{2.3.15}$$

$$\boldsymbol{S}_t^j = \boldsymbol{H}\boldsymbol{P}_{t|t-1}^j\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{Q}_m\,, \tag{2.3.16}$$

$$\tilde{\boldsymbol{y}}_t^j = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1}^j\,, \tag{2.3.17}$$

$$\boldsymbol{K}_t^j = \boldsymbol{P}_{t|t-1}^j\,\boldsymbol{H}^{\mathrm{T}}(\boldsymbol{S}_t^j)^{-1}\,, \tag{2.3.18}$$

$$\hat{\boldsymbol{x}}_t^j = \hat{\boldsymbol{x}}_{t|t-1}^j + \boldsymbol{K}_t^j\tilde{\boldsymbol{y}}_t^j\,, \tag{2.3.19}$$

$$\boldsymbol{P}_t^j = \boldsymbol{P}_{t|t-1}^j - \boldsymbol{K}_t^j\boldsymbol{H}\boldsymbol{P}_{t|t-1}^j\,. \tag{2.3.20}$$

Implicit in these relationships are the updating formulae provided by the Kalman filter. These provide, for each mode value, the conditional mean and covariance of the state at time t, given the new observation, $\boldsymbol{y}_t$, and the conditional mean and covariance of the state at the previous time $t-1$.

At this point, we also introduce a mapping of the likelihood of each mode-matched update, and mappings for the prediction and for the update equations only. Define:

$$\mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, r_t = j, t) \;=\; \mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j);\,, \tag{2.3.21}$$

$$\mathrm{KFpred}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, r_t = j, t) \;=\; (\hat{\boldsymbol{x}}_{t|t-1}^j, \boldsymbol{P}_{t|t-1}^j, \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j)\,, \tag{2.3.22}$$

$$\mathrm{KFupd}_x(\hat{\boldsymbol{x}}_{t|t-1}^j, \boldsymbol{P}_{t|t-1}^j, r_t = j, \boldsymbol{y}_t, t) \;=\; (\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j)\,, \tag{2.3.23}$$

$$\mathrm{KFupd}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t|t-1}^j, \boldsymbol{P}_{t|t-1}^j, r_t = j, t) \;=\; \mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j)\,. \tag{2.3.24}$$

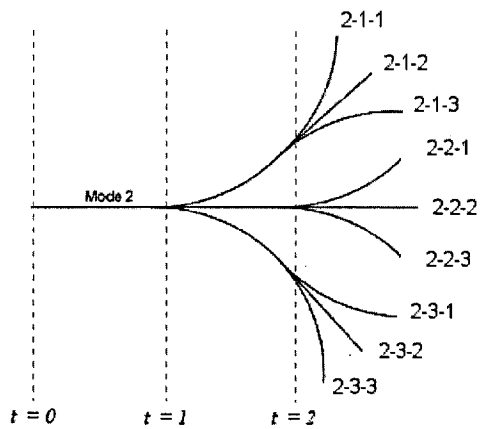


Figure 2.6: The figure shows the set of possible mode histories for the period $t = 0, 1, 2$, assuming the mode at time $t = 0$ is $r_0 = 2$.

2.3.2 The optimal multiple-model Bayesian solution

A recursive optimal Bayesian solution for target tracking problems based on the JMLS involves the estimation of all possible mode transitions, or mode-histories, up until the current time. Figure 2.6 shows the set of mode-histories for a three-mode system after only two time intervals. To obtain the optimal solution, an estimate would have to be evaluated (using Kalman filters) for each branch of this tree. The resulting mixture would provide the MMSE estimate.

Letting $D_{0:t}$ be the set of mode histories up to time t, it is possible to express the exact conditional density at time t as a weighted sum, or 'mixture', of densities in which $\mu_t^{r_{0:t}}$, where $\mu_t^{r_{0:t}} \in D^{r_{0:t}}$, represents the likelihood of each individual mode-history conditioned on all measurement histories, $\boldsymbol{y}_{1:t}$:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{r_{0:t} \in D_{0:t}} p(\boldsymbol{x}_t|r_{0:t}, \boldsymbol{y}_{1:t}) \mu_t^{r_{0:t}}\,, \tag{2.3.25}$$

where

$$D_{0:t} = \{r_{0:t} : r_0, \ldots, r_t \in \mathcal{S}\}\,, \tag{2.3.26}$$

$$\mu_t^{r_{0:t}} = p(r_{0:t}|\boldsymbol{y}_{1:t}), \quad \text{and} \quad \sum_{r_{0:t} \in D_{0:t}} \mu_t^{r_{0:t}} = 1\,. \tag{2.3.27}$$

This Gaussian mixture can also be expressed in terms of the weights $\mu_t^{r_{0:t}}$, the means, $\hat{\boldsymbol{x}}_t^{r_{0:t}}$, and the covariances, $\boldsymbol{P}_t^{r_{0:t}}$, of its constituent Gaussian densities as:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{r_{0:t}\in D_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{r_{0:t}}, \boldsymbol{P}_t^{r_{0:t}})\mu_t^{r_{0:t}}\,. \tag{2.3.28}$$

The densities in this sum can be calculated recursively using mode-matched Kalman filters, if the weights $\mu_{t-1}^{r_{0:t-1}}$, the means, $\hat{\boldsymbol{x}}_{t-1}^{r_{0:t-1}}$, and the covariances, $\boldsymbol{P}_{t-1}^{r_{0:t-1}}$, of the constituent Gaussian densities of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$ are known. Let:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}) = \sum_{r_{0:t-1}\in D_{0:t-1}} \mathcal{N}(\boldsymbol{x}_{t-1}; \hat{\boldsymbol{x}}_{t-1}^{r_{0:t-1}}, \boldsymbol{P}_{t-1}^{r_{0:t-1}})\mu_{t-1}^{r_{0:t-1}}\,, \tag{2.3.29}$$

then, for $r_t \in \mathcal{S}$, this can be achieved by means of the mappings introduced in (2.3.12) and (2.3.21), as:

$$(\hat{\boldsymbol{x}}_t^{r_{0:t}}, \boldsymbol{P}_t^{r_{0:t}}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^{r_{0:t-1}}, \boldsymbol{P}_{t-1}^{r_{0:t-1}}, \boldsymbol{y}_t, r_t, t)\,, \tag{2.3.30}$$

$$\tilde{\mu}_t^{r_{0:t}} = \mu_{t-1}^{r_{0:t-1}}\pi_{r_{t-1},r_t} KF_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{r_{0:t-1}}, \boldsymbol{P}_{t-1}^{r_{0:t-1}}, r_t, t)\,, \tag{2.3.31}$$

$$\mu_t^{r_{0:t}} = \frac{1}{c}\tilde{\mu}_t^{r_{0:t}}, \text{ where } c = \sum_{r_{0:t}\in D_{0:t}} \tilde{\mu}_t^{r_{0:t}}\,. \tag{2.3.32}$$

This optimal multiple-model implementation is quite clearly infeasible, due to the exponential growth in the number of mixture components (which is proportional to d^t). The following sections investigate target tracking algorithms that approximate this formula using a finite number of components.

2.3.3 Generalised pseudo-Bayes approximation techniques

Generalised pseudo-Bayes (GPB) approximation techniques (Bar-Shalom [10]-[11]) approximate the exact conditional distribution of all possible mode-histories (2.3.28), by merging together those components in the posterior conditional distribution whose mode-history is identical over the most recent time steps. Algorithms of this nature vary primarily with respect to the 'time-depth' considered for merging: first order algorithms consider only the latest time step (thus merging the distribution in a single Gaussian), second order algorithms consider the last two time intervals, and thus retain d histories. The most popular of these pseudo-Bayes techniques, known as the *interacting multiple model* (IMM) algorithm, introduces further approximations which enable it to retain a time-depth comparable to a second order algorithm, but with the computational complexity of a first order algorithm.

The most widely used of these techniques – first and second order GPB and the IMM – are now summarised. It should be noted that while higher order GPBs are easily developed, simulations (in [60] and [71]) have shown them to be qualitatively comparable to second order ones, but significantly more computationally intensive.

a) The first order generalised pseudo-Bayes algorithm

The *first order generalised pseudo-Bayes algorithm* (GPB1) is the crudest approximation of the optimal Bayesian solution. The conditional distribution,

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{r_{0:t}\in M_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{r_{0:t}}, \boldsymbol{P}_t^{r_{0:t}})\mu_t^{r_{0:t}},$$

is approximated using a single density:

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t, \boldsymbol{P}_t).$$

The algorithm has two main steps. Starting from the single Gaussian density, the algorithm uses a bank of d Kalman filters, each matched to a different mode, to calculate an estimate for all the possible mode transitions. These are then merged in a single Gaussian which enables the recursion to continue. The formulae involved in one iteration of the GPB1 algorithm are given in Appendix B.1.1.

b) The second order generalised pseudo-Bayes algorithm

The *second order generalised pseudo-Bayes algorithm* (GPB2) is a more refined approximation of the optimal Bayesian solution, in which the exact d^t-fold Gaussian mixture is approximated using d Gaussian densities, each assuming a different mode for the previous time interval. This allows combining mode-histories whose differences are found further back in the past. The exact density:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{r_{0:t}\in M_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{r_{0:t}}, \boldsymbol{P}_t^{r_{0:t}})\mu_t^{r_{0:t}},$$

is therefore approximated by:

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{j\in\mathcal{S}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j)\mu_t^j.$$

Figure 2.7 illustrates one iteration of the GPB2 (for a detailed description, see Appendix B.1.2). The algorithm firstly computes an estimate for all possible mode transitions at time t. Those estimates associated with a common mode for time t are then merged and the resulting d-fold mixture is propagated to the next iteration. This approximation brings about a reduction in the number of Kalman filter operations from d^{t+1} to d^2. At the same time this algorithm has the advantage of retaining the expectations associated with each mode.

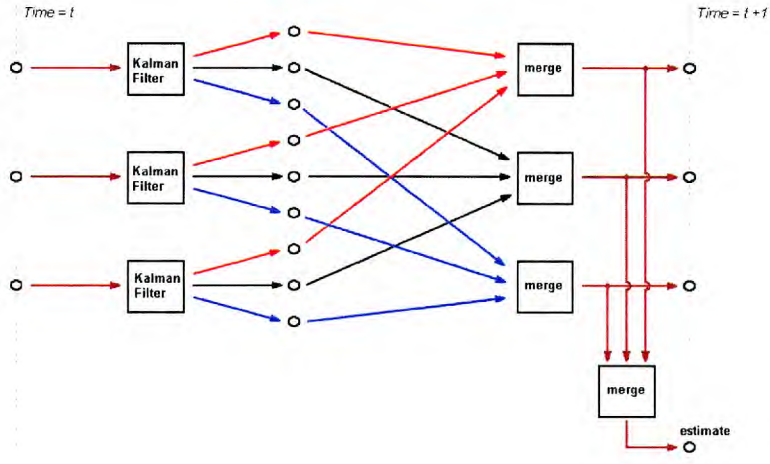


Figure 2.7: The second order generalised pseudo-Bayes algorithm (GPB2).

c) The interacting multiple model algorithm

The *interacting multiple model algorithm* (IMM) – first proposed by Blom and Bar-Shalom [17] – is a direct approximation of the GPB2. The IMM, in common with the GBP2 algorithm, approximates the evolving conditional density as a d-fold Gaussian mixture. However it introduces further approximations which reduce the algorithm's computational complexity by a factor of d without introducing significant degradation.

The key difference (as illustrated in Figure 2.8) is that the IMM, instead of letting the initial d-fold Gaussian mixture expand into a d^2-fold mixture, it precomputes the probabilities of each mode at time t, based on the prior information at time $t-1$. This procedure is known as 'mixing'. On the basis of these weights, the means and covariances are also 'mixed' to produce initial conditions which are then used by a bank of Kalman filters (matched to the appropriate mode), to directly generate a d-fold posterior

distribution.

The mixing step in the IMM is similar to the merging process of the GPB2 but, since it takes place before the measurement updates, it does not take into account the information of the latest measurement. The algorithm thus still retains the merging depth of a second-order algorithm, while its computational complexity has been reduced to that of a first order. Results (in [17],[60]) have shown how the IMM produces only minimally degraded estimates compared to the GPB2. The IMM algorithm is fully described in Appendix B.1.3.

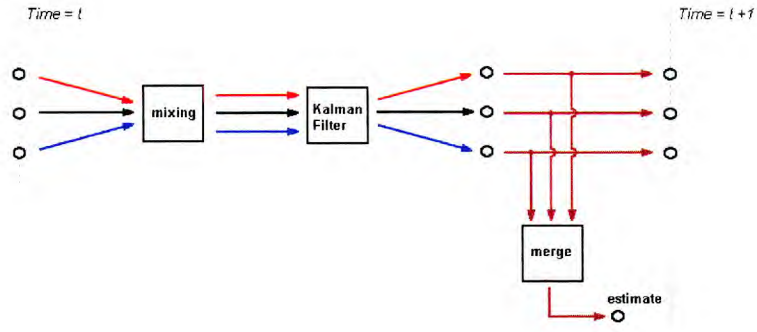


Figure 2.8: The interacting multiple model algorithm (IMM).

2.3.4 Particle filters for JMLS models

Sequential Monte Carlo techniques (or particle filters) were described in Section 2.2 for the general filtering problem. Particle filters are highly flexible and can be used for multiple-model problems as first described in [54]. Given the specific structure of this type of problems, the variance of such algorithms can be reduced by marginalising the state and sampling exclusively over the mode distribution. This process, known as *Rao-Blackwellisation*, leads to 'low variance' particle filters (also referred to as *mixture Kalman filters* [49]-[29]) which propagate mixtures of Gaussians.

a) The multiple-model bootstrap filter

The *multiple-model bootstrap filter* (MMBF) [54],[55] is a direct application of the standard bootstrap filter to multiple-model problems modelled using jump Markov linear systems. According to this approach, we use the prior distribution, $p(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t-1})$ as

the importance distribution. Suppose an empirical distribution:

$$\pi(\boldsymbol{x}_t, r_t) = \sum_{n=1}^{N} \mu_{t-1}^n \delta(\boldsymbol{x}_t^n, r_t^n) \tag{2.3.33}$$

is constructed by sampling from this distribution. By Bayes rule,

$$p(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t}) = \frac{1}{c_t} p(\boldsymbol{y}_t|\boldsymbol{x}_t) p(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t-1}), \tag{2.3.34}$$

where $c_t = p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$ is a normalisation constant. Therefore an approximation to the distribution $p(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t})$ is provided by:

$$\frac{1}{c_t} \sum_{n=1}^{N} \mu_{t-1}^n p(\boldsymbol{y}_t|\boldsymbol{x}_t) \delta(\boldsymbol{x}_t^n, r_t^n).$$

Thus, for $n \in \{1, \ldots, N\}$, the importance weights are:

$$\mu_t^n = \mu_{t-1}^n p(\boldsymbol{y}_t|\boldsymbol{x}_t^n).$$

The MMBF is an example of the general bootstrap filter, in which the signal process comprises discrete and continuous components. The MMBF is described in detail in Appendix B.1.4.

b) Rao-Blackwellised particle filters

An important refinement of the particle filter for multiple-model problems is the *Rao-Blackwellised particle filter* (RBPF) [29] (sometimes also known as *mixture Kalman filter* [22],[49]), a sequential Monte Carlo technique that uses 'Gaussian' particles to reduce the variance of the estimates. The RBPF algorithm makes use of the fact that the joint posterior distribution can be re-written as:

$$p(\boldsymbol{x}_{0:t}, r_{0:t}|\boldsymbol{y}_{1:t}) = p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, r_{0:t}) p(r_{0:t}|\boldsymbol{y}_{1:t}), \tag{2.3.35}$$

where $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, r_{0:t})$ is a Gaussian distribution which can be determined exactly using a Kalman filter for a given realisation of $r_{0:t}$. In order to use (2.3.35) we require also the distribution $p(r_{0:t}|\boldsymbol{y}_{1:t})$. However, as its exact evaluation is not feasible, approximations must be used.

The RBPF algorithm constructs an empirical distribution approximating $p(r_{0:t}|\boldsymbol{y}_{1:t})$ by sampling from the distribution:

$$\pi(r_{0:t}|\boldsymbol{y}_{1:t}) = p(r_0) \prod_{k=1}^{t} p(r_k|r_{0:k-1}, \boldsymbol{y}_{1:k}), \tag{2.3.36}$$

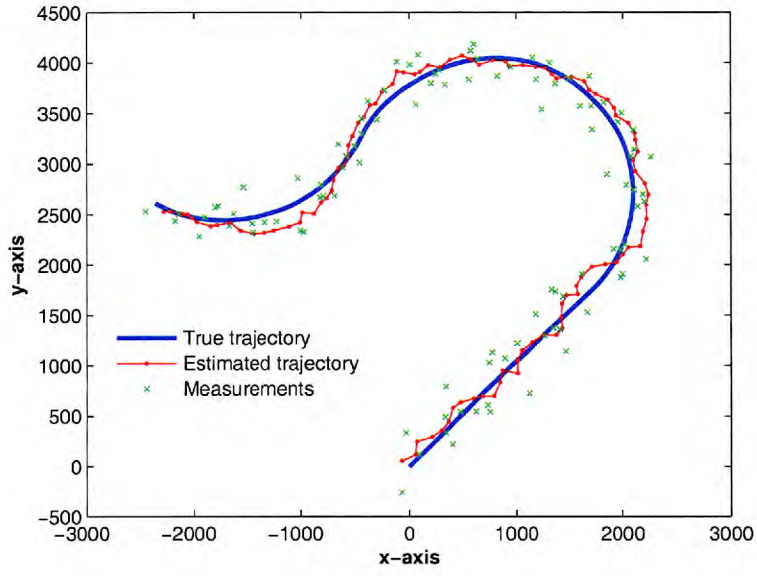


Figure 2.9: Sample realisation of Track 1. Estimates produced using the IMM.

and employing appropriate importance weights. The structure of this formula for $\pi(r_{0:t}|\boldsymbol{y}_{1:t})$ implies that the weights can be propagated recursively as time advances. As shown in [29], the product term in (2.3.36) can be calculated from the recursive formulae:

$$p(r_t = j|r_{0:t-1}, \boldsymbol{y}_{1:t}) = \frac{1}{c_t} p(\boldsymbol{y}_k|r_k = j, r_{0:k-1}, \boldsymbol{y}_{1:k-1}) p(r_k = j|r_{k-1}) \,, \tag{2.3.37}$$

where c_t is a normalising constant given by:

$$c_t = \sum_{j=1}^{d} p(\boldsymbol{y}_t|r_t = j, r_{0:t-1}, \boldsymbol{y}_{1:t-1}) p(r_t = j|r_{t-1}) \,.$$

The RBPF is fully described in Appendix B.1.5.

2.3.5 Preliminary comparisons

This section reports on preliminary simulation comparisons of the main multiple-model algorithms described in the previous sections. A sample track was generated using the coordinated turn model described in Section 2.3.1 with the following parameter values:

$$
\begin{aligned}
h &= 0.5 \text{ s} && \text{sample period,} \\
T &= 50 \text{ s} && \text{time horizon,} \\
\omega_0 &= 0.1353 \text{ rad s}^{-1} && \text{turn rate,} \\
q_s &= 20 \text{ m}^2 && \text{target perturbation intensity,} \\
\boldsymbol{u}_t^s &= [0\ 0\ 0\ 0]^T && \text{exogenous state input vector } (\forall\, t).
\end{aligned}
$$

The target is initialised at $\boldsymbol{x}_0 = [0, 150, 0, 150]^T$. It manoeuvres as follows: $r_{0:25} = 2$, $r_{26:75} = 3$, $r_{76:100} = 1$. As the actual mode switching probabilities are unknown to the observer, the following Markov transition matrix is used (simulations in [60] show that, within reason, the choice of parameters has little effect):

$$
\pi = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}.
$$

The measurement process was modelled using (2.3.2) and was assumed to be manoeuvring-mode independent. The following parameters were used:

$$
\begin{aligned}
\boldsymbol{H} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} && && \text{measurement transition matrix,} \\
\boldsymbol{Q}_m &= \begin{bmatrix} 24,000 & 0 \\ 0 & 24,000 \end{bmatrix} && \text{m}^2 && \text{measurement noise covariance matrix,} \\
\boldsymbol{u}_t^m &= [0\ 0]^T && && \text{exogenous measurement input vector } (\forall\, t).
\end{aligned}
$$

The track thus generated, which we refer to as 'Track 1', is shown in Figure 2.9 (along with a sample measurement and a sample estimate realisation produced using the IMM filter). One hundred sets of measurements were produced for comparison in a Monte Carlo simulation.

Figure 2.10 shows the root mean square (RMS) error comparison[2] of the *IMM* with

[2]All the estimation algorithms tested in this thesis make use of the same parameters and statistics as those used to generate the tracks and the measurements (unless otherwise stated). For further details on the way simulations are conducted, refer to Appendix C.

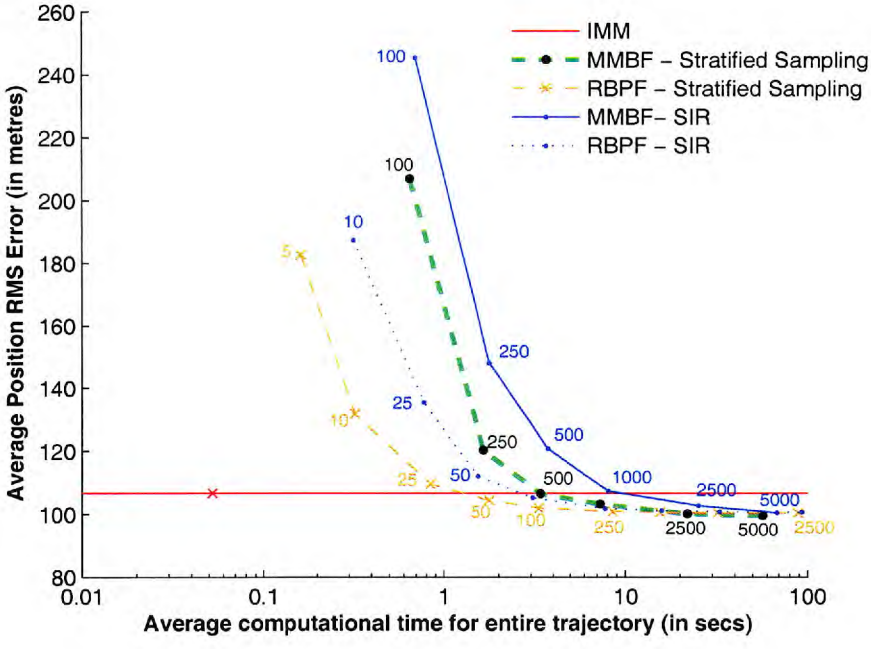


Figure 2.10: Position RMS error vs. computational cost comparisons for Track 1 and for varying number of mixture components (100 Monte Carlo average). Component numbers are indicated where possible. For the RBPF, the range tested is: [5, 10, 25, 50, 100, 250, 500, 1000, 2500]; for the MMBF: [100, 250, 500, 1000, 2500, 5000].

the *multiple model bootstrap filter* (MMBF) and with the *Rao-Blackwellised particle filter* (RBPF), both of which are tested using *sampling importance resampling* (SIR) and *stratified sampling* (SS) in the resampling stage. With the exception of the IMM, which links the number of mixture components to the cardinality of mode, the plot shows the RMS error produced vs. the computational time[3] required for increasing number of particles. The IMM's exact position on this chart is marked by a cross, but for illustration purposes, its RMS error is extended over the whole x-axis range.

For this basic scenario, Figure 2.10 shows the accuracy and remarkable computational efficiency of the IMM. The filter produces good quality estimates at a fraction of the cost of the particle filters. Since the only nonlinearity present in this scenario is the target mode dependence, clearly matching a mixture component to each manouevring mode is an effective strategy. The effectiveness of the IMM-related algorithms decreases when extra nonlinearities are introduced, and the estimator used to evaluate the mixture components is no longer optimal. This may happen when clutter is introduced (see Section 4.4) or when nonlinearities in the system equations require the use of a suboptimal estimator such as the EKF or the UKF [54] (this is further discussed in Section 5.4). The IMM also suffers in scenarios where the cardinality of the mode is very high (for example in joint classification and manoeuvring). In such cases, some of the modes might have very low likelihoods and retaining a component for every mode might no longer be an effective strategy.

Even for this simple scenario, specially tailored to the IMM algorithm, all of the sequential Monte Carlo algorithms eventually (i.e. following introduction of sufficiently large number of particles) manage to outperform it by nearly 10%. Figure 2.10 also reveals how the computational efficiency of the particle filters can be increased by means of improved resampling techniques (i.e. stratified sampling instead of sampling importance resampling) and by means of Rao-Blackwellisation. Chapter 3 introduces mixture filters which aim to outperform the IMM at an even smaller computational cost.

[3]As explained in further detail in Appendix C, the computational time used to illustrate the results presented in this thesis should be taken as indicative only, as it is inevitably dependent on the hardware and the programming language and style used.

Chapter 3

Sequential Gaussian Mixture Filters for Target Tracking

3.1 Introduction & review of mixture filters

Having covered some of the techniques most widely used for multiple-model target tracking problems, we now investigate the use of flexible sequential mixture filters, based on efficient mixture reduction algorithms. The techniques introduced in this section share similarities to both Rao-Blackwellised particle filters and to generalised pseudo-Bayes algorithms. In common with the former, these filters propagate a Gaussian mixture whose complexity is dictated by the computational resources available. In common with the latter, they calculate, for a given prior, the exact posterior distribution of the target by considering all possible hypotheses at that time step. While GPB techniques then reduce the distribution by deterministically merging those components that share a common 'mode history' (up to a certain time-depth), these mixture filters employ a more refined mixture reduction step, which allow for more customisation and flexibility.

In Section 3.1.1, the structure and general characteristics of these *mixture filters* are described, while in Section 3.2 we investigate in further detail the critical mixture reduction step. While mixture filters have been proposed before for manoeuvring target tracking problems ([69], [33]), the mixture reduction step involved has generally been limited to one of selecting a subset of the constituent densities of the posterior distribution on the sole basis of their *a posteriori* likelihood (or weight). One of the main

contributions of this thesis is to demonstrate how these methods are closely related to clustering algorithms normally presented in the context of multiple-measurement tracking, and to adapt the latter for application to multiple-model tracking problems. In the mixture reduction section, both types of techniques are described, and their application to both classes of problems is discussed. Furthermore, ideas from both types of techniques are combined to produce a 'hybrid' mixture reduction technique.

Lastly, in Section 3.3, we show how mixture filters can be modified to take account of the latest measurement information. The traditional method selects particles for use at time t based on the measurement information available up until time $t-1$; in the modified structure, the selection is delayed until receipt of the measurement at time t so that this information can be incorporated in the decision process. This 'look-ahead' method proves useful at recognising target manoeuvres, and particularly so when hardware resources allow for a limited use of mixture components.

3.1.1 The basic structure of mixture filters

The *mixture filter* (MF) approximates the optimal conditional distribution of (2.3.28):

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{r_{0:t}\in D_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{r_{0:t}}, \boldsymbol{P}_t^{r_{0:t}})\mu_t^{r_{0:t}} ,$$

by

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{n\in D_t} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{n}, \boldsymbol{P}_t^{n})\mu_t^{n} , \tag{3.1.1}$$

where D_t is a subset of $D_{0:t}$ (the set of all mode sequences $\{r_0,\ldots,r_t\}$) of cardinality N, a positive integer whose magnitude reflects available computational resources. When a new measurement becomes available, the *exact* $N\times d$-fold posterior distribution is calculated by considering all possible mode transitions for each prior component. This pdf is then approximated by a N-fold one via mixture reduction. This simple structure can be summarised in the following steps.

STEP 1: MODE-MATCHED KALMAN FILTERING

Given N prior mixture components with parameters $\{\hat{\boldsymbol{x}}_{t-1}^n,\ \boldsymbol{P}_{t-1}^n,\ \hat{r}_{t-1}^n,\ \mu_{t-1}^n\}$ (for $n\in$

$\{1,\ldots,N\}$), the Gaussian mixture filter calculates the exact posterior conditional distribution by using banks of mode-matched Kalman filters. For modes $j \in \mathcal{S} = \{1,\ldots,d\}$, this is done by means of the maps defined in (2.3.12),(2.3.21):

$$(\hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}_t^{nj}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, \boldsymbol{y}_t, j, t)\,, \tag{3.1.2}$$

$$\tilde{\mu}_t^{nj} = \mu_{t-1}^n \pi_{\hat{r}_{t-1}^n j} KF_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, j, t)\,, \tag{3.1.3}$$

$$\mu_t^{nj} = \frac{1}{c}\tilde{\mu}_t^{nj}, \text{ where } c = \sum_{n,j} \tilde{\mu}_t^{nj}\,. \tag{3.1.4}$$

STEP 2: STATE ESTIMATE & COVARIANCE CALCULATIONS

Next, the output state estimate, $\hat{\boldsymbol{x}}_t$, and covariance, $\boldsymbol{P}_t$, are computed by merging together all the constituent densities of the posterior conditional distribution:

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N}\sum_{j=1}^{d} \mu_t^{nj}\hat{\boldsymbol{x}}_t^{nj}\,, \tag{3.1.5}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N}\sum_{j=1}^{d} \mu_t^{nj}[\boldsymbol{P}_t^{nj} + (\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]\,. \tag{3.1.6}$$

It should be noted that the output estimates should be calculated prior to the mixture reduction step, as this inevitably introduces degradations.

STEP 3: MIXTURE REDUCTION

Finally the mixture is reduced back to N-fold. A number of possible mixture reduction algorithms are described in Section 3.2, along with a discussion of their merits and weaknesses. The mixture filter described in this section can incorporate only 'weight-based' mixture reduction methods. These methods involve selecting a subset of cardinality N from the $N \times d$-fold set of all the Gaussian components in the posterior conditional distribution. Selection is performed on the exclusive basis of their weights.

A different class of mixture reduction methods, which we refer to as 'distribution-based' (see Section 3.2.2) also involves component merging. As this may lead to the combination of components associated with different modes, a modification to (3.1.3) – which assumes a single prior mode per mixture component – is required. The resulting modified mixture filter is derived and outlined in Section 3.2.3(a).

3.1.2 Mixture filters & Rao-Blackwellised particle filters

The mixture filter (MF) shares clear similarities with the Rao-Blackwellised particle filter (RBPF) [22] described in Section 2.3.4. Both algorithms approximate the optimal Bayesian solution by means of a mixture of Gaussians in a manner which is both flexible and has a complexity that can be scaled to accommodate specific problem or hardware requirements. A careful analysis of the main differences between the MF and the RBPF (illustrated in Figure 3.1 along with a sample realisation of their sampling/selection processes) shows however that mixture filters have a number of clear advantages. Firstly,

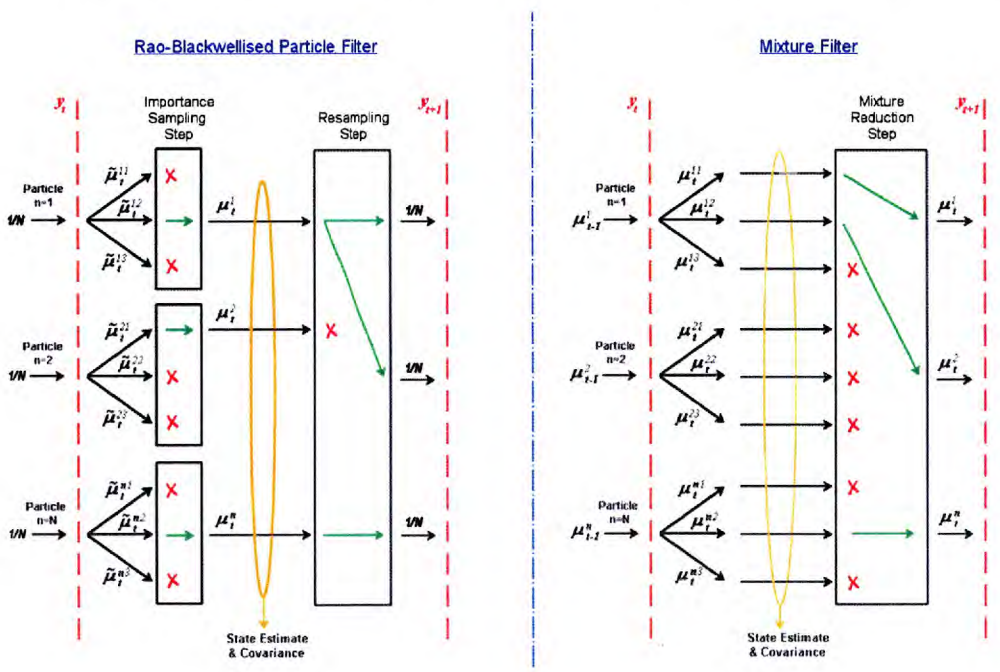


Figure 3.1: A comparison of the structure of the mixture filter with that of the Rao-Blackwellised particle filter. The diagram shows the calculation of the weights and the sampling/selection stages where samples are discarded or propagated (a sample realisation of the sampling processes is shown).

given an identical prior distribution, the updated state estimate produced by the RBPF is an approximation of the one produced by the MF. In fact, for any given prior, the MF calculates the exact posterior distribution by considering all possible mode transitions that each estimate could perform during the current iteration. On the other hand, the RBPF only considers a single, randomly selected, mode for each prior component. As a result, the estimates provided by the RBPF can be expected to be poorer than those

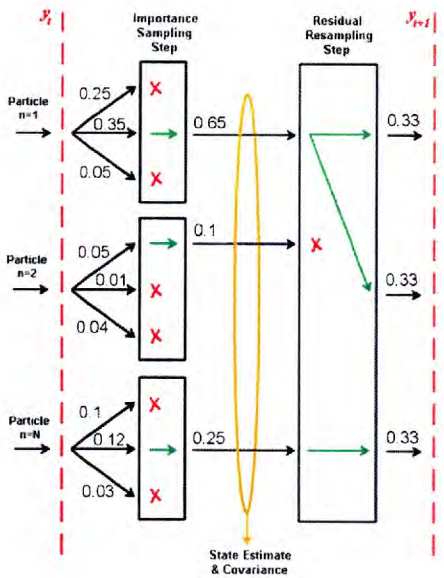


Figure 3.2: A sample realisation of the sampling and resampling stages of the RBPF shown using stylised structure schematics. The most likely outcome of the mixture filter for these same weight values is shown in Figure 3.1.

provided by the MF.

Furthermore, the full calculation of the posterior distribution requires only a modest increase in computations. Although the MF performs $N \times d$ full Kalman filter updates, compared to N of the RBPF, this analysis is misleading as, in order to compute the sampling probabilities, the algorithm needs to calculate $N \times d$ likelihoods for each possible mode transition. This involves the calculation of $\mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}^{nj}_{t|t-1}, \boldsymbol{S}^{nj}_t)$, which requires the evaluation of the entire Kalman filter prediction stage and the inversion of matrix $\boldsymbol{S}^{nj}_t$, itself one of the most computationally demanding parts of the Kalman filter. The computational savings of the RBPF amounts to $N \times (d-1)$ times the computational cost of evaluating (2.1.45)-(2.1.47) (with the exception of the inversion of $\boldsymbol{S}^{nj}_t$). The actual computation time difference varies depending on the dimension of the state and on the number of modes. For standard problems such as the one presented in Section 2.3.1, where $R = 4$ and $d = 3$, the computational saving of using the RBPF is of the order of 10-15%.

A second, more significant, advantage of mixture filters is the inclusion of a mixture

reduction step in place of the combination of the mode-sampling and resampling stages of the RBPF. Mixture reduction is explained in greater detail in the next section, but it generally involves the approximation of the $N \times d$-fold posterior distribution with a N-fold one. This approach has a clear advantage over the sampling/resampling combination of the RBPF, as the latter could result in the propagation of 'cloned' components with identical mean, covariance & weight. As shown in Figure 3.2 for a sample realisation chosen to illustrate the problem (but not uncommon), the sampling step in the RBPF forces one component to be selected from each 'parent' particle. This penalises parents with high-weight 'offsprings', as only one of them can be selected. When the resampling stage is next performed, it is likely that this high-weight offspring is selected more than once (i.e. 'cloned'). This is never the case in the mixture filter, where the mixture reduction stage considers all offsprings on a par, irrespectively of their parent particle, and avoids sample cloning. The most likely output selection for a (weight-based) mixture reduction algorithm for the numerical example of Figure 3.2, is shown in Figure 3.1.

3.2 Mixture Reduction

Mixture reduction is defined as the problem which occurs when a probability density function, which is a weighted sum of M Gaussian densities, needs to be approximated by another one involving only N Gaussian components, where $N < M$. The aim of mixture reduction algorithms is to perform this task while minimising, in some sense, the level of degradation introduced by the approximation. Figure 3.3 shows an example of mixture reduction in which a pdf comprising $M = 9$ components is approximated by one where $N = 3$.

In target tracking, mixture reduction techniques are applicable to a number of problems where the conditional density evolves as a Gaussian mixture with a growing number of terms. In the context of multiple-model target tracking, other approaches (such as pseudo-Bayes algorithms or particle filters) have generally been favoured, and only a few mixture reduction algorithms have been proposed ([69],[33]). These methods, which are described in further detail in Section 3.2.1, will be referred to as 'weight-based', as they select a subset of Gaussian components on the basis of *a posteriori* probabilities. The main assumption underlying these algorithms is that the weights are a good indi-

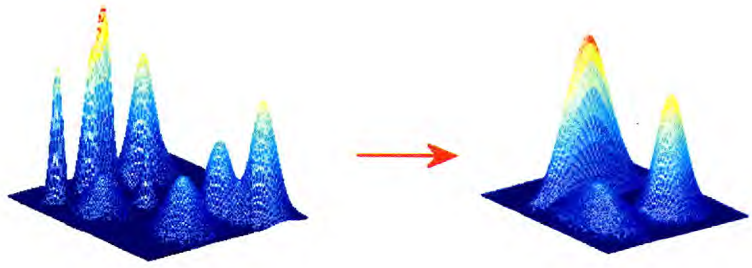


Figure 3.3: An example of mixture reduction.

cator of the usefulness of the information carried by the components. In Section 3.2.1 we present an empirical study of the performance of weight-based mixture reduction algorithms, and a novel algorithm that allows for their customisation for specific problems.

The interest in mixture reduction has been greater for *multiple-measurement* (or *data association*) problems, where hypotheses growth is caused by incidence of false measurements (referred to as clutter) and/or by multiple targets. To improve on simple techniques such as *probabilistic data association* [13], which retains a single mixture component, more sophisticated mixture reduction algorithms ([62], [73]) were developed. These techniques, which reduce the number of constituent densities by merging them on the basis of their spatial configuration, will be referred to as 'distribution-based' and are described in further detail in Section 3.2.2. For further information on the data association problem, see Appendix A.

Although weight-based and distribution-based mixture reduction algorithms were originally developed for multiple model and data association problems respectively, the task they perform is essentially the same. In Section 3.2.4, we combine aspects of weight-based and distribution-based algorithms to produce a novel 'hybrid' mixture reduction algorithm. Finally, in Section 3.2.5, we report on simulations comparing all mixture reduction methods for the two classes of problems.

3.2.1 Weight-based Mixture Reduction

We now describe the three main weight-based mixture reduction approaches. We also provide a simulation-based study which assesses these techniques, particularly with re-

spect to the trade-offs between randomisation and deterministic selection. An algorithm that allows the customisation of this trade-off is also proposed.

a) Randomised mixture reduction

The *random sampling algorithm* is a weight-based mixture reduction technique, first proposed by Akashi and Kumamoto [4], to solve a multiple model state estimation problem in which the measurement noise covariance parameters switch according to a Markov parameter. In the original algorithm of [4], a component is randomly selected from the set of 'offsprings' of each 'parent' (or prior) component. This approach constrains somewhat the efficiency of the randomised selection process, as the algorithm may be forced to propagate low weight samples.

We now present a variant of this algorithm in which this constraint is relaxed and in which a randomised lottery selects N of the M components irrespective of their origin. This approach (outlined in Algorithm 3.2.1) is essentially a generalisation of the *sampling importance resampling* (SIR) algorithm (which is described in Section 2.2.2 and which applies to the special case where $M = N$) and yields unbiased approximations to the density. A generalisation of low variance methods such as *stratified sampling* could also be used.

This randomised approach is however likely to result in sample cloning, which is wasteful as a single Gaussian component with a proportionally larger weight could carry the same information. To avoid this problem, a variant of this algorithm is also possible where the randomisation process is continued until N *distinct* samples are selected at least once. In this case, the weights of the selected components would be directly proportional to the number of times they are extracted by the randomised process. This approach is integral to Algorithm 3.2.6 and is further discussed in Section 3.2.1(d).

Algorithm 3.2.1 - Randomised mixture reduction

Step 1. Let $\mu_t(i)$ (for $i \in \{1,\ldots,M\}$) be the weights of the components of a M-fold Gaussian mixture. Draw N samples ψ_n (for $n \in \{1,\ldots,N\}$) from the uniform distribution $\mathcal{U}(0,1)$.

Step 2. Find the component G_n, (with possible values in the range $\{1,\ldots,M\}$) such that:

$$\sum_{j=1}^{G_n-1} \mu_t(j) < \psi_n \leq \sum_{j=1}^{G_n} \mu_t(j)\,.$$

Step 3. The reduced N-fold mixture is formed by the components indexed by G_n, all with weight $1/N$.

b) The detection-estimation (deterministic) approach

The *detection-estimation algorithm* (DEA) [68]-[69] takes a more deterministic approach in which the distribution is reduced by a simple selection of those N components which maximise *a posteriori* probability. In practice, the algorithm simply involves a permutation of the components in order of decreasing weights and a selection of the largest N ones. The original weights of the chosen samples are retained and are normalised to ensure that they add to one. Maskell [51] proposes a similar scheme in which the combined weight of the discarded components is equally redistributed amongst the 'surviving' ones.

Algorithm 3.2.2 - The detection-estimation algorithm

Step 1. Let $\mu_t(i)$ be the weights of the components of a M-fold Gaussian mixture. Permute weights $\mu_t(i)$ in decreasing order of weights, and let $\bar{\mu}_t(j)$ be the reordered set (for $i,j \in \{1,\ldots,M\}$).

Step 2. For $n \in \{1,\ldots,N\}$, find the component with the original label G_n for which $\bar{\mu}_t(n) = \mu_t(G_n)$.

Step 3. The reduced N-fold mixture is formed by the components indexed by G_n. The weights of the propagated samples is given by $\frac{\bar{\mu}_t(n)}{\sum_{j=1}^{N} \bar{\mu}_t(j)}$.

The clear advantage of the DEA algorithm is that, by choosing a maximum *a posteriori* probability approach, the change to the distribution at each iteration is reduced. However this deterministic selection criterion means that the state estimate is no longer unbiased, and we later show that a more randomised approach is sometimes favourable.

