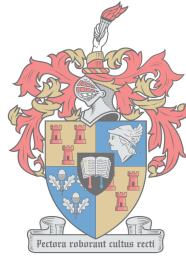


A Probabilistic Graphical Model Approach to Multiple Object Tracking

by

Everhard Johann Louw



UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

100
1918 · 2018

Thesis presented in partial fulfilment of the requirements for
the degree of Master of Engineering (Electrical & Electronic)
in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. J.A. du Preez

March 2018



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / Plagiarism Declaration

- 1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.
- 2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.
- 3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.
- 4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelike aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
- 5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

March 2018

Abstract

Probabilistic graphical models (PGMs) provide a framework for efficient probabilistic inference using graphs that correspond to factorised representations of high-dimensional probability distributions. The problem of tracking objects from noisy measurements is inherently a probabilistic one and the use of PGMs to solve this problem is therefore appropriate.

In this work, we investigate how PGMs can be used for tracking an unknown and varying number of targets in challenging scenarios. While many existing algorithms provide solutions to the multiple object tracking (MOT) problem, none of the established algorithms are framed as PGMs. In order to develop a graphical model for multiple object tracking, the connections between PGM theory and the Kalman filter algorithm, which is commonly used for single object tracking, are investigated. The PGM equivalent of the Kalman filter is used as a starting point for the development of the MOT PGM. The Kalman filter PGM is first expanded to allow a known and constant number of targets to be tracked, and Bayesian model selection is then used to allow the number of targets to be inferred automatically. In order to allow the model to track targets in the presence of false detections, a clutter classification model is developed and incorporated into the developed PGM. The efficiency of the model is improved through the use of an alternative model selection method. It is also shown that the tracking accuracy can be improved through more accurate Gaussian mixture approximations of the target state distributions.

The developed model is compared to a state-of-the-art method and is tested by way of a large number of simulations. We conclude that the model is capable of consistently and accurately tracking targets and that it offers advantages over some existing methods. Finally, the model output is compared to an industrial application with real radar data as input. The outputs of the two models are largely similar and the test results therefore indicate that the developed model can be used for real-world applications.

In order to create a general software resource for implementing the type of PGMs designed in this work, the University of Stellenbosch PGM library, EMDW was expanded. A large portion of the software created as part of this work is therefore not limited to the multiple object tracking problem, but useful for PGM inference in general.

Opsomming

Grafiese waarskynlikheidsmodelle (GWM'e) maak van gefaktoriseerde voorstellings van hoë-dimensionele waarskynlikheidsverdelings gebruik en bied 'n raamwerk vir die doeltreffende berekening van randverdelings. Om akkurate vooruitskattings van teikenposisies te maak aan die hand van metings waar ruis teenwoordig is, is inherent 'n waarskynlikheidsprobleem. GWM'e kan dus gebruik word om die probleem van teikenvolging effektief op te los.

In hierdie werk ondersoek ons hoe GWM'e vir die volg van 'n onbekende en wisselende aantal teikens in uitdagende scenario's gebruik kan word. Alhoewel daar baie bestaande algoritmes is wat oplossings vir die probleem van multi-teikenvolging (MTV) bied, word geen van die gevestigde algoritmes as GWM'e voorgestel nie. Ten einde die grafiese MTV-model te ontwikkel, word die verband tussen GWM-teorie en die Kalman-filter-algoritme, wat algemeen gebruik word vir enkel-teikenvolging, ondersoek. Die GWM-ekwivalent van die Kalman-filter word as uitgangspunt vir die ontwikkeling van die MTV-GWM gebruik. Die Kalman-filter-GWM word eers uitgebrei om 'n bekende en konstante aantal teikens te volg, en Bayesiese model-seleksie word dan gebruik om die aantal teikens outomaties te bepaal. 'n Vals meting-klassifikasie-model word ook ontwikkel om die model toe te laat om teikens in die teenwoordigheid van vals metings te volg. Die doeltreffendheid van die model word verbeter by wyse van 'n alternatiewe metode vir model-seleksie. Ons wys ook dat akkuraatheid deur 'n beter benadering tot die teikenverdelingsfunksies verbeter kan word.

Die ontwikkelde model word met 'n ultramoderne metode vergelyk en ook by wyse van 'n groot aantal simulaties getoets. Ons kom tot die gevolgtrekking dat die model daartoe in staat is om teikens konsekwent en met hoë akkuraatheid te volg en dat die ontwikkelde model 'n paar voordele bo sommige bestaande metodes bied. Laastens word die model-afvoer vergelyk met dié van 'n industriële toepassing met regte radar data as toevoer. Die afvoere van die twee modelle is meestal soortgelyk, en die toetsresultate dui dus daarop dat die ontwikkelde model vir werklike toepassings gebruik kan word.

Ten einde 'n algemene sagteware-hulpbron te skep vir die implementering van die tipe GWM'e wat in hierdie werk ontwerp is, is die Universiteit van Stellenbosch se GWM-biblioteek, EMDW, uitgebrei. 'n Groot gedeelte van die sagteware wat in hierdie werk ontwikkel is, is dus nuttig vir die gebruik van GWM'e in die algemeen en nie beperk tot die MTV-probleem nie.

Acknowledgements

I would like to thank Reutech Radar Systems (RRS) for funding this study and for supplying the real radar data (which was recorded in collaboration with the CSIR) that was used in this study. I would also like to thank