It should finally be noted that in a refinement of their basic DEA algorithm [69], the authors suggest the possibility of merging components which are closely related to each other before component pruning takes place. This is done in order to avoid propagating similar components when the same information could be carried forward by a single combined component. For components to qualify for this preliminary merging step, the authors suggest checking the Bhattacharyya distance [38] between the components against a pre-determined threshold. This approach is very similar to that used in distribution-based mixture reduction methods; while Tugnait and Haddad did not implement their refinement due to its computational complexity, their algorithm was clearly a first attempt to integrate weight-based decision making with the other parameters of the components.

c) The Fearnhead and Clifford approach

Fearnhead and Clifford ([32],[33]) recently proposed an algorithm which aims to provide an 'optimal' approach to weight-based mixture reduction. By focusing exclusively on the distribution of the weights, the authors approach the problem as one of approximating a discrete probability mass function $\mu_t(i)$ $(i \in \{1 \dots M\})$ with finite support of size M, with a stochastic probability mass function $w_t(i)$. The latter needs to satisfy three main conditions: *(i)* it should be unbiased (i.e. $E(w_t(i)) = \mu_t(i)$), *(ii)* it should not have more than N nonzero support points (where $N < M$) and *(iii)* it should be such that the L^2 difference between the distributions (i.e. $\sum_{i=1}^{M} (w_t(i) - \mu_t(i))^2$) is minimised. The procedure is 'optimal' in a sense made precise in the following theorem:

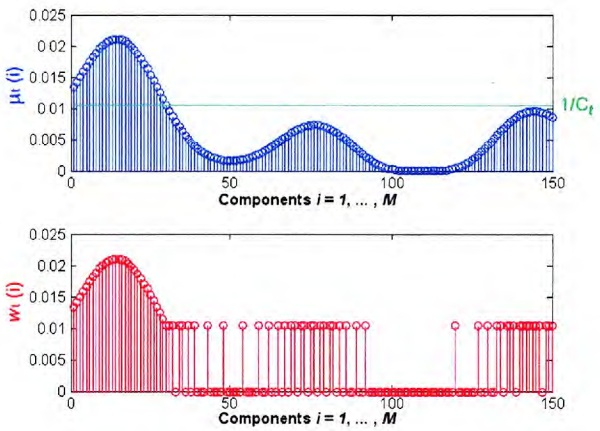


Figure 3.4: An example of sample selection using the Fearnhead and Clifford algorithm.

Theorem 3.2.3 Fix integers M and N ($N < M$). Let C_t be the unique solution of the equation $\sum_{i=1}^{M} \min(C_t \mu_t(i), 1) = N$. An approximation $w_t(i)$ minimises the expected L^2 error $E(\sum_{i=1}^{M} (w_t(i) - \mu_t(i))^2$ if the following conditions are satisfied:

- $E(w_t(i)) = \mu_t(i)$ for $i = 1, \ldots, M$.
- $w_t(i) = \begin{cases} \min(\mu_t(i), \frac{1}{C_t}) & \text{with probability } \min(1, C_t \mu_t(i)), \\ 0 & \text{otherwise}. \end{cases}$
- The number of nonzero support points in $w_t(i)$ must not be greater than N.

Proof. See [33] or, for a more detailed analysis, [32].

The distinctive feature of this algorithm is that terms with weights exceeding in magnitude the threshold $1/C_t$ (let these constitute set $\mathcal{A}_1$) are retained deterministically. From the set $\mathcal{A}_2$ of remaining terms, a random selection using stratified sampling (see Section 2.2.2) is used to fill the remaining 'slots'. $\mathcal{A}_3 \subset \mathcal{A}_2$ is the set of the samples chosen in this randomised manner; these are retained a weight of $1/C_t$. Algorithm 3.2.4 precisely describes this mixture reduction technique.

Algorithm 3.2.4 - the Fearnhead & Clifford algorithm

Step 1. Using Algorithm 3.2.5 below, calculate C_t as the unique solution to:

$$N = \sum_{i=1}^{M} \min(C_t \mu_t(i), 1)\,.$$

Step 2. Let $\mathcal{A}_1 = \{i | \mu_t(i) \geq 1/C_t\}$ and let $G = \text{card}(\mathcal{A}_1)$. Let $\mathcal{A}_2 = \{1, \ldots, M\} \backslash \mathcal{A}_1$.

Step 3. For $i \in \mathcal{A}_2$, using the *stratified sampling* procedure of Section 2.2.2(c), select $N - G$ distinct index values from $\mathcal{A}_2$ by sampling from:

$$\frac{\mu_t(i)}{\sum_{j \in \mathcal{A}_2} \mu_t(j)} \qquad i \in \mathcal{A}_2\,.$$

Let $\mathcal{A}_3 \subset \mathcal{A}_2$ be the set of components thus selected.

Step 4. The reduced N-fold mixture is obtained by combining those distributions indexed by $\mathcal{A}_1$ (with their original weight), and those indexed by the selected values in $\mathcal{A}_3$ (all with weight $1/C_t$). The components in $\mathcal{A}_2 \backslash \mathcal{A}_3$, whose new weight is zero, are discarded.

Figure 3.4 shows a realisation of the algorithm: the top chart displays the original weights of the distribution, and the threshold $1/C_t$; the bottom chart highlights how the components of set $\mathcal{A}_1$ retain their original weighting, those from $\mathcal{A}_3$ are set to $1/C_t$ and those in $\mathcal{A}_2 \backslash \mathcal{A}_3$ are set to 0. In this example the approximation has support on $N = 75$ nonzero points ($M = 150$). Finally note that while the original distribution in Figure 3.4 displays a well-defined shape resembling a Gaussian mixture, this has no significance and was chosen only for illustration purposes.

One way of understanding C_t is to see it as the threshold below which components can be randomly selected using stratified sampling without the possibility of selecting the same particle more than once. For this to be possible, we require 'Condition A':

$$\text{A)} \qquad \mu_t(i) < \frac{1}{C_t} \quad \text{for } i \in \mathcal{A}_2\,. \tag{3.2.1}$$

The choice of threshold should also ensure that the sum of the weights of the components in set $\mathcal{A}_1$ – which are propagated deterministically with their original weight (i.e. $w_t(i) = \mu_t(i)$ for $i \in \mathcal{A}_1$) – and of those in $\mathcal{A}_3$ – which are propagated randomly with weight $1/C_t$ (i.e. $w_t(i) = 1/C_t$ for $i \in \mathcal{A}_3$) – sum to 1. Mathematically, $\sum_{i\in\mathcal{A}_1} w_t(i) + \sum_{i\in\mathcal{A}_3} w_t(i) = 1$. For this to be possible, the total weight of the samples in $\mathcal{A}_2$ (i.e. $\sum_{\in\mathcal{A}_2} \mu_t(i)$) should be equal to the total weight of the selected samples in $\mathcal{A}_3$ (i.e. $\sum_{\in\mathcal{A}_3} w_t(i) = \frac{N-G}{C_t}$). Thus we introduce a second requirement on C_t (which we denote as 'Condition B') to ensure that:

$$\text{B)} \qquad \sum_{i\in\mathcal{A}_2} \mu_t(i) = \frac{N-G}{C_t}, \tag{3.2.2}$$

where $G = \text{card}(\mathcal{A}_1)$. Conditions A and B can be used in the calculation of the unique solution of C_t by means of the following algorithm:

Algorithm 3.2.5 - Calculation of C_t

Step 1. Let $\bar{\mu}_t$ be the permutation of μ_t which ensures decreasing order of weights. Set $G = 1$.

Step 2. Set $\tilde{C}_t = \frac{N-G}{\sum_{i=G+1}^{M} \bar{\mu}_t(i)}$ such that Condition B (3.2.2) is satisfied.

Step 3. Let Ω be the number of components for which $\bar{\mu}_t(i) > 1/\tilde{C}_t$, for $i = \{(G+1), \ldots, M\}$.

Step 4.

- If $\Omega = 0$, Condition A (3.2.1) is also satisfied, then set $C_t = \tilde{C}_t$ and terminate the algorithm.
- If $\Omega > 0$, Condition A (3.2.1) is not satisfied, then set $G = G + \Omega$ and go to Step 2.

The Fearnhead & Clifford algorithm is an 'unbiased' selection procedure (similar to the random sampling approach), which however reduces the variance of the randomisation process by introducing a significant component of determinateness. This is achieved at the same time as minimising a cost function based on the squared difference between the original and the approximated weight distributions, and avoiding the possibility of sample 'clones'. Furthermore, these favourable properties come at an affordable computational cost, particularly if compared to the distribution-based mixture reduction methods of Section 3.2.2.

d) A simulation study on weight-based algorithms

In this section, an empirical analysis is carried out to assess how the degree of randomisation in weight-based mixture reduction algorithms affects tracking errors in a manoeuvring target tracking scenario. On a scale ranging from pure randomness to pure determinateness, the random sampling approach and the DEA would place themselves respectively at the two extremes, while the Fearnhead and Clifford algorithm would be somewhere in the middle.

The α-algorithm

The first step to this analysis is the development of an algorithm based on a parameter, α, which governs the degree of randomness in the selection procedure. The algorithm works by thresholding the components' weights by $1/\alpha N$, those larger than this threshold are propagated deterministically with their own weights, those smaller are instead randomised. This algorithm is similar to that of Fearnhead and Clifford (where $1/C_t$ is now replaced by $1/\alpha N$), but unlike their method, the threshold is no longer a function of the original weight distribution and it can be pre-determined to model different degrees of randomness.

It is acknowledged that the $1/C_t$-threshold has some very desirable properties not present for other threshold values: if $1/\alpha N < 1/C_t$, stratified sampling on set $\mathcal{A}_2$ will potentially lead to the same sample being multiply selected; if on the other hand $1/\alpha N > 1/C_t$, more than the required N samples could be chosen deterministically. To deal with these conditions while still being able to test the entire spectrum, two *ad-hoc* decisions are made. Firstly, when more than N components exceed the threshold, only the N largest are selected (such as in the DEA). Secondly, to ensure that no component is needlessly carried forward more than once, random sampling is carried out using a version of Algorithm 3.2.1 modified to ensure that the lottery continues until N *distinct* components are selected. In this case the weights of the samples selected is made proportional to the number of extractions.

This choice of randomisation procedure is the main *ad-hoc* element in these 'α-algorithm' investigations, and might not be suitable for all types of applications. In fact, for prob-

lems where the weight distribution is likely to be very skewed (some weights very large and some others very small), computation times might sometimes be significantly increased since randomisation is not complete until N *distinct* components are selected at least once. This problem is not noticeable in the manoeuvring target tracking scenarios tested in this thesis, but could arise, for example, in tracking problems involving clutter. An alternative solution would be to allow identical samples to be propagated more than once (as is the case in Algorithm 3.2.1). However, it was deemed that the first approach would allow a fairer comparison with the Fearnhead and Clifford algorithm, and has been followed in this empirical analysis.

It should finally be noted that when $\alpha = 0$, the algorithm reduces to purely random selection (as the threshold approaches infinity), while when $\alpha \rightarrow \infty$ the threshold approaches zero and the selection procedure is purely deterministic and identical to the DEA. In practice, to ensure that at least N components are above the threshold, α was set to some high value (typically $\alpha = 1,000$).

Algorithm 3.2.6 - The α-algorithm

Step 1. Let $\mu_t(i)$ be the weights of the components of a M-fold Gaussian mixture. Let $\mathcal{A}_1 = \{i|\mu_t(i) \geq \frac{1}{\alpha N}\}$ be the set of components chosen deterministically, and let $G = \text{card}(\mathcal{A}_1)$. If $G > N$, select the N largest components and terminate the algorithm.

Step 2. Let $\mathcal{A}_2 = \{1, \ldots M\} \backslash \mathcal{A}_1$ be the set of residual components. Let $\tilde{N}_i = 0 \ \forall i \in \mathcal{A}_2$, and let:

$$\bar{\mu}_t(i) = \frac{\mu_t(i)}{\sum_{j \in \mathcal{A}_2} \mu_t(j)} \qquad i \in \mathcal{A}_2 \, .$$

Step 3. Draw ψ from the uniform distribution $\mathcal{U}(0,1)$ and find the component R, with possible values in $\mathcal{A}_2$, such that:

$$\sum_{j=1}^{R-1} \bar{\mu}_t(j) < \psi \leq \sum_{j=1}^{R} \bar{\mu}_t(j) \, .$$

Let also $\tilde{N}_R = \tilde{N}_R + 1$.

Step 4. If the number of components selected at least once (i.e. those $i \in A_2$ for which $\tilde{N}_i > 0$) is equal to $N - G$, go to Step 5. Otherwise repeat Step 3.

Step 5. Let $\mathcal{A}_3$ be the set (of cardinality $N - G$) of components for which $\tilde{N}_i > 0$. Then the weights of the components in $\mathcal{A}_3$ are given by:

$$w_t(i) = \tilde{N}_i \left(\frac{1 - \sum_{j \in \mathcal{A}_1} \mu_t(j)}{\sum_{j \in \mathcal{A}_3} \tilde{N}_j} \right) \qquad i \in \mathcal{A}_3 \, .$$

Step 6. The reduced N-fold mixture is obtained by combining those distributions indexed by $\mathcal{A}_1$ (with their original weight), and those indexed by the selected values in $\mathcal{A}_3$ (all with weight $w_t(i)$, for $i \in \mathcal{A}_3$). The components in $\mathcal{A}_2 \backslash \mathcal{A}_3$, whose new weight is zero, are eliminated.

Effects of randomness and determinateness

The first test in this section aims to illustrate the expected difference between the purely randomised and deterministic versions of the algorithm. Deterministic algorithms, such as the DEA, select components on the basis of maximum *a posteriori* probability and are thus likely to retain samples which are most closely related to the latest measurements

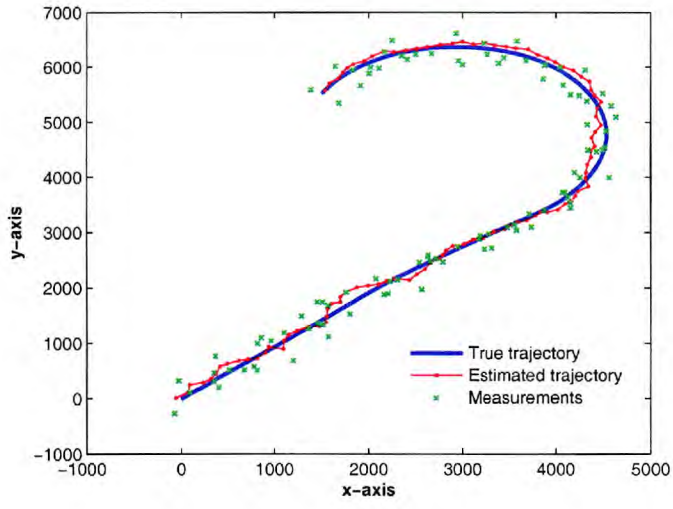


Figure 3.5: Sample realisation of Track 2 using the mixture filter and the Fearnhead & Clifford mixture reduction algorithm (N=50).

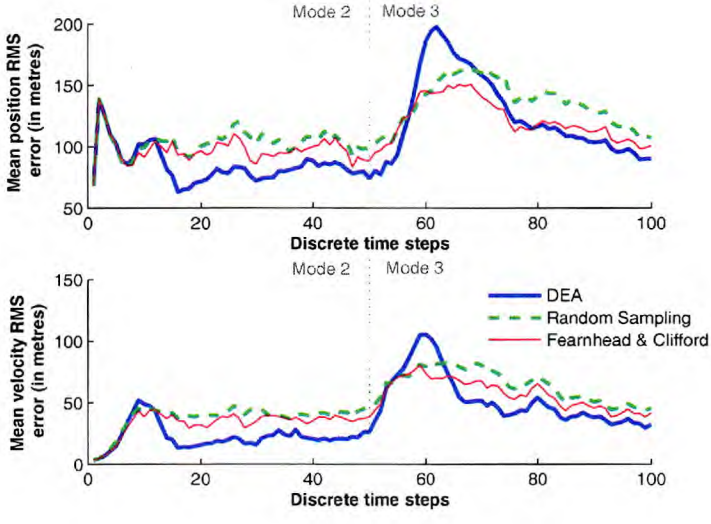


Figure 3.6: RMS error comparison of different weight-based mixture reduction techniques for Track 2 using $N = 10$ samples (100 Monte Carlo average).

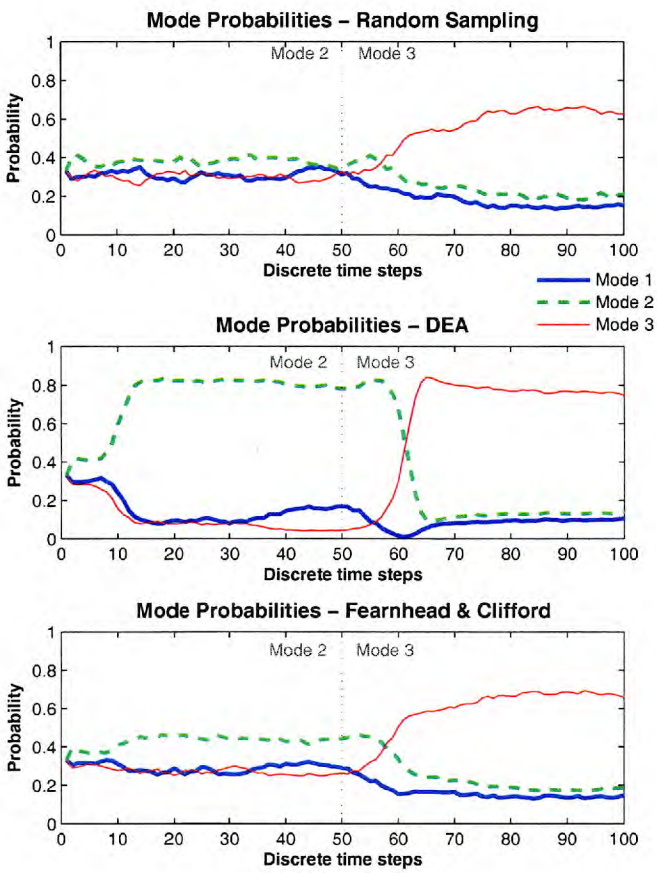


Figure 3.7: Mode probabilities comparison of different weight-based mixture reduction techniques for Track 2 using $N = 10$ samples (100 Monte Carlo average).

and which are likely to be similar to each other. The advantage of this approach should be that, when the system is stable on a certain mode for a length of time, the deterministic selection should help minimise errors. On the other hand, randomised algorithms are likely to retain a richer mix of samples which will include a number of components associated with modes inconsistent with the latest measurements. The advantage of this approach should be that as soon as a mode transition is recognised, the algorithm should already be likely to have in store a number of samples associated with the new mode.

Track 2 (shown in Figure 3.5), generated using the JMLS model of Section 2.3.1, was designed to highlight these differences. The target settles on a non-manoeuvring mode ($r_{0:50} = 2$) for 50 consecutive time steps, then switches to a left turn for the remainder of the trajectory ($r_{51:100} = 3$). This simple manoeuvring structure was chosen to highlight how different resampling algorithms 'settle' on a specific mode, and how the same algorithms react to a mode change after a long stable period. 100 Monte Carlo simulations were carried out using Algorithm 3.2.6 using $\alpha = 0$, to test the randomised hypothesis, and using $\alpha = 1000$ to test the DEA; the Fearnhead & Clifford algorithm was also tested. The parameters used in these simulations are identical to those used in Section 2.3.5.

Figure 3.6 shows the average RMS error of the three algorithms over the duration of the trajectory, and Figure 3.7 shows the mode probabilities over the same period. As predicted, the deterministic algorithm has the lowest error over long periods of time during which the mode is constant, and it attaches a high probability to the correct mode. Immediately after the mode changes however, as only very few samples model the correct mode (the total probability is around 0.1), the DEA gives a very high peak error.

On the other hand, the fully randomised algorithm shows higher errors during 'stationary' periods, but owing to the significant number of samples it retains associated with other modes, it 'damps' the error after the transition. Finally, the Fearnhead and Clifford algorithm shows properties midway between the two extremes, however in this scenario its performance is more similar to that of the randomised filter.

Testing the α-range

The entire range of parameter α, which measures the degree of randomness, is now tested for Track 1 (which was described in Section 2.3.5). As can be observed in Figure 3.8, pure randomisation causes the highest average error and, while deterministic selection provides significantly lower errors, a minimum can be found at around $\alpha = 6$. The Fearnhead and Clifford method also outperforms both extremes but not as evidently. Figure 3.9 shows the RMS position and velocity errors for this track for the three standard weight-based methods as well as for the minimising value of α ($\alpha = 6$). Figure 3.10 shows the respective mode probabilities. This choice of α seems to strike the right balance between tracking the correct mode and recovering after a transition (as reflected by the mode transition probabilities of Figure 3.10), and produces low estimation errors throughout the track.

Finally, in Figure 3.11 the RMS errors for these mixture reduction schemes are plotted against computational time for varying number of mixture components. These results are superimposed on the results of some of the algorithm already tested in Section 2.3.5 (Figure 2.10). We can see how the random sampling technique ($\alpha = 0$) behaves similarly to the RBPF with stratified sampling, however the more deterministic algorithms have a clear advantage. In this scenario, the Fearnhead & Clifford algorithm has similar performance to the DEA, while the α-algorithm with $\alpha = 6$ shows remarkable further improvements.

Unfortunately, the value of α producing minimum error is problem-dependent. It will vary depending on the target and on the frequency of its transitions between modes, which is an unknown. Empirically, it was found that – for an average track with number of manoeuvres reasonably consistent with the expected Markov chain mode transition probabilities – values in the range between $3 < \alpha < 7$ generally reduce the minimum error compared to the other methods, but again we stress that the actual value is problem dependent.

A number of *ad hoc* algorithms, adaptively varying the value of α to ensure that the

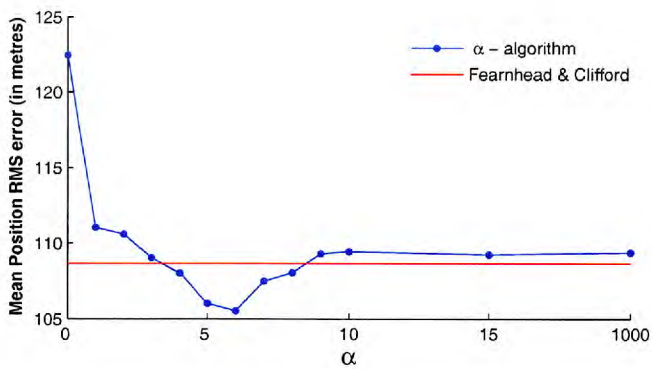


Figure 3.8: Mean RMS position error produced by the α-algorithm for varying values of α. These results were produced for Track 1 using $N = 10$ samples (100 Monte Carlo average). The average RMS error produced by the Fearnhead & Clifford algorithm is also shown (as a straight red line).

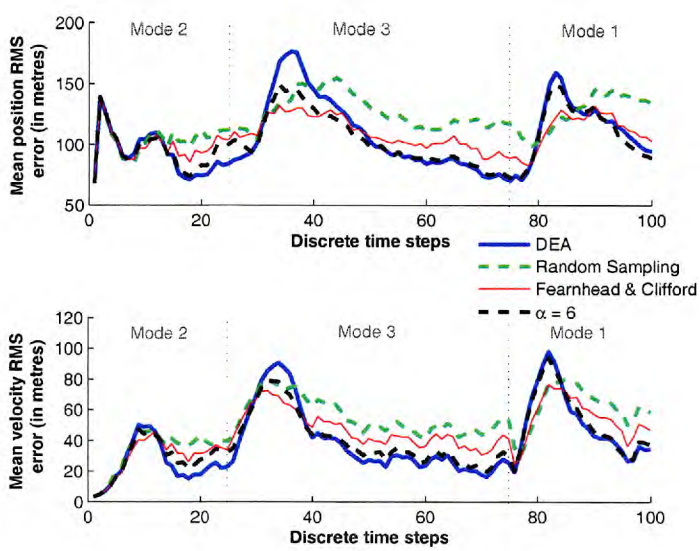


Figure 3.9: RMS error comparison of different weight-based mixture reduction techniques for Track 1 using $N = 10$ samples (100 Monte Carlo average).

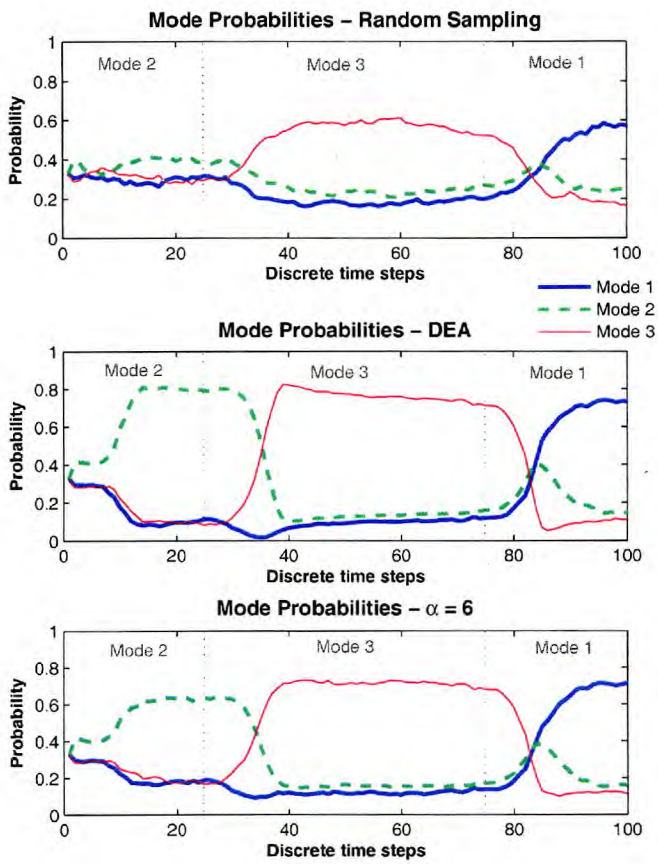


Figure 3.10: Mode probabilities comparison of different weight-based mixture reduction techniques for Track 1 using $N = 10$ samples (100 Monte Carlo average).

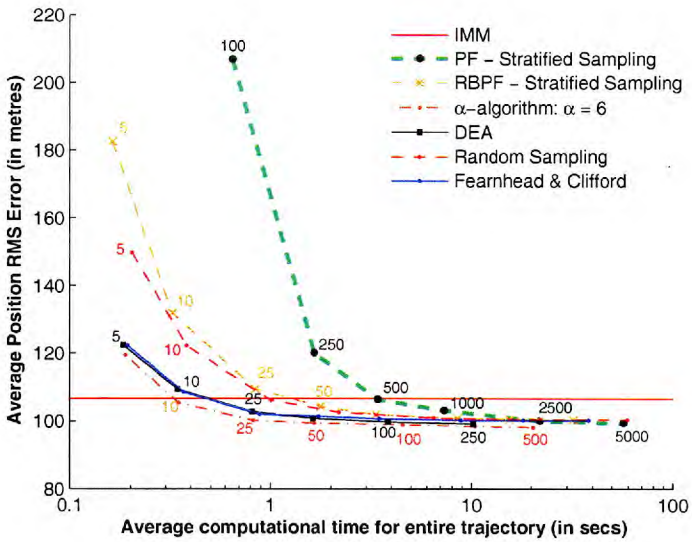


Figure 3.11: Position RMS error vs. computational cost comparisons for Track 3 and for varying number of mixture components. Component numbers are indicated where possible. For the mixture filters and RBPF, the range shown is: [5, 10, 25, 50, 100, 250, 500, 1000]; for the MMBF: [100, 250, 500, 1000, 2500, 5000].

mode probabilities rapidly converge on a given mode, while also retaining a certain presence in the other modes, can also be constructed. While not always as effective as the α-algorithm with the appropriate value of α, the Fearnhead & Clifford algorithm provides a good compromise between randomness and determinateness and is used as the 'weight-based' mixture reduction algorithm of reference in the remaining sections of this thesis.

3.2.2 Distribution-based mixture reduction

So far, the mixture reduction techniques investigated reduce the mixture on the basis of the component weights alone. A wealth of other information could however be incorporated in this selection process, and in this respect the spatial configuration of the constituent densities (the location of their means and the nature of their covariance matrices) is of particular significance. This section discusses techniques which attempt to minimise cost functions that take into consideration this extra information. These methods are referred to as 'distribution-based' mixture reduction techniques.

a) Salmond's mixture reduction algorithm

Distribution-based mixture reduction techniques were first introduced by Salmond [61]-[62], who proposed two algorithms aimed at improving on the popular *probabilistic data association filter* (PDAF) [13] for the problem of tracking a single non-manoeuvring target in clutter. Salmond's algorithms, known as the *joining* and the *clustering algorithms*, work on the principle that approximating a Gaussian mixture distribution using multiple Gaussian terms was preferable to using a single component, as in the PDAF. As such, his sub-optimal algorithms were designed to approximate a Gaussian mixture by a lower order one, while at the same time reducing information loss. This section outlines Salmond's algorithms.

The Mahalanobis distance

Salmond's algorithms reduce a Gaussian mixture pdf by combining those constituents that are, in some sense, similar. In order to determine which particular reduction combination will cause a minimal degradation to the original pdf, Salmond [62] suggests the use of a cost function which measures the modification to the structure of the Gaussian mixture.

Consider a mixture with M Gaussian constituent densities with mean, covariance and weights given by $\{\hat{\boldsymbol{x}}_i, \boldsymbol{P}_i, \mu_i\}, i \in \{1, \ldots, N\}$. Then the mean and the covariance of this distribution are:

$$\begin{aligned} \bar{\boldsymbol{x}} &= \sum_{i=1}^{M} \mu_i \hat{\boldsymbol{x}}_i \,, \\ \bar{\boldsymbol{P}} &= \boldsymbol{P}^w + \boldsymbol{P}^b \\ &= \sum_{i=1}^{M} \mu_i \boldsymbol{P}_i + \sum_{i=1}^{M} \mu_i (\hat{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\hat{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})^T \,. \end{aligned} \tag{3.2.3}$$

The structure of the covariance of the mixture is composed of two components: the covariance 'within' constituent densities, $\boldsymbol{P}^w$, and the covariance 'between' constituent densities, $\boldsymbol{P}^b$. If any subset, $\mathcal{C}$, of the constituent densities is now replaced by a single

Gaussian density with mean, covariance and weight given by:

$$\bar{\boldsymbol{x}}_{\mathcal{C}} = \frac{1}{\bar{\mu}_{\mathcal{C}}} \sum_{j \in \mathcal{C}} \mu_j \hat{\boldsymbol{x}}_j \,, \tag{3.2.4}$$

$$\bar{\boldsymbol{P}}_{\mathcal{C}} = \frac{1}{\bar{\mu}_{\mathcal{C}}} \sum_{j \in \mathcal{C}} \mu_j (\boldsymbol{P}_j \,, + (\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_{\mathcal{C}})(\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_{\mathcal{C}})^T) \,, \tag{3.2.5}$$

$$\bar{\mu}_{\mathcal{C}} = \sum_{j \in \mathcal{C}} \mu_j \,, \tag{3.2.6}$$

then the covariance of the resulting Gaussian mixture, which has cardinality equal to $(M - \text{card}(\mathcal{C}) + 1)$, is still equal to $\bar{\boldsymbol{P}}$. However – unless the constituents of $\mathcal{C}$ are identical in mean and covariance – the structure of $\bar{\boldsymbol{P}}$ changes: $\boldsymbol{P}^w$ increases, while $\boldsymbol{P}^b$ decreases. The size of this shift can be calculated as:

$$\begin{aligned} \boldsymbol{L} &= \bar{\mu}_{\mathcal{C}} \bar{\boldsymbol{P}}_{\mathcal{C}} - \sum_{j \in \mathcal{C}} \mu_j \boldsymbol{P}_j \\ &= \bar{\mu}_{\mathcal{C}} \frac{1}{\bar{\mu}_{\mathcal{C}}} \sum_{j \in \mathcal{C}} \mu_j (\boldsymbol{P}_j + (\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_{\mathcal{C}})(\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_{\mathcal{C}})^T) - \sum_{j \in \mathcal{C}} \mu_j \boldsymbol{P}_j \\ &= \sum_{j \in \mathcal{C}} \mu_j (\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}})(\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_{\mathcal{C}})^T \,. \end{aligned}$$

This shift in the covariance structure reflects the distortion introduced to the Gaussian mixture, and can be used as a cost function to assess viable reduction strategies. As shown in [62], if two constituent components, i and j, are considered, the increase in $\boldsymbol{P}^w$ that is caused by their combination is given by:

$$\boldsymbol{L}_{ij} = \frac{\mu_i \mu_j}{\mu_i + \mu_j} (\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j)(\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j)^T \,. \tag{3.2.7}$$

The trace of $\boldsymbol{L}_{ij}$ would then be a good cost function, if not for that fact that it is problem dependent. A preferable choice, in which the distances are normalised, is known as the Mahalanobis [38] distance, which is given by:

$$\Delta_{ij}^2 = \frac{\mu_i \mu_j}{\mu_i + \mu_j} (\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j)^T \bar{\boldsymbol{P}}^{-1} (\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j) \,. \tag{3.2.8}$$

Here $\bar{\boldsymbol{P}}$ is the combined covariance of the original pdf as given in (3.2.3). This measure is equivalent to $\text{tr}(\bar{\boldsymbol{P}}^{-1} \boldsymbol{L}_{ij})$.

The joining algorithm

The first algorithm proposed by Salmond, known as the *joining algorithm*, is an agglomerative clustering method, which reduces the mixture to the desired number of terms by

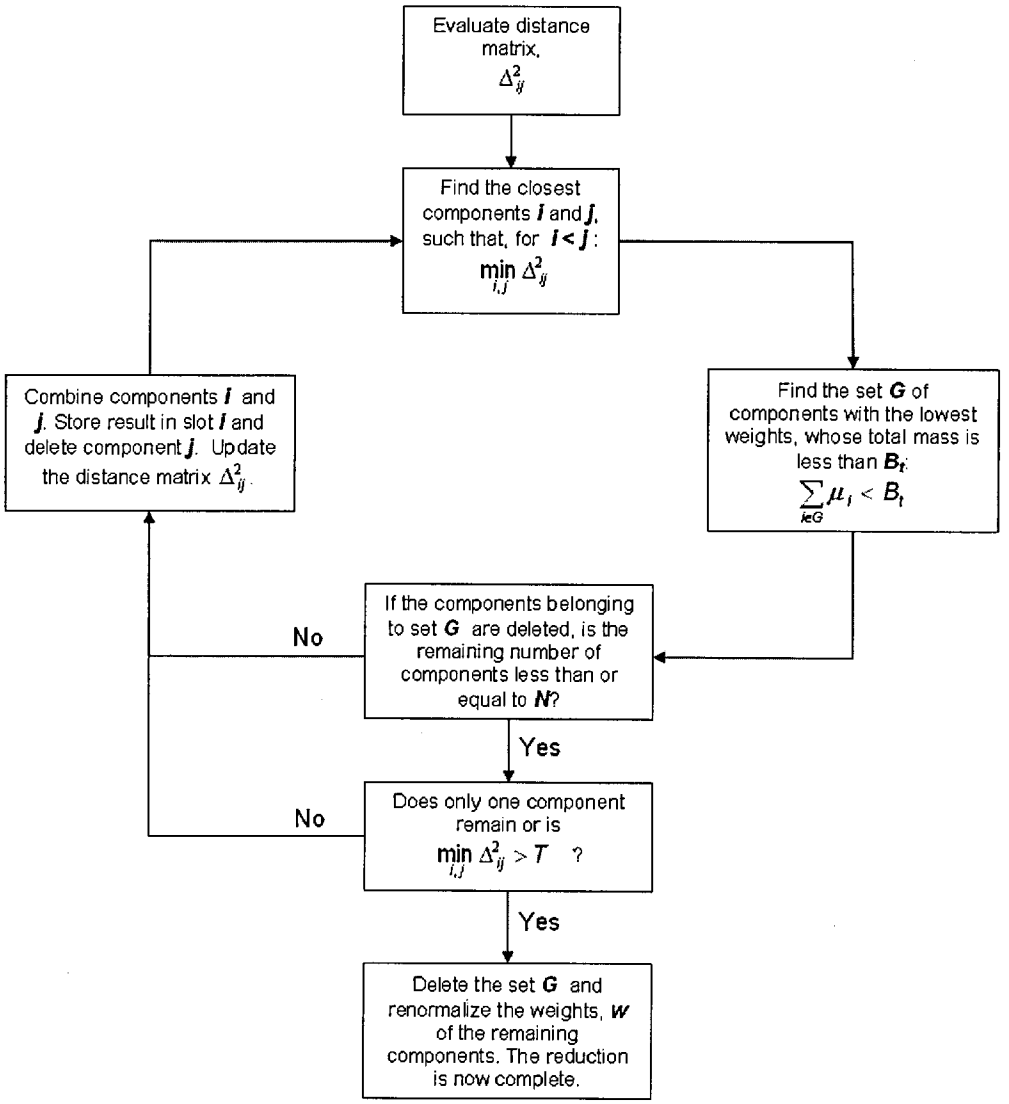


Figure 3.12: The structure of the joining algorithm.

combining the two closest components at each iteration.

Figure 3.12[1] shows a flowchart of the joining algorithm. At the beginning of each recursion, the algorithm calculates the distance matrix, $\Delta_{ij}^2 \; \forall \, i, j \in \{1, \ldots, M\}$ using (3.2.8), and combines the two components which are closest to each other (i.e. those components i and j which satisfy $\min(\Delta_{ij}^2)$). The distance matrix is then amended and the process

[1]Figures adapted from [61].

repeated until the number of components is reduced to the required level or until the modification to the original mixtures exceeds a certain threshold (see Salmond [61] for more information on threshold selection).

Although the joining algorithm choses the minimum cost strategy at each iteration, there is no guarantee that the final mixture will have caused the minimum possible distortion (global minimum cost). Furthermore, given its recursive nature, the algorithm can be computationally intensive, particularly if compared to weight-based mixture reduction methods.

The clustering algorithm

The *clustering algorithm* attempts to reduce the computational demands of the joining algorithm by merging constituent components in groups rather than in pairs. As shown in Figure 3.13[1], the algorithm starts by ranking components in decreasing order of weights, following the assumption that these carry the most important information. The algorithm then considers the first component (i.e. the one with the largest weight) to create a new cluster, and computes the distance of all the remaining components. This is done using a modified version of the Mahalanobis distance (see (3.2.8)), in which the distance is normalised by means of the covariance of the cluster instead of that of the entire pdf. Thus the distance of component i from the cluster centre is now given by:

$$\Delta^2_{\mathcal{C}i} = \frac{\mu_i \mu_{\mathcal{C}}}{\mu_i + \mu_{\mathcal{C}}} (\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{\mathcal{C}})^T \boldsymbol{P}_{\mathcal{C}}^{-1} (\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{\mathcal{C}}) \,, \tag{3.2.9}$$

where $\hat{\boldsymbol{x}}_{\mathcal{C}}$, $\boldsymbol{P}_{\mathcal{C}}$, and $w_{\mathcal{C}}$ – calculated as in (3.2.4)-(3.2.6)– are respectively the mean, the covariance and the weight of a cluster created by merging the densities in $\mathcal{C}$, a subset of the constituents Gaussian densities of the original pdf. Once this step is completed, any component whose distance is below a threshold T is merged with the cluster. Salmond suggests $T = 0.05T_1$, where T_1 determines the hyperellipsoid in state space containing 1% of the probability mass of the principal component of the cluster and, as shown in [61], can be calculated using χ^2 tables such that $T = 0.01485$.

The algorithm then checks the total combined probability of the remaining densities: if this is below a minimum threshold, B_t, (see [61]), these are discarded, if not the

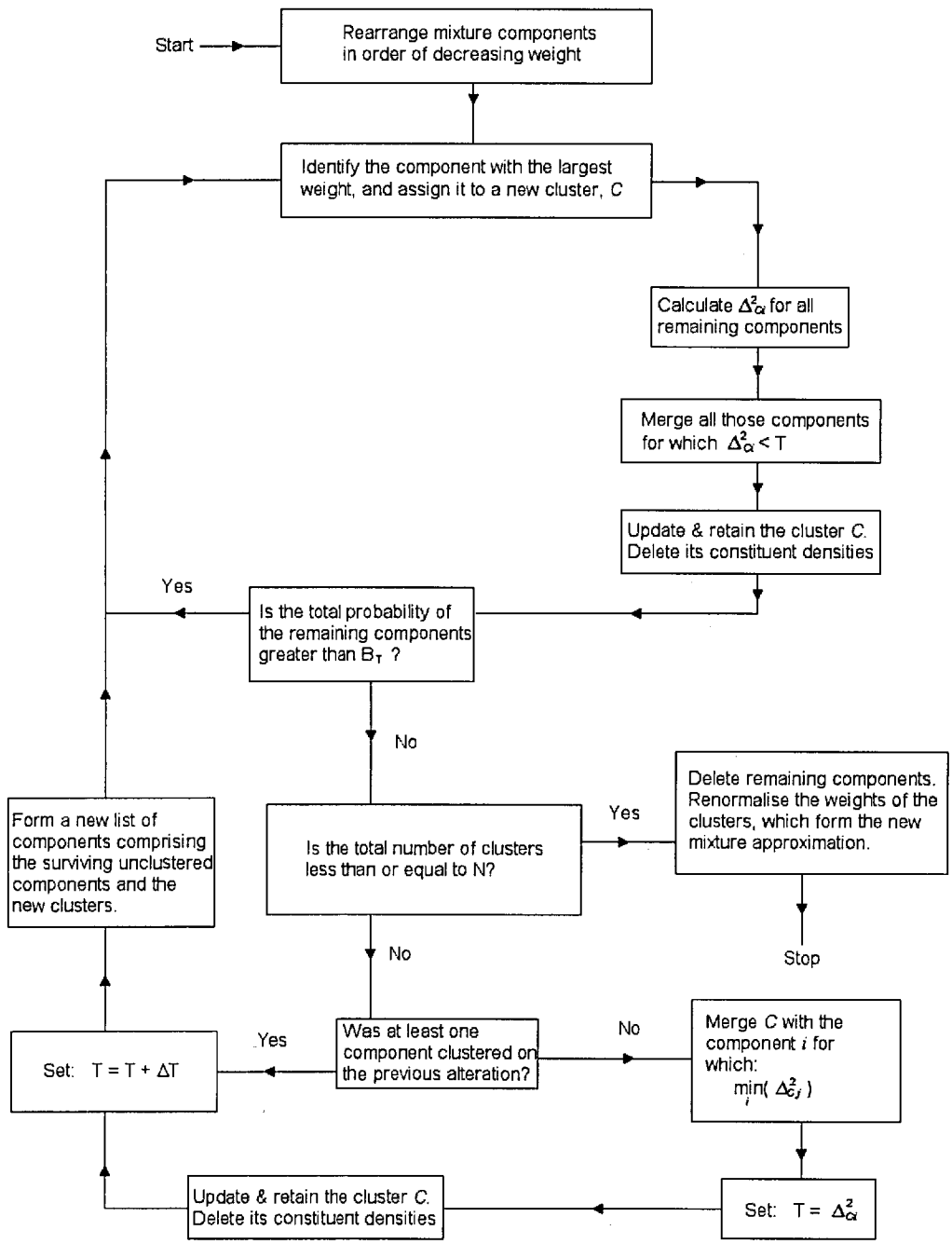


Figure 3.13: The structure of the clustering algorithm.

process is repeated with the surviving largest weight component forming a new cluster. If by the end of the recursion the number of components is still too large, the clustering algorithm repeats the entire process on the current approximation using a larger thresh-

old. Salmond suggests incrementing T by $\Delta T = 0.05\Delta T_1$, where $T_1 + \Delta T_1$ defines the hyperellispoid containing 6% of the probability of mass of the principal component of the cluster. Using χ^2 tables, $\Delta T = 0.02065$. As shown in Figure 3.13, Salmond also introduces an override to ensure that at least a single reduction takes place at every iteration.

Although the clustering algorithm tends to be considerably faster than the joining algorithm (and particularly so for large mixtures), its approximations do not tend to be as accurate. Furthermore if enough reduction strategies have a cost smaller than the threshold, the algorithm approximates the Gaussian mixture with less than N components. When this happens, the degree of approximations is wastefully high.

b) The integral square difference algorithms

Williams and Maybeck [72]-[73] have proposed two algorithms to obtain a solution to the mixture reduction problem, aimed at achieving greater accuracy. These extremely computational demanding techniques make use of a different cost function, known as *integral square difference* (ISD) which takes into consideration the entire spatial configuration of the elements of the pdf, and not only the shift in the 'within' covariance of the approximation. The first algorithm, known as *ISD initialisation* (ISDI) algorithm, produces an initial approximation. The second algorithm is an optimisation algorithm which improves the initial approximations.

The ISD cost function

The *integral square difference* (ISD) cost function, as proposed by Williams and Maybeck [72]-[73], is a measure of the square difference between two pdfs (3.2.10). It can be used in mixture reduction in order to measure the difference between the original distribution and its reduced form, and to measure how much degradation is introduced. In this fashion, the ISD cost function can be used as a parameter to be minimised in the development of mixture reduction algorithms.

Following the notation of [73], the ISD cost function can be expressed as:

$$J = \int (f(x_t|\Omega_t^o) - f(x_t|\Omega_t^r))^2 dx_t \,, \tag{3.2.10}$$

where:

$$f(\boldsymbol{x}_t|\Omega_t^o) = \sum_{i=1}^{M} \mu_i^o \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_i^o, \boldsymbol{P}_i^o)\,, \tag{3.2.11}$$

and

$$f(\boldsymbol{x}_t|\Omega_t^r) = \sum_{i=1}^{N} \mu_i^r \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_i^r, \boldsymbol{P}_i^r)\,, \tag{3.2.12}$$

are the joint pdfs of the state given the parameters of the components of the M-fold original and of the N-fold approximated distributions (referred to as Ω_t^o and Ω_t^r respectively). To distinguish between the two pdfs, the mean, covariance and weights of the components of the original distribution will be labelled as ($\hat{\boldsymbol{x}}^o$,$\boldsymbol{P}_i^o$,μ_i^o) respectively, while those of the reduced distribution as ($\hat{\boldsymbol{x}}^r$,$\boldsymbol{P}_i^r$,μ_i^r).

By simplifying the product of two Gaussians to:

$$\mathcal{N}(x; \hat{\boldsymbol{x}}_1, \boldsymbol{P}_1)\mathcal{N}(x; \hat{\boldsymbol{x}}_2, \boldsymbol{P}_2) = \alpha \mathcal{N}(x; \hat{\boldsymbol{x}}_3, \boldsymbol{P}_3)\,, \tag{3.2.13}$$

where:

$$\alpha = \mathcal{N}(\hat{\boldsymbol{x}}_1; \hat{\boldsymbol{x}}_2, \boldsymbol{P}_1 + \boldsymbol{P}_2)\,, \tag{3.2.14}$$

$$\boldsymbol{P}_3 = (\boldsymbol{P}_1^{-1} + \boldsymbol{P}_2^{-1})^{-1}\,, \tag{3.2.15}$$

$$\hat{\boldsymbol{x}}_3 = \boldsymbol{P}_3(\boldsymbol{P}_1^{-1}\hat{\boldsymbol{x}}_1 + \boldsymbol{P}_2^{-1}\hat{\boldsymbol{x}}_2)^{-1}\,, \tag{3.2.16}$$

Williams [72] shows how the cost function can be expressed in terms of:

$$J = J_{MM} - 2J_{MN} + J_{NN}\,. \tag{3.2.17}$$

These terms simplify to:

$$J_{MM} = \int f(\boldsymbol{x}_t|\Omega_t^o)^2 d\boldsymbol{x}_t \tag{3.2.18}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{M} \mu_i^o \mu_j^o \mathcal{N}(\hat{\boldsymbol{x}}_i^o; \hat{\boldsymbol{x}}_j^o, \boldsymbol{P}_i^o + \boldsymbol{P}_j^o)\,, \tag{3.2.19}$$

$$J_{MN} = \int f(\boldsymbol{x}_t|\Omega_t^o) f(\boldsymbol{x}_t|\Omega_t^r) d\boldsymbol{x}_t \tag{3.2.20}$$

$$= \sum_{i=1}^{M}\sum_{j=1}^{N} \mu_i^o \mu_j^r \mathcal{N}(\hat{\boldsymbol{x}}_i^o; \hat{\boldsymbol{x}}_j^r, \boldsymbol{P}_i^o + \boldsymbol{P}_j^r)\,, \tag{3.2.21}$$

$$J_{NN} = \int f(\boldsymbol{x}_t|\Omega_t^r)^2 d\boldsymbol{x}_t \tag{3.2.22}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \mu_i^r \mu_j^r \mathcal{N}(\hat{\boldsymbol{x}}_i^r; \hat{\boldsymbol{x}}_j^r, \boldsymbol{P}_i^r + \boldsymbol{P}_j^r)\,. \tag{3.2.23}$$

J_{MM} and J_{NN} are respectively the *self-likeness* of the original and of the reduced distributions, while J_{MN} is known as the *cross-likeness*. The latter balances the former terms by cancelling out any common part, causing this cost function to have the desirable property of being zero when the distributions are identical. J reaches its maximum value when the product of the two distributions is zero at every point.

A further property of this L^2 cost function is that higher importance is given to those mixture components which are more concentrated (i.e. with a lower covariance) and less to those which are more spread out. According to Williams [72], this is desirable as areas with larger errors (i.e. higher variance) are given lower weights than more accurate ones.

The ISD initialisation algorithm

With minimisation of the ISD cost function in mind, Williams and Maybeck develop a two-stage mixture reduction algorithm which attempts to be optimal. The first stage, known as the *ISD initialisation* (ISDI) algorithm, seeks out an initial approximation to the original pdf. The ISDI works according to the same principles of Salmond's joining algorithm, except for using the ISD cost function instead of the Mahalanobis distance. It also considers the option of deleting mixture components, and not just joining them.

At each iteration, the initialisation algorithm starts by calculating the number of possible reduction strategies (all the possible merging combinations as well as all the possible deletions) and for each one it calculates the resulting ISD cost. The best strategy is then implemented, the ISD cost table is updated, and the operation is repeated until the number of components drops to the desired level.

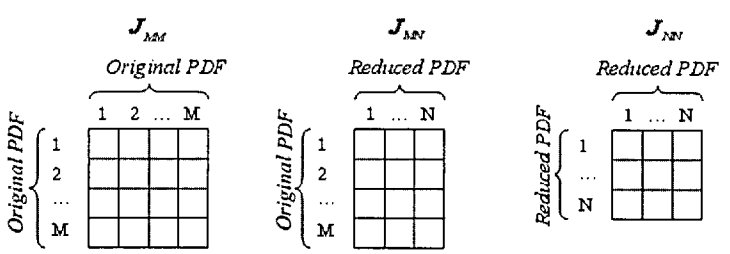


Figure 3.14: Decomposing the ISD cost function (adapted from [72]).

While this may sound computationally prohibitive, an analysis of the structure of J shows how considerable computational savings can be achieved. As shown in Figure 3.14, notice how J_{MM} is an $M \times M$ matrix, J_{MN} a $M \times N$ matrix and finally J_{NN} a $N \times N$ matrix. However J_{MM} is independent of the reduction being carried out and can therefore be computed only once. Furthermore J_{MN} and J_{NN} share most of their components with it. As – when merging components – a single reduction is carried out per iteration, the only elements that need to be re-evaluated are those in one column of J_{MN}, and those in one column and one row of J_{NN}. When deleting components, the calculation of the ISD cost is even simpler: the columns and rows related to the component being considered simply get discarded.

Despite computational economies achievable in this way, this algorithm remains extremely computationally intensive (even compared to Salmond's joining algorithm).

The iterative ISD optimisation algorithm

The second algorithm proposed by Williams [72] is an iterative optimisation algorithm which, using gradient techniques, modifies the means, covariance and weights of the components produced using the ISDI algorithm with the aim of having them converge to a cost minimum. Thus the iterative *ISD optimisation* (ISDO) algorithm essentially fine-tunes the reduced components in an effort to minimise the difference between the original and the reduced mixtures (see [72] for mode details).

Needless to say, this optimisation step further increases the already significant computational load required by the initialisation step, making the use of this algorithm impracticable in many real scenarios. Furthermore, according to Williams [72], no clear performance improvements are observed when using this extra step, and the initialisation estimates were even shown to frequently outperform the optimised parameters. This implies that the ISDO often converges to a local minimum instead of the global one. This local minimum, while reducing the overall ISD cost with respect to the starting point, might not necessarily represent the best solution from a tracking point of view.

3.2.3 Applying distribution-based algorithms to multiple-model problems

When using distribution-based techniques in multiple-model tracking problems such as manoeuvring target tracking, an extra difficulty arises in the treatment of clusters formed from the combination of components associated with different modes. As illustrated in Figure 3.15, there are two possible ways of dealing with this extra complication: allowing individual components to be associated with more than one mode, or restricting clustering to components of the same mode. We focus on the former approach and propose a modification to the standard mixture filter in which each mixture component is associated with a vector of mode probabilities, rather than on a single mode.

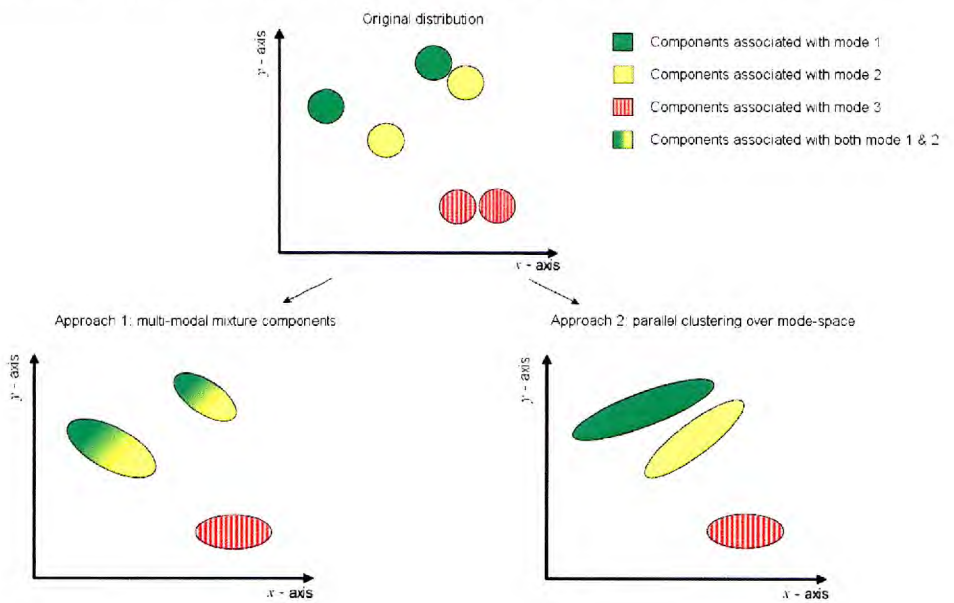


Figure 3.15: Clustering strategies for multiple-model problems.

a) The multi-modal mixture component approach

The first approach involves exact calculations of the mode transition probabilities at time t given that the prior components $\hat{\boldsymbol{x}}_{t-1}^{n}$ are clusters of terms belonging to different modes. This can be done by generalising the mixture filter model presented in Section 3.1.1 so that the mode transition probability calculations no longer assume that parent particles are associated with a single mode. Consider a M-fold Gaussian mixture representing an approximation to the conditional distribution at time $t-1$. Generally, for the multiple-

model problem investigated in this thesis, $M = N \times d$, however for tidiness of notation we here replace the double index $n \in \{1, \ldots, N\}, j \in \{1, \ldots, d\}$, with the single index $i \in \{1, \ldots, M = N \times d\}$. Let the constituent components of this conditional distribution have mean, covariance, mode and weight described by:

$$\left\{\hat{\boldsymbol{x}}_{t-1}^{i}, \boldsymbol{P}_{t-1}^{i}, \hat{r}_{t-1}^{i}, \mu_{t-1}^{i}\right\}.$$

Let us now suppose that a mixture reduction algorithm (that takes no account of the mode value) creates N clusters, $\mathcal{C}_{t-1}^{n}$, for $n \in 1, \ldots, N$. The components to be clustered are then:

$$\{\hat{\boldsymbol{x}}_{t-1}^{k}, \boldsymbol{P}_{t-1}^{k}, \hat{r}_{t-1}^{k}, \mu_{t-1}^{k}\}_{k \in \mathcal{C}_{t-1}^{n}}.$$

If $\mathcal{C}_{t-1}^{0}$ is the set of those components to be discarded, then:

$$\bigcup_{n=0}^{N} \mathcal{C}_{t-1}^{n} = \{1, \ldots, M\}.$$

Each of these clusters can be replaced by a single weighted Gaussian density $\left\{\hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, \mu_{t-1}^{n}\right\}$, where:

$$\hat{\boldsymbol{x}}_{t-1}^{n} = \sum_{k \in \mathcal{C}_{t-1}^{n}} \mu_{t-1}^{k} \hat{\boldsymbol{x}}_{t-1}^{k}, \tag{3.2.24}$$

$$\boldsymbol{P}_{t-1}^{n} = \sum_{k \in \mathcal{C}_{t-1}^{n}} \mu_{t-1}^{k} \left(\boldsymbol{P}_{t-1}^{k} + (\hat{\boldsymbol{x}}_{t-1}^{k} - \hat{\boldsymbol{x}}_{t-1}^{n})(\hat{\boldsymbol{x}}_{t-1}^{k} - \hat{\boldsymbol{x}}_{t-1}^{n})^{T}\right), \tag{3.2.25}$$

$$\mu_{t-1}^{n} = \sum_{k \in \mathcal{C}_{t-1}^{n}} \mu_{t-1}^{k}. \tag{3.2.26}$$

The treatment of the modes is more complicated. Now, several mode values (and corresponding weights) are associated with each sub-mixture associated with the set of index values $\mathcal{C}_{t-1}^{n}$, and this information needs to be carried forward in the reduction process.

This can be done by constructing $\boldsymbol{\gamma}_{t-1}^{n}$, a $(d \times 1)$-vector containing the 'mode probabilities' of the elements in a given cluster $\mathcal{C}_{t-1}^{n}$. The probability of a given mode l based on these components is then the (normalised) sum of the weights of all those components $k \in \mathcal{C}_{t-1}^{n}$ whose mode $\hat{r}_{t-1}^{k} = l$. Introduce the Kronecker delta , $\delta(\hat{r}_{t-1}^{k}, l)$, where:

$$\delta(\hat{r}_{t-1}^{k}, l) = \begin{cases} 1 & \text{if } \hat{r}_{t-1}^{k} = l \\ 0 & \text{otherwise} \end{cases}. \tag{3.2.27}$$

Then $\boldsymbol{\gamma}_{t-1}^n$ can be written as:

$$\boldsymbol{\gamma}_{t-1}^n = \frac{1}{\mu_{t-1}^n}\left[\sum_{k\in\mathcal{C}_{t-1}^n}\mu_{t-1}^k\delta(\hat{r}_{t-1}^k,1),\ldots,\sum_{k\in\mathcal{C}_{t-1}^n}\mu_{t-1}^k\delta(\hat{r}_{t-1}^k,d)\right]^T, \tag{3.2.28}$$

where μ_{t-1}^n ensures that the sum of the mode-probabilities for a given cluster add to 1. In the modified algorithm we propose in this section, instead of associating each component from the reduced N-fold distribution with a single mode r_{t-1}^n, we associate it instead with the vector $\boldsymbol{\gamma}_{t-1}^n$ of mode probabilities. Thus, we propagate to the next iteration the N-fold mixture with the following parameters:

$$\left\{\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, \boldsymbol{\gamma}_{t-1}^n, \mu_{t-1}^n\right\}_{n\in\{1,\ldots,N\}}.$$

At time t, the weights associated with each posterior mixture component (returning to the double-index notation) are calculated as:

$$\mu_t^i = \mu_t^{nj} = \frac{1}{c}\mu_{t-1}^n\left(\sum_{l=1}^d \gamma_{t-1}^{nl}\pi_{lj}\right)\mathrm{KF}_y\left(\boldsymbol{y}_t;\hat{\boldsymbol{x}}_{t-1}^n,\boldsymbol{P}_{t-1}^n,j,t\right).$$

Here $\left(\sum_{l=1}^d \gamma_{t-1}^{nl}\pi_{lj}\right)$ (where γ_{t-1}^{nl} refers to the l^{th} element of $\boldsymbol{\gamma}_{t-1}^n$) replaces $\pi_{\hat{r}_{t-1}^n j}$ in the traditional form of this equation (3.1.3) in order to take into account all the possible modes that can be associated with the component. Clearly, this is a generalisation of the old formula, as in the case of a component being associated with a single mode $\hat{r}_{t-1}^n$, $\gamma_{t-1}^{n(l\neq\hat{r}_{t-1}^n)} = 0$ and $\gamma^{n(l=\hat{r}_{t-1}^n)} = 1$. In this case, $\left(\sum_{l=1}^d \gamma_{t-1}^{nl}\pi_{lj}\right)$ reduces to $\pi_{\hat{r}_{t-1}^n j}$. Figure 3.16 shows a sample calculation of $\boldsymbol{\gamma}_t^n$. A similar treatment of mode transition likelihoods was briefly investigated in [72].

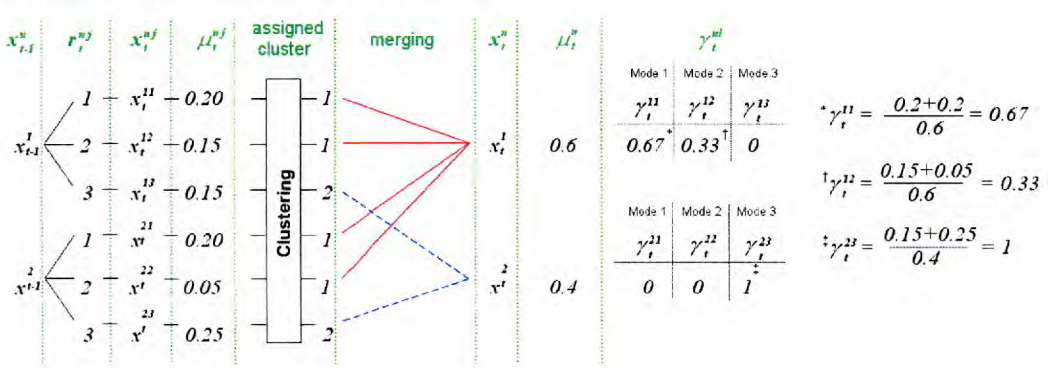


Figure 3.16: Calculation of $\boldsymbol{\gamma}_t^n$: an illustrative example (with $N = 2$, $d = 3$).

It should be noted that the computational increase imposed by this modification is

negligible, as the summation term can efficiently be implemented as a single matrix operation:

$$\left(\sum_{l=1}^{d} \gamma_{t-1}^{nl} \pi_{lj}\right) = \boldsymbol{\pi}_{:j}^{T} \boldsymbol{\gamma}_{t-1}^{n}\,, \tag{3.2.29}$$

where $\boldsymbol{\pi}_{:j}$ refers to the j^{th} column of $\boldsymbol{\pi}$, the $d \times d$ Markov matrix of mode transition probabilities (2.3.3).

We now summarise the modified version of the mixture filter that allows for clustering of components irrespectively of their mode:

STEP 1: MODE-MATCHED KALMAN FILTERING

Given N prior mixture components with mean, covariance, weights and mode probabilities $\hat{\boldsymbol{x}}_{t-1}^{n}$, $\boldsymbol{P}_{t-1}^{n}$, μ_{t-1}^{n}, and $\boldsymbol{\gamma}_{t-1}^{n}$ (for $n \in \{1, \ldots, N\}$), the mixture filter calculates the exact $N \times d$-fold posterior conditional distribution. For modes $j \in \mathcal{S}$, this is done by means of the maps defined in (2.3.12),(2.3.21):

$$(\hat{\boldsymbol{x}}_{t}^{nj}, \boldsymbol{P}_{t}^{nj}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, \boldsymbol{y}_t, j, t)\,, \tag{3.2.30}$$

$$\tilde{\mu}_{t}^{nj} = \mu_{t-1}^{n} \left(\sum_{l=1}^{d} \gamma_{t-1}^{nl} \pi_{lj}\right) KF_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, j, t)\,, \tag{3.2.31}$$

$$\mu_{t}^{nj} = \frac{1}{c}\,\tilde{\mu}_{t}^{nj}, \text{ where } c = \sum_{n,j} \tilde{\mu}_{t}^{nj}\,, \tag{3.2.32}$$

$$\hat{r}_{t}^{nj} = j\,, \tag{3.2.33}$$

where γ_{t-1}^{nl} refers to the l^{th} element of $\boldsymbol{\gamma}_{t-1}^{n}$.

STEP 2: STATE ESTIMATE & COVARIANCE CALCULATIONS

Next, the output state estimate, $\hat{\boldsymbol{x}}_t$, and covariance, $\boldsymbol{P}_t$, are computed by merging together all the constituent densities of the posterior conditional distribution:

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \mu_{t}^{nj} \hat{\boldsymbol{x}}_{t}^{nj}\,, \tag{3.2.34}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \mu_{t}^{nj} [\boldsymbol{P}_{t}^{nj} + (\hat{\boldsymbol{x}}_{t}^{nj} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_{t}^{nj} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]\,. \tag{3.2.35}$$

It should be noted that the output estimates should be calculated prior to the mixture reduction as this inevitably introduces degradation.

STEP 3: MIXTURE REDUCTION

Finally, to reduce the mixture, it is easiest to relabel all mixture components indexed by $n \in \{1, \ldots, N\}$ and by $j \in \mathcal{S}$, with a single index $i \in \{1, \ldots, N \times d\}$. Using any of the mixture reduction techniques presented in this thesis, N sets of index values, $\mathcal{C}_t^n$, should be produced indicating which components are to be clustered together. Components in $\mathcal{C}_t^0$ are discarded.

The N clustered components to be propagated to the next iteration are described by $\{\hat{\boldsymbol{x}}_t^n, \boldsymbol{P}_t^n, \boldsymbol{\gamma}_t^n, \mu_t^n\}$, where:

$$\hat{\boldsymbol{x}}_t^n = \sum_{k \in \mathcal{C}_t^n} \mu_t^k \hat{\boldsymbol{x}}_t^k \,, \tag{3.2.36}$$

$$\boldsymbol{P}_t^n = \sum_{k \in \mathcal{C}_t^n} \mu_t^k \left(\boldsymbol{P}_t^k + (\hat{\boldsymbol{x}}_t^k - \hat{\boldsymbol{x}}_t^n)(\hat{\boldsymbol{x}}_t^k - \hat{\boldsymbol{x}}_t^n)^T \right), \tag{3.2.37}$$

$$\mu_t^n = \sum_{k \in \mathcal{C}_t^n} \mu_t^k \,, \tag{3.2.38}$$

$$\boldsymbol{\gamma}_t^n = \frac{1}{\mu_t^n} \left[\sum_{k \in \mathcal{C}_t^n} \mu_t^k \delta(\hat{r}_t^k, 1), \ldots, \sum_{k \in \mathcal{C}_t^n} \mu_t^k \delta(\hat{r}_t^k, d)) \right]^T . \tag{3.2.39}$$

b) Parallel clustering over mode-space

Alternatively the problem of 'mixed' clusters can be avoided altogether by carrying out parallel mixture reductions procedures for each manoeuvring mode independently of the others. This method would effectively turn the mixture algorithm into an evolution of the 2nd orded GPB (see Section 2.3.3), where a mixture of components, instead of a single one, is retained for each mode.

The main disadvantage of this approach is that this discrimination between components of different modes is somewhat artificial, and overall clustering efficiency might suffer as a result (Figure 3.15 shows an example where this might be the case). This is most likely to occur in problems where the mode distributions are themselves multi-peaked (such as manoeuvring target tracking in clutter) and where the need for mixture algorithms is greatest.

It should finally be noted than in a recent conference paper, Maybeck and Smith [53],

chose this technique to apply ISDI clustering techniques to the problem of manoeuvring target tracking in clutter (more discussion on this problem is found in Chapter 4).

3.2.4 Hybrid mixture reduction

In this section a new mixture reduction method is proposed that aims to combine aspects of weight-based and of distribution-based mixture reduction. One of the most obvious advantages of distribution-based methods is the fact that, by favouring merging over pruning, very little information is ever completely discarded. This generally implies that the true 'hypothesis-path' of the optimal Bayesian solution is incorporated in any mixture approximation (although it might be merged with other paths). On the other hand, weight-based methods are computationally efficient, and those that have a degree of randomisation have also the desirable property of rejuvenating the components of the distribution by randomly revaluing low-weight components. We showed in section 3.2.1(d) that this feature helps to reduce the estimation error caused by mode-transitions.

The advantages of both methods are clearly compatible and mutually beneficial if properly combined. As doing so analytically is problematic, a variety of *ad-hoc* schemes can be designed and tailored to specific problems and computational resources. However if the methods are kept as general as possible, two distinct approaches become clear: the integration of weight-based methods with a degree of clustering (to reduce the level of component pruning), and the integration of clustering methods with an element of randomisation. We investigate the first of these two approaches in its simplest form, although more sophisticated algorithms could surely be implemented. It should be noted that this algorithm requires the modification to the standard mixture algorithm presented in Section 3.2.3(a) if it is to be used for multiple-model target tracking problems.

a) The algorithm

The simplest and most intuitive way to integrate weight-based mixture reduction with a degree of clustering, which would ensure that little information is ever completely discarded, is to combine all those components destined to be discarded into a single *residuals component* (while the other $N-1$ components are chosen normally).

This cluster of residuals is likely to have a high covariance, which should also generally increase the robustness of the algorithms. The effect of this extra residuals component should be most significant when a small number of densities is used, while for high N the difference with the unmodified weight-based algorithms should be less noticeable.

Even for such a simple scheme, a variety of design decisions need to be made, the most significant of which is clearly the choice of the underlying weight-based mixture reduction method. As the Fearnhead and Clifford algorithm (see Section 3.2.1) optimises a cost function and was shown to be an effective randomised technique, it is used in this section as the basis of this *hybrid mixture reduction* method. Any of the other weight-based methods, including the α-algorihtm, could also be used.

The second design decision concerns the treatment of the weight of the clustered density with respect to the other densities. One possibility would be to weigh the residuals component as the sum of the weights of its constituent densities. However when the weights of the original distribution are reasonably balanced, the resultant weight of the residual component could be significantly larger than the weight of the resampled particles. This is clearly undesirable as it would distort the approximation.

The alternative, which is adopted here, is to use the weight structure of the Fearnhead and Clifford algorithm. Their technique assigns the original weight to the largest components chosen deterministically, and it assigns a uniform weight to the components chosen randomly. This weight, given by $1/C_t$, is equal to the total residual weight divided by the number of components to be resampled randomly. Following the notation introduced in Section 3.2.1(c):

$$\frac{1}{C_t} = \frac{1 - \sum_{i \in \mathcal{A}_1} \mu_t(i)}{N - G}, \tag{3.2.40}$$

where G is the cardinality of $\mathcal{A}_1$, the set of densities chosen deterministically. This structure can be readily modified to incorporate the extra residuals component by assigning it with weight equal to that of the samples resampled randomly. When N components are to be retained, the resulting algorithm is essentially the same as the Fearnhead & Clifford algorithm calculated for $N-1$, with the remaining slot being filled by the 'residual' particle. The new algorithm is summarised as follows:

Algorithm 3.2.7 - Hybrid mixture reduction

Step 1. Using Algorithm 3.2.5, calculate C_t as the unique solution to:

$$N - 1 = \sum_{i=1}^{M} \min(C_t \mu_t(i), 1) .$$

Step 2. Let $\mathcal{A}_1 = \{i | \mu_t(i) \geq 1/C_t\}$ and let $G = \text{card}\{\mathcal{A}_1\}$. Let $\mathcal{A}_2 = \{1, \ldots, M\} \backslash \mathcal{A}_1$.

Step 3. Using the stratified sampling procedure of Section 2.2.2(c), select $(N - G - 1)$ distinct index values from $\mathcal{A}_2$ by sampling from:

$$\frac{\mu_t(i)}{\sum_{j \in \mathcal{A}_2} \mu_t(j)} \qquad i \in \mathcal{A}_2 .$$

Step 4. Let set $\mathcal{A}_3 \subset \mathcal{A}_2$ be the set of samples selected as part of Step 3. Then the components in the set $\mathcal{A}_4 = \mathcal{A}_2 \backslash \mathcal{A}_3$ are merged. The mean and covariance of the residuals component are given by:

$$\begin{aligned} \hat{\boldsymbol{x}}_t^m &= \sum_{i \in \mathcal{A}_4} \mu_t(i) \hat{\boldsymbol{x}}_t^i , \\ \boldsymbol{P}_t^m &= \sum_{i \in \mathcal{A}_4} \mu_t(i) (\boldsymbol{P}_t^i + (\hat{\boldsymbol{x}}_t^i - \hat{\boldsymbol{x}}_t^m)(\hat{\boldsymbol{x}}_t^i - \hat{\boldsymbol{x}}_t^m)^T) . \end{aligned}$$

Step 5. Calculate the weight, w_t, of the components in set $\mathcal{A}_3$ and of the residuals component $\mathcal{N}(\hat{\boldsymbol{x}}_t^m, \boldsymbol{P}_t^m)$ as:

$$w_t = \frac{1 - \sum_{i \in \mathcal{A}_1} \mu_t(i)}{N - G} .$$

Step 6. The reduced N-fold mixture is obtained by combining those distributions indexed by $\mathcal{A}_1$ (with their original weight), those indexed by $\mathcal{A}_3$ (all with weight w_t) and the residuals component $\mathcal{N}(\hat{\boldsymbol{x}}_t^m, \boldsymbol{P}_t^m)$ also with weight w_t.

3.2.5 Simulation comparisons

The mixture reduction algorithms described above are now tested for two target tracking problems: the target in clutter problem and the manoeuvring target problem. Although the former has normally been solved with distribution-based mixture reduction methods and the latter using weight-based methods, all reduction algorithms – including the novel

one presented in Section 3.2.4 – are here tested for both problems. This should highlight areas of overlap as well as the strengths and weaknesses of the different techniques.

a) The target in clutter scenario

The first scenario considered aims to replicate the results of [62] and [72], whose original purpose was to demonstrate the effectiveness of distribution-based mixture reduction methods for the problem of tracking non-manouevring targets in heavy clutter. This problem, which is outlined in detail in Appendix A, involves nonlinearity in the form of multiple measurements. Of these, only one may belong to the true target; the rest are 'false alarms' (generally referred to as 'clutter' measurements) which may be erroneously mistaken for the true measurement.

The nature of clutter measurements varies with the type of sensors being used, but is generally associated with reflections of the true target or, especially in the case of passive sensors, with interferences produced by unrelated sources. Although some types of clutter measurements (such as reflections) may exhibit correlation, false measurements are generally assumed to be uniformly distributed over the sensor's surveillance region, while the frequency of their appearance is modelled using a Poisson distribution. Generally, $P_c(Z)$, the probability of Z clutter measurements at a given time step, is given by $P_c(Z) = \frac{(\lambda V)^Z e^{-\lambda V}}{Z!}$, where λ is the spatial density of the clutter and V is the volume of the observation region. For a more detailed formulation of the data association problem see Appendix A.1.

The chief goal of these simulations is the evaluation of the performance of weight-based mixture reduction methods in scenarios normally solved using distribution-based methods. To this end, the Fearnhead and Clifford algorithm is tested along with the modified hybrid reduction scheme of Section 3.2.4. Due to the computationally intensive nature of the simulations associated with the ISD initialisation (ISDI) algorithm, 50 Monte Carlo simulations are used instead of the 100 used in previous comparisons.

These simulations compare the algorithms on the basis of the 'track-life', the average number of iterations until a 'track-loss' is registered. A track-loss, as defined in [72],

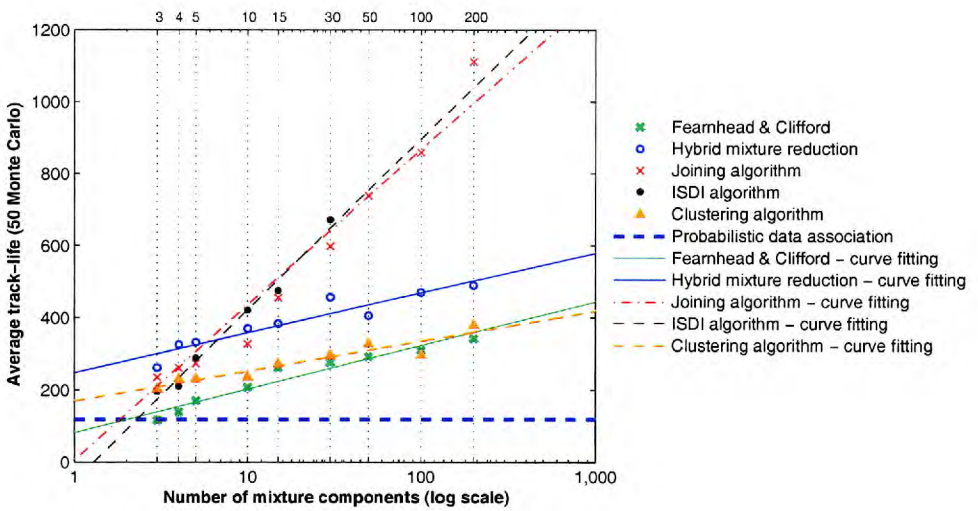


Figure 3.17: Average track-life per mixture component (50 Monte Carlo average). Exact component numbers are marked at the top of the chart.

takes place if the true measurement does not appear in any of the mixture component gates[2] for five consecutive time steps, or if all hypotheses are 10σ from the true target position for five consecutive steps. Here σ is defined as the standard deviation of the estimation error a Kalman filter using only the correct measurements would produce. To allow for such a comparison, a linear track is generated for $t = 1, 2, \ldots, 10,000$, and a very high density of clutter measurements ($\lambda = 0.012$) is used. As in [62] and [72], the sampling time h, q_m (such that $\boldsymbol{Q}_m = q_m \boldsymbol{I}_{2\times 2}$) and q_s (as given in (2.3.9)) are all normalised to unity. Given the length of the track, the measurements are simulated in a box of size $200\sqrt{q_m}$ around the true track position. Although Salmond [62] suggests using a large gate of size determined by $P_g = 0.001$, in order to keep the simulations more computationally manageable, the larger value of $P_g = 0.01$ (suggested by Williams [72]) is used (see Appendix A.2 for a description of how P_G, the probability of the correct measurement being rejected by the gating process, affects the gate size). This choice ensures that all tracking algorithms will lose track long before the track is complete.

Figure 3.17 shows the average track-life of the different algorithms plotted against the

[2] A 'gate' is a coarse data association tool used to reduce the computational complexity of tracking algorithms. Gates work by rejecting measurements that are highly unlikely to be true based on current expectations. Appendix A.2 provides further details on gating techniques.

number of mixture components retained at each iteration. As shown by the regression lines, all these algorithms seem to display a roughly logarithmic relationship between track-life and number of mixture components. These results match well with those presented in [72] although, in the case of the ISDI algorithm, there does not seem to be significant evidence to suggest an exponential relationship between N and the track-life. The ISDI would rather seem to follow logarithmic patterns more similar to those of the other algorithms (the regression line is remarkably similar to that of the joining algorithm). However the difficulty in testing this algorithm for $N > 30$ (due to prohibitive computational requirements) leaves a degree of uncertainty.

In this setting, the Fearnhead and Clifford algorithm performs comparably to the clustering algorithm, however both these faster algorithms display a low-gradient logarithmic relationship (between number of components and track-life) which fails to be competitive with either the Joining or the ISDI algorithms for all values of N.

On the other hand, integrating the Fearnhead and Clifford algorithm with a component collecting all the discarded residuals provides significant benefits. The hybrid mixture reduction algorithm introduced in Section 3.2.4, while displaying a similar gradient to that of the Fearnhead and Clifford algorithm, has a positive offset which grants it a track-life of around 200 steps more for all values of N. This improvement means that for a low number of mixture components, this scheme produces the highest track-life of all algorithms. This is even more evident if track-life is plotted against the average computational time of each recursion, shown in Figure 3.18[3]. Clearly up until 10-15 mixture components, the most efficient technique is the hybrid method, above which it is outperformed by the joining algorithm.

As pointed out in [72], the computational efficiency of the ISDI algorithm could be improved using techniques such as U-D factorisation[4] or by means of a C++ implemen-

[3] As the most computational intensive plots on this chart had to be computed on parallel machines, the values shown should be taken as indicative rather than exact. Every effort was made to try to keep these values consistent, however minor variations are nonetheless possible.

[4] U-D Factorisation is a technique which enables the fast calculation of the inverse of a positive definite matrix.

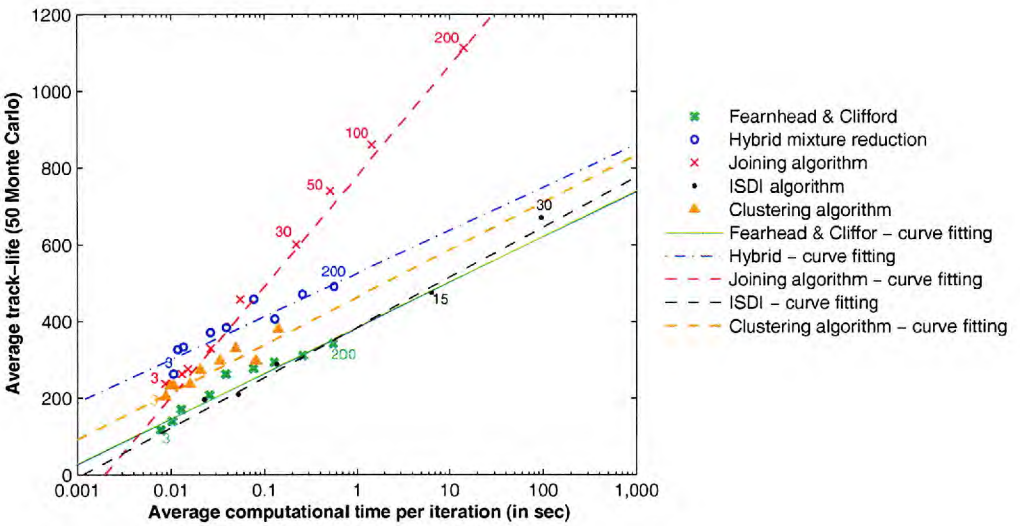


Figure 3.18: Average track-life vs. average computational time of each iteration (50 Monte Carlo average). Component numbers are indicated where possible: for ISDI joining algorithm, the range shown is: [3, 4, 5, 10, 15, 30]; for the LAMF using the joining algorithm: [2, 3, 5, 10, 25]; for the RBPF: [10, 25, 50, 100, 250, 500]; for all other filters: [3, 4, 5, 10, 15, 30, 50, 100, 200].

tation. For consistency – as the other algorithms do not use these techniques – these were not implemented in these simulations. Although methods such U-D factorisation would benefit the ISDI method more than other algorithms (given its heavy use of matrix inversion needed to calculate the ISD costs), it is unlikely that the ISDI would replace the joining algorithm as the most cost-effective given their similar performance per particle (Figure 3.17).

It should be noted that the results presented are somewhat affected by the relatively small gate size used in these simulations ($P_g = 0.01$). This was chosen so that simulations would be possible in a reasonable time. If the value of $P_g = 0.001$ (as used in [62]) was chosen instead, the tracking performance of all algorithms would greatly increase. The smaller gate size was used as, not only the larger number of measurements satisfying gating conditions would increase the computational cost per iteration, but especially the longer track-lives would dramatically increase testing times.

b) The manoeuvring target tracking scenario

The mixture reduction algorithms are now tested in the multiple-model scenario already introduced in Section 2.3.5. The modified mixture filter presented in Section 3.2.3(a), which allows for a mixture component to be associated with more than one mode, is used. This is necessary in order for the distribution-based algorithm to merge components irrespectively of their associated modes.

Figure 3.19 shows the average RMS estimation error of the IMM, the multiple model bootstrap filter (MMBF) and the Rao-Blackwellised particle filter (RBPF) using stratified sampling in the resampling step, as was already presented in Figure 2.10. To these we superimpose the results produced using the mixture filter tested using a number of different mixture reduction methods: the Fearnhead and Clifford, the joining algorithm, and the hybrid scheme of Section 3.2.4. Having proved to be the most efficient of the distribution-based methods in the data association simulations, the joining algorithm was chosen as the representative of the whole category.

It can be seen how all mixture reduction methods outperform the PF and the RBPF by achieving asymptotic convergence with a computational cost smaller by nearly an order of magnitude. Clearly the IMM is still the most efficient method for very low computational costs, but – if needed – the new techniques allow it to be outperformed much more cheaply. Of the mixture reduction methods, the hybrid scheme proves to be the most efficient for small number of components and is closely followed by the Fearnhead and Clifford scheme and by the joining algorithm. The latter produces estimates marginally inferior to the IMM's with only 3 mixture components, and outperforms it with only 5. Its overall computational efficiency is decreased by its relatively higher cost per mixture component.

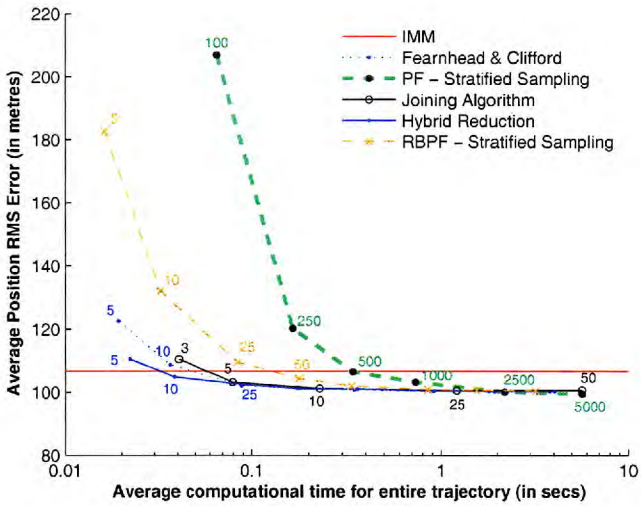


Figure 3.19: Position RMS error vs. computational cost comparisons for Track 1 and for varying number of mixture components / particles (100 Monte Carlo average). Component numbers are indicated where possible. For the mixture filters and the RBPF, the range shown is: [5, 10, 25, 50, 100, 250, 500, 1000, 2500]; for the particle filter: [100, 250, 500, 1000, 2500, 5000]; finally for the mixture filter using the joining algorithm: [3, 5, 10, 25, 50].

3.3 Look-ahead mixture filters

In this section the structure of the standard mixture filter presented in section 3.1.1 is further modified to allow for the very latest measurement information to be incorporated in the mixture reduction process, thus ensuring that the approximations are more strictly related to the current conditional distribution of the target. The key motivation behind this enhancement, which we refer to as 'look-ahead', is that the standard filter tends to approximate the optimal conditional density of all mode histories without direct reference to the degradation being caused to future estimates, but rather tends to concentrate exclusively on the error on current estimates.

By looking at the structure of the standard mixture filter (simplified in Figure 3.20), we note how the mixture reduction process – in which the posterior distribution is approximated by one with support on a lower number of terms – is the last step of each iteration (say at time t), and is computed just prior to the receipt of the measurement used at the next cycle (time $t+1$). However as mixture reduction does not affect the state estimate

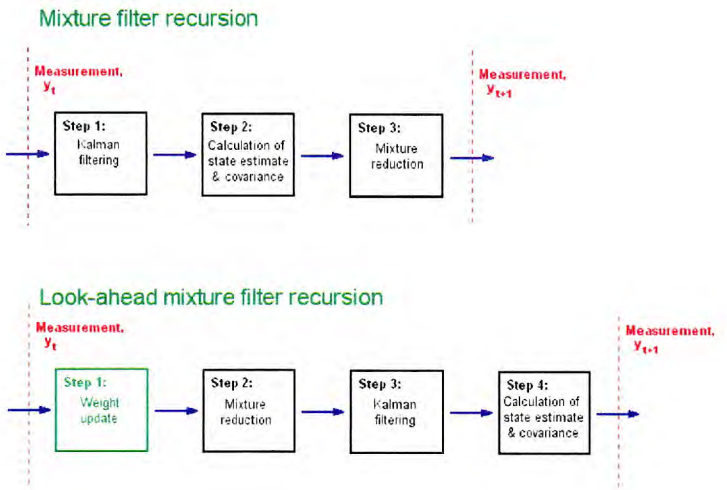


Figure 3.20: Structure modifications introduced by the look-ahead mixture filter.

at time t, it is exclusively relevant to the future recursions, and particularly so to the next one at time $t+1$. The approximations to the conditional distribution could then be seen as being based on *past* information, as the samples to be used at time $t+1$ are chosen exclusively on the basis of measurement data up until time t.

Figure 3.20 shows how by slightly shifting the recursion of the standard filter, the *look-ahead mixture filter* delays the mixture reduction step until it is possible to update the weights of the mixture components using current measurement information. This ensures that the approximations to the conditional distribution are carried out with *current* information.

3.3.1 Derivation of the look-ahead mixture filter (LAMF)

Suppose that, at time $t-1$, as a result of previous updating, a Gaussian mixture is used to describe the posterior distribution of the target, $p(\boldsymbol{x}_{t-1}, r_{t-1}|\boldsymbol{y}_{1:t-1})$. This will be of the form:

$$\tilde{p}(\boldsymbol{x}_{t-1}, r_{t-1}|\boldsymbol{y}_{1:t-1}) = \sum_{n=1}^{N}\sum_{j=1}^{d} \mu_{t-1}^{nj}\mathcal{N}(\boldsymbol{x}_{t-1}; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}) .$$

In order to control the growth of mixture components, at this point we normally produce $\bar{p}(\boldsymbol{x}_{t-1}, r_{t-1}|\boldsymbol{y}_{1:t-1})$, an N-fold Gaussian mixture approximation to the $N \times d$-fold Gaussian mixture $\tilde{p}(\boldsymbol{x}_{t-1}, r_{t-1}|\boldsymbol{y}_{1:t-1})$. For simplicity, we shall for the moment assume that the mixture reduction step is weight-based, however the same concepts can be ap-

plied to the distribution-based mixture reduction methods by applying the modifications of Section 3.2.3(a).

Suppose now that the measurement $\boldsymbol{y}_t$ is available prior to the mixture reduction step. We could then use $\tilde{p}(\boldsymbol{x}_{t-1}, r_{t-1}|\boldsymbol{y}_{1:t-1})$ as the prior distribution, and by means of mode-matched Kalman filters, we could generate a $N \times d^2$,-fold mixture approximation to $p(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t})$. For $n \in \{1, \ldots, N\}$ and for $j, k \in \mathcal{S}$, this would be:

$$\hat{p}(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t}) = \sum_{n=1}^{N}\sum_{j=1}^{d}\sum_{k=1}^{d} \mu_t^{njk}\mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{njk}, \boldsymbol{P}_t^{njk}),$$

where

$$\left(\hat{\boldsymbol{x}}_t^{njk}, \boldsymbol{P}_t^{njk}\right) = \mathrm{KF}_x(\hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, \boldsymbol{y}_t, k, t), \tag{3.3.1}$$

$$\mu_t^{njk} = (1/c)\mu_{t-1}^{nj}\pi_{jk}\mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, k, t), \tag{3.3.2}$$

and where c is a normalisation constant. This represents a further step into the optimal Bayesian solution tree, where the weights, μ_t^{njk}, represent the updated probabilities of each mode history one step ahead (i.e. at time t).

Although contructing $\hat{p}(\boldsymbol{x}_t, r_t|\boldsymbol{y}_{1:t})$ is not desirable if the complexity of the mixture approximation is to be kept manageable, weights μ_t^{njk} could be used to improve the effectiveness of the mixture reduction step. To do so, instead of selecting components on the basis of μ_{t-1}^{nj}, we instead use $\tilde{\mu}_t^{nj}$, the updated weights including the 'future' information. These are given by:

$$\begin{aligned}\tilde{\mu}_t^{nj} &= \sum_{k=1}^{d} \mu_t^{njk} \\ &= \frac{1}{c}\mu_t^{nj}\sum_{k=1}^{d}\pi_{jk}\mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, k, t).\end{aligned} \tag{3.3.3}$$

This version is unsatisfactory, however, because of the computational burden of calculating μ_t^{njk} for all $N \times d^2$ values of the indices n, j, k. However, in typical models of manouevring targets:

$$\pi_{ii} \gg \pi_{ij}, \quad \text{for } j \neq i, \tag{3.3.4}$$

(i.e. the transition matrix favours no change of mode). We also note that the target model is arrived at by a time discretisation of a continuous model (with sampling period,

h). When h is small:

$$\boldsymbol{Q}_s(i) \simeq \boldsymbol{Q}_s(j), \qquad \boldsymbol{F}(i) \simeq \boldsymbol{F}(j), \qquad \text{for all } i, j.$$

In these circumstances, it is reasonable to approximate (3.3.3) by:

$$\begin{aligned} \tilde{\mu}_t^{nj} &= \mu_t^{nj(k=j)} \\ &= \frac{1}{\tilde{c}} \mu_{t-1}^{nj} \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, j, t), \end{aligned} \tag{3.3.5}$$

since the difference between these quantities can be expressed as the product of two small numbers (here $\tilde{c}$ is another normalising variable).

These approximations bring about a reduction in the number of Kalman filtering operations involved in calculating the weights by a factor of approximately $1/d$, and thus ensure that the condensation phase of the algorithm imposes a minor additional computational burden to the overall scheme.

For systems for which condition (3.3.4) is violated, instead of retaining a constant mode, a mode value could be sampled from the mode transition probabilities. If the newly sampled mode is $\hat{r}_t^{nj}$, the resulting look-ahead weights are modified to:

$$\tilde{\mu}_t^{nj} = (1/\bar{c}) \mu_{t-1}^{nj} \pi_{j\hat{r}_t^{nj}} \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, \hat{r}_t^{nj}, t), \tag{3.3.6}$$

where $\bar{c}$ is a normaliser.

3.3.2 The LAMF algorithm

The *look-ahead mixture filter* (LAMF) is now described in detail. The algorithm starts by retaining the exact $N \times d$-fold mixture from the previous step $(t-1)$, and its first task involves updating the weights of its components using the current measurement, $\boldsymbol{y}_t$. At this point, the mixture reduction step is performed and the posterior distribution at time t is then computed. The final stage of the algorithm is the calculation of the state estimate and covariance.

It should be noted that, assuming the mixture reduction algorithm used is weight-based, some of the Kalman filter operations required to calculate some terms of the posterior

distribution, have already been evaluated as part of the evaluation of 'look-ahead' probabilities computations (one d^{th}, to be exact) and do not need to be recomputed.

STEP 1: WEIGHT UPDATE

Assume a prior $N \times d$-fold Gaussian mixture, whose components have parameters given by $\{\hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, \hat{r}_t^{nj} = j, \mu_{t-1}^{nj}\}$ (for $n \in \{1, \ldots, N,\ j \in \mathcal{S}\}$), is available. The first step of the look-ahead mixture filter is to update weights μ_{t-1}^{nj} (current to measurement $t-1$) with the latest measurement $\boldsymbol{y}_t$ so that the mixture reduction step may be carried out with current information. The updated weights $\tilde{\mu}_t^{nj}$ are calculated as:

$$\tilde{\mu}_t^{nj} = (1/\tilde{c})\mu_{t-1}^{nj} \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, j, t)\,, \tag{3.3.7}$$

where

$$\tilde{c} = \sum_{n,j} \mu_{t-1}^{nj} \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{nj}, \boldsymbol{P}_{t-1}^{nj}, j, t)\,.$$

STEP 2: MIXTURE REDUCTION

Next, the algorithm reduces the distribution from $N \times d$-fold to N-fold on the basis of the updated weights, $\tilde{\mu}_t^{nj}$. An extensive discussion of mixture reduction techniques is found in Section 3.2, but all possible techniques produce a distribution whose components are defined by mean $\hat{\boldsymbol{x}}_{t-1}^{n}$, covariance $\boldsymbol{P}_{t-1}^{n}$ and μ_{t-1}^{n}. These algorithms should also produce a vector $\boldsymbol{\gamma}_{t-1}^{n}$ of mode probabilities as described in Section 3.2.3(a).

STEP 3: MODE-MATCHED KALMAN FILTERING

From this point, the algorithm is identical to the standard mixture filter of Section 3.1.1. Given N prior mixture components, the algorithm calculates the exact posterior conditional distribution by using banks of mode-matched Kalman filters which generate a weighted sum of estimates that considers each possible mode transition. For modes $j \in \mathcal{S}$, this is done by means of the maps defined in (2.3.12) and (2.3.21):

$$(\hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}_t^{nj}) = \mathrm{KF}_x(\hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, \boldsymbol{y}_t, r_t = j, t)\,, \tag{3.3.8}$$

$$\mu_t^{nj} = \frac{1}{c}\mu_{t-1}^{n} \left(\sum_{l=1}^{d} \gamma_{t-1}^{nl} \pi_{lj} \right) \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, j, t)\,, \tag{3.3.9}$$

where:

$$c = \sum_{n,j} \mu_t^{n} \left(\sum_{l=1}^{d} \gamma_{t-1}^{nl} \pi_{lj} \right) \mathrm{KF}_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t-1}^{n}, j, t)\,, \tag{3.3.10}$$

and where γ_{t-1}^{nl} refers to the l^{th} element of γ_{t-1}^{n}.

STEP 4: STATE ESTIMATE & COVARIANCE CALCULATIONS

The final stage consists in the calculation of the output state estimate and covariance. These are computed by further merging together all the elements of the posterior conditional distribution.

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N}\sum_{j=1}^{d} \mu_t^{nj} \hat{\boldsymbol{x}}_t^{nj}, \tag{3.3.11}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N}\sum_{j=1}^{d} \mu_t^{nj} [\boldsymbol{P}_t^{nj} + (\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]. \tag{3.3.12}$$

3.3.3 Simulation comparisons

The look-ahead version of the mixture filter was initially tested for Track 1, the manoeuvring target tracking scenario already investigated in previous sections (see Section 2.3.5 for simulation parameters). Although the 'look-ahead' algorithms outperforms the normal versions of the algorithm for equal number of particles, Figure 3.21 shows how both versions of the algorithms produce similar tracking errors per computational cost (i.e. the error reduction obtained using the look-ahead update is generally offset by its cost).

The main exception is the case where the joining algorithm is used in the mixture reduction step. This type of reduction is in fact considerably more computationally intensive than other methods, and takes up a significantly higher proportion of each iteration's computational time. The look-ahead update therefore introduces a lower proportional increase in computational requirements (from around +50%, in the case of the weight based methods, to around +25% for the joining algorithm). This is enough for the look-ahead version of the joining algorithm to outperform its normal counterpart. This is particularly evident for low number of mixture components, and with $N = 3$, this technique produces lower errors than the IMM algorithm (which retains a mixture of equal size).

Given their nature, look-ahead filters should generally be expected to provide estimate improvements during the manoeuvring phase. In order to fully appreciate the advantages they can offer, a new track featuring more intense manoeuvring was designed (Track 3).

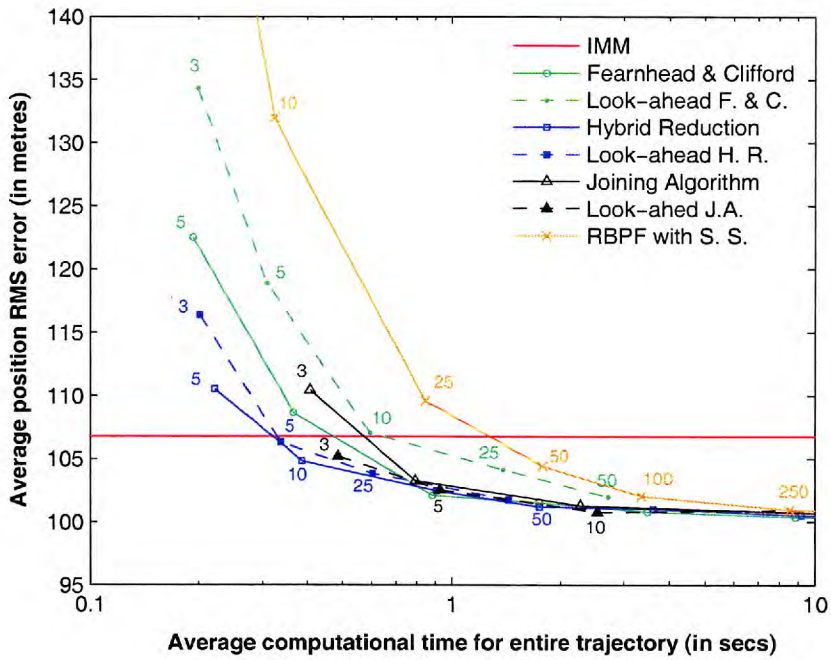


Figure 3.21: Position RMS error vs. computational cost comparisons for Track 1 and for varying number of mixture components / particles (100 Monte Carlo average). Component numbers are indicated where possible. For the filters using the joining algorithm, the range shown is: [3, 5, 10]; for the RBPF: [10, 25, 50, 100, 250]. For all other filters: [5, 10, 25, 50, 100].

Figure 3.22 shows a realisation of this new target trajectory (Track 3), which uses the same parameters as those of Track 1 but in which the target now manoeuvres as follows: $r_{0:25} = 2$, $r_{26:55} = 3$, $r_{56:85} = 1$, $r_{86:115} = 3$, $r_{116:145} = 1$, $r_{146:175} = 3$, $r_{176:200} = 2$.

When the algorithms are tested for this track (Figure 3.23), the look-ahead versions are more competitive. In particular, the most efficient algorithm of all is now the LAMF using the joining algorithm in the reduction step.

The type of chart considered so far shows the averaged error over the entire trajectory, however during manoeuvring the look-ahead versions of all algorithms outperform their counterparts. Figure 3.24 shows the average RMS error throughout the trajectory for the LAMF using the hybrid mixture reduction technique (with 5 mixture components). It can be seen that the errors it produces during manoeuvring are lower than

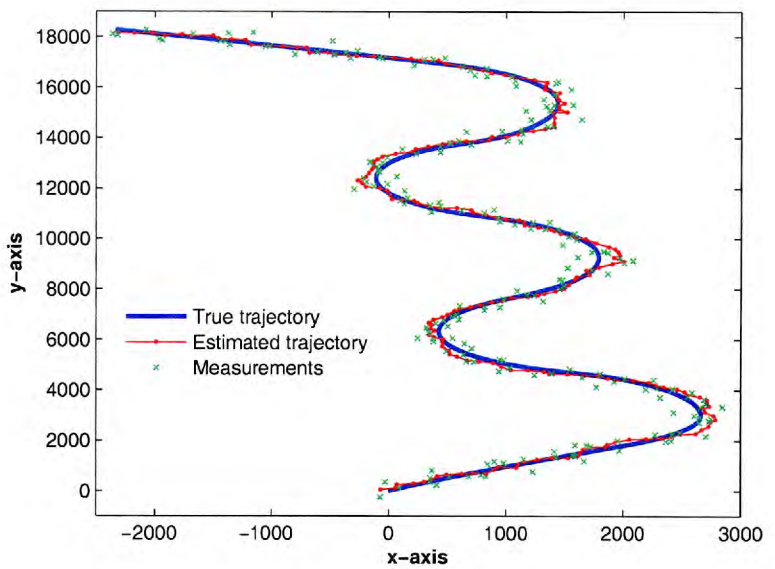


Figure 3.22: Sample realisation of Track 3 using the mixture filter and the Fearnhead & Clifford mixture reduction algorithm (N=50).

both the normal version with 5 mixture components, as well as the normal version with 8 mixture components (which is computationally equivalent). Figure 3.25 also shows the superiority of the look-ahead version in estimating the correct modes: mode transitions are identified about 5 steps earlier than in the normal version.

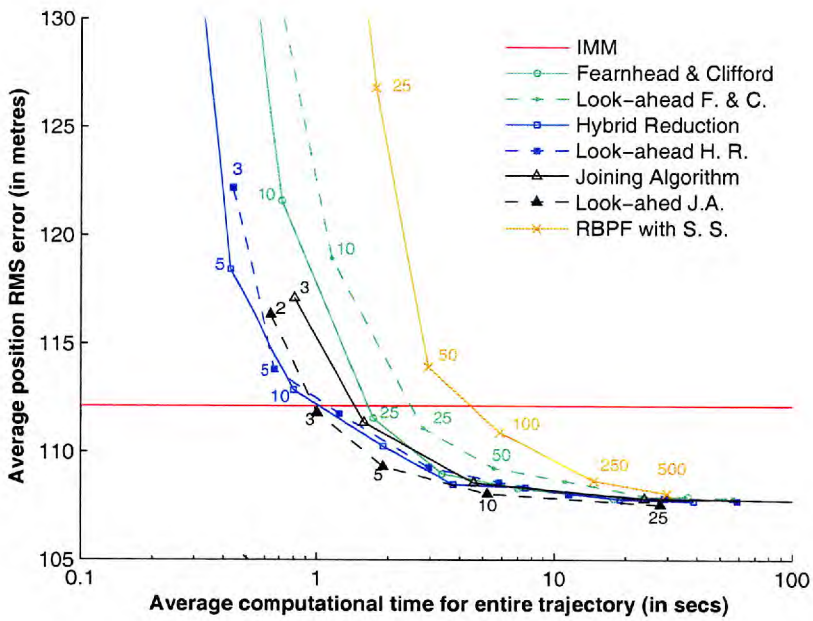


Figure 3.23: Position RMS error vs. computational cost comparisons for Track 3 and for varying number of mixture components / particles (100 Monte Carlo average). Component numbers are indicated where possible. For the mixture filter using the joining algorithm, the range shown is: [3, 5, 10, 25]; for the LAMF using the joining algorithm: [2, 3, 5, 10, 25]; for the RBPF: [10, 25, 50, 100, 250, 500]. For all other filters: [3, 5, 10, 25, 50, 100, 250].

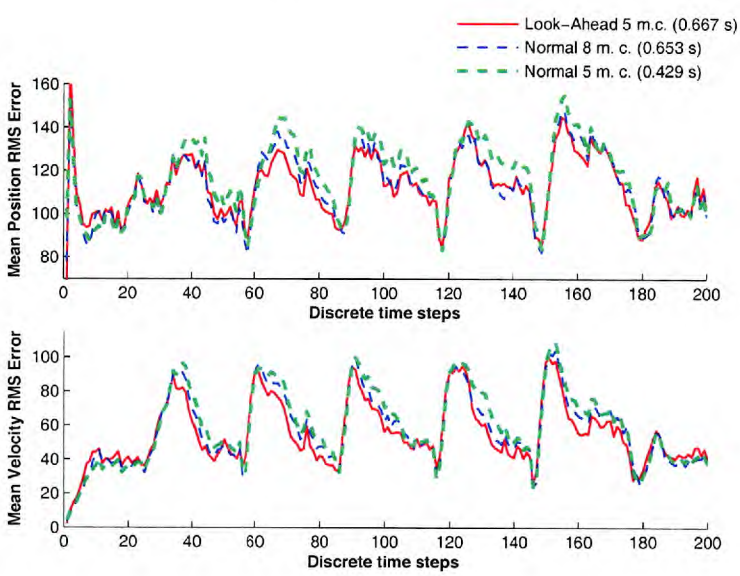


Figure 3.24: RMS error comparison of the LAMF with the MF for Track 3 (100 MC average).

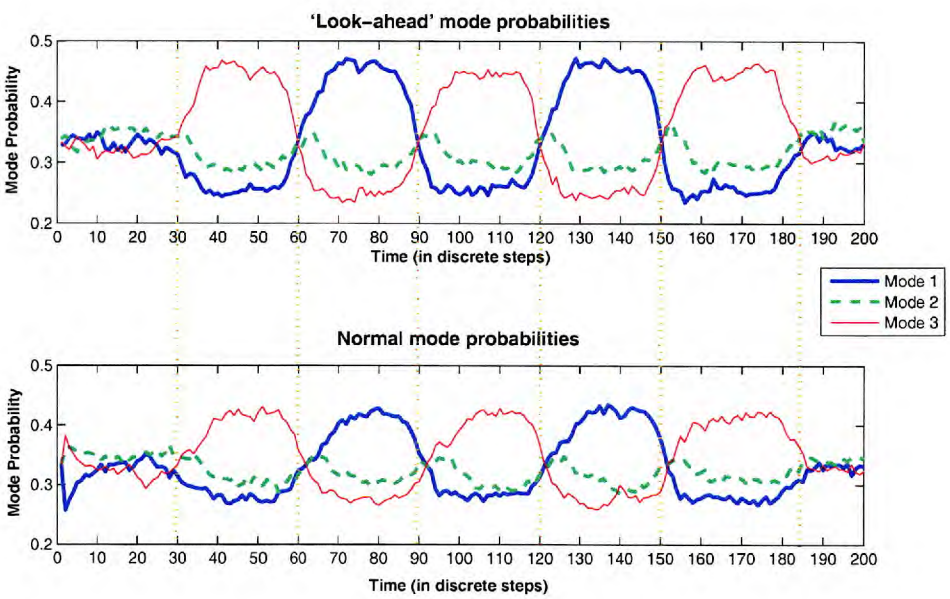


Figure 3.25: Mode probabilities comparison of the LAMF with the MF for Track 3 (100 MC average). Both filters use 5 mixture components.)

Chapter 4

Tracking a Manoeuvring Target in Clutter

In Section 3.2, two classes of mixture reduction algorithms, which were originally designed for different types of problems (weight-based algorithms for multiple model tracking problems and distribution-based algorithms for data association problems), were shown to be mutually applicable to both types of problems. These two problems are now combined, and we show how the simple structure of the mixture filter can be extended to accomodate this extra complexity and provide an effective solution for the manoeuvring target in clutter tracking problem.

The *multiple measurement mixture filter* (MMMF) we propose, not only allows for 'fair' testing of the mixture reduction methods presented earlier, but especially provides a novel and valid alternative to the existing techniques to deal with this difficult problem. Normally this problem is tackled by integrating a multiple model tracker with data association techniques (see Appendix A). The most popular algorithm of this type of approach is the *interacting multiple model probabilistic data association* (IMMPDA) [18], [20] – a combination of the *probabilistic data association filter* (PDAF) (Appendix A.3.2) with the IMM algorithm. More recently, the problem has also been successfully tackled using particle filters ([19],[46]). Many variants of these algorithms also exist to extend the problem to multiple targets (see for example [20], [19], [46] or [15]).

The use of mixture filters which use a mixture reduction step to control component

growth, has not been presented in this context before (as far as the author is aware of), and provides a useful alternative. In a recent paper [53], however, Maybeck and Smith suggest the integration of the ISDI mixture reduction algorithm with the IMM to retain multiple densities for each target mode. A similar approach, which involves a separate clustering process for each mode, was described in Section 3.2.3 and is nonetheless different from the one proposed in this section. Our method makes use of the modification to the standard mixture filter introduced in Section 3.2.3 to allow for the clustering of mixture components associated with different modes.

In Section 4.1, we introduce the problem of tracking a manoeuvring target with multiple measurements, and in Section 4.2 we discuss the optimal Bayesian solution. In Section 4.3 we derive the MMMF and finally in Section 4.4 we report on simulation studies comparing the MMMF with other techniques, as well as comparing the performance of the mixture reduction methods described earlier in this thesis.

4.1 The problem

A manoeuvring target in clutter can be modelled using the jump Markov linear system in which the measurement process is subject to the presence of extra false meausurements. The following problem definition is based on the description of the multiple-model (Section 2.3.1) and data association (Appendix A.1) problems. The state equations can be described as:

$$\boldsymbol{x}_t = \boldsymbol{F}(r_t)\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s\,, \tag{4.1.1}$$

$$\boldsymbol{\mathcal{Y}}_t = \cup \begin{cases} \boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m & \text{with probability } P_d\,, \\ \boldsymbol{z}_t(i) = \mathcal{U}[\boldsymbol{R}_{\min}, \boldsymbol{R}_{\max}] & \text{for } i \in \{1, \ldots, Z_t\}\,. \end{cases} \tag{4.1.2}$$

Here $r_t \in \{1, \ldots, d\}$ is the hidden Markov chain mode process with transition probabilities $\pi_{ij} = \Pr(r_t = j | r_{t-1} = i)$. $\boldsymbol{F}(r_t)$ and $\boldsymbol{Q}_s(r_t)$ are mode-dependent system matrices which model the dynamics of different manoeuvring modes using the coordinated turn model. In the scenarios considered in this section, we have assumed that all other transition matrices are time and mode invariant.

To deal with clutter measurements, we introduce $\boldsymbol{\mathcal{Y}}_t$, which is the set of measurements at

time t. This consists of the true measurement $\boldsymbol{y}_t$, detected with probability P_d, and a set of clutter measurements $z_t(i)$, for $i \in \{1, \ldots, Z_t\}$. The number of clutter measurements, Z_t is governed by a Poisson distribution such that the probability of Z_t measurements is given by $P_c(Z_t) = \frac{(\lambda V_t)^{Z_t} e^{-\lambda V_t}}{Z_t!}$. Here λ is the spatial density of the clutter process, and V_t is the volume of the rectangular observation region described by $\boldsymbol{R}_{\min}$, $\boldsymbol{R}_{\max}$. These are $L \times 1$ vectors containing respectively the minimum and maximum coordinates of this region. Assuming the true target is detected, the total number of measurements at time t is then given by $Y_t = Z_t + 1$, otherwise $Y_t = Z_t$. Each clutter measurement is assumed to be uniformly distributed within the surveillance region.

4.2 The optimal Bayesian solution

When extending the multiple-model tracking problem to include multiple-measurements, the true conditional density is subject to an extra layer of growth at each iteration. The exact solution not only needs to take account of all the possible 'mode-histories' of the target up until time t, but also its possible 'measurement association histories'. As shown in Figure 4.1, the evolution of the conditional density can now be described using a tree of mode and measurement 'association hypotheses' (or simply 'hypotheses'), each branch of which can be evaluated using Kalman filters with the correct history of modes and measurements. As explained in further detail in Appendix A.3.1, to take into account the possibility that the true measurement may not be detected, an extra 'null-hypothesis' representing this eventuality (and whose mean and covariance are the unupdated predictions) is added at each iteration. The optimal solution at time t is then a weighted sum of the $N_t = d^t \prod_1^t (Y_t + 1)$ Gaussian estimates corresponding to all association hypotheses, where Y_t is the total number of measurements at time t.

Define $\mathcal{D}_{1:t}$ to be the set of measurement association histories up until time t, and $D_{0:t}$ the set of mode histories. The exact conditional density at time t can then be expressed

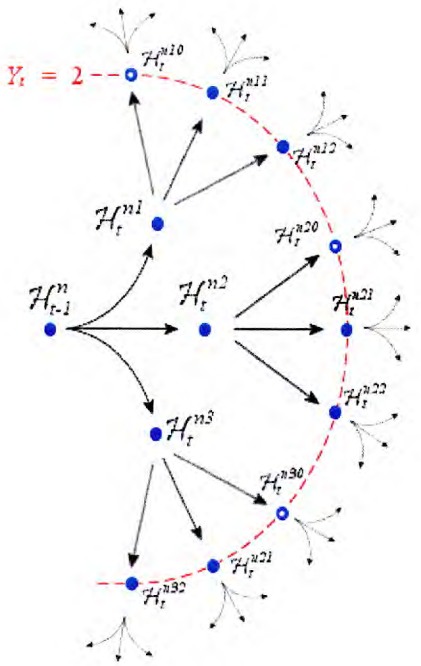


Figure 4.1: Multiple-model/multiple-measurement problem: exponential growth in association hypotheses

as the weighted sum of the N_t densities corresponding to each possible hypothesis:

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \sum_{r_{0:t}\in D_{0:t}} p(\boldsymbol{x}_t|q_{1:t}, r_{0:t}, \boldsymbol{\mathcal{Y}}_{1:t})\mu_t^{q_{1:t},r_{0:t}}\,, \tag{4.2.1}$$

where

$$\mathcal{D}_{1:t} = \{q_{1:t} : q_1, \ldots, q_t \in \{0, \ldots, Y_t\}\}\,, \tag{4.2.2}$$

$$D_{0:t} = \{r_{0:t} : r_1, \ldots, r_t \in \{1, \ldots, d\}\}\,, \tag{4.2.3}$$

$$\mu_t^{q_{1:t},r_{0:t}} = p\left(q_{1:t}, r_{0:t}|\boldsymbol{\mathcal{Y}}_{1:t}\right), \quad \text{and} \quad \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \sum_{r_{0:t}\in D_{0:t}} \mu_t^{q_{1:t},r_{0:t}} = 1\,. \tag{4.2.4}$$

This Gaussian mixture can also be expressed in terms of the weights $\mu_t^{q_{1:t},r_{0:t}}$, the means, $\hat{\boldsymbol{x}}_t^{q_{1:t},r_{0:t}}$, and the covariances, $\boldsymbol{P}_t^{q_{1:t},r_{0:t}}$, of the constituent Gaussian densities:

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \sum_{r_{0:t}\in D_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{q_{1:t},r_{0:t}}, \boldsymbol{P}_t^{q_{1:t},r_{0:t}})\mu_t^{q_{1:t},r_{0:t}}\,. \tag{4.2.5}$$

For the appropriate mode r_t, if knowledge of the prior distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{\mathcal{Y}}_{1:t-1})$ is available, the densities in this sum corresponding to $q_t = 0$ are simple predictions. The densities for $q_t = \{1, \ldots, Y_t\}$ are instead calculated using the Kalman filter update

equations. If the prior distribution is given by:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{\mathcal{Y}}_{1:t-1}) = \sum_{q_{1:t-1}\in\mathcal{D}_{0:t-1}} \sum_{r_{0:t-1}\in D_{0:t-1}} \mathcal{N}(\boldsymbol{x}_{t-1}; \hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1},r_{0:t}}, \boldsymbol{P}_{t-1}^{q_{1:t-1},r_{0:t}})\mu_{t-1}^{q_{1:t-1},r_{0:t}}, \tag{4.2.6}$$

where $\mu_{t-1}^{q_{1:t-1},r_{0:t-1}}$, $\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1},r_{0:t-1}}$ and $\boldsymbol{P}_{t-1}^{q_{1:t-1},r_{0:t-1}}$ are respectively the weights, the means and the covariance of its constituent Gaussian densities, then the densities in the posterior conditional distribution at t can be expressed as:

for $r_t \in \mathcal{S}$:

- for $q_t = 0$:

$$\hat{\boldsymbol{x}}_t^{q_{1:t},r_{0:t}} = \boldsymbol{F}(r_t)\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1},r_{0:t-1}} + \boldsymbol{u}_t^s, \tag{4.2.7}$$

$$\boldsymbol{P}_{t-1}^{q_{1:t},r_{0:t}} = \boldsymbol{F}(r_t)\boldsymbol{P}_{t-1}^{q_{1:t-1},r_{0:t-1}}\boldsymbol{F}(r_t)^{\mathrm{T}} + \boldsymbol{Q}^s(r_t), \tag{4.2.8}$$

- for $q_t \in \{1,\ldots,Y_t\}$:

$$(\hat{\boldsymbol{x}}_t^{q_{1:t},r_{0:t}}, \boldsymbol{P}_t^{q_{1:t},r_{0:t}}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1}}, \boldsymbol{P}_{t-1}^{q_{1:t-1}}, \boldsymbol{y}_t(q_t), r_t, t), \tag{4.2.9}$$

where $\boldsymbol{y}_t(q_t) \in \boldsymbol{\mathcal{Y}}_t$. The calculation of the posterior weights is complicated by the possibility of misdetecting the true measurement. To simplify notation, let $\mathcal{H}_{t-1}^n$ be the index of the prior hypotheses $q_{1:t-1}, r_{1:t-1} = n \in \{1,\ldots,N_{t-1}\}$, and let $\mathcal{H}_t^{njk}$ be the index of the posterior hypotheses for $r_t = j \in \mathcal{S}$ and $q_t = k \in \{0,\ldots,Y_t\}$. Then, by Bayes rule:

$$\begin{aligned}\mu_t^{q_{1:t},r_{0:t}} &= p\left(q_{1:t}, r_{0:t}|\boldsymbol{\mathcal{Y}}_{1:t}\right) = p\left(\mathcal{H}_t^{njk}|\boldsymbol{\mathcal{Y}}_{1:t}\right) = \mu_t^{njk} \\ &= \frac{p(\mathcal{H}_{t-1}^n)p(r_t = j|\mathcal{H}_{t-1}^n)\Pr(q_t = k|\mathcal{H}_{t-1}^n)p(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{njk})}{p(\boldsymbol{\mathcal{Y}}_t)}.\end{aligned} \tag{4.2.10}$$

Here $p(\boldsymbol{\mathcal{Y}}_t)$ is a normalising constant and $p(\mathcal{H}_{t-1}^n) = \mu_{t-1}^{q_{1:t-1},r_{0:t-1}}$ is the prior weight of hypothesis $\mathcal{H}_{t-1}^n$. $p(r_t = j|\mathcal{H}_{t-1}^n) = \pi_{r_{t-1}^n j}$ is the probability of the system switching to mode j given the prior mode, r_{t-1}^n being associated with density n. Consider now $p(q_t = k|\mathcal{H}_{t-1}^n)$ and define $P_{FA}(m)$ to be the probability of exactly m of the valid measurements being false. Then if the clutter incidence has a Poisson distribution:

$$P_{FA}(m) = \frac{(\lambda V)^m e^{-\lambda V}}{m!}. \tag{4.2.11}$$

Assuming that at most one of the measurements can be true (or they could all be false), $p(q_t = k|\mathcal{H}_{t-1}^n)$ can be computed as:

$$p(q_t = 0|\mathcal{H}_{t-1}^n) = (1 - P_D)P_{FA}(Y_t)\,, \tag{4.2.12}$$

$$p(q_t = (k \neq 0)|\mathcal{H}_{t-1}^n) = P_D\left(\frac{P_{FA}(Y_t - 1)}{Y_t}\right). \tag{4.2.13}$$

As the measurements in $\boldsymbol{\mathcal{Y}}_t$ are independent of one another, the final term in (4.2.10) is given by:

$$p\left(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{njk}\right) = \prod_{k=1}^{Y_t} p(\boldsymbol{y}_t(k)|\mathcal{H}_t^{njk})\,. \tag{4.2.14}$$

For hypothesis $k = 0$, all measurements are false alarms. As these are uniformly distributed over the surveillance region:

$$p(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{nj0}) = \frac{1}{V^{Y_t}}\,. \tag{4.2.15}$$

For hypotheses $k > 0$, only one measurement is valid (and Normally distributed), while the others are false. Therefore:

$$p(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{njk}) = \frac{\mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj})}{V^{Y_t - 1}}\,. \tag{4.2.16}$$

Substituting (4.2.12) and (4.2.15) into (4.2.10), μ_t^{nj0} becomes:

$$\mu_t^{nj0} = \mu_{t-1}^n \pi_{r_{t-1}^n j} \frac{(1 - P_D)(\lambda V)^{Y_t} e^{-\lambda V}}{\bar{c} Y_t! V^{Y_t}} \tag{4.2.17}$$

$$= \mu_{t-1}^n \pi_{r_{t-1}^n j} \frac{(1 - P_D)\lambda^{Y_t} e^{-\lambda V}}{\bar{c} Y_t!}\,. \tag{4.2.18}$$

Similarly by substituting (4.2.13) and (4.2.16) into (4.2.10), μ_t^{njk} becomes:

$$\mu_t^{njk} = \mu_t^n \pi_{r_{t-1}^n j} \frac{P_D(\lambda V)^{Y_t - 1} e^{-\lambda V} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj})}{\bar{c}(Y_t - 1)! Y_t V^{Y_t - 1}} \tag{4.2.19}$$

$$= \mu_t^n \pi_{r_{t-1}^n j} \frac{P_D \lambda^{Y_t - 1} e^{-\lambda V} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj})}{\bar{c} Y_t!}\,, \tag{4.2.20}$$

where $\bar{c}$ is the normaliser defined as:

$$\bar{c} = \frac{\lambda^{Y_t - 1} e^{-\lambda V}}{Y_t!}\left(\sum_{njk} P_D \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) + (1 - P_D)\lambda\right). \tag{4.2.21}$$

Let:

$$c = \frac{\bar{c} P_D Y_t!}{\lambda^{Y_t - 1} e^{-\lambda V}} \tag{4.2.22}$$

$$= \sum_{njk} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) + \frac{(1 - P_D)\lambda}{P_D}\,. \tag{4.2.23}$$

Then $\mu_t^{q_{1:t}}$ can be simplified to:

$$\mu_t^{q_{1:t}} = \mu_t^{njk} = \frac{1}{c}\mu_t^n \pi_{r_{t-1}^n j} \begin{cases} \frac{(1-P_D)\lambda}{P_D} & \text{for } k = 0\,, \\ \mathcal{N}(\boldsymbol{y}_t^k; \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) & \text{for } k = 1 \ldots Y_t\,. \end{cases} \tag{4.2.24}$$

This optimal solution is clearly not computationally viable, due to the exponentially growing number of terms in the Gaussian mixture. In Appendix B.3 we describe algorithms, such as the IMMPDA and properly constructed particle filters which approximate this conditional density. In the following section we propose an algorithm based on Gaussian mixture approximation which is a direct extensions of the mixture filter presented in Section 3.1.1.

4.3 The multiple measurement mixture filter

In the same manner as the mixture filter described in Section 3.1.1, the *multiple measurement mixture filter* (MMMF) presented in this section is designed to approximate the exact conditional distribution presented in (4.2.5):

$$p(\boldsymbol{x}_t|\mathcal{Y}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \sum_{r_{0:t}\in D_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{q_{1:t},r_{0:t}}, \boldsymbol{P}_t^{q_{1:t},r_{0:t}})\mu_t^{q_{1:t},r_{0:t}}\,, \tag{4.3.1}$$

using a finite number of number of components:

$$\hat{p}(\boldsymbol{x}_t|\mathcal{Y}_{1:t}) = \sum_{n\in\mathcal{M}_t} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^n, \boldsymbol{P}_t^n)\mu_t^n\,. \tag{4.3.2}$$

Here, $\mathcal{M}_t \subset \mathcal{D}_{1:t}D_{0:t}$ is a subset of cardinality N of the set of the joint mode and measurement histories up until time t. When a new set of measurements $\mathcal{Y}_t$ becomes available, the MMMF calculates the exact posterior density as a $N \times d \times (Y_t + 1)$-fold Gaussian mixture, which is then approximated using an N-fold one using mixture reduction techniques.

It should be noted that, as for most other algorithms, gating techniques are needed to carry out a preliminary selection of the measurements in order to discard those which are highly unlikely to be true and hence maintain the computational cost of the iteration manageable. Most gating techniques (which are described in greater detail in Appendix A.2) reject or accept measurements on the basis of the filter's measurement predictions. We integrate the MMMF with the most commonly used ellipsoidal gates, which allow

for a customised measurement selection for each mixture component.

Let $\hat{\boldsymbol{y}}_{t|t-1}^{nj}$ and $\boldsymbol{S}_t^{nj}$ be the measurement prediction and measurement prediction covariance for prior component $n \in \{1,\ldots,N\}$ and mode $j \in \mathcal{S}$, which are calculated using (2.3.13)-(2.3.16) and which can be summarised given as:

$$\hat{\boldsymbol{y}}_{t|t-1}^{nj} = \boldsymbol{H}\big(\boldsymbol{F}(r_t)\hat{\boldsymbol{x}}_{t-1}^{n} + \boldsymbol{u}_t^s\big) + \boldsymbol{u}_t^m\,, \tag{4.3.3}$$

$$\boldsymbol{S}_t^{nj} = \boldsymbol{H}\big(\boldsymbol{F}(r_t)\boldsymbol{P}_{t-1}^{n}(\boldsymbol{F}(r_t))^T + \boldsymbol{Q}_s(r_t)\big)\boldsymbol{H}^T + \boldsymbol{Q}_m\,. \tag{4.3.4}$$

Then the preliminary gating step produces $N \times d$ subsets $\boldsymbol{\mathcal{Z}}_t^{nj}$ of the set of all measurements $\boldsymbol{\mathcal{Y}}_t$ for which the following gating condition is satisfied:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^{nj})^T(\boldsymbol{S}_t^{nj})^{-1}(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^{nj}) \leq G\,. \tag{4.3.5}$$

Here $\boldsymbol{y}_t(i) \in \boldsymbol{\mathcal{Y}}_t$ for $i \in \{1,\ldots,Y_t\}$, and G is the size of the gate. The latter can be calculated using χ^2 tables by means of P_G, a design parameter representing the acceptable probability of the true measurement being rejected. When this form of gating is used, if $Z_t^{nj} = \text{card}(\boldsymbol{\mathcal{Z}}_t^{nj})$, the posterior distribution is a Gaussian mixture comprising $\sum_{n,j}(Z_t^{nj}+1)$ densities (when gating is used, the calculation of the posterior distribution is no longer strictly *exact*). We also note that, as the gating step introduces the concrete possibility of the true measurement being erroneously discarded, P_D, the probability of detection, is updated to $P_{DG} = P_D(1-P_G)$.

Finally, our mixture filter contains the modification of Section 3.2.3(a) which allows distribution-based mixture reduction methods to merge components associated with different modes.

4.3.1 The MMMF algorithm

The multiple measurement mixture filter (MMMF) is presented here in detail.

STEP 1: STATE AND COVARIANCE PREDICTIONS

Given N prior mixture components with mean, covariance, weights and mode probabilities defined by $\hat{\boldsymbol{x}}_{t-1}^{n}$, $\boldsymbol{P}_{t-1}^{n}$, μ_{t-1}^{n} and $\boldsymbol{\gamma}_{t-1}^{n}$ respectively (for $n \in \{1,\ldots,N\}$), the Kalman filter prediction step is performed for each prior component and for every possible mode transition, $j \in \mathcal{S}$, using prediction maps (2.3.22):

$$\{\hat{\boldsymbol{x}}_{t|t-1}^{nj}, \boldsymbol{P}_{t|t-1}^{nj}, \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}\} = \text{KFpred}(\hat{\boldsymbol{x}}_{t-1}^{n}, \boldsymbol{P}_{t|t-1}^{n}, j, t)\,. \tag{4.3.6}$$

STEP 2: MEASUREMENT GATING

Gating is now performed separately for each individual prediction such that $N \times d$ measurement subsets $\mathcal{Z}_t^{nj} \subset \mathcal{Y}_t$ are produced. For each $n \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, d\}$, measurement $\boldsymbol{y}_t(i) \in \mathcal{Y}_t$ is accepted and included in $\mathcal{Z}_t^{nj}$ if:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^{nj})^T (\boldsymbol{S}_t^{nj})^{-1} (\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^{nj}) \leq G\,. \tag{4.3.7}$$

Let $Z_t^{nj} = \text{card}(\mathcal{Z}_t^{nj})$.

STEP 3: MEASUREMENT UPDATE

Using banks of Kalman filters, each of the $N \times d$ predictions is now updated using the measurements $\boldsymbol{y}_t(k) \in \mathcal{Z}_t^{nj}$, which have satisfied its gating requirements. Using the mappings defined in (2.3.23), and for $k \in \{1, \ldots, Z_t^{nj}\}$:

$$\{\hat{\boldsymbol{x}}_t^{njk}, \boldsymbol{P}_t^{njk}\} = \text{KFupd}_x(\hat{\boldsymbol{x}}_{t|t-1}^{nj}, \boldsymbol{P}_{t|t-1}^{nj}, j, \boldsymbol{y}_t(k), t)\,. \tag{4.3.8}$$

Furthermore, the algorithm retains the unupdated state predictions (and their covariance) to model the hypothesis where all the measurements received are false and the true measurement is not detected. Therefore for $k = 0$:

$$\hat{\boldsymbol{x}}_t^{nj0} = \hat{\boldsymbol{x}}_{t|t-1}^{nj}\,, \tag{4.3.9}$$

$$\boldsymbol{P}_t^{nj0} = \boldsymbol{P}_{t|t-1}^{nj}\,. \tag{4.3.10}$$

For $k \in \{0, \ldots, Z_t^{nj}\}$, $\hat{r}_t^{njk} = j$.

STEP 4: WEIGHT CALCULATIONS

If the mode-probability vector of prior component $n \in \{1, \ldots, N\}$ is given by $\boldsymbol{\gamma}_{t-1}^n$, then the weights of the components just generated are calculated using the Kalman filter update map (2.3.24) as follows. For $k \in \{1, \ldots, Z_t^{nj}\}$:

$$\mu_t^{njk} = \frac{1}{c}\mu_{t-1}^n \left(\sum_{l=1}^{d} \pi_{lj}\, \gamma_{t-1}^{nl} \right) \text{KFupd}_y(\boldsymbol{y}_t(k); \hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}^{nj}, j, t)\,, \tag{4.3.11}$$

where γ_{t-1}^{nl} refers to the l^{th} element of $\boldsymbol{\gamma}_{t-1}^n$. If λ is the clutter density and P_{DG} is the probability of the true measurement not being part of set $\mathcal{Z}_t^{nj}$, the probability of the null-hypothesis (i.e. that all measurements are false) is calculated as follows:

$$\mu_t^{nj0} = \frac{1}{c}\mu_{t-1}^n \left(\sum_{l=1}^{d} \pi_{lj}\, \gamma_{t-1}^{nl} \right) \frac{(1 - P_{DG})\lambda}{P_{DG}}\,. \tag{4.3.12}$$

In both equations, c is a normalising term given by:

$$c = \sum_{n,j,k} \mu_{t-1}^{n} \left(\sum_{l=1}^{d} \pi_{lj} \gamma_{t-1}^{nl} \right) \left(\frac{(1 - P_{DG})\lambda}{P_{DG}} + \mathrm{KFupd}_{y}(\boldsymbol{y}_t(k); \hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}^{nj}, j, t) \right). \quad (4.3.13)$$

STEP 5: STATE ESTIMATE & COVARIANCE CALCULATIONS

Next, the output state estimate and covariance are computed by merging together all the elements of the posterior conditional distribution. It should be noted that the output estimates should be calculated prior to the mixture reduction as this inevitably introduces degradations.

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \sum_{k=0}^{Z_t^{nj}} \mu_t^{njk} \hat{\boldsymbol{x}}_t^{njk}, \quad (4.3.14)$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \sum_{k=0}^{Z_t^{nj}} \mu_t^{njk} [\boldsymbol{P}_t^{njk} + (\hat{\boldsymbol{x}}_t^{njk} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{njk} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]. \quad (4.3.15)$$

STEP 6: MIXTURE REDUCTION

The final step of the mixture filter involves the use of a mixture reduction algorithm. A number of these are described in Section 3.2, along with a discussion of their merits and weaknesses. These methods produce an N-fold mixture whose components have parameters $\{\boldsymbol{x}_t^n, \boldsymbol{P}_t^n, \gamma_t^n, \mu_t^n\}$ as described in Section 3.2.3(a).

4.4 Simulation results

We now present simulations that assess the performance of the MMMF compared to other popular techniques for tracking manoeuvring targets in clutter, namely the IMM-PDA and a particle filter. This type of problem, also presents a valuable opportunity to see how the mixture reduction methods compare for a problem involving both data association and multiple modes.

Similarly to the simulation scenario used for the data association problem, the problem considered in this section compares the algorithms on the basis of the 'track-life' (the average time until the tracker 'loses' the target). A 'track-loss' is now registered if the true measurement does not appear in any of the sets of gated measurements $\boldsymbol{\mathcal{Z}}_t^{nj}$

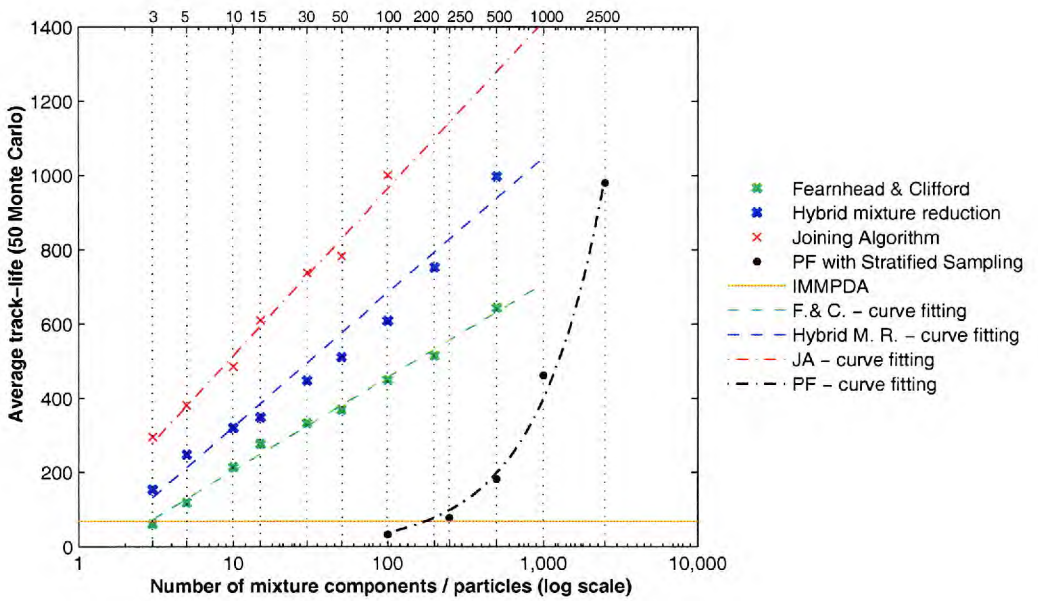


Figure 4.2: Manoeuvring target in clutter: average track-life vs. number of mixture components (50 Monte Carlo average). Exact component/particle numbers are marked at the top of the chart.

for five consecutive time steps, or if all hypotheses are 10σ from the true target position for five consecutive steps. Here σ is defined as the standard deviation of the estimation error a Kalman filter, using only the correct measurements and modes, would produce.

The track is generated for $t = 1, 2, \ldots, 5000$ using a repeated mode sequence $\{2, 3, 1, 2, 3, 1, 2, \ldots\}$ in which each mode transition takes place at intervals of 100 time steps. As for problems presented in [72] and in [62], a large value $\lambda = 0.012$ is used and the sampling time h, q_m (such that $\boldsymbol{Q}_m = q_m \boldsymbol{I}_{2\times 2}$) and q_s (as given in (2.3.9)) are all normalised to unity. The rectangular surveillance region (in which clutter measurements are generated), has sides equal to $200\sqrt{q_m}$. This value was chosen large enough such that, if the filter loses track of the target, a track-loss is recorded long before the estimates can approach its edges. As in [72], $P_D = 1$, and $P_G = 0.01$ (which results in a relatively small gate) are used.

Figure 4.2 shows the track-life vs. the number of mixture components used for the MMMF (using three different mixture reduction techniques), as well as the performance

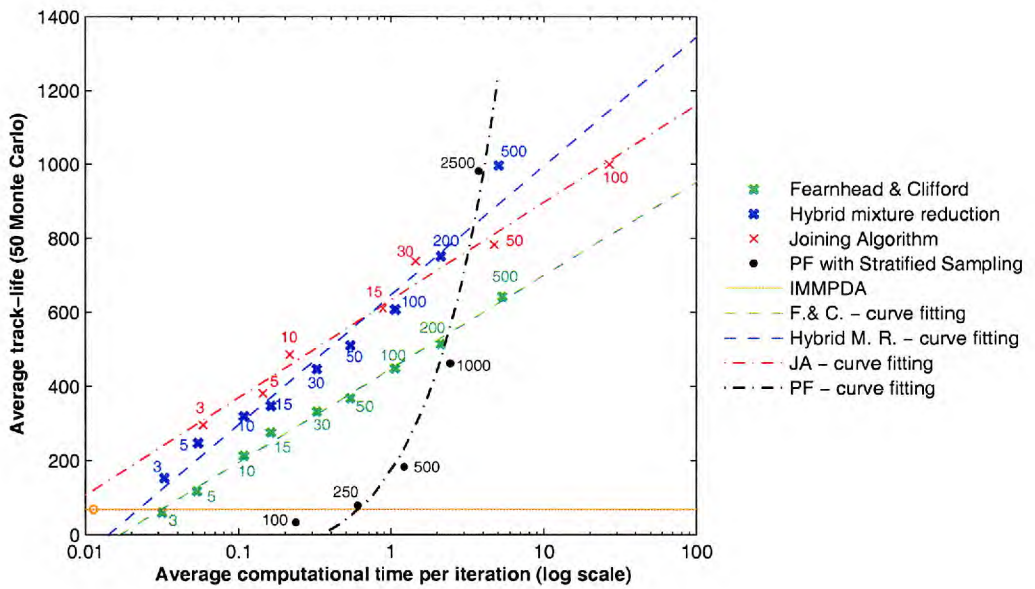


Figure 4.3: Manoeuvring target in clutter: average track-life vs. average computational time of each iteration (50 Monte Carlo average).

of the IMMPDA and the perfomance per particle of a particle filter (these algorithms are implemented as described in Appendix B.3). The mixture reduction algorithms tested were Salmond's joining algorithm (which emerged as the most efficient of the distribution-based methods), the Fearnhead and Clifford algorithm (which was chosen to represent weight-based methods) and the new hybrid mixture reduction scheme.

All the mixture reduction methods outperform the IMMPDA with only few mixture components, and display a (roughly) logarithmic relationship between track-life and N. The joining algorithm provides the best results per mixture components (and the steepest gradient), followed by the hybrid method and by the Fearnhead & Clifford algorithm. However, once the same track-lives are plotted against the computational cost of one iteration, the lead of the joining algorithm substantially narrows, such that its performance is comparable to that of the hybrid method. In this chart, the latter also exhibits the steepest gradient.

It should be noted that the particle filter displays a linear (instead of logarithmic) relationship between particle number/computational cost and track-life. This clearly

makes it the algorithm of choice when abundant computational resources are available. Nonetheless the MMMF (using either the joining algorithm, or the hybrid method) provides the most efficient solution for a large range of the computational time spectrum, and provides a valid alternative for problems in which IMMPDA is not adequate but for which the cost of a particle filter is too high.

Chapter 5

Bearings-only Manoeuvring Target Tracking

This chapter shows how mixture filters, if designed with appropriate estimators, are capable of solving problems in which the nonlinearity is no longer limited to multiple modes or measurements, but in which the system equations themselves are nonlinear. The problem investigated, bearing-only manoeuvring target tracking, includes nonlinearities in the measurement equation as well as mode switching in the state dynamics. A mixture filter using as estimator the *shifted Rayleigh filter*, a recently proposed moment-matching filter, is developed and shown to outperform traditional particle filters in terms of track accuracy and computational efficiency.

5.1 Traditional methods for bearings-only tracking

Bearings-only target tracking is a difficult problem faced whenever passive sensors are used. These sensors scan specific portions of the electromagnetic spectrum (usually the acoustic, infrared or microwave) for possible target emissions, and as such usually allow a measurement of the direction of the target's signal but not of its distance. Bearings-only measurements produced by these devices therefore provide only limited information about the target and as such introduce nonlinearity.

Figure 5.1 shows a sample bearings-only tracking scenario in which a moving passive observer takes measurements of a moving target. From the plot, it is straight-forward

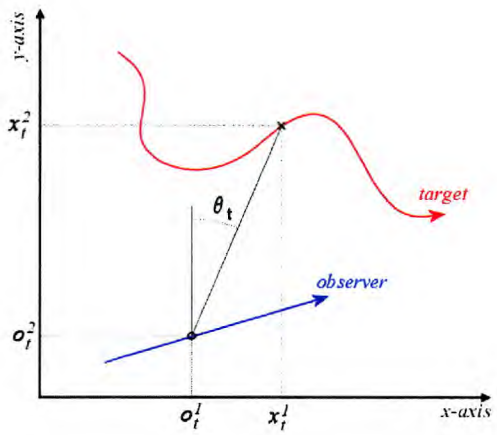


Figure 5.1: The bearings-only measurement (shown with no noise).

to derive an equation for the measurement as a function of the relative displacement $\boldsymbol{d}_t = [d_t^1, d_t^2]^T$ between the position of the target ($\boldsymbol{x}_t$) and that of the observer ($\boldsymbol{o}_t$). Let:

$$d_t^1 = x_t^1 - o_t^1\,, \tag{5.1.1}$$

$$d_t^2 = x_t^2 - o_t^2\,, \tag{5.1.2}$$

then, if $\eta_t \sim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian noise process, the angle θ_t, measured clockwise from the North (or positive y-axis), is given by:

$$\theta_t = \arctan\left(\frac{d_t^1}{d_t^2}\right) + \eta_t\,. \tag{5.1.3}$$

To deal with the severe nonlinearity presented in (5.1.3), a number of sub-optimal filtering approaches have been presented in the literature and can generally be grouped in two main filter categories: *moment-matching filters* and *density approximation filters*. Moment-matching techniques, such as the extended Kalman filter (EKF - discussed in Section 2.1.3) or the unscented Kalman filter (UKF- see Section 2.1.4), generally match a Gaussian distribution to the first and second moment of the state distribution. By contrast filters in the 'density approximation' category, mainly particle filters (Section 2.2), propagate a number of random samples from the prior distribution to produce a representation of the posterior whose accuracy is dependent on the sample size used. Particle filters are generally very accurate but are significantly more computationally intensive then the simpler moment-matching techniques.

Although the EKF and the UKF can be directly applied to bearings-only tracking problems, given the significance of this type of problem, a number of algorithms have been designed to deal with the specific nonlinearity featured in (5.1.3). The majority of these algorithms, such as the 'modified polar coordinates extended Kalman filter' (MPEKF) [2], are extensions to the standard (Cartesian coordinate) EKF that in some way attempt to overcome its limitations.

In the case of the MPEKF, the nonlinearity is shifted from the measurement equation to the state, by transforming it to polar coordinates. If $\phi_t = \arctan(d_t^1/d_t^2)$ is the true bearing, then the polar coordinates state, $\boldsymbol{x}_t$, becomes:

$$\boldsymbol{x}_t = \left[\phi_t,\ \dot{\phi}_t,\ 1/r_t,\ \dot{r}_t/r_t\right]^T , \tag{5.1.4}$$

and the measurement equation simplifies to:

$$\theta_t = [1\ 0\ 0\ 0]^T \boldsymbol{x}_t + \eta_t . \tag{5.1.5}$$

According to [2], this method prevents filter instability by decoupling observable components from the unobservable. In spite of this improvement over the standard (Cartesian coordinate) EKF, the nature of the algorithm remains unchanged and the algorithm still suffers from the problems described in Section 2.1.3, including possible divergence. Furthermore, the MPEKF tends to be sensitive to initialisation errors, which can be problematic as the initial range of the target is generally unknown.

Another approach, known as the *range-parameterised extended Kalman filter* (RPEKF) [57], is to employ banks of MPEKF filters, each associated with a different initial range estimate. The RPEKF is effectively a simple mixture filter carrying a fixed number of weighted estimates. After a few iterations, the likelihood of most of the components normally falls to negligible levels, and these estimates are discarded. The algorithm may eventually reduce to a normal MPEKF, as all but one initial range estimates are eliminated. With these arrangements, the RPEKF was shown to produce accurate estimates while being robust to initial conditions [8].

In an entirely different approach, known as the *pseudomeasurement filter* [1]-[3], Aidala *et al.* propose an algorithm which, for the linear state equation of (2.1.14) and the non-

linear measurement equation of (5.1.3), makes use of a 'virtual' Cartesian measurement (known as 'pseudomeasurement') in the form of:

$$\boldsymbol{y}_t^* = \boldsymbol{H}_t^* \boldsymbol{x}_t + \mu_t(\boldsymbol{x}_t, \eta_t) \,, \tag{5.1.6}$$

where:

$$\boldsymbol{H}_t^* = [\cos(\theta_t),\ 0,\ -\sin(\theta_t),\ 0] \,, \tag{5.1.7}$$

$$\mu_t(\boldsymbol{x}_t, \eta_t) = r_t \sin(\eta_t) \,. \tag{5.1.8}$$

$\mu_t(\boldsymbol{x}_t, \eta_t)$ is zero-mean and in its covariance, $\boldsymbol{Q}_t^*$, is a function of the bearing measurement's variance, σ_t^2.

Since the predicted virtual measurement is simply given by $\hat{\boldsymbol{y}}_t^* = \boldsymbol{H}_t^* \hat{\boldsymbol{x}}_{t|t-1}$, the pseudomeasurement filter makes use of the linear Kalman filter equations presented in Section 2.1.2 where the Cartesian measurement is replaced by the pseudomeasurement.

The performance of the pseudomeasurement filter is discussed in [1], [3], [24], [65], and it is generally agreed that, although it is generally stable, it can give rise to biased estimates in difficult scenarios characterised by long target range and/or significantly noisy measurements. In more benign scenarios, the pseudomeasurent filter can however produce stable, high-quality tracks.

5.2 The shifted Rayleigh filter

In this section the 'shifted Rayleigh filter' (SRF), a technique recently introduced by Clark *et al* [24] to solve the bearings-only problem, is presented. The SRF is a moment-matching technique which enables the *exact* calculation of the non-Gaussian posterior distribution of the target (given a Gaussian prior), to which a Normal distribution is then fitted by matching its first and second moments. The latter is the only approximation used by this technique, which otherwise involves only exact calculations. This implies that if the prior conditional distribution is close to Normal, then the algorithm produces estimates comparable to the MMSE estimator.

5.2.1 Measurement treatment

The exact calculation of the posterior distribution is made possible by a treatment of the measurement which is slightly different from its traditional formulation in (5.1.3) where white noise is added directly to the angle. In the modified formulation, white noise is added to a non-observable 'virtual' measurement $\boldsymbol{y}_t^*$ (not to be confused with that of the pseudomeasurement filter), which is then converted to angular form via the 'arctan' function. Assuming the position of the observer is incorporated in the exogenous input $\boldsymbol{u}_t^m$, the measurement equation is now given by:

$$\boldsymbol{y}_t^* = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + (\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^m\,, \tag{5.2.1}$$

$$\theta_t = \arctan\left(\frac{y_t^{*1}}{y_t^{*2}}\right)\,, \tag{5.2.2}$$

$$\boldsymbol{b}_t = [\sin\theta_t, \cos\theta_t]^T\,. \tag{5.2.3}$$

where $\boldsymbol{\nu}_t^m \sim \mathcal{N}(0, \boldsymbol{I}_{2\times 2})$, and where $\boldsymbol{b}_t$ – the *bearing vector* – is the projection of the measurement on the unit circle. This treatment of the measurement causes the white noise to also undergo the nonlinear transformation, which facilitates calculations.

As shown in [24], this approach is not dissimilar to the standard measurement treatment. Traditionally the bearing error $\theta_t - \phi_t$, where ϕ_t is the true bearing given by:

$$\phi_t \;=\; \arctan\left(\frac{y_t^1}{y_t^2}\right) \quad \text{where} \quad \boldsymbol{y}_t \;=\; \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m\,,$$

is modelled by a zero-mean Gaussian random variable with variance σ_t^2 (i.e. a 'wrapped' normal density on $[-\pi, +\pi]$). However in the shifted Rayleigh filter the measurement error is taken to be the difference between the bearing of $(\sin\phi_t, \cos\phi_t)$ and the bearing of the projection onto the unit circle of the random variable $(\sin\phi_t, \cos\phi_t) + \tilde{\boldsymbol{\nu}}_t$, where $\tilde{\boldsymbol{\nu}}_t$ is a two dimensional $\mathcal{N}(0, \sigma_t^2\boldsymbol{I}_{2\times 2})$ random variable independent of ϕ_t. The measurement error, in this sense, is

$$\arctan\left(\frac{\sin\phi_t + \tilde{\nu}_t^1}{\cos\phi_t + \tilde{\nu}_t^2}\right) - \phi_t\,.$$

If $\alpha_\sigma(\theta_t)$ is the bearing density of this error, it can be shown (see [24] and Section 5.3.1 for more details) that if $\sigma^2 \leq 1$ (in which case the wrapped normal density is virtually indistinguishable from the Normal density restricted to $[-\pi, +\pi]$), the difference between the density $\alpha_\sigma(\theta_t)$ and the wrapped normal density never exceeds 0.04 (see Figure 5.2).

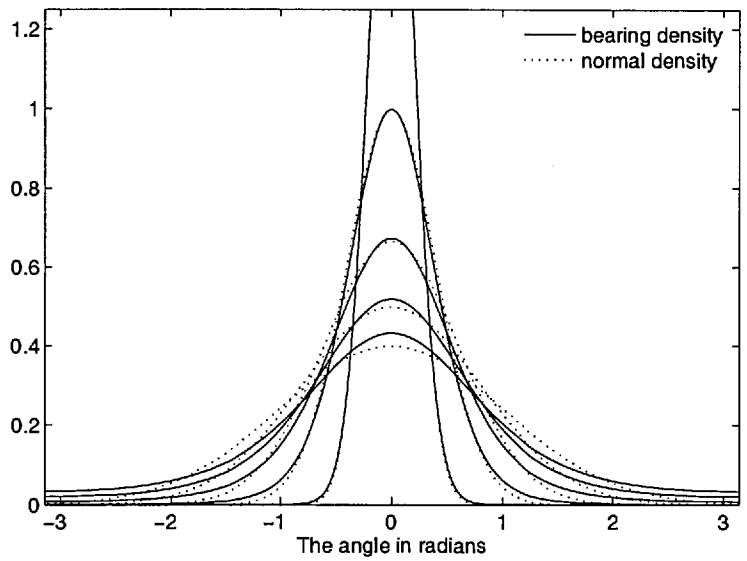


Figure 5.2: The bearing probability density $\alpha_\sigma(\theta_t)$ compared with the Normal $\mathcal{N}(0, \sigma^2)$ density, for $\sigma = 0.2, 0.4, \ldots, 1$. (For this range of values the wrapped Normal density is indistinguishable from the Normal density.)

Given this modified description of the noise, the one-step predicted covariance of the measurement noise, $\boldsymbol{Q}_t^m$, can be calculated by making use of the fact that the projection onto the unit circle of $(\sin\phi_t, \cos\phi_t) + \tilde{\boldsymbol{\nu}}_t$ is equivalent to the projection of $\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + E[\|\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m\|^2]\tilde{\boldsymbol{\nu}}_t$ onto the unit circle. If an extra symmetric matrix $\boldsymbol{Q}_t^{tr}$ is added to take account of random translations in sensor location, the predicted covariance of the measurement noise becomes:

$$\boldsymbol{Q}_t^m = \boldsymbol{Q}_t^{tr} + \sigma^2 E[\|\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}^m\|^2]\boldsymbol{I}_{2\times 2}. \tag{5.2.4}$$

Here, $E[\|\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m\|^2]$ has the function of range of estimate and can be estimated as:

$$E[\|\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m\|^2] = \left\|\boldsymbol{H}\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{u}_t^m\right\|^2 + \mathrm{tr}[\boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T]\,, \tag{5.2.5}$$

where $\hat{\boldsymbol{x}}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$ are linear predictions.

5.2.2 Derivation of the shifted Rayleigh filter

Using the measurement model just described, the equations of the *shifted Rayleigh filter* (SRF) are now derived and we show how they can provide a recursive solution based on

exact calculations of the conditional distribution of the target. The following presentation, based largely on [24] and [75], aims at presenting the concepts in as intuitive and logical way as possible while still rigorously deriving the filter's formulae. For a more complete and formal presentation including derivations for three dimensional bearings-only problems, the reader is referred to [24] or [75].

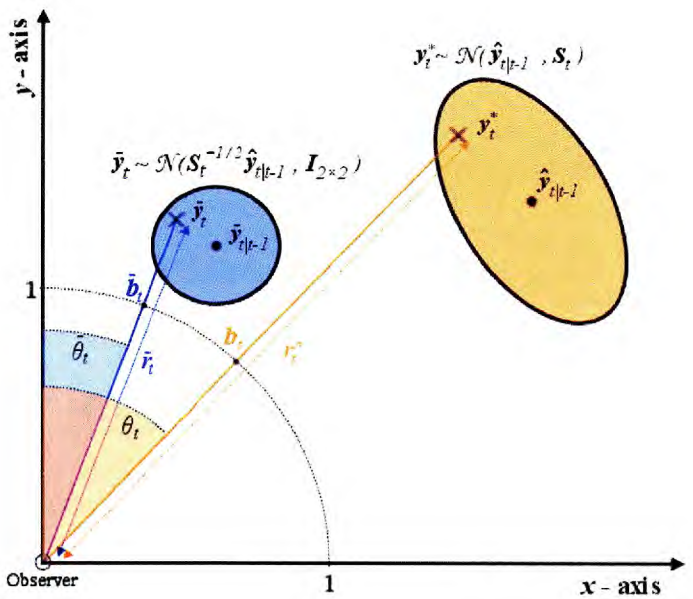


Figure 5.3: Normalisation of the virtual measurement.

Consider the 'virtual' measurement, $\boldsymbol{y}_t^*$, of (5.2.1). If this was directly measurable, the system would reduce to linear and Gaussian. The state could then be described in terms of the Kalman filter equation (2.1.47) as:

$$\boldsymbol{x}_t = \boldsymbol{\xi}_t + \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t^* - \hat{\boldsymbol{y}}_{t|t-1}), \tag{5.2.6}$$

where $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \boldsymbol{P}_t)$ is the estimation error. The objective of the shifted Rayleigh filter is to use the information available (i.e the bearing θ_t and the bearing vector $\boldsymbol{b}_t = (\sin\theta_t, \cos\theta_t)^T$) to estimate the virtual measurement so that properly adjusted versions of the Kalman filter may be used to estimate the state. The following relationships are

known:

1) $y_t^* = r_t^* b_t$, where r_t^* is the unknown range of the virtual measurement, (5.2.7)

2) $y_t^* \sim \mathcal{N}(\hat{y}_{t|t-1}, S_t)$. (5.2.8)

Consider now a normalised version of the virtual measurement (see Figure 5.3) such that its covariance is set to unity. The 'normalised virtual measurement', $\bar{y}_t \sim \mathcal{N}(S_t^{-1/2}\hat{y}_{t|t-1}, I_{2\times 2})$, can then be written in terms of:

$$\bar{y}_t = S_t^{-1/2} y_t^* \tag{5.2.9}$$

$$= \bar{r}_t \bar{b}_t \,. \tag{5.2.10}$$

Here, $\bar{r}_t$ and $\bar{b}_t$ are respectively the range and the bearing vector of the normalised virtual measurement, and are given by:

$$\bar{r}_t = \left\| S_t^{-1/2} y_t^* \right\| , \tag{5.2.11}$$

$$\bar{b}_t = \bar{r}_t^{-1} S_t^{-1/2} y_t^* . \tag{5.2.12}$$

By substituting (5.2.7) and (5.2.11) in this equation, the normalised bearing vector can be expressed in terms of known parameters:

$$\begin{aligned} \bar{b}_t &= \left\| r_t^* S_t^{-1/2} b_t \right\|^{-1} r_t^* S_t^{-1/2} b_t \\ &= \left\| S_t^{-1/2} b_t \right\|^{-1} S_t^{-1/2} b_t \\ &= (b_t^T S_t^{-1} b_t)^{-1/2} S_t^{-1/2} b_t . \end{aligned} \tag{5.2.13}$$

If we now rearrange (5.2.9) and substitute for $\bar{y}_t$ using (5.2.10), the virtual measurement can be rewritten as:

$$\begin{aligned} y_t^* &= S_t^{1/2} \bar{y}_t \\ &= \bar{r}_t S_t^{1/2} \bar{b}_t \,. \end{aligned} \tag{5.2.14}$$

If this is substituted back into (5.2.6), the conditional expectation of x_t given the measured bearing vector b_t is given by:

$$E[x_t|b_t] = E[\xi_t] + \hat{x}_{t|t-1} + K_t(E[\bar{r}_t|\bar{b}_t] S_t^{1/2} \bar{b}_t - \hat{y}_{t|t-1}) \tag{5.2.15}$$

$$= \hat{x}_{t|t-1} + K_t(E[\bar{r}_t|\bar{b}_t] S_t^{1/2} \bar{b}_t - \hat{y}_{t|t-1}) , \tag{5.2.16}$$

since $\boldsymbol{\xi}_t$ is zero-mean and independent of $\boldsymbol{y}_t^*$. This formulation of the problem has the advantage of narrowing the scope of the estimation to $E[\bar{r}_t|\bar{\boldsymbol{b}}_t]$, which is the first moment of $p(\bar{r}_t|\bar{\boldsymbol{b}}_t)$: the conditional density of $\bar{r}_t$ given $\bar{\boldsymbol{b}}_t$ (the latter being computable using known parameters as shown in (5.2.13)).

Using conditional probability laws, $p(\bar{r}_t|\bar{\boldsymbol{b}}_t)$ can be written in terms of:

$$p(\bar{r}_t|\bar{\boldsymbol{b}}_t) = \frac{p(\bar{r}_t, \bar{\boldsymbol{b}}_t)}{p(\bar{\boldsymbol{b}}_t)}, \tag{5.2.17}$$

where the joint density $p(\bar{r}_t, \bar{\boldsymbol{b}}_t)$ can be calculated by transforming the Normal density of the normalised virtual measurement from its Cartesian form (shown in Figure 5.3) to polar form. If $J(\bar{y}_t^1, \bar{y}_t^2)$ is the Jacobian of the transformation, then:

$$p(\bar{r}_t, \bar{\boldsymbol{b}}_t) = p_{\bar{\boldsymbol{y}}_t}(\bar{y}_t^1, \bar{y}_t^2) \left|J(\bar{y}_t^1, \bar{y}_t^2)\right| . \tag{5.2.18}$$

Here, $p_{\bar{\boldsymbol{y}}_t}(\bar{y}_t^1, \bar{y}_t^2)$ is Normal and 2-dimensional, and is given by:

$$p_{\bar{\boldsymbol{y}}_t}(\bar{y}_t^1, \bar{y}_t^2) = \mathcal{N}(\boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1}, \boldsymbol{I}_{2\times 2}) \tag{5.2.19}$$

$$= \frac{1}{2\pi} e^{-\frac{1}{2}(\bar{\boldsymbol{y}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})(\bar{\boldsymbol{y}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})^T} . \tag{5.2.20}$$

As the transformation involves a change of variables from $(\bar{y}_t^1, \bar{y}_t^2)$ to $(\bar{r}_t, \bar{\theta}_t)$, where $\bar{\theta}_t$ is the bearing of the normalised virtual measurement, the relationship between these variables is given by:

$$\bar{y}_t^1 = \bar{r}_t \sin\bar{\theta}_t \qquad\qquad \bar{y}_t^2 = \bar{r}_t \cos\bar{\theta}_t . \tag{5.2.21}$$

It follows that the Jacobian of the transformation is given by:

$$J(\bar{y}_t^1, \bar{y}_t^2) = \begin{vmatrix} \frac{\partial \bar{y}_t^1}{\partial \bar{r}_t} & \frac{\partial \bar{y}_t^1}{\partial \bar{\theta}_t} \\ \frac{\partial \bar{y}_t^2}{\partial \bar{r}_t} & \frac{\partial \bar{y}_t^2}{\partial \bar{\theta}_t} \end{vmatrix} = \begin{vmatrix} \sin(\bar{\theta}_t) & \bar{r}_t\cos(\bar{\theta}_t) \\ \cos(\bar{\theta}_t) & -\bar{r}_t\sin(\bar{\theta}_t) \end{vmatrix}$$

$$= -\bar{r}_t \sin(\bar{\theta}_t)^2 - \bar{r}_t \cos(\bar{\theta}_t)^2 = -\bar{r}_t . \tag{5.2.22}$$

Replacing (5.2.20) and (5.2.22) into (5.2.18), the joint density $p(\bar{r}_t, \bar{\boldsymbol{b}}_t)$ is obtained as:

$$\begin{aligned} p(\bar{r}_t, \bar{\boldsymbol{b}}_t) &= p_{\bar{\boldsymbol{y}}_t}(\bar{y}_t^1, \bar{y}_t^2) \left|J(\bar{y}_t^1, \bar{y}_t^2)\right| \\ &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{\boldsymbol{y}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})(\bar{\boldsymbol{y}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})^T} \\ &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r}_t\bar{\boldsymbol{b}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})(\bar{r}_t\bar{\boldsymbol{b}}_t - \boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})^T} \\ &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r}_t^2\bar{\boldsymbol{b}}_t^T\bar{\boldsymbol{b}}_t - \bar{r}_t\bar{\boldsymbol{b}}_t^T\boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1} - \bar{r}_t(\boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1})^T\bar{\boldsymbol{b}}_t + \hat{\boldsymbol{y}}_{t|t-1}^T\boldsymbol{S}_t^{-1}\hat{\boldsymbol{y}}_{t|t-1})} . \end{aligned}$$

Note that $\bar{\boldsymbol{b}}_t^T \bar{\boldsymbol{b}}_t = \sin(\bar{\theta}_t)^2 + \cos(\bar{\theta}_t)^2 = 1$. Also, $\bar{r}_t \bar{\boldsymbol{b}}_t^T \boldsymbol{S}_t^{-1/2} \hat{\boldsymbol{y}}_{t|t-1} = \bar{r}_t (\boldsymbol{S}_t^{-1/2} \hat{\boldsymbol{y}}_{t|t-1})^T \bar{\boldsymbol{b}}_t$, and let:

$$z_t = \bar{\boldsymbol{b}}_t^T \boldsymbol{S}_t^{-1/2} \hat{\boldsymbol{y}}_{t|t-1} \,. \tag{5.2.23}$$

Then the joint density can be simplified to:

$$\begin{aligned} p(\bar{r}_t, \bar{\boldsymbol{b}}_t) &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r}^2 - 2\bar{r}z_t + \hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1})} \\ &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r}^2 - 2\bar{r}z_t + z_t^2 + \hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \\ &= \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r} - z_t)^2} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \,. \end{aligned} \tag{5.2.24}$$

From this expression, it follows that the conditional distribution of $\bar{r}_t$ given $\bar{\boldsymbol{b}}_t$ is a 'shifted' Rayleigh density[1] of the form:

$$p(\bar{r}_t | \bar{\boldsymbol{b}}_t) = \frac{p(\bar{r}_t, \bar{\boldsymbol{b}}_t)}{p(\bar{\boldsymbol{b}}_t)} = \frac{\bar{r}_t e^{-\frac{1}{2}(\bar{r} - z_t)^2}}{\int_0^\infty \bar{r}_t e^{-\frac{1}{2}(\bar{r} - z_t)^2} d\bar{r}_t} \,. \tag{5.2.25}$$

As it was shown in (5.2.16), in order to generate an estimate of the state given the bearing vector (i.e. $E[\boldsymbol{x}_t | \boldsymbol{b}_t]$), it is necessary to calculate the first moment of $p(\bar{r}_t | \bar{\boldsymbol{b}}_t)$. Let this be the function of z_t, $\rho(z_t)$. Then:

$$\begin{aligned} \rho(z_t) = E[\bar{r}_t | \bar{\boldsymbol{b}}_t] &= \int_0^\infty \bar{r}_t p(\bar{r}_t | \bar{\boldsymbol{b}}_t) \, d\bar{r}_t \\ &= \frac{\int_0^\infty \bar{r}_t^2 e^{-\frac{1}{2}(\bar{r} - z_t)^2} d\bar{r}_t}{\int_0^\infty \bar{r}_t e^{-\frac{1}{2}(\bar{r} - z_t)^2} d\bar{r}_t} \,. \end{aligned} \tag{5.2.26}$$

In order to calculate $\rho(z_t)$ in computable form, consider the *cumulative distribution function* of $\mathcal{N}(0,1)$, and let:

$$F(z_t) = \frac{1}{2} \mathrm{erfc}\left(\frac{-z_t}{\sqrt{2}}\right) = \int_{-\frac{z_t}{\sqrt{2}}}^\infty e^{-s^2} ds \,. \tag{5.2.27}$$

Here, $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$ is the *complementary error function.* Next define the integrals :

$$J_m(z_t) = \int_{-z_t}^\infty \bar{r}_t^m e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t \,, \tag{5.2.28}$$

---

[1]The Rayleigh distribution of a random variable, r, is $f(r) = \frac{re^{-\frac{r^2}{2\sigma^2}}}{\sigma^2}$, and has mean $\sigma\sqrt{\pi/2}$ and variance $2\pi\sigma^2$.

where:

$$J_0(z_t) = \int_{-z_t}^{\infty} e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t \qquad \text{(let } s^2 = \bar{r}_t^2/2\text{, then } d\bar{r}_t = \sqrt{2}ds\text{)}$$
$$= \sqrt{2}\int_{-\frac{z_t}{\sqrt{2}}}^{\infty} e^{-s^2} ds = \sqrt{2\pi}F(z_t)\,, \tag{5.2.29}$$
$$J_1(z_t) = \int_{-z_t}^{\infty} \bar{r}_t e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t$$
$$= -e^{-\frac{\bar{r}_t^2}{2}}\Big|_{-z_t}^{\infty} = e^{-\frac{z_t^2}{2}}\,, \tag{5.2.30}$$

and where, for $m > 1$:

$$J_m(z_t) = \int_{-z_t}^{\infty} \bar{r}_t^m e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t = \int_{-z_t}^{\infty} \bar{r}_t^{m-1} \bar{r}_t e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t\,. \tag{5.2.31}$$

By use of integration by parts, let:

$$u = \bar{r}_t^{m-1} \qquad dv = \bar{r}_t e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t$$
$$du = (m-1)\bar{r}_t^{m-2} d\bar{r}_t \qquad v = -e^{-\frac{\bar{r}_t^2}{2}}\,.$$

Therefore:

$$J_m(z_t) = udv = uv - vdu$$
$$= -\bar{r}_t^{m-1} e^{-\frac{\bar{r}_t^2}{2}}\Big|_{-z_t}^{\infty} + (m-1)\int_{-z_t}^{\infty} \bar{r}_t^{m-2} e^{-\frac{\bar{r}_t^2}{2}} d\bar{r}_t$$
$$= 0 - z_t^{m-1} e^{-\frac{z_t^2}{2}} + (m-1)J_{m-2}(z_t)\,. \tag{5.2.32}$$

Returning to the evaluation of $\rho(z_t)$, the numerator of (5.2.26) is equal to:

$$\int_0^{\infty} \bar{r}_t^2 e^{-\frac{1}{2}(\bar{r}-z_t)^2} d\bar{r}_t = \qquad \text{(let } \bar{r}_t = s + z_t\text{, then } d\bar{r}_t = ds\text{)}$$
$$= \int_{-z_t}^{\infty} (s+z_t)^2 e^{-\frac{s^2}{2}} ds = \int_{-z_t}^{\infty} (s^2 + 2sz_t + z_t^2) e^{-\frac{s^2}{2}} ds$$
$$= \int_{-z_t}^{\infty} s^2 e^{-\frac{s^2}{2}} ds + 2z_t \int_{-z_t}^{\infty} s e^{-\frac{s^2}{2}} ds + z_t^2 \int_{-z_t}^{\infty} e^{-\frac{s^2}{2}} ds$$
$$= z_t^2 J_0(z_t) + 2z_t J_1(z_t) + \int_{-z_t}^{\infty} s^2 e^{-\frac{s^2}{2}} ds\,.$$

By using (5.2.29), (5.2.30) and (5.2.32), where $m = 2$, the numerator of $\rho(z_t)$ simplifies to:

$$\int_0^{\infty} \bar{r}_t^2 e^{-\frac{1}{2}(\bar{r}-z_t)^2} d\bar{r}_t = \sqrt{2\pi} z_t^2 F(z_t) + 2z_t e^{-\frac{z_t^2}{2}} - z_t e^{-\frac{z_t^2}{2}} + \sqrt{2\pi}F(z_t)$$
$$= z_t e^{-\frac{z_t^2}{2}} + \sqrt{2\pi}(z_t^2 + 1)F(z_t)\,. \tag{5.2.33}$$

Similarly, the denominator of $\rho(z_t)$:

$$\begin{aligned}\int_0^\infty \bar{r}_t e^{-\frac{1}{2}(\bar{r}-z_t)^2} d\bar{r}_t &= \int_{-z_t}^\infty (s+z_t) e^{-\frac{s^2}{2}} ds \\ &= \int_{-z_t}^\infty s e^{-\frac{s^2}{2}} ds + z_t \int_{-z_t}^\infty e^{-\frac{s^2}{2}} ds \\ &= J_1(z_t) + z_t J_0(z_t) \\ &= e^{-\frac{z_t^2}{2}} + \sqrt{2\pi} z_t F(z_t)\,. \end{aligned} \tag{5.2.34}$$

By substituting (5.2.33) and (5.2.34) into (5.2.26), the formula for the first moment of $p(\bar{r}_t|\bar{\boldsymbol{b}}_t)$ (i.e $\rho(z_t)$) is given by:

$$\begin{aligned}\rho(z_t) &= \frac{z_t e^{-\frac{z_t^2}{2}} + \sqrt{2\pi}(z_t^2+1)F(z_t)}{e^{-\frac{z_t^2}{2}} + \sqrt{2\pi} z_t F(z_t)} \\ &= \frac{z_t + \sqrt{2\pi}(z_t^2+1)e^{\frac{z_t^2}{2}} F(z_t)}{1 + \sqrt{2\pi} z_t e^{\frac{z_t^2}{2}} F(z_t)}\,. \end{aligned} \tag{5.2.35}$$

While $\rho(z_t)$ is sufficient to calculate the conditional expectation of the state given the bearing vector (i.e. $E[\bar{r}_t|\boldsymbol{b}_t]$), in order to calculate the conditional covariance, we also need to calculate, $\sigma^2(z_t)$, the second moment of $p(\boldsymbol{x}_t|\bar{\boldsymbol{b}}_t)$. This can be done by an analysis of the relationship between moments. Recall (5.2.24):

$$p(\bar{r}_t|\bar{\boldsymbol{b}}_t) = \frac{\bar{r}_t e^{-\frac{1}{2}(\bar{r}_t - z_t)^2}}{\int_0^\infty \bar{r}_t e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} d\bar{r}_t} = \frac{1}{K(z)} \bar{r}_t e^{-\frac{1}{2}(\bar{r}_t-z_t)^2}\,, \tag{5.2.36}$$

where $K(z)$ is the normalising integral. The n^{th} moment of this distribution is then equal to:

$$\begin{aligned} E[\bar{r}_t^n] &= K(z)^{-1} \int_0^\infty \bar{r}_t^{n+1} e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} d\bar{r}_t \\ &= K(z)^{-1} \int_0^\infty \bar{r}_t^n e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} (\bar{r}_t - z_t + z_t) d\bar{r}_t \\ &= K(z)^{-1} \left(\int_0^\infty \bar{r}_t^n (\bar{r}_t - z_t) e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} d\bar{r}_t + z \int_0^\infty \bar{r}_t^n e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} d\bar{r}_t \right) \\ &= K(z)^{-1} \int_0^\infty \bar{r}_t^n (\bar{r}_t - z_t) e^{-\frac{1}{2}(\bar{r}_t-z_t)^2} d\bar{r}_t + z E[\bar{r}_t^{n-1}]\,. \end{aligned}$$

By use of integration by parts, let:

$$\begin{aligned} u &= \bar{r}_t^n & dv &= (\bar{r}_t - z_t) e^{-\frac{(\bar{r}_t-z_t)^2}{2}} d\bar{r}_t \\ du &= (n)\bar{r}_t^{n-1} d\bar{r}_t & v &= -e^{-\frac{(\bar{r}_t-z_t)^2}{2}}\,. \end{aligned}$$

Therefore:

$$\begin{aligned}\int_0^\infty \bar{r}_t^n(\bar{r}_t - z_t)e^{-\frac{1}{2}(\bar{r}_t - z_t)^2}d\bar{r}_t &= udv = uv - vdu \\ &= -\bar{r}_t^n e^{-\frac{(\bar{r}_t - z_t)^2}{2}}\Big|_0^\infty + n\int_0^\infty \bar{r}_t^{n-1}e^{-\frac{(\bar{r}_t - z_t)^2}{2}}d\bar{r}_t \\ &= (0-0) + nE[\bar{r}_t^{n-2}]\,. \end{aligned} \tag{5.2.37}$$

The n^{th} moment is then equal to:

$$E[\bar{r}_t^n] = nE[\bar{r}_t^{n-2}] + z_t E[\bar{r}_t^{n-1}]\,. \tag{5.2.38}$$

In the special case of $n = 2$, the variance of $p(\bar{r}_t|\bar{\boldsymbol{b}}_t)$ is determined as:

$$\begin{aligned}\sigma^2(z_t) &= E[\bar{r}_t^2] - \rho(z_t)^2 \\ &= 2E[\bar{r}_t^0] + z_t E[\bar{r}_t] - \rho(z_t)^2 \\ &= 2 + z_t\rho(z_t) - \rho(z_t)^2\,. \end{aligned} \tag{5.2.39}$$

With these moments worked out, it is finally possible to return to the conditional expectation of $\boldsymbol{x}_t$ given $\boldsymbol{b}_t$. Recall from (5.2.16), that:

$$\begin{aligned}E[\boldsymbol{x}_t|\boldsymbol{b}_t] &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(E[\bar{r}_t|\bar{\boldsymbol{b}}_t]\boldsymbol{S}_t^{1/2}\bar{\boldsymbol{b}}_t - \hat{\boldsymbol{y}}_{t|t-1}) \\ &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\rho(z_t)\boldsymbol{S}_t^{1/2}\bar{\boldsymbol{b}}_t - \hat{\boldsymbol{y}}_{t|t-1})\,. \end{aligned} \tag{5.2.40}$$

We showed in (5.2.13) how $\bar{\boldsymbol{b}}_t$ can be expressed in terms of the bearing and the measurement prediction covariance as $\bar{\boldsymbol{b}}_t = (\boldsymbol{b}_t^T\boldsymbol{S}_t^{-1}\boldsymbol{b}_t)^{-1/2}\boldsymbol{S}_t^{-1/2}\boldsymbol{b}_t$. Also, by letting:

$$\gamma_t = (\boldsymbol{b}_t^T\boldsymbol{S}_t^{-1}\boldsymbol{b}_t)^{-1/2}\rho(z_t)\,, \tag{5.2.41}$$

the conditional expectation simplifies to:

$$E[\boldsymbol{x}_t|\boldsymbol{b}_t] = \hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\gamma_t\boldsymbol{b}_t - \hat{\boldsymbol{y}}_{t|t-1})\,. \tag{5.2.42}$$

This is clearly equivalent to the Kalman filter where the virtual measurement is estimated by means of γ_t, itself the estimate of the range r_t^*.

The condition covariance of the state given the bearing vector (i.e. $\text{cov}[\boldsymbol{x}_t|\boldsymbol{b}_t] = E[(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t})^T]$) can similarly be calculated. Using (5.2.6) and (5.2.16), we can write:

$$(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t}) = \boldsymbol{\xi}_t + (\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])\boldsymbol{K}_t\boldsymbol{S}_t^{1/2}\bar{\boldsymbol{b}}_t\,. \tag{5.2.43}$$

Therefore the conditional covariance becomes:

$$\begin{aligned}\mathrm{cov}[\boldsymbol{x}_t|\boldsymbol{b}_t] &= E[(\boldsymbol{\xi}_t + (\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])\boldsymbol{K}_t\boldsymbol{S}^{1/2}\bar{\boldsymbol{b}}_t)(\boldsymbol{\xi}_t + (\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])\boldsymbol{K}_t\boldsymbol{S}^{1/2}\bar{\boldsymbol{b}}_t)^T] \\ &= E[\boldsymbol{\xi}_t^2] + E[(\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])(\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])^T]\boldsymbol{K}_t\boldsymbol{S}_t^{1/2}\bar{\boldsymbol{b}}_t\bar{\boldsymbol{b}}_t^T(\boldsymbol{S}_t^{1/2})^T\boldsymbol{K}_t^T\,.\end{aligned} \tag{5.2.44}$$

Here $E[\boldsymbol{\xi}_t^2]$ is the covariance of the estimation, as normally calculated by means of the Kalman filter (see (2.1.47)), and $E[(\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])(\bar{r}_t - E[\bar{r}_t|\bar{\boldsymbol{b}}_t])^T] = \sigma^2(z_t)$ is the conditional variance of the range of the normalised virtual measurement given its bearing vector. By letting δ_t be the scaled conditional variance given by:

$$\delta_t = (\boldsymbol{b}_t^T\boldsymbol{S}_t^{-1}\boldsymbol{b}_t)^{-1}\sigma^2(z_t)\,, \tag{5.2.45}$$

it is possible to simplify the conditional covariance of the state to:

$$\mathrm{cov}[\boldsymbol{x}_t|\boldsymbol{b}_t] = \boldsymbol{P}_{t|t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{P}_{t|t-1} + \delta_t\boldsymbol{K}_t\boldsymbol{b}_t\boldsymbol{b}_t^T\boldsymbol{K}_t^T\,. \tag{5.2.46}$$

This is clearly equivalent to the Kalman filter covariance update equation with the extra non-negative term $\delta_t\boldsymbol{K}_t\boldsymbol{b}_t\boldsymbol{b}_t^T\boldsymbol{K}_t^T$ added. The latter increases the state covariance to compensate for the fact that only partial information is available about the virtual measurement (i.e. only its bearing).

This concludes the derivation of the shifted Rayleigh filter which, given a Normal prior distribution, was shown to exactly calculate the conditional density of the bearing (which is then approximated by a Normal distribution matching its first two moments) . It should be noted that the SRF can be directly extended for 3-dimensional problems (see either [24] or [75]).

5.2.3 Summary of the shifted Rayleigh filter equations

The equations of the Shifted Rayleigh filter are now presented concisely for future reference. Furthermore, in order to simplify the explanation of the next section, the state equation is assumed to be mode dependent as in (2.3.10). As the algorithm presented so far is single-mode, consider for this case that $d = 1$.

In an effort to keep notation tidy, the time indicator for the state transition matrix and the state noise covariance matrix (which are normally time-invariant) has been dropped.

Similarly the mode indicator has been omitted from those variables introduced exclusively to tidy the formulation of others (i.e. $\boldsymbol{V}_t$, $\boldsymbol{Q}_t^m$, $\boldsymbol{K}_t$, ζ_t, z_t, $\rho(z_t)$, γ_t, and δ_t) and which would not normally be required independently.

PREDICTION STAGE: (5.2.47)

$$\hat{\boldsymbol{x}}_{t|t-1}^j = \boldsymbol{F}(j)\hat{\boldsymbol{x}}_{t-1} + \boldsymbol{u}_t^s\,, \tag{5.2.48}$$

$$\boldsymbol{P}_{t|t-1}^j = \boldsymbol{F}(j)\boldsymbol{P}_{t-1}\boldsymbol{F}(j)^T + \boldsymbol{Q}_s(j)\,, \tag{5.2.49}$$

$$\hat{\boldsymbol{y}}_{t|t-1}^j = \boldsymbol{H}\,\hat{\boldsymbol{x}}_{t|t-1}^j + \boldsymbol{u}_t^m\,, \tag{5.2.50}$$

$$\boldsymbol{V}_t = \boldsymbol{H}\,\boldsymbol{P}_{t|t-1}^j\,\boldsymbol{H}^T\,, \tag{5.2.51}$$

$$\boldsymbol{Q}_t^m = \boldsymbol{Q}_t^{tr} + \sigma^2\,, \left(\left\|\hat{\boldsymbol{y}}_{t|t-1}^j\right\| + \mathrm{tr}[\boldsymbol{V}_t]\right)\,, \tag{5.2.52}$$

$$\boldsymbol{S}_t^j = \boldsymbol{V}_t + \boldsymbol{Q}_t^m\,. \tag{5.2.53}$$

UPDATE STAGE:

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}^j\,\boldsymbol{H}^T(\boldsymbol{S}_t^j)^{-1}\,, \tag{5.2.54}$$

$$\zeta_t = (\boldsymbol{b}_t^T\,(\boldsymbol{S}_t^j)^{-1}\,\boldsymbol{b}_t)^{-1/2}\,, \tag{5.2.55}$$

$$z_t = \zeta_t\,\boldsymbol{b}_t^T\,\boldsymbol{S}_t^j\,\hat{\boldsymbol{y}}_{t|t-1}^j\,, \tag{5.2.56}$$

$$F(z_t) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{-z_t}{\sqrt{2}}\right)\,, \tag{5.2.57}$$

$$\rho(z_t) = \frac{z_t e^{-\frac{z_t^2}{2}} + \sqrt{2\pi}(z_t^2+1)F(z_t)}{e^{-\frac{z_t^2}{2}} + \sqrt{2\pi}z_t F(z_t)}\,, \tag{5.2.58}$$

$$\gamma_t = \zeta_t\rho(z_t)\,, \tag{5.2.59}$$

$$\delta_t = \zeta_t^2\left(2 + z_t\rho(z_t) - \rho(z_t)^2\right)\,, \tag{5.2.60}$$

$$\hat{\boldsymbol{x}}_t^j = \hat{\boldsymbol{x}}_{t|t-1}^j + \boldsymbol{K}_t(\gamma_t\boldsymbol{b}_t - \hat{\boldsymbol{y}}_{t|t-1}^j)\,, \tag{5.2.61}$$

$$\boldsymbol{P}_t^j = \boldsymbol{P}_{t|t-1}^j - \boldsymbol{K}_t\boldsymbol{H}\,\boldsymbol{P}_{t|t-1}^j + \delta_t\boldsymbol{K}_t\boldsymbol{b}_t\boldsymbol{b}_t^T\,\boldsymbol{K}_t^T\,. \tag{5.2.62}$$

For a concise formulation of the SRF, the following mappings are also introduced. Define, for $j \in \{1,\ldots,d\}$:

$$\mathrm{SR}_x(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, \theta_t, r_t = j, t) \;=\; (\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j)\,, \tag{5.2.63}$$

$$\mathrm{SRupd}_x(\hat{\boldsymbol{x}}_{t|t-1}, \boldsymbol{P}_{t|t-1}, \theta_t, r_t = j, t) \;=\; (\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j)\,. \tag{5.2.64}$$

While the first map includes the entire shifted Rayleigh cycle (and updates the state estimate for a given prior), the second map includes only its update stage and is useful

when describing an algorithm with multiple sensor updates in Section 5.4.

5.3 Extension to bearings-only manoeuvring tracking

The extension of bearings-only target tracking to incorporate manoeuvring targets is a problem including multiple nonlinearities in the form of partial measurements and switching modes. The system equations for such problems is usually of the form:

$$\boldsymbol{x}_t = \boldsymbol{F}(r_t)\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s\,, \tag{5.3.1}$$

$$\theta_t = \arctan\left(\frac{d_t^1}{d_t^2}\right) + \eta_t\,, \tag{5.3.2}$$

where d_t^1 and d_t^2 are as defined in (5.1.1) and (5.1.2) respectively. A similar formulation of the form:

$$\boldsymbol{x}_t = \boldsymbol{F}(r_t)\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s\,, \tag{5.3.3}$$

$$\boldsymbol{y}_t^* = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + (\boldsymbol{Q}_t^m)^{1/2}\boldsymbol{\nu}_t^m\,, \tag{5.3.4}$$

$$\theta_t = \arctan\left(\frac{y_t^{*1}}{y_t^{*2}}\right)\,, \tag{5.3.5}$$

$$\boldsymbol{b}_t = [\sin(\theta_t)\cos(\theta_t)]^T\,, \tag{5.3.6}$$

where $\boldsymbol{u}_t^m$ incorporates the observer's coordinates, allows the problem to be adapted for use with algorithms based on the shifted Rayleigh filter (SRF).

The traditional approach to problems of this type has been to use the interacting multiple model (IMM) algorithm with banks of extended Kalman filters (EKF) to carry out the estimation of each mode. This technique, generally referred to as the IMM-EKF features in a number of papers (see for example, [48], [59] and [7]) and has been widely adopted. More recently, the IMM-UKF, a technique first appearing in [59], presents a similar approach but in which the IMM uses the unscented Kalman filters (UKF) as estimator.

In [59], Ristic and Aralumpalam also introduce the use of particle filters for the problem. Given their simulation based approach, these techniques are particularly well suited for the approximation of a number of nonlinearities. Although these solutions are considerably more computationally intensive than the IMM-based algorithms, by employing

a sufficiently large number of samples they can produce accurate solutions. The use of particle filters for this problem was later expanded by Arulampalam *et al.* in [7], where more refined particle filters, including a Rao-Blackwellised version using the EKF as the state estimator, are also presented.

In [59], and more extensively in [7], the authors compare the performance of these algorithms and conclude that the particle filters provide 'superior performance', largely due to their scalability and their ability to deal with nonstandard information. However the computational complexity of these algorithms is considerably greater than that of the IMM-based techniques, and hence there is a clear need for scalable algorithms which can perform at a lower cost.

In this section, the use of mixture filters of the form presented in Section 3.1.1 is proposed and their performance is compared with that of other algorihtms. While a mixture filter based on the UKF is also tested, the key result is the development of the *shifted Rayleigh mixture filter* (SRMF). This technique, first presented in [23], makes use of the exact bearings-only calculations of the SRF and is shown to be as accurate as particle filters while being of an order of magnitude cheaper computationally.

5.3.1 Evaluation of the bearing density

In order for the shifted Rayleigh filter to be useful as an estimator in a mixture filter, we need to be able to evaluate the density of the bearing, so that the different components of the mixture can be weighted and ranked. For the linear measurement case, the measurement density was Gaussian and described by:

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t, \boldsymbol{P}_t)) = \mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}_{t|t-1}, \boldsymbol{S}_t)\,. \tag{5.3.7}$$

In the case of the nonlinear measurement model described in (5.3.4)-(5.3.6), the joint density of the range and bearing vector of the normalised virtual measurement, $p(\bar{r}_t, \bar{\boldsymbol{b}}_t)$, can be used to provide a similar measure. Recall that in (5.2.24), this density was derived to be:

$$p(\bar{r}_t, \bar{\boldsymbol{b}}_t) = \frac{\bar{r}_t}{2\pi} e^{-\frac{1}{2}(\bar{r}-z_t)^2} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)}\,. \tag{5.3.8}$$

The density of the normalised bearing vector can then be calculated by integrating out the range:

$$p(\bar{\boldsymbol{b}}_t) = \int_0^\infty p(\bar{r}_t, \bar{\boldsymbol{b}}_t) d\bar{r}_t \tag{5.3.9}$$

$$= \frac{1}{2\pi} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \int_0^\infty \bar{r}_t e^{-\frac{1}{2}(\bar{r}-z_t)^2} d\bar{r}_t \,. \tag{5.3.10}$$

Let $\bar{r}_t = s + z_t$ and $d\bar{r}_t = dt$. Then:

$$p(\bar{\boldsymbol{b}}_t) = \frac{1}{2\pi} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \int_0^\infty (s + z_t) e^{-\frac{1}{2}s^2} ds \tag{5.3.11}$$

$$= \frac{1}{2\pi} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \left(\int_{-z_t}^\infty s e^{-\frac{1}{2}s^2} ds + z_t \int_{-z_t}^\infty e^{-\frac{1}{2}s^2} ds \right) . \tag{5.3.12}$$

Using results in (5.2.29) and (5.2.30), the bearing density simplifies to:

$$p(\bar{\boldsymbol{b}}_t) = \frac{1}{2\pi} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \left(J_1(z_t) + z_t J_0(z_t) \right) \tag{5.3.13}$$

$$= \frac{1}{2\pi} e^{-\frac{1}{2}(\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1} - z_t^2)} \left(e^{-\frac{z_t^2}{2}} + \sqrt{2\pi} z_t F(z_t) \right) \tag{5.3.14}$$

$$= \frac{1}{2\pi} e^{-\frac{1}{2\sigma_t^2}} \left(1 + \sqrt{2\pi} z_t e^{\frac{z_t^2}{2}} F(z_t) \right) , \tag{5.3.15}$$

where $F(z_t) = \frac{1}{2} \operatorname{erfc}\left(\frac{-z_t}{\sqrt{2}}\right)$ is the cumulative distribution function of $\mathcal{N}(0,1)$ and where $\sigma_t = (\hat{\boldsymbol{y}}_{t|t-1}^T \boldsymbol{S}_t^{-1} \hat{\boldsymbol{y}}_{t|t-1})^{-1/2}$ is the reciprocal of the length of the normalised measurement. Now consider the case in which the true bearing is 0, and where the mean of the bearing vector is $\hat{\boldsymbol{y}}_{t|t-1} = (0,1)^T$. In this case, from (5.2.23):

$$z_t = \left(\sin(\bar{\theta}_t), \cos(\bar{\theta}_t)\right) \sigma_t^{-1} (0,1)^T \tag{5.3.16}$$

$$= \cos(\bar{\theta}_t) \sigma_t^{-1} . \tag{5.3.17}$$

Let $\alpha_{\sigma_t}(\bar{\theta}_t)$ be the resulting distribution, this can then be written as:

$$\alpha_{\sigma_t}(\bar{\theta}_t) = \frac{1}{2\pi} e^{-\frac{1}{2\sigma_t^2}} \left(1 + \sqrt{2\pi} \frac{\cos(\bar{\theta}_t)}{\sigma_t} e^{\frac{1}{2}\frac{\cos(\bar{\theta}_t)}{\sigma_t}} F\left(\frac{\cos(\bar{\theta}_t)}{\sigma_t} \right) \right) . \tag{5.3.18}$$

Now consider the general case for any true bearing. As illustrated in Figure 5.4, the angle of interest is given by $\bar{\theta}_t - \bar{\tilde{\theta}}_t$, where, for $[a_{ij}] = \boldsymbol{S}_t^{-1/2}$:

$$\bar{\theta}_t = \angle \left(\boldsymbol{S}_t^{-1/2} \boldsymbol{y}_t \right) \tag{5.3.19}$$

$$= \angle \begin{pmatrix} a_{11} \sin\theta_t + a_{21} \cos\theta_t \\ a_{21} \sin\theta_t + a_{22} \cos\theta_t \end{pmatrix} \tag{5.3.20}$$

$$= \arctan \left(\frac{a_{11} \sin\theta_t + a_{21} \cos\theta_t}{a_{21} \sin\theta_t + a_{22} \cos\theta_t} \right) , \tag{5.3.21}$$

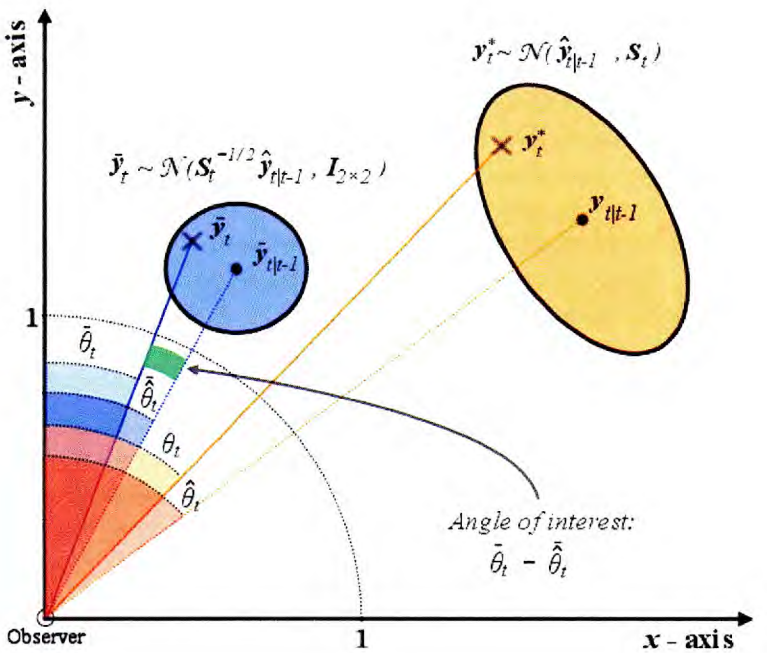


Figure 5.4: The angle of interest when calculating the bearing density.

is the bearing of normalised virtual measurement, and where:

$$\hat{\bar{\theta}}_t = \angle\left(\boldsymbol{S}_t^{-1/2}\hat{\boldsymbol{y}}_{t|t-1}\right) \tag{5.3.22}$$

$$= \angle\begin{pmatrix} a_{11}\sin\hat{\theta}_t + a_{21}\cos\hat{\theta}_t \\ a_{21}\sin\hat{\theta}_t + a_{22}\cos\hat{\theta}_t \end{pmatrix} \tag{5.3.23}$$

$$= \arctan\left(\frac{a_{11}\sin\hat{\theta}_t + a_{21}\cos\hat{\theta}_t}{a_{21}\sin\hat{\theta}_t + a_{22}\cos\hat{\theta}_t}\right), \tag{5.3.24}$$

is the bearing of the normalised predicted measurement. Here $\hat{\theta}_t = \arctan(\hat{y}^1_{t|t-1}/\hat{y}^2_{t|t-1})$. Clearly the density of $\bar{\theta}_t - \hat{\bar{\theta}}_t$ can be found using $\alpha_{\sigma_t}(\bar{\theta}_t - \hat{\bar{\theta}}_t)$. For this general case, let the reciprocal length of the normalised predicted measurement be $\sigma_t = g(\bar{\theta}_t) = (\bar{\boldsymbol{b}}_t^T\,(\boldsymbol{S}_t^j)^{-1}\,\bar{\boldsymbol{b}}_t)^{-1/2}$. Since the distribution of the measurement involves a transformation from the normalised measurement, the bearing density becomes equal to:

$$f(\theta_t|\hat{\boldsymbol{x}}_t^j) = \frac{g(\theta_t)^2}{|\boldsymbol{S}_t|^{1/2}}\,\alpha_{g(\bar{\theta}_t)}(\bar{\theta}_t - \hat{\bar{\theta}}_t) \tag{5.3.25}$$

$$= \frac{\zeta_t^2}{|\boldsymbol{S}_t|^{1/2}}\,\alpha_{g(\bar{\theta}_t)}(\bar{\theta}_t - \hat{\bar{\theta}}_t)\,, \tag{5.3.26}$$

where the $\zeta_t^2 |\boldsymbol{S}_t|^{-1/2}$ term is the Jacobian of the transformation.

It should be noted however that, as shown in Figure 5.2, $\alpha_{g(\bar{\theta}_t)}(\bar{\theta}_t - \bar{\hat{\theta}}_t)$ becomes identical to a Normal distribution for small values of $g(\bar{\theta}_t)$. Since the latter is more easily computed, it is advised that $\mathcal{N}(\bar{\theta}_t; \bar{\hat{\theta}}_t, g(\bar{\theta}_t)^2)$ is used when $g(\bar{\theta}_t) < T$. In the simulations in this thesis, T was chosen to be 0.04, thus using the Gaussian approximation exclusively when the degradation caused is truly negligible. This threshold value could be increased in benign scenarios where a trade-off between computational cost and bearing density accuracy might be desirable (note that, as shown in [24], the maximum possible error is of the order of 4%).

Before moving on to the next section, it is convenient to introduce mapping for the likelihood of the shifted Rayleigh update. Define:

$$\mathrm{SR}_\theta(\theta_t; \hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, r_t = j, t) = f(\theta_t | \hat{\boldsymbol{x}}_t^j)\,, \tag{5.3.27}$$

$$\mathrm{SRupd}_\theta(\theta_t; \hat{\boldsymbol{x}}_{t|t-1}, \boldsymbol{P}_{t|t-1}, r_t = j, t) = f(\theta_t | \hat{\boldsymbol{x}}_t^j)\,. \tag{5.3.28}$$

5.3.2 The shifted Rayleigh mixture filter

A mixture filter based on the shifted Rayleigh filter is now developed for those bearings-only problems involving manoeuvring targets. First presented in [23], the *shifted Rayleigh mixture filter* (SRMF) follows the standard mixture filter structure presented in Section 3.1.1 where the Kalman filter is now replaced by the SRF.

At each step, the algorithm processes an N-fold Gaussian mixture approximation to the prior conditional distribution of the state, to produce an $N{\times}d$-fold Gaussian mixture approximation to the updated conditional distribution in light of the new measurement. Each term in the posterior has mean and covariance calculated using the shifted Rayleigh update equations (5.2.61) and (5.2.62). The final stage involves reducing the $N{\times}d$-fold Gaussian mixture to an N-fold Gaussian mixture, in preparation for the next recursion. For this purpose, we include in the following algorithm the modification of Section 3.2.3 to allow the use of distribution-based mixture reduction methods.

STEP 1: MODE-MATCHED SHIFTED RAYLEIGH FILTERING

Given N prior mixture components with mean, covariance, weights and mode probabilities defined by $\hat{\boldsymbol{x}}_{t-1}^n$, $\boldsymbol{P}_{t-1}^n$, μ_{t-1}^n, and γ_{t-1}^n respectively (for $n \in \{1, \ldots, N\}$), the SRMF calculates the exact posterior conditional distribution by using banks of mode-matched shifted Rayleigh filters to generate a weighted sum of estimates for each possible mode transition for each prior component. For modes $j \in \mathcal{S}$, this is done by means of the maps defined in (5.2.63) and (5.3.27):

$$(\hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}_t^{nj}) = \mathrm{SR}_x(\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, \theta_t, j, t)\,, \tag{5.3.29}$$

$$\tilde{\mu}_t^{nj} = \mu_{t-1}^n \left(\sum_{l=1}^{d} \gamma_{t-1}^{nl} \pi_{lj} \right) \mathrm{SR}_\theta(\theta_t; \hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, j, t)\,, \tag{5.3.30}$$

$$\mu_t^{nj} = \frac{1}{c} \tilde{\mu}_t^{nj}, \text{ where } c = \sum_{n,j} \tilde{\mu}_t^{nj}\,. \tag{5.3.31}$$

Here, γ_{t-1}^{nl} refers to the l^{th} element of γ_{t-1}^n.

STEP 2: STATE ESTIMATE & COVARIANCE CALCULATIONS

At this stage, the output state estimate and covariance are computed by further merging together all the elements of the posterior conditional distribution.

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \mu_t^{nj} \hat{\boldsymbol{x}}_t^{nj}\,, \tag{5.3.32}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \sum_{j=1}^{d} \mu_t^{nj} [\boldsymbol{P}_t^{nj} + (\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]\,. \tag{5.3.33}$$

STEP 3: MIXTURE REDUCTION

The final step of the Gaussian mixture filter involves a mixture reduction algorithms. A number of these are described in Section 3.2, along with a discussion of their merits and weaknesses. These methods produce an N-fold mixture whose components have parameters $\{\boldsymbol{x}_t^n, \boldsymbol{P}_t^n, \gamma_t^n, \mu_t^n\}$ as described in Section 3.2.3(a).

5.4 Assessment of the SRMF in a multi-sensor target tracking problem

In this section, simulations compare the performance of the shifted Rayleigh mixture filter with that of a particle filter and also of a nonlinear variant of the mixture filter presented in Section 3.1.1, in which the unscented Kalman filter is used as estimator. We also compare our filter with versions of the IMM using the SRF and the UKF as nonlinear estimators.

5.4.1 Problem definition and state model

In order to compare the algorithms in a challenging scenario likely to highlight differences, the bearings-only tracking problem considered in this thesis is a variant of the problem earlier investigated in [25], [45] and [50]. The motion of a submerged manoeuvring target is tracked by three drifting sonabuoys providing noisy bearings measurements of the target position, which are subject to clutter in the form of *vacuous measurements* uniformly distributed in the range $[-\pi, +\pi]$. The reader should note that this definition of 'clutter' is different from the one used in earlier parts of this thesis, where 'clutter' referred to *extra measurements* ('false alarms').

A mobile airborne vehicle provides noisy, clutter-free bearing measurements of the sonabuoys' positions. Each of three sonabuoy sensors moves as the sum of a low intensity Brownian motion and an integrated Brownian motion common to all three, that represents the effect of a 'bulk' drift.

The target equation is as presented in (5.3.1), where three possible mode values ($d = 3$) are used (the exogenous state term is zero):

$$\boldsymbol{x}_t = \boldsymbol{F}(r_t)\boldsymbol{x}_{t-1} + (\boldsymbol{Q}_s(r_t))^{1/2}\boldsymbol{\nu}_t^s \,. \tag{5.4.1}$$

Here the continuous component of the state (describing the positions and velocities of the target and the observation platforms for each mode) is the 12-dimensional vector:

$$\boldsymbol{x}_t = [x^o(t), \dot{x}^o(t), y^o(t), \dot{y}^o(t), x^1(t), y^1(t), x^2(t), y^2(t), x^3(t), y^3(t), u^1(t), u^2(t)]^T \,.$$

Here x and y denote coordinates of position in the plane, and $\dot{x}$ and $\dot{y}$ the corresponding velocities. The target and the three sonabuoys are labelled by the superscripts $o, 1, 2$ and 3 respectively. The variables (u^1, u^2) generate the common bulk drift term in the equations governing the positions of the three sonabuoys.

In (5.4.1), the normalized input noise term $\boldsymbol{\nu}_t^s$ has density $\mathcal{N}(0, \boldsymbol{I}_{12\times 12})$ and

$$\boldsymbol{F}(r_t) = \mathrm{diag}(\boldsymbol{A}(r_t), \boldsymbol{I}_{4\times 4}, \boldsymbol{I}_{4\times 4}) + h\left[\boldsymbol{O}_{12\times 10},\ \boldsymbol{D}^T\right] \,.$$

Here $\boldsymbol{O}_{m\times n}$ is an $m \times n$ matrix of zeros and

$$\boldsymbol{D} \ = \ [\boldsymbol{O}_{2\times 4},\ \boldsymbol{I}_{2\times 2},\ \boldsymbol{I}_{2\times 2},\ \boldsymbol{I}_{2\times 2},\ \boldsymbol{O}_{2\times 2}] \,.$$

The 4×4 matrix $\boldsymbol{A}(r_t)$ describes the dynamics of the target, which are modelled using the coordinated turn model of Section 2.3.1(a). For a real number ω_0, representing the target's fastest turn rate,

$$\begin{aligned}\boldsymbol{A}(r_t = 1) &= \boldsymbol{A}_{-\omega_0}\,,\\ \boldsymbol{A}(r_t = 2) &= \boldsymbol{A}_0\,,\\ \boldsymbol{A}(r_t = 3) &= \boldsymbol{A}_{+\omega_0}\,,\end{aligned}$$

where A_ω is as defined in (2.3.7). Similarly the mode dependent system noise covariance matrices are

$$\boldsymbol{Q}_s(r_t) \;=\; \mathrm{diag}\,(q_1\boldsymbol{R}(r_t),\, q_2\boldsymbol{I}_{2\times 2},\, q_2\boldsymbol{I}_{2\times 2},\, q_2\boldsymbol{I}_{2\times 2},\, q_3\boldsymbol{I}_{2\times 2})\;,$$

where:

$$\begin{aligned}\boldsymbol{R}(r_t = 1) &= \boldsymbol{R}_{-\omega_0}\,,\\ \boldsymbol{R}(r_t = 2) &= \boldsymbol{R}_0\,,\\ \boldsymbol{R}(r_t = 3) &= \boldsymbol{R}_{+\omega_0}\,,\end{aligned}$$

and $\boldsymbol{R}_\omega$ is as defined in (2.3.8).

5.4.2 Multisensor measurement model

The problem considered clearly involves multiple sensors (as well as clutter, and in order to adapt the tracking algorithms to this type of problem, it is necessary to modify the measurement equations presented in (5.3.4)-(5.3.6). Firstly clutter is incorporated by modelling the measurement as ψ_t, where:

$$\psi_t = (1 - \kappa_t)\theta_t + \kappa_t U_t\,. \tag{5.4.2}$$

Here θ_t is considered to be the true measurement as described in (5.3.5), and the random variables U_t are independent and uniformly distributed on the interval $[-\pi, +\pi]$ and describe a (possible) random clutter measurement. Its occurrence is dictated by the random variable κ_t, which is independent and has common mass function:

$$\Pr(\kappa_t = 1) \;=\; P_c \quad \text{and} \quad \Pr(\kappa_t = 0) \;=\; 1 - P_c\,.$$

The constant P_c , $0 \leq P_c \leq 1$ represents the probability of a clutter measurement.

Next, if J is the total number of sensors, the 3 measurement equations (5.3.4)-(5.3.6) are replaced by $4 \times J$ equations for the sets of virtual measurements $\{\boldsymbol{y}_t^k\}_{k=1}^J$, noisy measurements $\{\theta_t^k\}_{k=1}^J$ and clutter corrupted measurements $\{\psi_t^k\}_{k=1}^J$. For $k \in \{1 \ldots J\}$:

$$\boldsymbol{y}_t^k = \boldsymbol{H}^k \boldsymbol{x}_t + \boldsymbol{u}_t^m + (\boldsymbol{Q}_t^{mk})^{1/2} \boldsymbol{\nu}_t^k , \tag{5.4.3}$$

$$\theta_t^k = \arctan\left(y_t^{k,1} / y_t^{k,2}\right) , \tag{5.4.4}$$

$$\psi_t^k = (1 - \kappa_t^k)\theta_t^k + \kappa_t^k U_t^k , \tag{5.4.5}$$

$$\boldsymbol{b}_t^k = [\sin(\psi_t^k), \cos(\psi_t^k)]^T . \tag{5.4.6}$$

In this particular problem, $J = 6$ simultaneous measurements are made at each time step. The three monitor-sensor measurements are identified by index values $k = \{1, 2, 3\}$, and the three sensor-target measurements by index values $k = \{4, 5, 6\}$. Notice that the order assigned to the measurements is a design decision, and in some cases one ordering might produce better estimates than another. One sensible procedure for assigning an order would be to choose measurements according to their predicted error variances, with the most accurate first.

Let the displacements associated with the measurements, $\boldsymbol{d}_t^k = [d_t^{k1}, d_t^{k2}]^T$, be:

$$\boldsymbol{d}_t^k = \begin{cases} \left[x_t^k - x_t^{ms}\,,\; y_t^k - y_t^{ms}\right]^T & \text{for } k = \{1, 2, 3\}, \\ \left[x_t^0 - x_t^{k-3}\,,\; y_t^0 - y_t^{k-3}\right]^T & \text{for } k = \{4, 5, 6\}. \end{cases}$$

Here x_t^{ms} and y_t^{ms} are the x and y positions of the monitoring sensor platform, which we regard in this exercise as the deterministic 'exogenous input' $\boldsymbol{u}_t^{mk} = [x_t^{ms}, y_t^{ms}]^T$ for $k = \{1, 2, 3\}$. (For $k = \{4, 5, 6\}$, $\boldsymbol{u}_t^{mk} = \boldsymbol{O}_{2\times 1}$.)

The measurements ψ_t^k are generated by the following equations:

$$\theta_t^k = \arctan(d_t^{k1} / d_t^{k2}) + \tilde{\sigma}_k \eta_t^k , \tag{5.4.7}$$

$$\psi_t^k = (1 - \kappa_t^k)\theta_t^k + \kappa_t^k U_t^k ,$$

for $k = 1, \ldots, 6$. Here η_t^k is a scalar $\mathcal{N}(0, 1)$ distributed noise variable. κ_t^k is a discrete variable taking values 0 or 1 with:

$$\Pr(\kappa_t^k = 1) = P_c^k ,$$

and U_t^k is a variable, uniformly distributed on $[-\pi, \pi]$. All the noise and clutter processes are assumed to be independent. We set:

$$\tilde{\sigma}_k = \begin{cases} \sigma_1 & \text{for } k = 1,2,3\,, \\ \sigma_2 & \text{for } k = 4,5,6\,, \end{cases}$$

and

$$P_c^k = \begin{cases} 0 & \text{for } k = 1,2,3\,, \\ P_c & \text{for } k = 4,5,6\,, \end{cases}$$

(i.e. clutter is present on sensor-target measurements, but not on sensor-monitor measurements).

Notice that the measurement model adopted in the above formulation of the tracking problem is of the traditional kind (bearing angle of the displacement plus scalar noise). For purposes of implementing the SRMF algorithm, we approximate it, replacing (5.4.7) by the two equations:

$$\boldsymbol{y}_t^k = \boldsymbol{d}_t^k + (\boldsymbol{Q}_t^{mk})^{1/2}\boldsymbol{\nu}_t^k\,, \tag{5.4.8}$$

$$\theta_t^k = \arctan(\boldsymbol{y}_t^{k1}/\boldsymbol{y}_t^{k2})\,. \tag{5.4.9}$$

Here $\boldsymbol{\nu}_t^k$ is an $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{2\times 2})$ distributed noise term and:

$$\boldsymbol{Q}_t^{mk}(r) = E\left[||\boldsymbol{d}_t^k||^2 \,\middle|\, \psi_{1:t-1}^k\right] \tilde{\sigma}_k^2 \boldsymbol{I}_{2\times 2}\,.$$

In all our simulations, the traditional measurement model (5.4.7) was used to generate the data. It was also used in the construction of the other algorithms which are compared to the SRMF. These choices put the SRMF at a disadvantage, because there was a mismatch between the measurement model implicit in the filter and the model generating the measurements, as compared with the other algorithms, where there was no such mismatch.

5.4.3 The SRMF for multisensor tracking with vacuous measurement

To adapt the shifted Rayleigh mixture filter for use with this model, the algorithm presented in Section 5.3.2 needs to be modified to allow for multiple sensor updates. Assume an arbitrary ordering of the sensor measurements at each update time (the position of a particular sensor measurement in the ordering is indicated by the value of the index k), and regard the occurrence of sensor measurements as 'separated' in time by a period of

zero seconds.

Fix an estimate update time t and suppose there is available a Gaussian mixture approximation to $p(\boldsymbol{x}_{t-1}|\psi^1_{1:t-1},\ldots,\psi^J_{1:t-1})$. Then a Gaussian mixture approximation to $p(\boldsymbol{x}_t|\psi^1_{1:t})$ (the updated density based on the first measurement) is obtained by applying one step of the SRMF to the state and measurement equations (5.4.1) and (5.4.3)-(5.4.6) (with $k=1$).

Next, a Gaussian mixture approximation to $p(\boldsymbol{x}_t|\psi^1_{1:t},\psi^2_{1:t})$ is obtained by applying one step of the SRMF to the state equation:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + 0 + 0\,,$$

$$\pi_{ij} = \begin{cases} 1 & \text{if } i=j\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

and measurements equations (5.4.3)-(5.4.5) (with $k=2$). The prior state distribution here is taken to be the previously calculated Gaussian mixture approximation to $p(\boldsymbol{x}_t|\psi^1_{1:t})$. (The trivial nature of the system equations reflects the fact that the time period is of zero seconds duration.) Estimators based on all remaining measurements are calculated in the same manner.

As already pointed out, the order assigned to measurements is a design decision which could affect performance. An alternative, more refined, approach which involves a more exact calculation of the likelihoods, has been investigated in [25] for the non-manoeuvring target case. This approach is similar to the multiple-measurement approach of Chapter 4, and it involves the formulation of a hypothesis-tree for various measurement combinations. However, since the number of measurements per sensor is equal to 1, branching differentiates only between the updated state and the predicted state at each sensor update. This implies that all hypotheses are likely to be fairly similar and, as confirmed in [25], this scheme is not cost-effective and the sequential measurements model is used here instead.

Using the update maps defined in (5.2.64)-(5.3.28), the shifted Rayleigh mixture filter, modified and adapted for this type of problem is now summarised:

1. **Initialisation**, Set $t = 0$. For $n = 1, \ldots, N$ set

   Choose $\hat{\boldsymbol{x}}_0^n, \boldsymbol{P}_0^n, \hat{r}_0^n$. Let:

   $$\gamma_0^{nl} = \begin{cases} 1 & \text{if } l = \hat{r}_0^n\,, \\ 0 & \text{otherwise}, \end{cases} \qquad \mu_0^n = 1/N.$$

   Set $t = 1$.

2. **State and Covariance Predictions**

   For $n = 1 \ldots, N$, for $j = 1, \ldots, d$, calculate:

   $$\begin{aligned} \hat{\boldsymbol{x}}_{t|t-1}^{nj} &= \boldsymbol{F}(j)\hat{\boldsymbol{x}}_{t-1}^n + \boldsymbol{u}_t^s\,, \\ \boldsymbol{P}_{t|t-1}^{nj} &= \boldsymbol{F}(j)\boldsymbol{P}_{t-1}^n \boldsymbol{F}(j)^T + \boldsymbol{Q}_t^s(j)\,. \end{aligned}$$

3. **First Measurement Update**

   On receipt of ψ_t^1, evaluate:

   $$\begin{aligned} \left(\tilde{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj}, \tilde{\boldsymbol{P}}_{t|t-1,\psi_t^1}^{nj}\right) &= \mathrm{SRupd}_x\left(\hat{\boldsymbol{x}}_{t|t-1}^{nj}, \boldsymbol{P}_{t|t-1}^{nj}, \psi_t^1, j, t\right)\,, \\ f_t^{nj1}(\psi_t^1) &= \mathrm{SRupd}_\theta\left(\psi_t^1; \hat{\boldsymbol{x}}_{t|t-1}^{nj}, \boldsymbol{P}_{t|t-1}^{nj}, j, t\right)\,, \\ L_t^{nj1} &= (1 - P_c^1) f_t^{nj1}(\psi_t^1) + P_c^1/2\pi\,, \\ q_t^{nj1}(0) &= (L_t^{nj1})^{-1}(1 - P_c^1) f_t^{nj1}(\psi_t^1)\,, \\ q_t^{nj1}(1) &= (2\pi L_t^{nj1})^{-1} P_c^1\,, \\ \hat{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj} &= q_t^{nj1}(0)\tilde{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj} + q_t^{nj1}(1)\hat{\boldsymbol{x}}_{t|t-1}^{nj}\,, \\ \boldsymbol{P}_{t|t-1,\psi_t^1}^{nj} &= q_t^{nj1}(0)\left[\tilde{\boldsymbol{P}}_{t|t-1,\psi_t^1}^{nj} + \left(\bar{\boldsymbol{x}}_t^{nj1}(0)\right)\left(\bar{\boldsymbol{x}}_t^{nj1}(0)\right)^T\right] \\ &\quad + q_t^{nj1}(1)\left[\boldsymbol{P}_{t|t-1}^{nj} + \left(\bar{\boldsymbol{x}}_t^{nj1}(1)\right)\left(\bar{\boldsymbol{x}}_t^{nj1}(1)\right)^T\right]\,, \end{aligned}$$

   where:

   $$\begin{aligned} \bar{\boldsymbol{x}}_t^{nj1}(0) &= \tilde{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj} - \hat{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj}\,, \\ \bar{\boldsymbol{x}}_t^{nj1}(1) &= \hat{\boldsymbol{x}}_{t|t-1}^{nj} - \hat{\boldsymbol{x}}_{t|t-1,\psi_t^1}^{nj}\,. \end{aligned}$$

4. **Successive Measurement Updates**

On receipt of $\psi_t^{2:J}$, for $k = 2, \ldots, J$, evaluate:

$$\begin{aligned}
\left(\tilde{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}}, \tilde{\boldsymbol{P}}^{nj}_{t|t-1,\psi_t^{1:k}}\right) &= \mathrm{SRupd}_x\left(\hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k-1}}, \boldsymbol{P}^{nj}_{t|t-1,\psi_t^{1:k-1}}, \psi_t^k, j, t\right), \\
f_t^{njk}(\psi_t^k) &= \mathrm{SRupd}_\psi\left(\psi_t^k; \hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k-1}}, \boldsymbol{P}^{nj}_{t|t-1,\psi_t^{1:k-1}}, j, t\right), \\
L_t^{njk} &= (1 - P_c^k) f_t^{njk}(\psi_t^k) + P_c^k/2\pi, \\
q_t^{njk}(0) &= (L_t^{njk})^{-1}(1 - P_c^k) f_t^{njk}(\psi_t^k), \\
q_t^{njk}(1) &= (2\pi L_t^{njk})^{-1} P_c^k, \\
\hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}} &= q_t^{njk}(0)\tilde{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}} + q_t^{njk}(1)\hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k-1}}, \\
\boldsymbol{P}^{nj}_{t|t-1,\psi_t^{1:k}} &= q_t^{njk}(0)\left[\tilde{\boldsymbol{P}}^{nj}_{t|t-1,\psi_t^{1:k}} + \left(\bar{\boldsymbol{x}}_t^{njk}(0)\right)\left(\bar{\boldsymbol{x}}_t^{njk}(0)\right)^T\right] \\
&\quad + q_t^{njk}(1)\left[\boldsymbol{P}^{nj}_{t|t-1,\psi_t^{1:k-1}} + \left(\bar{\boldsymbol{x}}_t^{njk}(1)\right)\left(\bar{\boldsymbol{x}}_t^{njk}(1)\right)^T\right],
\end{aligned}$$

where:

$$\begin{aligned}
\bar{\boldsymbol{x}}_t^{njk}(0) &= \tilde{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}} - \hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}}, \\
\bar{\boldsymbol{x}}_t^{njk}(1) &= \hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k-1}} - \hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:k}}.
\end{aligned}$$

5. **Particle and Weight Update**

Let: $\hat{\boldsymbol{x}}_t^{nj} = \hat{\boldsymbol{x}}^{nj}_{t|t-1,\psi_t^{1:J}}$ and $\boldsymbol{P}_t^{nj} = \boldsymbol{P}^{nj}_{t|t-1,\psi_t^{1:J}}$, and evaluate:

$$\mu_t^{nj} = c_t^{-1}\mu_{t-1}^n\left(\sum_{l=1}^d \gamma_{t-1}^{nl}\pi_{lj}\right)\left(\prod_{k=1}^J L_t^{njk}\right),$$

where: $c_t = \sum_{n=1}^N \sum_{j=1}^d \mu_{t-1}^n \left(\sum_{l=1}^d \gamma_{t-1}^{nl}\pi_{lj}\right)\left(\prod_{k=1}^J L_t^{njk}\right)$.

6. **Updated State Estimate and Covariance:**

Evaluate:

$$\begin{aligned}
\hat{\boldsymbol{x}}_t &= \sum_{n=1}^N \sum_{j=1}^d \mu_t^{nj}\hat{\boldsymbol{x}}_t^{nj}, \\
\boldsymbol{P}_t &= \sum_{n=1}^N \sum_{j=1}^d \mu_t^n[\boldsymbol{P}_t^n + (\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{nj} - \hat{\boldsymbol{x}}_t)^T].
\end{aligned}$$

7. **Mixture Reduction**

Select N components $(\{\hat{\boldsymbol{x}}_t^n, \boldsymbol{P}_t^n, \mu_t^n\}_{n=1}^N)$ from the set $(\{\hat{\boldsymbol{x}}_t^{nj}, \boldsymbol{P}_t^{nj}, \mu_t^{nj}\}_{n,j=1}^{N,d})$ according to weights μ_t^{nj} by using one of the mixture reduction algorithm of choice

(see Section 3.2). For each mixture components, mode probabilities γ_t^n should also be computed as described in Section 3.2.3.

Set $t = t + 1$ and go to Step 2.

We also develop an *unscented mixture filter*, which is identical to this filter, but where the shifted Rayleigh update equations are replaced by the uscented Kalman filter update equations as described in Section 2.1.4.

5.4.4 Simulation results

In the simulations in this section, the model and the algorithm just described are setup with the following parameters:

h	=	5 s	sample period
T	=	500 s	time horizon
ω_0	=	0.02 rad s^{-1}	turn rate
q_1	=	0.005	target perturbation intensity
q_2	=	30	sensor perturbation intensity
q_3	=	0.005	bulk drift perturbation intensity
σ_1	=	0.283 rad($\sim 16°$)	standard deviation of sensor -target measurement noise
σ_2	=	0.028 rad($\sim 1.6°$)	standard deviation of monitor -sensor measurement noise
P_c	=	0.5	probability that the sens.-target bearing is clutter (in most sims.)

We generate Track 4, where the monitoring sensor platform is initially at the location $(500, -2500)$ and moves uniformly in a straight line to the location $(500, 500)$ (all distances are in metres). The changing coordinates x_t^{ms}, y_t^{ms} of the monitoring sensor platform appear as terms in $\boldsymbol{u}_t^m$. The prior state mean is:

$$\hat{\boldsymbol{x}}_{0|0} = [1500, -2.5, -1500,\ 5.5,\ 1000, -1500, -500, -2500, -1000,\ -1000,\ 2,\ 2]^T,$$

and the prior state covariance is:

$$\boldsymbol{P}_{0|0} = \text{diag}([10000,\ 1,\ 10000,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.02,\ 0.02]).$$

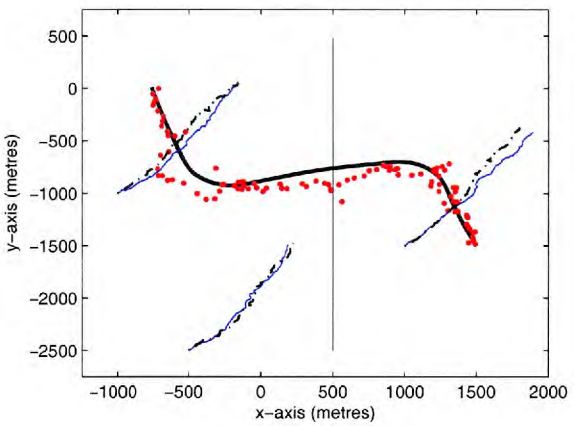


Figure 5.5: Sample realisation of Track 4 using the SRMF and the Fearnhead & Clifford mixture reduction algorithm (N=50). This figure shows the target's trajectory (thick black line) and a sample realisation of the SRMF (dots). The position of the three sonabuoys (thin lines) and their estimated position ('dash dot' lines) is also displayed. The thin straight line shows the trajectory of the monitor.

The target manoeuvres as follows: $r_{0:20} = 2$, $r_{21:35} = 3$, $r_{36:65} = 2$, $r_{66:80} = 1$ and $r_{81:100} = 2$. As the actual mode switching probabilities are unknown to the observer, we develop a SRMF using the following Markov transition matrix (simulations show that, within reason, the choice of parameters has little effect):

$$\boldsymbol{\pi} = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}.$$

Figure 5.5 shows a sample realisation of the SRMF for Track 4 with $P_c = 0.5$. The number of mixture components used is $N = 50$. We can see that the filter successfully tracks the manoeuvring target.

Figure 5.6 provides a comparison of the average RMS errors for the SRMF (tested using a variety of mixture reduction techniques), the UMF, a particle filter (featuring stratified sampling of Section 2.2.2(c) in the resampling phase) and versions of the IMM using as estimators both the shifted Rayleigh filter (the 'SRIMM') and the unscented Kalman filter (the 'UIMM'). These average errors are plotted against computational time and for varying number of mixture components.

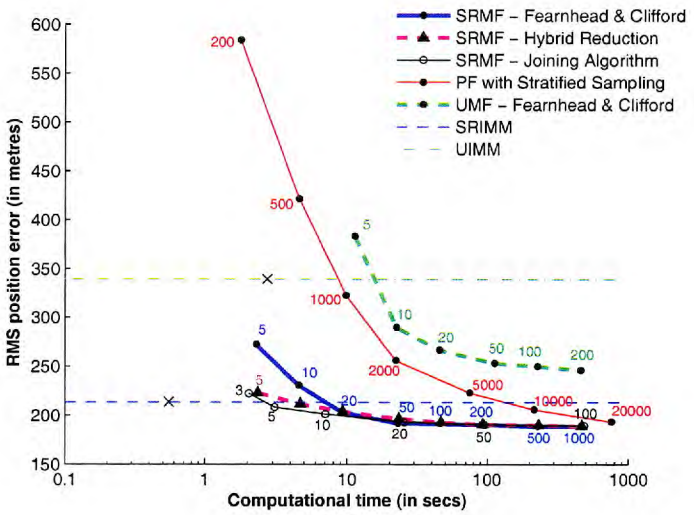


Figure 5.6: Position RMS error vs. computational cost comparisons for Track 4 and for varying number of mixture components / particles (100 Monte Carlo average).

As the graph shows, the SRMF achieves asymptotic convergence with as little as 20 particles for all the tested mixture reduction techniques, and is as such the most efficient algorithm. In this scenario, the joining algorithm provides the most cost-efficient estimates for low number of particles, and it outperforms the SRIMM with as little as 5 mixture components. The hybrid mixture reduction method provides comparable results.

The particle filter – while eventually achieving similar asymptotic levels – requires a very large number of samples to deal with all the non-linearities present in this scenario (10,000 for the same accuracy of 5 samples of the SRMF using the joining algorithm). While it is true that each individual particle update is comparably faster to calculate, the sheer number of particles that have to be considered makes the algorithm less efficient.

The UMF (using the Fearnhead and Clifford method) does not achieve the tracking accuracy of its competitors and also imposes significantly greater computational demands than the SRMF. Each particle update of the UMF takes over 5 times longer to

compute than that of the SRMF, largely because of the high-dimensionality of the state (which requires the use of a large number of σ-points).

It is however worth pointing out how the UMF is significantly more efficient than the UIMM (which is outperformed with less than 10 samples), than the SRMF is of the SRIMM (outperformed with nearly 20 samples) for the same mixture reduction method. This points out the robustness of mixture filters (compared to pseudo-Bayes methods) for tracking problems where the estimators being used are far from being exact. Although in this scenario the SRMF is the preferable option, in other nonlinear problems where the only available estimators are the UKF or the EKF, mixture filters should provide a more competitive alternative to particle filters than the IMM based algorithms.

In Figure 5.7 and 5.8 we also show the average RMS error (position and velocity respectively) for the three algorithms using component numbers requiring comparable computational time (500 for the SRMF, 100 for the UMF and 10000 for the PF). Both mixture filters make use of the Fearnhead and Clifford mixture reduction technique. Figure 5.9 also illustrates the effect of clutter on the estimates provided by the SRMF and the particle filter, matched to impose the same computational load. We can see that both algorithms can tolerate a high level of clutter (i.e. two thirds of measurements are clutter). Above that, tracking performance becomes erratic and the increase in RMS error reflects occasional 'track losses'.

In conclusion, the simulations presented in this section demonstrate the effectiveness of the SRMF in a challenging scenario. It achieves the same accuracy of a particle filter but this can be achieved at a fraction of the computational cost. A related mixture filter, based on the unscented Kalman filter was also developed, and although this does not share the accuracy of the algorithm based on shifted Rayleigh filter for bearings-only target tracking, it can be applied to other manoeuvring target tracking problems featuring nonlinear state equations. Furthermore these results have shown how, in the absence of an accurate estimator, mixture filters are more robust than algorithms based on pseudo-Bayes approximations.

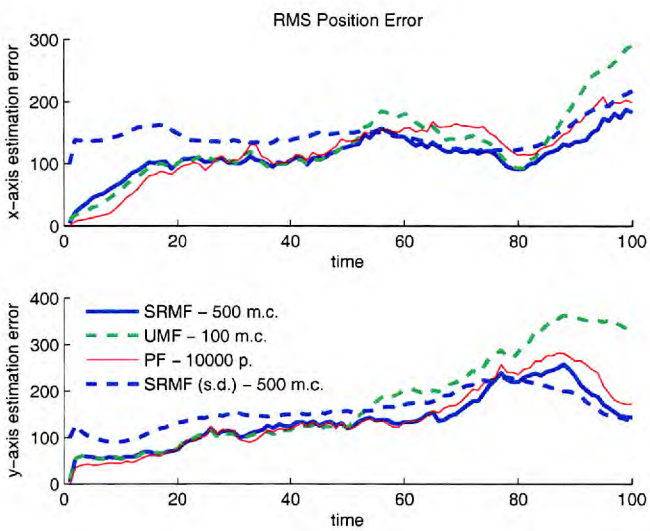


Figure 5.7: Position RMS estimation error (100 Monte Carlo average). This figure shows the average $x-$ and $y-$axis RMS position estimation error throughout the trajectory as estimated using 500 particles of the SRMF, 100 particles of the UMF and 10000 particles of the PF (the computation times of the three filters are comparable). The additional graphs are the RMS estimated standard deviations for the position coordinates generated by the SRMF.

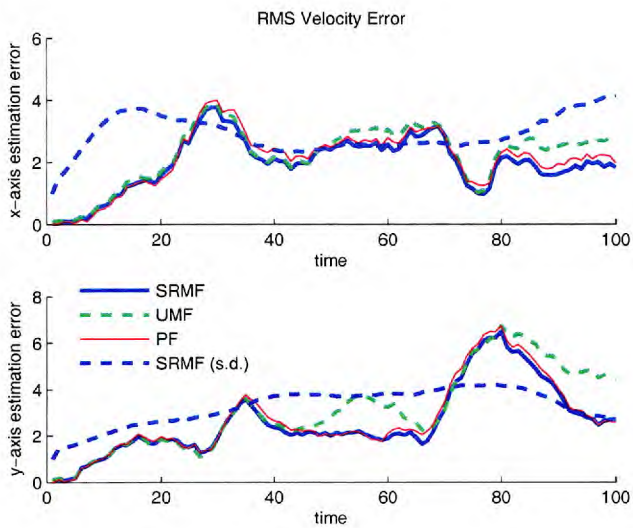


Figure 5.8: Velocity RMS estimation error (100 Monte Carlo simulations). This figure shows the average $x-$ and $y-$axis RMS velocity error throughout the trajectory as estimated using 500 particles of the SRMF, 100 particles of the UMF and 10000 particles of the PF (the computation times of the three filters are comparable). The additional graphs are the RMS estimated standard deviations for the velocity coordinates generated by the SRMF.

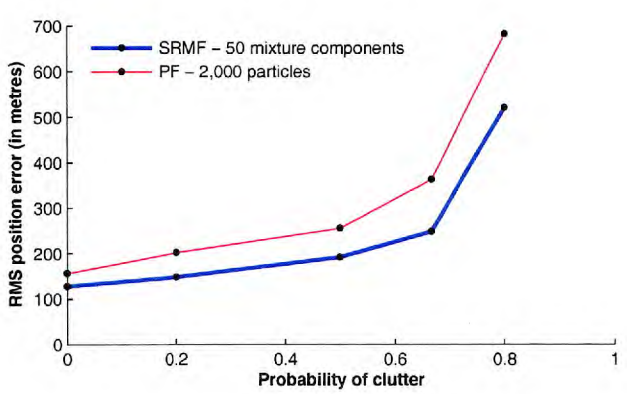


Figure 5.9: Mean RMS position error vs. probability of clutter (100 Monte Carlo simulations). This figure compares a SRMF using 50 particles with a particle filter using 2000 particles. The computing times for the two filters are approximately the same.

Chapter 6

Conclusion

This thesis has investigated the use of mixture filters for use in target tracking problems where the conditional distribution of the target is well suited to approximation by mixtures of Gaussians. These techniques provide an accurate representation of the posterior distribution of the target by using efficient mixture reduction techniques which permit the complexity of the conditional distribution to expand and contract. Furthermore, we showed how mixture filters can be directly extended to take account of extra nonlinearities, such as multiple and/or nonlinear measurements, while still retaining their simplicity and their computational efficiency. The main contributions of this thesis, in the order they were presented, are now listed.

- An in-depth empirical analysis was carried out to highlight the trade-offs between randomised and deterministic versions of weight-based mixture reduction algorithms. A new technique, known as the *α-algorithm* was introduced to allow for the possibility of customising this trade-off, and therefore tailoring the mixture reduction step for specific multiple-model tracking problems. (See Section 3.2.1.)

- The standard structure of the mixture filter was modified to allow for the possibility that each constituent density may be associated with more than one manoeuvring mode. This was necessary in order to apply distribution-based mixture reduction techniques, which are normally used for data association problems, to the multiple-model problems. (See Section 3.2.3(d).)

- We have shown how weight-based mixture reduction techniques, normally used for multiple-model problems, and distribution-based mixture reduction algorithms,

normally used for data association problems, can be successfully applied to both classes of problems. This is particularly significant in the multiple model context, as a whole new set of algorithms has been made available for use. (See Section 3.2.5.)

- A *hybrid mixture reduction* algorithm was developed to combine the computational efficiency and 'rejuvenating' nature of randomised weight-based mixture reduction methods, with the accuracy of distribution-based methods. (See Section 3.2.4.) The resulting technique was shown to outperform all other mixture reduction methods for multiple-model problems, and to provide a valid alternative to distribution-based algorithms for data association problems in which computational constraints allow for a limited use of mixture components. (See Section 3.2.5.)

- A *look-ahead mixture filter* (LAMF) was derived to allow the mixture reduction process to select components on the basis of current, rather than prior, measurement data. This allows an earlier identification of mode-transitions, which reduces the estimation error when the target manoeuvres. This modified filter was found to be particularly effective combined with distribution-based mixture reduction algorithms, and can produce high-quality estimates with very few mixture components. (See Section 3.3.)

- A *multiple measurement mixture filter* (MMMF) was developed for the problem of tracking a manoeuvring target in clutter, the combination of the multiple-model and data association problems. The MMMF was shown to easily outperform the IMMPDA (one of the most widely used algorithms for this type of problem) and to be the most competitive algorithm for a large range of computational values. Only at a very large computational cost it is outperformed by a particle filter. (See Chapter 4.)

- The use of mixture filters was then extended to solve multiple-model problems involving nonlinear/non-Gaussian dynamics. For the difficult problem of tracking a manoeuvring target using bearings-only measurements, the *shifted Rayleigh mixture filter* (SRMF) was derived and shown to be particularly competitive compared to other methods. The SRMF, which makes use of the accurate moment-

matching calculations of the shifted Rayleigh filter, was also extended and tested in a complex scenario involving multiple sensors and clutter in the form of vacuous measurements. A mixture filter based on the unscented Kalman filter was also constructed to show the advantages of mixture filters over generalised pseudo-Bayes techniques when suboptimal estimators are used. (See Section 5.)

In conclusion, mixture filters based on efficient mixture reduction techniques – a class of algorithms often overlooked in the literature – have been shown to provide a valuable alternative for target tracking problems involving multiple models. In particular, these algorithms seem to fill a 'gap' between traditional pseudo-Bayes algorithms and sequential Monte Carlo methods, in that they outperform the former at a fraction of the computational cost of the latter. These algorithms can be constructed even for sophisticated problems, and their mixture reduction step allows for a great degree of customisation to suit particular problem specifications. This, along with their scalable nature, makes mixture filters very cost effective and flexible algorithms that should be considered for the solution of target tracking problems involving switching parameters.

Suggestions for Future Work

By establishing a clear connection between weight-based and distribution-based mixture reduction methods, this thesis has shown how a new class of 'hybrid' mixture reduction methods could play a significant role in the solution of problems involving multiple-models. A simple algorithm of this type was shown to be very competitive, however more sophisticated versions might further improve performance. More specifically, it would be interesting to see if existing distribution-based algorithms could directly benefit from a degree of randomisation. For example, an algorithm where only the most obviously similar components are clustered, while the rest are randomised, could benefit from increased robustness during mode-transitions, as well as from a computational point of view.

The development of the α-algorithm has also shown some very interesting results, which could lead to more sophisticated mixture reduction techniques. An empirical analysis has in fact shown how there seems to be an ideal ratio between randomness and deter-

minateness, whose exact value is unfortunately problem dependent. A class of adaptive techniques based on the α-algorithm, which adjusts this ratio on the basis of past measurement and estimation data, could however exploit this property independently of the type of problem being considered. A variety of different approaches, including 'soft' techniques such as neural networks or fuzzy control, could be applied in the development of these adaptive algorithms.

The look-ahead mixture filter has also shown potential in reducing errors during manoeuvring by using 'future' information in its decision process. This concept might be further extended to develop 'two-' or even 'three-steps-ahead' filters. Such types of algorithm might smooth estimates during mode-transitions and might prove valuable for scenarios involving heavy manoeuvring.

Finally, mixture filters were also shown to be very flexible and applicable to a variety of problems beyond the basic multiple-model form. Mixture filters could undoubtedly be derived for problems other than those discussed in this thesis.

Appendix A

The data association problem: tracking in clutter

Data association is the problem of dealing with the uncertainty arising when multiple, simultaneous, measurements are detected which could belong to targets already being tracked, to new targets or to false measurements (clutter). A wide variety of algorithms have been proposed to deal with different aspects of this problem. Here, we briefly describe the simplest one, which involves the tracking on a single target with linear dynamics in clutter. The techniques presented in this section can readily be extended to deal with multiple targets, by adjusting the association probabilities to reflect the fact that a measurement cannot be true for more than one track. This idea is the basis of techniques such as *joint probabilistic data association* ([9],[31]). A number of algorithms can also be implemented to initiate and terminate tracks (see, for example, [56] or [10]).

A.1 The target in clutter problem

In this problem nonlinearity is introduced by the sensors, which at each iteration may detect multiple measurements. Of these, at most one may belong to the true target, the rest being 'false alarms' which may be erroneously mistaken for the true measurement and which are generally referred to as 'clutter' measurements. The probability of detecting the true target, P_D, is also introduced to take account of false detections.

The nature of clutter measurements varies with the type of sensors being used but

is generally associated with reflections of the true target or, especially in the case of passive sensors, with interferences produced by unrelated sources. Although some type of clutter measurements (such as reflections) may exhibit correlation, most target tracking techniques are based on the assumption that clutter measurements are uniformly distributed over the measurement space, and whose frequency is governed by a Poisson distribution of the form:

$$P_c(Z) = \frac{(\lambda V)^Z e^{-\lambda V}}{Z!}, \tag{A.1.1}$$

where $P_c(Z)$ is the probability of Z clutter measurements (in a given measurement cycle), λ is the spatial density of the clutter and V is the volume of the observation region. For 2-dimensional measurements, this information can be combined with the linear state space equations of (2.1.14)-(2.1.15) to produce the following system equations:

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{u}_t^s + \boldsymbol{Q}_s^{1/2}\boldsymbol{\nu}_t^s, \tag{A.1.2}$$

$$\boldsymbol{\mathcal{Y}}_t = \cup \begin{cases} \boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{u}_t^m + \boldsymbol{Q}_m^{1/2}\boldsymbol{\nu}_t^m & \text{with probability} P_d, \\ \boldsymbol{z}_t(i) = \mathcal{U}[\boldsymbol{R}_{\min}, \boldsymbol{R}_{\max}] & \text{for } i \in \{1, \ldots, Z_t\}. \end{cases} \tag{A.1.3}$$

Here $\boldsymbol{\mathcal{Y}}_t$ is the set of measurements at time t, $\boldsymbol{R}_{\min}$ and $\boldsymbol{R}_{\max}$ are vectors containing the lower and upper limits of the surveillance region (assumed to be rectangular), and Z_t is the random number of Poisson measurements with probability $P_c(Z_t)$. Assuming the true target is detected, the total number of measurements at time t is then given by $Y_t = Z_t + 1$.

A.2 Measurement gating

In most practical tracking problems, the surveillance region is generally very large compared to the variance of the true measurement process, with the implication that a large number of clutter measurement fall in regions where there is little uncertainty concerning their (false) origin. The computational efficiency of any algorithm can then generally be improved by not including these measurements in the data association process.

In order to do so, a preliminary coarse selection technique, generally referred to as 'measurement gating', or simply 'gating', is usually implemented. This involves comparing the location of all the measurements with that of a 'gate' about the expected

location of the true measurement. Any observation falling within this region is considered by the data association algorithms for track update, those falling outside the gate are discarded. Generally the gate's size and shape is chosen so as to eliminate as many unlikely measurements as possible, while still trying to ensure that true measurement is not erroneously discarded.

A variety of different gating techniques have been proposed, but these generally fall into two broad categories: rectangular gates [63] and ellipsoidal gates [15]. Considering each measurement $\boldsymbol{y}_t(i) \in \boldsymbol{\mathcal{Y}}_t$, recall the measurement innovation (2.1.44) and the predicted measurement covariance (2.1.43) given by:

$$\begin{aligned}\tilde{\boldsymbol{y}}_t(i) &= \boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}\,,\\ \boldsymbol{S}_t(i) &= \boldsymbol{H}\boldsymbol{P}_{t|t-1}\boldsymbol{H}^T + \boldsymbol{Q}_m\,.\end{aligned}$$

The simplest form of gating involves defining a rectangular region about the measurement $\boldsymbol{y}_t(i)$ such that all of the elements of the innovation vector ($\tilde{y}_t^l(i)$ for $l \in \{1,\ldots,L\}$, where L is the dimension of the measurement) satisfy the thresholding condition:

$$\left|\tilde{y}_t^l(i)\right| \leq \sigma_l K_l\,. \tag{A.2.1}$$

Here, $\sigma_l = \sqrt{S_t^{ll}(i)}$ is the standard deviation of the innovation and K_l is the gate size, whose choice is discussed in detail in [63]-[64].

A more refined technique, which is used in all the algorithms considered in this thesis, makes use of ellipsoidal gates that ensure that the norm of the innovation vector satisfies the condition:

$$\tilde{\boldsymbol{y}}_t(i)\boldsymbol{S}_t^{-1}\tilde{\boldsymbol{y}}_t(i) \leq G\,. \tag{A.2.2}$$

As discussed in [16], since the false measurements are uniformly distributed, this test is essentially a likelihood ratio test. Here the threshold G is a design parameter which can be obtained using χ^2 tables (with L degrees of freedom) to ensure that P_G, the acceptable probability of the true measurement falling outside the gate, is satisfied. In order to avoid a significant number of true measurement losses, P_G is generally chosen to be very small (normally in the range $0.01 > P_G > 0.001$). This gating approach, however, allows for the possibility of misdetections even when $P_D = 1$, and therefore the

detection probability should be updated to $P_{DG} = P_D(1 - P_G)$.

Ellipsoidal measurement gating of this form is generally more computationally intensive than coarser methods (mainly due to the matrix inversion operations involved) and some methods have been proposed where two-stage gating is performed. As suggested in [16], for example, a first stage employing large rectangular gates could be used to eliminate most false measurements followed by a second stage using ellipsoidal gates to refine the selection. Alternatively, in [27], a 'binning' approach in which the measurement space is split into grids, is used as a preliminary selection method. It should be noted that these multi-stage gating techniques are primarily useful in scenarios with large observation regions for which the computational savings may be significant.

A.3 Data association techniques

A.3.1 The optimal Bayesian solution

The analysis involved in applying the Bayesian approach to the multiple-measurement data association problem is remarkably similar to that of the multiple-model tracking problem described in Section 2.3.2. In both cases, the conditional density involves a Gaussian mixture with an exponentially growing number of terms.

In data association a tree of measurement-histories (which in this report will generally be referred to as 'association hypotheses', or simply 'hypotheses') can be constructed (see Figure A.1) by calculating a Kalman filter update for each feasible measurement at each iteration. To take into account the possibility that the true measurement may not be detected or may be discarded by the preliminary gating process (if $P_{DG} < 1$), an extra 'null-hypothesis' representing this eventuality (and whose mean and covariance are the unupdated predictions) is added at each iteration. The optimal solution at time t is then a weighted sum of the $N_t = \prod_1^t (Y_t + 1)$ Gaussian estimates corresponding to all association hypotheses, where Y_t is the total number measurements satisfying gating conditions at time t.

Define $\mathcal{D}_{1:t}$ to be the set of measurement association histories up until time t. The exact conditional density at time t can be expressed as the weighted sum of the $N_t =$

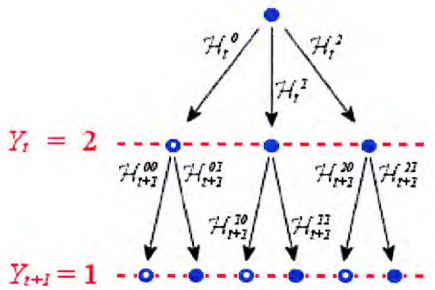


Figure A.1: Multiple-measurement problem: exponential growth in association hypotheses.

$\prod_1^t (Y_t + 1)$ densities corresponding to the possible association hypotheses:

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{1:t}} p(\boldsymbol{x}_t|q_{1:t},\boldsymbol{\mathcal{Y}}_{1:t})\mu_t^{q_{1:t}}\,, \tag{A.3.1}$$

where

$$\mathcal{D}_{1:t} = \{q_{1:t} : q_1,\ldots,q_t \in \{0,\ldots,Y_t\}\}\,, \tag{A.3.2}$$

$$\mu_t^{q_{1:t}} = p\left(q_{1:t}|\boldsymbol{\mathcal{Y}}_{1:t}\right), \quad \text{and} \quad \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \mu_t^{q_{1:t}} = 1\,. \tag{A.3.3}$$

This Gaussian mixture can also be expressed in terms of the weights $\mu_t^{q_{1:t}}$, the means, $\hat{\boldsymbol{x}}_t^{q_{1:t}}$, and the covariances, $\boldsymbol{P}_t^{q_{1:t}}$, of the constituent Gaussian densities:

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{1:t}} \mathcal{N}(\boldsymbol{x}_t;\hat{\boldsymbol{x}}_t^{q_{1:t}},\boldsymbol{P}_t^{q_{1:t}})\mu_t^{q_{1:t}}\,. \tag{A.3.4}$$

For $q_t = 0$, the corresponding densities in this sum are simple predicted densities. For $q_t = \{1,\ldots,Y_t\}$, they can be calculated recursively using the Kalman filter update equations, which involve knowledge of the weights $\mu_{t-1}^{q_{0:t-1}}$, the means, $\hat{\boldsymbol{x}}_{t-1}^{q_{0:t-1}}$, and the covariances, $\boldsymbol{P}_{t-1}^{q_{0:t-1}}$, of the constituent Gaussian densities of $p(\boldsymbol{x}_{t-1}|\boldsymbol{\mathcal{Y}}_{1:t-1})$ in:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{\mathcal{Y}}_{1:t-1}) = \sum_{q_{1:t-1}\in\mathcal{D}_{0:t-1}} \mathcal{N}(\boldsymbol{x}_{t-1};\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1}},\boldsymbol{P}_{t-1}^{q_{1:t-1}})\mu_{t-1}^{q_{1:t-1}}\,. \tag{A.3.5}$$

Then the mean and covariance of the constituent normal densities in the posterior con-

ditional distribution at t can be expressed as:

for $q_t = 0$:

$$\hat{\boldsymbol{x}}_t^{q_{1:t}} = \boldsymbol{F}\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1}} + \boldsymbol{u}_t^s\,, \tag{A.3.6}$$

$$\boldsymbol{P}_{t-1}^{q_{1:t}} = \boldsymbol{F}\boldsymbol{P}_{t-1}^{q_{1:t-1}}\boldsymbol{F}^{\mathrm{T}} + \boldsymbol{Q}_s\,, \tag{A.3.7}$$

for $q_t = \{1, \ldots, Y_t\}$:

$$(\hat{\boldsymbol{x}}_t^{q_{1:t}}, \boldsymbol{P}_t^{q_{1:t}}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^{q_{1:t-1}}, \boldsymbol{P}_{t-1}^{q_{1:t-1}}, \boldsymbol{y}_t(q_t), t)\,. \tag{A.3.8}$$

The calculation of the posterior weights of the components is more complicated than in multiple model tracking problems, as the probabilities of misdetection need to be taken into account. To simplify notation, let $\mathcal{H}_{t-1}^n$ be the index of the prior hypotheses $q_{1:t-1} = n \in \{1, \ldots, N_{t-1}\}$, and let $\mathcal{H}_t^{nk}$ be the index of the posterior hypotheses for $q_t = k \in \{0, \ldots, Y_t\}$. Then, by Bayes rule:

$$\mu_t^{q_{1:t}} = p\left(q_{1:t}|\boldsymbol{\mathcal{Y}}_{1:t}\right) = p\left(\mathcal{H}_t^{nk}|\boldsymbol{\mathcal{Y}}_{1:t}\right) = \mu_t^{nk} \tag{A.3.9}$$

$$= \frac{p(\mathcal{H}_{t-1}^n)p(q_t = k|\mathcal{H}_{t-1}^n)p(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{nk})}{p(\boldsymbol{\mathcal{Y}}_t)}\,. \tag{A.3.10}$$

Here $p(\boldsymbol{\mathcal{Y}}_t)$ is normalising constant and $p(\mathcal{H}_{t-1}^n) = \mu_{t-1}^{q_{1:t-1}}$ is the prior weight of hypothesis $\mathcal{H}_{t-1}^n$. Consider now $p(q_t = k|\mathcal{H}_{t-1}^n)$ and define $P_{FA}(m)$ to be the probability of exactly m of the valid measurements being false. Then if the clutter variable has a Poisson distribution:

$$P_{FA}(m) = \frac{(\lambda V)^m e^{-\lambda V}}{m!}\,. \tag{A.3.11}$$

Assuming that at most one of the measurements can be true (or they could all be false), then $p(q_t = k|\mathcal{H}_{t-1}^n)$ can be computed as:

$$p(q_t = 0|\mathcal{H}_{t-1}^n) = (1 - P_{DG})P_{FA}(Y_t)\,, \tag{A.3.12}$$

$$p(q_t = (k \neq 0)|\mathcal{H}_{t-1}^n) = P_{DG}\left(\frac{P_{FA}(Y_t - 1)}{Y_t}\right)\,. \tag{A.3.13}$$

As the measurements in $\boldsymbol{\mathcal{Y}}_t$ are independent of one another, the final term in (A.3.9), $p(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{nk})$, is given by:

$$p\left(\boldsymbol{\mathcal{Y}}_t|\mathcal{H}_t^{nk}\right) = \prod_{k=1}^{Y_t} p(\boldsymbol{y}_t(k)|\mathcal{H}_t^{nk})\,. \tag{A.3.14}$$

For hypothesis $k = 0$, all measurements are false alarms. As these are uniformly distributed over the surveillance region:

$$p(\mathcal{Y}_t|\mathcal{H}_t^{n0}) = \frac{1}{V^{Y_t}}. \tag{A.3.15}$$

For hypotheses $k > 0$, only one measurement is valid (and Normally distributed), while the others are false. Therefore:

$$p(\mathcal{Y}_t|\mathcal{H}_t^{nk}) = \frac{\mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n)}{V^{Y_t-1}}. \tag{A.3.16}$$

Substituting (A.3.12) and (A.3.15) into (A.3.9), μ_t^{n0} becomes:

$$\mu_t^{n0} = \mu_{t-1}^n \frac{(1-P_{DG})(\lambda V)^{Y_t} e^{-\lambda V}}{\bar{c} Y_t! V^{Y_t}} \tag{A.3.17}$$

$$= \mu_{t-1}^n \frac{(1-P_{DG})\lambda^{Y_t} e^{-\lambda V}}{\bar{c} Y_t!}. \tag{A.3.18}$$

Similarly by substituting (A.3.13) and (A.3.16) into (A.3.9), μ_t^{nk} becomes

$$\mu_t^{nk} = \mu_t^n \frac{P_{DG}(\lambda V)^{Y_t-1} e^{-\lambda V} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n)}{\bar{c}(Y_t-1)! Y_t V^{Y_t-1}} \tag{A.3.19}$$

$$= \mu_t^n \frac{P_{DG}\lambda^{Y_t-1} e^{-\lambda V} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n)}{\bar{c} Y_t!}, \tag{A.3.20}$$

where $\bar{c}$ is the normaliser defined as:

$$\bar{c} = \frac{\lambda^{Y_t-1} e^{-\lambda V}}{Y_t!} \left(\sum_{nk} P_{DG} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n) + (1-P_{DG})\lambda \right). \tag{A.3.21}$$

Let:

$$c = \frac{\bar{c} P_{DG} Y_t!}{\lambda^{Y_t-1} e^{-\lambda V}} \tag{A.3.22}$$

$$= \sum_{nk} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n) + \frac{(1-P_{DG})\lambda}{P_{DG}}, \tag{A.3.23}$$

then $\mu_t^{q_{1:t}}$ can be written as:

$$\mu_t^{q_{1:t}} = \mu_t^{nk} = \frac{1}{c} \mu_t^n \begin{cases} \frac{(1-P_{DG})\lambda}{P_{DG}} & \text{for } k = 0, \\ \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n) & \text{for } k = 1 \ldots Y_t. \end{cases} \tag{A.3.24}$$

This optimal solution is not computationally viable, due to the exponentially growing number of terms in the Gaussian mixture. The following sections investigate target tracking algorithms which propose alternative techniques to approximate this formula.

A.3.2 Probabilistic data association filter (PDAF)

One of the most common tracking data association tracking algorithms, *probabilistic data Association filter* (PDAF), first introduced by Bar-Shalom and Tse [13], is the data association equivalent of the GPB1 algorithm (see Section 2.3.3). In fact, the key feature of the PDAF involves the approximation the optimal posterior conditional distribution of (A.3.1),

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{q_{1:t}}, \boldsymbol{P}_t^{q_{1:t}})\mu_t^{q_{1:t}}\,,$$

using a single Gaussian term (which is effectively the weighted sum of all the association hypotheses at each iteration):

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t, \boldsymbol{P}_t)\,.$$

The common implementation of PDAF does this rather efficiently by combining the innovations $\tilde{\boldsymbol{y}}_t^k$ of all the valid measurements and then updating the term using a single Kalman filter update. The key steps of the PDAF are summarised in Appendix B.2.1.

A.3.3 Mixture filters based on mixture reduction

These techniques, first introduced by Salmond [62], are an elaboration of the PDAF method in which multiple terms are retained at each iteration. The exact conditional distribution:

$$p(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_{0:t}} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{q_{1:t}}, \boldsymbol{P}_t^{q_{1:t}})\mu_t^{q_{1:t}}\,,$$

is approximated as:

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{\mathcal{Y}}_{1:t}) = \sum_{q_{1:t}\in\mathcal{D}_t} \mathcal{N}(\boldsymbol{x}_t; \hat{\boldsymbol{x}}_t^{q_{1:t}}, \boldsymbol{P}_t^{q_{1:t}})\mu_t^{q_{1:t}}\,,$$

where $\mathcal{D}_t$ is a subset $\mathcal{D}_{0:t}$ of cardinality N. Mixture filters recursively approximate the exact mixture, using a fixed number of terms. As a new measurement set becomes available, estimates for association hypotheses $\mathcal{H}_t^{nk}$ for $n \in \{1,\ldots,N\}$ and $k \in \{0,\ldots,Y_t\}$ are calculated. In order to keep the number of terms controlled to N, the posterior conditional distribution is reduced to a N-fold one via mixture reduction techniques. These methods, an important part of this thesis, are discussed in further detail in Chapter 3. The mixture filter for data association is very similar to the one for multiple-model, and is detailed in Appendix B.2.2.

Appendix B

Algorithms implementation

In this Appendix, we outline the recursion of some of the target tracking algorithms presented in this thesis.

B.1 Multiple-model target tracking algorithms

B.1.1 The first order GPB algorithm

We now outline the *first order generalised pseudo-Bayes estimator* (GPB1) in detail. For a description of the algorithm, see Section 2.3.3(a).

STEP 1: MODE-MATCHED KALMAN FILTERING

A single prior density with mean $\hat{\boldsymbol{x}}_t$, covariance $\boldsymbol{P}_t$ and mode probabilities μ_{t-1}^i, is expanded into a d-fold mixture using the mode-matched Kalman maps (2.3.12) and (2.3.21). For $j \in \mathcal{S}$:

$$(\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j) = KF_x(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, \boldsymbol{y}_t, j, t)\,, \tag{B.1.1}$$

$$\tilde{\mu}_t^j = \left(\sum_{i=1}^{d} \pi_{ij}\mu_{t-1}^i\right) KF_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1}, j, t)\,, \tag{B.1.2}$$

$$\mu_t^j = \frac{1}{c}\tilde{\mu}_t^j, \text{ where } c = \sum_j \tilde{\mu}_t^j\,. \tag{B.1.3}$$

STEP 2: STATE ESTIMATE & COVARIANCE CALCULATIONS

The state estimate and covariance, which are then propagated along with μ_t^j to the next

iteration, are now calculated as:

$$\hat{\boldsymbol{x}}_t = \sum_{j=1}^{d} \mu_t^j \hat{\boldsymbol{x}}_t^j , \tag{B.1.4}$$

$$\boldsymbol{P}_t = \sum_{j=1}^{d} \mu_t^j [\boldsymbol{P}_t^j + (\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}] . \tag{B.1.5}$$

B.1.2 The second order GPB algorithm

We now outline the *second order generalised pseudo-Bayes estimator* (GPB2) in detail. For a description of the algorithm, see Section 2.3.3(b).

STEP 1: MODE-MATCHED KALMAN FILTERING

Given d prior densities with mean, covariance, mode and weights defined by $\hat{\boldsymbol{x}}_{t-1}^i$, $\boldsymbol{P}_{t-1}^i$, $\hat{r}_{t-1}^i = i$ and μ_{t-1}^i respectively (for $i \in \mathcal{S}$), the GPB2 estimator firstly uses mode-matched Kalman filter maps (described in (2.3.12),(2.3.21)) to compute all possible mode transitions. This generates the following d^2 densities defined (for $i, j \in \mathcal{S}$) by:

$$(\hat{\boldsymbol{x}}_t^{ij}, \boldsymbol{P}_t^{ij}) = KF_x(\hat{\boldsymbol{x}}_{t-1}^i, \boldsymbol{P}_{t-1}^i, \boldsymbol{y}_t, j, t) , \tag{B.1.6}$$

$$\tilde{\mu}_t^{ij} = \mu_{t-1}^i \pi_{ij} KF_y(\boldsymbol{y}_t; \hat{\boldsymbol{x}}_{t-1}^i, \boldsymbol{P}_{t-1}^i, j, t) , \tag{B.1.7}$$

$$\mu_t^{ij} = \frac{1}{c_j} \tilde{\mu}_t^{ij}, \text{ where } c_j = \sum_i \tilde{\mu}_t^{ij} . \tag{B.1.8}$$

STEP 2: MODE-BASED MERGING

We next merge those estimates that assume a common mode for time t. The resultant densities are then propagated to the next iteration.

$$\hat{\boldsymbol{x}}_t^j = \sum_{i=1}^{d} \mu_t^{ij} \hat{\boldsymbol{x}}_t^{ij} , \tag{B.1.9}$$

$$\boldsymbol{P}_t^j = \sum_{i=1}^{d} \mu_t^{ij} [\boldsymbol{P}_t^{ij} + (\hat{\boldsymbol{x}}_t^{ij} - \hat{\boldsymbol{x}}_t^j)(\hat{\boldsymbol{x}}_t^{ij} - \hat{\boldsymbol{x}}_t^j)^{\mathrm{T}}] , \tag{B.1.10}$$

$$\mu_t^j = \frac{1}{c} \sum_{i=1}^{d} \tilde{\mu}_t^{ij} = \frac{c_j}{c} , \qquad \text{where, } c = \sum_{j=1}^{d} c_j . \tag{B.1.11}$$

STEP 3: STATE ESTIMATE & COVARIANCE CALCULATIONS

The current state estimate and covariance are computed by further merging the elements

of the posterior conditional distribution:

$$\hat{\boldsymbol{x}}_t = \sum_{j=1}^{d} \mu_t^j \hat{\boldsymbol{x}}_t^j , \tag{B.1.12}$$

$$\boldsymbol{P}_t = \sum_{j=1}^{d} \mu_t^j [\boldsymbol{P}_t^j + (\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}] . \tag{B.1.13}$$

B.1.3 The IMM algorithm

We now outline the *interacting multiple model algorithm* (IMM) in detail. For a description of the algorithm, see Section 2.3.3(c).

STEP 1: ESTIMATE MIXING

Firstly, given d prior densities with parameters $\{\hat{\boldsymbol{x}}_{t-1}^i, \boldsymbol{P}_{t-1}^i, \hat{r}_{t-1}^i = i, \mu_{t-1}^i\}$ (for $i \in \mathcal{S}$), the IMM determines the mixing probabilities. These describe the probabilities of each possible mode-transition based on prior information only. For $i, j \in \mathcal{S}$:

$$\mu_t^{ij} = \frac{1}{\bar{c}_j} \mu_{t-1}^i \pi_{ij} , \tag{B.1.14}$$

where,

$$\bar{c}_j = \sum_{i=1}^{d} \mu_{t-1}^i \pi_{ij} . \tag{B.1.15}$$

Secondly, the algorithm mixes the state estimates and covariances related to the modes as follows:

$$\tilde{\boldsymbol{x}}_t^j = \sum_{i=1}^{d} \hat{\boldsymbol{x}}_{t-1}^i \mu_t^{ij} , \tag{B.1.16}$$

$$\tilde{\boldsymbol{P}}_t^j = \sum_{i=1}^{d} \mu_t^{ij} [\boldsymbol{P}_{t-1}^i + (\hat{\boldsymbol{x}}_{t-1}^i - \tilde{\boldsymbol{x}}_t^j)(\hat{\boldsymbol{x}}_{t-1}^i - \tilde{\boldsymbol{x}}_t^j)^{\mathrm{T}}] . \tag{B.1.17}$$

STEP 2: MODE-MATCHED KALMAN FILTERING

The IMM then directly computes a posterior d-fold mixture by using the mixed Gaussians as the inputs to the Kalman filter. Using the Kalman filter maps of (2.3.12) and (2.3.21), the d densities are computed as:

$$(\hat{\boldsymbol{x}}_t^j, \boldsymbol{P}_t^j) = KF_x(\tilde{\boldsymbol{x}}_{t-1}^j, \tilde{\boldsymbol{P}}_{t-1}^j, \boldsymbol{y}_t, j, t) , \tag{B.1.18}$$

$$\tilde{\mu}_t^j = \bar{c}_j KF_y(\boldsymbol{y}_t; \tilde{\boldsymbol{x}}_{t-1}^j, \tilde{\boldsymbol{P}}_{t-1}^j, j, t) , \tag{B.1.19}$$

$$\mu_t^j = \frac{1}{c} \tilde{\mu}_t^j, \text{ where } c = \sum_j \tilde{\mu}_t^j . \tag{B.1.20}$$

STEP 3: STATE ESTIMATE & COVARIANCE CALCULATIONS

The last step of the IMM is identical to that of the GPB2. The algorithm computes the state estimate and covariance to be used exclusively as the output of the algorithm. These are given by:

$$\hat{\boldsymbol{x}}_t = \sum_{j=1}^{d} \mu_t^j \hat{\boldsymbol{x}}_t^j , \tag{B.1.21}$$

$$\boldsymbol{P}_t = \sum_{j=1}^{d} \mu_t^j [\boldsymbol{P}_t^j + (\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}] . \tag{B.1.22}$$

B.1.4 The multiple model bootstrap filter (MMBF)

We now outline the *multiple model bootstrap filter* (MMBF) in detail. For a description of the algorithm, see Section 2.3.4(a).

STEP 1: SEQUENTIAL IMPORTANCE SAMPLING

Assuming the availability of N i.i.d. equally weighted samples from the prior distribution, each associated with mode $\hat{r}_{t-1}^n$ and with point mass located on $\hat{\boldsymbol{x}}_{t-1}^n$ (for $n \in \{1, \ldots, N\}$), the MMBF first simulates N state noise values, $\tilde{\boldsymbol{\nu}}_t^s(n)$, and N mode values, $\hat{r}_{t-1}^n$, at time t. This is done by sampling $\tilde{\boldsymbol{\nu}}_t^s(n) \sim \mathcal{N}(0, \boldsymbol{Q}_m)$, and $\tilde{r}_t^n \sim p(r_t | r_{t-1} = \hat{r}_{t-1}^n) = \{\pi_{\hat{r}_{t-1}^n j}\}_{j=1}^d$. Here, $\tilde{r}_t^n$ is selected as the mode for which the uniformly distributed random sample $u_n \sim \mathcal{U}(0,1)$ satisfies:

$$\sum_{j=1}^{\tilde{r}_t^n - 1} \pi_{\hat{r}_{t-1}^n j} \leq u_n \leq \sum_{j=1}^{\tilde{r}_t^n} \pi_{\hat{r}_{t-1}^n j} . \tag{B.1.23}$$

Using these values, $\tilde{\boldsymbol{x}}_t^n$ is then evaluated (using (2.3.1)) as:

$$\tilde{\boldsymbol{x}}_t^n = \boldsymbol{F}(\tilde{r}_t^n)\hat{\boldsymbol{x}}_{t-1}^n + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(\tilde{r}_t^n))^{1/2}\tilde{\boldsymbol{\nu}}_t^s(n) . \tag{B.1.24}$$

For each sample, the relevant importance weight is given by:

$$\mu_t^n = \frac{1}{c} p(\boldsymbol{y}_t | \tilde{\boldsymbol{x}}_t^n) = \mathcal{N}(\boldsymbol{y}_t; \boldsymbol{H}\tilde{\boldsymbol{x}}_t^n + \boldsymbol{u}_t^m, \boldsymbol{Q}_m) , \tag{B.1.25}$$

where $c = \sum_{n=1}^{N} \mathcal{N}(\boldsymbol{y}_t; \boldsymbol{H}_t\tilde{\boldsymbol{x}}_t^n + \boldsymbol{u}_t^m, \boldsymbol{Q}_m)$ is a normalising constant. As pointed out in [55], the mode transition probabilities are omitted in this equation, since they have already been accounted for by the mode-sampling process.

STEP 2: STATE ESTIMATE CALCULATION

The output state estimate, $\hat{\boldsymbol{x}}_t$, is computed as a weighted sum of all particles:

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \mu_t^n \tilde{\boldsymbol{x}}_t^n . \tag{B.1.26}$$

STEP 3: RESAMPLING

For $n \in \{1, \ldots, N\}$, resample with replacement N particles $\{\hat{\boldsymbol{x}}_t^n, \hat{r}_t^n\}$ from the set of samples $\{\tilde{\boldsymbol{x}}_t^n, \tilde{r}_t^n\}$ according to importance weights μ_t^n, using one of the techniques described in Section 2.2.2.

B.1.5 The Rao-Blackwellised particle filter (RBPF)

We now outline the *Rao-Blackwellised particle filter* (RBPF) in detail. For a description of the algorithm, see Section 2.3.4(b).

STEP 1: EVALUATION OF THE OPTIMAL SAMPLING DISTRIBUTION

Assuming N i.i.d equally weighted samples $\{\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, \hat{r}_{t-1}^n\}$ from the prior distribution are available, for particle $n \in \{1, \ldots, N\}$ and mode $j \in \{1, \ldots, d\}$ calculate (using the Kalman filter prediction map (2.3.22)):

$$(\hat{\boldsymbol{x}}_{t|t-1}^{nj}, \boldsymbol{P}_{t|t-1}^{nj}, \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) = \mathrm{KFpred}(\hat{\boldsymbol{x}}_{t-1}^n, \boldsymbol{P}_{t-1}^n, j, t) . \tag{B.1.27}$$

The sampling probabilities are then calculated as:

$$\tilde{\mu}_t^{nj} = \frac{\mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) \pi_{\hat{r}_{t-1}^n j}}{\sum_{j=1}^{d} \mathcal{N}(\boldsymbol{y}_t; \hat{\boldsymbol{y}}_{t|t-1}^{nj}, \boldsymbol{S}_t^{nj}) \pi_{\hat{r}_{t-1}^n j}} . \tag{B.1.28}$$

STEP 2: MODE SAMPLING

For each $n \in \{1, \ldots, N\}$, a mode is sampled from $\{\tilde{\mu}_t^{nj}\}_{j=1}^{d}$. The selected mode M_n is the one for which the uniformly distributed random sample $u_n \sim \mathcal{U}(0, 1)$ satisfies:

$$\sum_{j=1}^{M_n - 1} \tilde{\mu}_t^{nj} \leq u_n \leq \sum_{j=1}^{M_n} \tilde{\mu}_t^{nj} . \tag{B.1.29}$$

STEP 3: KALMAN FILTER UPDATE

For each sampled mode, the state and covariance of each Gaussian particle are updated by means of Kalman filter map (2.3.23):

$$(\tilde{\boldsymbol{x}}_t^n, \tilde{\boldsymbol{P}}_t^n) = \mathrm{KFupd}_x(\hat{\boldsymbol{x}}_{t|t-1}^{nM_n}, \boldsymbol{P}_{t|t-1}^{nM_n}, M_n, \boldsymbol{y}_t, t) . \tag{B.1.30}$$

The weight of each updated sample is given by:

$$\mu_t^n = \sum_{j=1}^{d} \tilde{\mu}_t^{nj} . \tag{B.1.31}$$

STEP 4: STATE ESTIMATE & COVARIANCE CALCULATIONS

The output state estimate, $\hat{\boldsymbol{x}}_t$, and covariance, $\boldsymbol{P}_t$, are now computed by merging all particles:

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \mu_t^n \tilde{\boldsymbol{x}}_t^n , \tag{B.1.32}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \mu_t^n [\tilde{\boldsymbol{P}}_t^n + (\tilde{\boldsymbol{x}}_t^n - \hat{\boldsymbol{x}}_t)(\tilde{\boldsymbol{x}}_t^n - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}] . \tag{B.1.33}$$

STEP 5: RESAMPLING

For $n \in \{1, \ldots, N\}$, resample with replacement N particles $\{\hat{\boldsymbol{x}}_t^n, \boldsymbol{P}_t^n, \hat{r}_t^n\}$ from the set of samples $\{\tilde{\boldsymbol{x}}_t^n, \tilde{\boldsymbol{P}}_t^n, M_n\}$ according to importance weights μ_t^n, using one of the techniques described in Section 2.2.2.

B.2 Multiple measurement target tracking algorithms

B.2.1 The probabilistic data association filter (PDAF)

We now outline the *probabilistic data association filter* (PDAF) in detail. For a description of the algorithm, see Section A.3.2.

STEP 1: STATE AND MEASUREMENT PREDICTIONS

Given a Gaussian prior, $\mathcal{N}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1})$, the first step of the PDAF is the calculation of the predicted state, $\hat{\boldsymbol{x}}_{t|t-1}$, predicted state covariance, $\boldsymbol{P}_{t|t-1}$, predicted measurement, $\hat{\boldsymbol{y}}_{t|t-1}$ and predicted measurement covariance, $\boldsymbol{S}_t$ as described in (2.1.40)-(2.1.43).

STEP 2: MEASUREMENT GATING

Gating is then performed to discard those measurements that are highly unlikely to be true. Let $\boldsymbol{\mathcal{Z}}_t \subset \boldsymbol{\mathcal{Y}}_t$ be the subset of measurements $\boldsymbol{y}_t(i) \in \boldsymbol{\mathcal{Y}}_t$ that satisfy:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1})^T (\boldsymbol{S}_t)^{-1} (\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}) \leq G . \tag{B.2.1}$$

Let $Z_t = \mathrm{card}(\boldsymbol{\mathcal{Z}}_t)$.

STEP 3: HYPOTHESES PROBABILITY CALCULATIONS

Next, the PDAF calculates the probability of each association hypothesis, $\mathcal{H}_t^k$, for $k \in \{0, \ldots, Z_t\}$. This is done using (A.3.24) for the special case in which the prior consists of a single Gaussian:

$$\mu_t^k = \frac{1}{c} \begin{cases} \frac{(1-P_{DG})\lambda}{P_{DG}} & \text{for } k = 0\,, \\ \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}, \boldsymbol{S}_t) & \text{for } k = 1, \ldots, Z_t\,. \end{cases} \tag{B.2.2}$$

STEP 4: RESIDUALS COMBINATION

Next, the PDAF calculates a weighted sum of the innovation (or residual) of each measurement. This is done as follows:

$$\bar{\boldsymbol{y}}_t = \sum_{k=0}^{Z_t} \mu_t^k \tilde{\boldsymbol{y}}_t^k\,, \tag{B.2.3}$$

where:

$$\tilde{\boldsymbol{y}}_t^k = \boldsymbol{y}_t(k) - \hat{\boldsymbol{y}}_{t|t-1}\,. \tag{B.2.4}$$

Note that $\tilde{\boldsymbol{y}}_t^k$ were already calculated as part of the evaluation of $\mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}, \boldsymbol{S}_t)$ in Step 2.

STEP 5: STATE AND COVARIANCE UPDATE

Finally, the predictions calculated in Step 1 are updated using the combined residual. If $\boldsymbol{K}_t$ is calculated as in (2.1.45), then:

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t \bar{\boldsymbol{y}}_t\,, \tag{B.2.5}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - (1 - \mu_t^0)\boldsymbol{K}_t \boldsymbol{H} \boldsymbol{P}_{t|t-1} + \boldsymbol{K}_t \tilde{\boldsymbol{P}}_t \boldsymbol{K}_t^T\,, \tag{B.2.6}$$

where:

$$\tilde{\boldsymbol{P}}_t = \sum_{k=1}^{Z_t} \left[\mu_t^k (\tilde{\boldsymbol{y}}_t^k)(\tilde{\boldsymbol{y}}_t^k)^T - (\bar{\boldsymbol{y}}_t)(\bar{\boldsymbol{y}}_t)^T \right]. \tag{B.2.7}$$

Here $\tilde{\boldsymbol{P}}_t$ is a positive semidefinite matrix which represents an increase in the covariance caused by the spread of the observations.

B.2.2 The data association mixture filter

We now outline the mixture filter, based on mixture reduction, used for the data association problem. For a description of the algorithm, see Section A.3.3.

STEP 1: STATE AND MEASUREMENT PREDICTIONS

Given N prior mixture component with mean, covariance and weights defined by $\hat{\boldsymbol{x}}_{t-1}^n$, $\boldsymbol{P}_{t-1}^n$, and μ_{t-1}^n (for $n \in \{1, \ldots, N\}$), state and measurement predictions $\hat{\boldsymbol{x}}_{t|t-1}^n$, $\boldsymbol{P}_{t|t-1}^n$, $\hat{\boldsymbol{y}}_{t|t-1}^n$ and $\boldsymbol{S}_t^n$ are calculated using (2.1.40)-(2.1.43).

STEP 2: MEASUREMENT GATING

Gating is now performed separately for each individual measurement prediction such that N measurement subsets $\boldsymbol{\mathcal{Z}}_t^n \subset \boldsymbol{\mathcal{Y}}_t$ are produced. For each $n \in \{1, \ldots, N\}$, measurement $\boldsymbol{y}_t(i) \in \boldsymbol{\mathcal{Y}}_t$ is accepted and included in $\boldsymbol{\mathcal{Z}}_t^n$ if:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^n)^T (\boldsymbol{S}_t^n)^{-1} (\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^n) \leq G\,. \tag{B.2.8}$$

Let $Z_t^n = \text{card}(\boldsymbol{\mathcal{Z}}_t^n)$.

STEP 3: MEASUREMENT UPDATE

Using banks of Kalman filters, each of the N predictions is now updated using the measurements $\boldsymbol{y}_t(k) \in \boldsymbol{\mathcal{Z}}_t^n$, which have satisfied its gating requirements. For $k \in \{1, \ldots, Z_t^n\}$, $\hat{\boldsymbol{x}}_t^{nk}$ and $\boldsymbol{P}_t^{nk}$ are calculated using (2.1.44)-(2.1.47). Furthermore, the algorithm retains the unupdated state predictions (and their covariance) to model the hypotheses where all the measurements received are false and the true measurement is not detected. Therefore for $k = 0$:

$$\hat{\boldsymbol{x}}_t^{n0} = \hat{\boldsymbol{x}}_{t|t-1}^n\,, \tag{B.2.9}$$

$$\boldsymbol{P}_t^{n0} = \boldsymbol{P}_{t|t-1}^n\,. \tag{B.2.10}$$

STEP 4: WEIGHT CALCULATIONS

For $k \in \{1, \ldots, Z_t^{nj}\}$, the weights of each of these hypotheses are calculated as:

$$\mu_t^{nk} = \frac{1}{c} \mu_{t-1}^n \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{S}_t^n)\,. \tag{B.2.11}$$

If λ is the clutter density and P_{DG} is the probability of true measurement being part of set $\boldsymbol{\mathcal{Z}}_t^n$, the probability of the null-hypothesis (i.e. that all gated measurements are false) is calculated as follows:

$$\mu_t^{n0} = \frac{1}{c} \mu_{t-1}^n \frac{(1 - P_{DG})\lambda}{P_{DG}}\,. \tag{B.2.12}$$

In both equations, c is a normalising term given by:

$$c = \sum_{n,k} \mu_{t-1}^{n} \left(\frac{(1 - P_{DG})\lambda}{P_{DG}} + \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^{n}, \boldsymbol{S}_t^{n}) \right). \tag{B.2.13}$$

STEP 5: STATE ESTIMATE & COVARIANCE CALCULATIONS

At this stage, the output state estimate and covariance are computed by merging together all the elements of the posterior conditional distribution. It should be noted that the output estimates should be calculated prior to the mixture reduction as this inevitably introduces degradations.

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \sum_{k=0}^{Z_t^n} \mu_t^{nk} \hat{\boldsymbol{x}}_t^{nk}, \tag{B.2.14}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \sum_{k=0}^{Z_t^n} \mu_t^{n} [\boldsymbol{P}_t^{nk} + (\hat{\boldsymbol{x}}_t^{nk} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^{nk} - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}]. \tag{B.2.15}$$

STEP 6: MIXTURE REDUCTION

The final part of this data association mixture filter involves a mixture reduction step. A number of mixture reduction techniques can be implemented, and these are described in Section 3.2. These algorithms input the expanded Gaussian mixture with $\sum_n (Z_t^n + 1)$ densities with means, covariance and weights given by $\hat{\boldsymbol{x}}_t^{nk}$, $\boldsymbol{P}_t^{nk}$ and μ_t^{nk}, and output a N-fold mixture of Gaussians with mean, covariance and weights $\hat{\boldsymbol{x}}_t^{n}$, $\boldsymbol{P}_t^{n}$ and μ_t^{n}.

B.3 Multiple-model/multiple-measurements target tracking algorithms

B.3.1 The IMMPDA algorithm

We now outline the *interacting multiple model probabilistic data association* algorithm (IMMPDA) in detail.

STEP 1: ESTIMATE MIXING

Firstly, given d prior densities with parameters $\{\hat{\boldsymbol{x}}_{t-1}^{i}, \boldsymbol{P}_{t-1}^{i}, \hat{r}_{t-1}^{i} = i, \mu_{t-1}^{i}\}$ (for $i \in \mathcal{S}$), the IMMPDA determines the mixing probabilities. These describe the probabilities of

each possible mode-transition based on prior information only. For $i, j \in \mathcal{S}$:

$$\mu_t^{ij} = \frac{1}{\bar{c}_j} \mu_{t-1}^i \pi_{ij} \,, \tag{B.3.1}$$

where,

$$\bar{c}_j = \sum_{i=1}^{d} \mu_{t-1}^i \pi_{ij} \,. \tag{B.3.2}$$

Secondly, the algorithm mixes the state estimates and covariances related to the modes as follows:

$$\tilde{\boldsymbol{x}}_t^j = \sum_{i=1}^{d} \hat{\boldsymbol{x}}_{t-1}^i \, \mu_t^{ij} \,, \tag{B.3.3}$$

$$\tilde{\boldsymbol{P}}_t^j = \sum_{i=1}^{d} \mu_t^{ij} [\boldsymbol{P}_{t-1}^i + (\hat{\boldsymbol{x}}_{t-1}^i - \tilde{\boldsymbol{x}}_t^j)(\hat{\boldsymbol{x}}_{t-1}^i - \tilde{\boldsymbol{x}}_t^j)^{\mathrm{T}}] \,. \tag{B.3.4}$$

STEP 2: STATE AND MEASUREMENT PREDICTIONS

Next state and measurement predictions are calculated for each mixed component using the mapping described in 2.3.22:

$$(\hat{\boldsymbol{x}}_{t-1}^j, \boldsymbol{P}_{t-1}^j, \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j) = \mathrm{KFpred}(\tilde{\boldsymbol{x}}_t^j, \tilde{\boldsymbol{P}}_t^j, j, t) \,. \tag{B.3.5}$$

STEP 3: MEASUREMENT GATING

The IMMPDA next gates the measurements. The approach used here follows the one presented in [18], where a single gate is used for all mixture components (alternatively a separate gate could be configured for each). Let m, taking possible values in $\mathcal{S}$, be the mode value for which the determinant of the measurement prediction covariance is largest. For $j \in \mathcal{S}$:

$$m = \mathrm{argmax}_j(\det(\boldsymbol{S}_t^j)) \,.$$

Then the measurements to be considered for update are those in $\boldsymbol{\mathcal{Z}}_t \subset \boldsymbol{\mathcal{Y}}_t$ which satisfy:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^m)^T (\boldsymbol{S}_t^m)^{-1} (\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^m) \leq G \,. \tag{B.3.6}$$

Let $Z_t = \mathrm{card}(\boldsymbol{\mathcal{Z}}_t)$.

STEP 4: LIKELIHOOD CALCULATIONS

Next, the IMMPDA performs a PDAF update step for each mixture component. The

algorithm calculates the probability of each association hypothesis, $\mathcal{H}_t^{jk}$, for $j \in \mathcal{S}$ and $k \in \{0, \ldots, Z_t\}$:

$$\mu_t^{jk} = \frac{1}{c} \begin{cases} \frac{(1-P_{DG})\lambda}{P_{DG}} & \text{for } k = 0\,, \\ \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j) & \text{for } k = 1, \ldots, Z_t\,. \end{cases} \tag{B.3.7}$$

STEP 5: RESIDUALS COMBINATION

Next, for each component, the IMMPDA calculates a weighted sum of the innovation (or residual) of each measurement. This is done as follows:

$$\bar{\boldsymbol{y}}_t^j = \sum_{k=0}^{Z_t} \mu_t^k \tilde{\boldsymbol{y}}_t^{jk}\,, \tag{B.3.8}$$

where:

$$\tilde{\boldsymbol{y}}_t^{jk} = \boldsymbol{y}_t(k) - \hat{\boldsymbol{y}}_{t|t-1}^j\,. \tag{B.3.9}$$

Note that $\tilde{\boldsymbol{y}}_t^{jk}$ should have already been calculated as part of the evaluation of $\mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^j, \boldsymbol{S}_t^j)$ in Step 4.

STEP 6: COMPONENT UPDATE

The predictions calculated in Step 2 are now updated using the combined residuals. If $\boldsymbol{K}_t^j$ is calculated as in (2.3.18), then:

$$\hat{\boldsymbol{x}}_t^j = \hat{\boldsymbol{x}}_{t|t-1}^j + \boldsymbol{K}_t^j \bar{\boldsymbol{y}}_t^j\,, \tag{B.3.10}$$

$$\boldsymbol{P}_t^j = \boldsymbol{P}_{t|t-1}^j - (1 - \mu_t^{j0}) \boldsymbol{K}_t^j \boldsymbol{H} \boldsymbol{P}_{t|t-1}^j + \boldsymbol{K}_t^j \tilde{\boldsymbol{P}}_t^j (\boldsymbol{K}_t^j)^T\,, \tag{B.3.11}$$

where:

$$\tilde{\boldsymbol{P}}_t^j = \sum_{k=1}^{Z_t} \left[\mu_t^{jk} (\tilde{\boldsymbol{y}}_t^{jk})(\tilde{\boldsymbol{y}}_t^{jk})^T - (\bar{\boldsymbol{y}}_t^j)(\bar{\boldsymbol{y}}_t^j)^T \right]. \tag{B.3.12}$$

The weight of each of these components is:

$$\mu_t^j = \sum_{k=0}^{Z_t} \mu_t^{jk}\,. \tag{B.3.13}$$

STEP 7: STATE ESTIMATE & COVARIANCE CALCULATIONS

The final step is the calculation of the output state estimate and covariance, which are

computed by merging together all the elements of the posterior conditional distribution.

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{j} \mu_t^j \hat{\boldsymbol{x}}_t^j , \tag{B.3.14}$$

$$\boldsymbol{P}_t = \sum_{n=1}^{N} \mu_t^j [\boldsymbol{P}_t^j + (\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_t^j - \hat{\boldsymbol{x}}_t)^{\mathrm{T}}] . \tag{B.3.15}$$

B.3.2 A multiple-measurement/multiple-model particle filter

We now outline a multiple-measurement/multiple-model particle filter. Many versions of this type of filter have been proposed, the one we present here is a direct extension of the multiple-model Bootstrap filter (see Section 2.3.4) to allow for multiple measurements, which is largely based on [46].

STEP 1: IMPORTANCE SAMPLING

Assuming the availability of N i.i.d. equally weighted samples from the prior distribution, each associated with mode $\hat{r}_{t-1}^n$ and with point mass located on $\hat{\boldsymbol{x}}_{t-1}^n$ (for $n \in \{1, \ldots, N\}$), the particle filter first simulates N state noise values, $\tilde{\boldsymbol{\nu}}_t^s(n)$, and N mode values, $\tilde{r}_{t-1}^n$, at time t. This is done by sampling $\tilde{\boldsymbol{\nu}}_t^s(n) \sim \mathcal{N}(0, \boldsymbol{Q}_m)$, and $\tilde{r}_t^n \sim p(r_t | r_{t-1} = \hat{r}_{t-1}^n) = \{\pi_{\hat{r}_{t-1}^n j}\}_{j=1}^d$. Here, $\tilde{r}_t^n$ is selected as the mode for which the uniformly distributed random sample $u_n \sim \mathcal{U}(0,1)$ satisfies:

$$\sum_{j=1}^{\tilde{r}_t^n - 1} \pi_{\hat{r}_{t-1}^n j} \leq u_n \leq \sum_{j=1}^{\tilde{r}_t^n} \pi_{\hat{r}_{t-1}^n j} . \tag{B.3.16}$$

Using these values, $\tilde{\boldsymbol{x}}_t^n$ and $\hat{\boldsymbol{y}}_{t|t-1}^n$ are evaluated as:

$$\tilde{\boldsymbol{x}}_t^n = \boldsymbol{F}(\tilde{r}_t^n)\hat{\boldsymbol{x}}_{t-1}^n + \boldsymbol{u}_t^s + (\boldsymbol{Q}_s(\tilde{r}_t^n))^{1/2}\tilde{\boldsymbol{\nu}}_t^s(n) , \tag{B.3.17}$$

$$\hat{\boldsymbol{y}}_{t|t-1}^n = \boldsymbol{H}\hat{\boldsymbol{y}}_{t|t-1}^n + \boldsymbol{u}_t^m . \tag{B.3.18}$$

STEP 2: MEASUREMENT GATING

Gating is now performed separately for each individual particle such that N measurement subsets $\boldsymbol{\mathcal{Z}}_t^n \subset \boldsymbol{\mathcal{Y}}_t$ are produced. For each $n \in \{1, \ldots, N\}$, measurement $\boldsymbol{y}_t(i) \in \boldsymbol{\mathcal{Y}}_t$ is accepted and included in $\boldsymbol{\mathcal{Z}}_t^n$ if:

$$(\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^n)^T (\boldsymbol{Q}_m)^{-1} (\boldsymbol{y}_t(i) - \hat{\boldsymbol{y}}_{t|t-1}^n) \leq G . \tag{B.3.19}$$

Let $Z_t^n = \text{card}(\boldsymbol{\mathcal{Z}}_t^n)$.

STEP 3: LIKELIHOOD CALCULATIONS

Next, the likelihood of each particle is evaluated. For $y_t(k) \in \mathcal{Z}_y^n t$, this is given by:

$$\mu_t^n = \frac{1}{c}\left(\frac{(1-P_{DG})\lambda}{P_{DG}} + \sum_{k=1}^{Z_t^n} \mathcal{N}(\boldsymbol{y}_t(k); \hat{\boldsymbol{y}}_{t|t-1}^n, \boldsymbol{Q}_m)\right), \tag{B.3.20}$$

where c ensures $\sum_n \mu_t^n = 1$.

STEP 4: STATE ESTIMATE CALCULATION

The output state estimate, $\hat{\boldsymbol{x}}_t$, is computed as a weighted sum of all particles:

$$\hat{\boldsymbol{x}}_t = \sum_{n=1}^{N} \mu_t^n \tilde{\boldsymbol{x}}_t^n . \tag{B.3.21}$$

STEP 5: RESAMPLING

For $n \in \{1, \ldots, N\}$, resample with replacement N particles $\{\hat{\boldsymbol{x}}_t^n, \hat{r}_t^n\}$ from the set of samples $\{\tilde{\boldsymbol{x}}_t^n, \tilde{r}_t^n\}$ according to importance weights μ_t^n, using one of the techniques described in Section 2.2.2.

Appendix C

Conduct of simulations

This appendix provides further details on the conduct of the simulations presented in this thesis.

The entire simulation environment was developed by the author using Matlab version 7. Most of the simulation results presented in this thesis were executed on a dual processor AMD Opteron 246 machine (2.0GHz) running Redhat Enterprise Linux 4 (64-bit edition) with 2GB of RAM, although some of the most computationally intensive Monte Carlo tasks were performed in parallel on a twin machine.

As most of the simulations presented make use of computational time as a performance metric, every effort was made to ensure fair and comparable results. However, although all algorithms were programmed and executed in as identical manner as possible, the simulation times presented should be taken as indicative only as these are inevitably highly dependent on programming language and style (as well as hardware, of course). Similar care was taken to minimise the effects of randomisation: common sets of measurements were pre-generated and used by all algorithms.

The simulation environment has now been passed on for further development to the Control & Power group of the Electrical & Electronic Engineering department at Imperial College London. However the author would be happy to provide further information if needed, and may be contacted at stefano_robbiati@hotmail.com.

Bibliography

[1] V. J. Aidala. Kalman filter behavior in bearings-only tracking applications. *IEEE Trans. Aerosp. Electron. Syst.*, 15(1):29–39, Jan. 1979.

[2] V. J. Aidala and S. E. Hammel. Utilization of modified polar coordinates for bearings-only tracking. *IEEE Trans. on Automatic Control*, 28(3):283–294, March 1983.

[3] V. J. Aidala and S. C. Nardone. Biased estimation properties of the pseudolinear tracking filter. *IEEE Trans. Aerosp. Electron. Syst.*, 18:432–441, Jul. 1982.

[4] H. Akashi and H. Kumamoto. Random sampling approach to state estimation in switching environments. *Automatica*, 13:429–434, 1977.

[5] B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Information and System Science Series. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

[6] D. Angelova and L. Mihaylova. Sequential Monte Carlo algorithms for joint target tracking and classification using kinematic radar information. In *Fusion 2004*, pages 709–716, Stockholm.

[7] M. S. Arulampalam, B. Ristic, N. Gordon, and T. Mansell. Bearings-only tracking of manoeuvring targets using particle filters. *EURASIP Journal on Applied Signal Processing*, 15:2351–2365, 2004.

[8] S. Arulampalam and B. Ristic. Comparison of the particle filter with range-parameterised and modified polar EKFs for angle-only tracking. *Proc. of SPIE*, 4048:288–299.

[9] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association.* Academic Press, Orlando (FL, USA), 1988.

[10] Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking: Principles, Techniques, and Software.* Artech House, Norwood (MA, USA), 1993.

[11] Y. Bar-Shalom and X.-R. Li. *Multitarget-multisensor Tracking: Principles and Techniques.* YBS Publishing, Storrs (CT, USA), 1995.

[12] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software.* John Wiley & Sons, New York, 2001.

[13] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11:451–460, September 1975.

[14] J.M. Bernardo and A.F.M. Smith. *Bayesian Theory.* John Wiley & Sons, Chicester (West Sussex, UK), 1994.

[15] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artech House, Norwood (MA, USA), 1999.

[16] S. S. Blackman. *Multiple Target Tracking with Radar Applications.* Artech House, Norwood (MA, USA), 1986.

[17] H. A. P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with Markovian switchin coefficients. *IEEE Transactions on Automatic Control*, August 1988.

[18] H. A. P. Blom and E. A. Bloem. Combining IMM and PDA for tracking multiple maneuvering targets in clutter. In *Proc. 5th Int. Conf. on Information Fusion.*

[19] H. A. P. Blom and E. A. Bloem. Joint IMMPDA particle filter. In *Proc. 6th Int. Conf. on Information Fusion.*

[20] H. A. P. Blom and E. A. Bloem. Tracking multiple maneuvering targets by joint combinations of IMM and PDA. In *IEEE Conf. on Dec. and Control*, volume 42.

[21] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings* -, 146(1):2–7.

[22] R. Chen and J. S. Liu. Mixture Kalman filters. *J. of the Royal Statistical Society B*, 62(3):493–508, 2000.

[23] J. M. C. Clark, S. A. Robbiati, and R. B. Vinter. The shifted Rayleigh mixture filter for bearings-only tracking of manoeuvring targets. *Accepted for publication in IEEE Transactions on Signal Processing*, 2006.

[24] J. M. C. Clark, R. B. Vinter, and M. M. Yaqoob. The shifted Rayleigh filter: a new algorithm for bearings-only tracking. *Submitted to IEEE Trans. on Aerospace and Electronic Systems.*, 2005, available at http://www3.imperial.ac.uk/pls/portallive/docs/1/3853901.PDF.

[25] J. M. C. Clark, R. B. Vinter, and M.M. Yaqoob. The shifted Rayleigh filter for bearings only tracking. Presented at the 8th International Conference on Information Fusion, Philadelphia, PA, USA, 2005.

[26] W. G. Cochran. *Sampling techniques.* John Wiley & Sons, London, 3rd edition, 1963.

[27] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate Gaussian distributed states. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):909–916, 1992.

[28] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, New York (NY, USA), 2001.

[29] A. Doucet, N. J.Gordon, and V. Krishnamurthy. Particle filter for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, March 2001.

[30] M. Efe and D.P. Atherton. Maneuvering target tracking using adaptive turn rate models in the interacting multiple model algorithm. In *Decision and Control, 1996, Proceedings of the 35th IEEE*, volume 3, pages 3151–3156 vol.3, 1996.

[31] T.E. Fortmann et al. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Ocean Engineering*, 8:173–184, Jul. 1983.

[32] P. Fearnhead. *Sequential Monte Carlo methods in filter theory.* D.Phil thesis, University of Oxford, 1998.

[33] P. Fearnhead and P. Clifford. On-line inference for hidden Markov models via particle filters. *J. of the Royal Statistical Society B*, 65(4):887–899, 2003.

[34] R. J. Fitzgerald. Error divergence in optimal filtering theory. In *2nd IFAC symp. AC space*, Vienna, Austria, 1968.

[35] J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 24:1317–1399, 1989.

[36] W. R. Gilks, S.Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice.* Chapman & Hall, Bungay (Surrey, UK), 1996.

[37] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[38] D. J. Hand. *Discrimination and Classification.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, Chichester, 1981.

[39] S. Haykin and N. deFreitas. Special issue on sequential state estimation. In *Proceedings of the IEEE.*

[40] A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[41] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *Automatic Control, IEEE Transactions on*, 45(3):477–482.

[42] S. Julier and J. K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford - Available at http://www.robots.ox.ac.uk/ siju, 1996.

[43] S. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL, 1997.

[44] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. AMSE, Journal of Basic Engineering*, 82(D):35–45, 1960.

[45] A. D. Karan and S. E. Wang. Bearings-only tracking with sea trial sonar data from multiple asynchronous sonobuoys. In *Proc. Underwater Defence Technology Conference*, Sydney, Australia, 1998.

[46] R. Karlsson and F. Gustafsson. Monte Carlo data association for multiple target tracking. In *IEE Target Tracking: Algorithms & Applications*, volume 1, pages 13/1 – 13/5, 2001.

[47] T.H. Kerr. Streamlining measurement iteration for EKF target tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 27(2):408–421.

[48] Qiang Le, L.M. Kaplan, and J.H. McClellan. Multiple-mode Kalman filtering with node selection using bearings-only measurements. In *System Theory, 2004. Proceedings of the Thirty-Sixth Southeastern Symposium on*, pages 185–189, 2004.

[49] J. S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1022–1031, 1998.

[50] A. D. Marrs. Asynchronous multi-sensor tracking in clutter with uncertain sensor locations using Bayesian sequential Monte Carlo methods. In *Proc. IEEE Aerospace Conference*, volume 5, pages 2171–2178, Big Sky, MT, 2001.

[51] S. Maskell. *Sequentially Structured Bayesian Solutions.* PhD thesis, Cambridge University, 2004.

[52] P. S. Maybeck. *Stochastic Models, Estimation and Control: Volume 1*, volume 141-1 of *Mathematics in Science and Engineering.* Academic Press, San Diego - London, 1979.

[53] P. S. Maybeck and B. D. Smith. Multiple model tracker based on Gaussian mixture reduction for maneuvring targets in clutter. In *Fusion 2005*, Presented at the 8th International Conference on Information Fusion, Philadelphia, PA, USA, 2005.

[54] S. McGinnity and G.W. Irwin. Multiple model bootstrap filter for maneuvering target tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 36(3):1006–1012.

[55] S. McGinnity and G.W. Irwin. Manoeuvring target tracking using a multiple-model bootstrap filter. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 479–497. Springer-Verlag, New York (NY - USA), 2001.

[56] D. Musicki and et al. Integrated probabilistic data association. *IEEE Transactions on Automatic Control*, 39(6):1237–1241, 1994.

[57] N. Peach. Bearings-only tracking using a set of range-parameterised extended Kalman filters. *IEE Proc. Contr. Th. Appl.*, 142:73–80, Jan. 1995.

[58] M. K. Pitt and N. Shepherd. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[59] Branko Ristic and M. Sanjeev Arulampalam. Tracking a manoeuvring target using angle-only measurements: algorithms and performance. *Signal Processing*, 83(6):1223–1238, 2003.

[60] A. Rychlinski. High performance target tracking. M.Eng thesis, Department of Electrical & Electronic Engineering, Imperial College London, London (UK), June 2003.

[61] D. J. Salmond. *Tracking in uncertain environments.* D.Phil thesis, University of Sussex, 1989.

[62] D. J. Salmond. Mixture reduction algorithms for target tracking in clutter. *SPIE, Signal and Data Processing of Small Targets*, 1305:434–444, 1990.

[63] R. G. Sea. Optimal correlation of sensor data with tracks in surveillance systems. *Proc. of 6th Int. Conf on System Sciences*, pages 424–426, Jan., 9–11 1973.

[64] S. K. Singh. Ellipsoid gates for an airborne track while scan radar. *Proc. of IEEE Int. Radar Conference*, pages 334–339, May, 8–11 1995.

[65] T. L. Song and J. L. Speyer. A stochastic analysis of a modified gain extended Kalman filter with applications to estimation with bearings only measurements. *IEEE Trans. on Automatic Control*, 30(10):940–949, Oct. 1985.

[66] H. W. Sorenson. Kalman filtering techniques. In C. T. Leondes, editor, *Advances in Control Systems Theory and Applications*, volume 3, pages 219–292. Academic Press Inc., 1966.

[67] N. A. Thacker and A. J. Lacey. Tutorial: the Kalman filter, TINA, University of Manchester, 1998, Available at http://www.tina-vision.net/docs/memos/1996-002.pdf.

[68] J. K. Tugnait. Detection and estimation for abruptly changing systems. *Automatica*, 18:607–615, 1982.

[69] J. K. Tugnait and A. H. Haddad. A detection-estimation scheme for state estimation in switching environments. *Automatica*, 15(4):477–481, 1979.

[70] E. A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158.

[71] E. K. Westwood. Filtering algorithms for the linear estimation problem with switching parameters. M.S. thesis, University of Texas, Austin (US), 1984.

[72] J. L. Williams. *Gaussian Mixture Reduction for Tracking Multiple Maneuvring Targets in Clutter.* M.S.E.E. thesis, Air Force Institute of Technology, 2003.

[73] J. L. Williams and P.S. Maybeck. Cost-function-based Gaussian mixture reduction for target tracking. In *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, volume 2, pages 1047–1054.

[74] R. P. Wishner, J. A. Tabaczynski, and M. Athans. A comparison of three non-linear filters. *Automatica*, 5(4):487–496.

[75] M. M. Yaqoob. Moment matching trackers for difficuly targets. MPhil-to-PhD transfer report, Imperial College, London, UK, 2005.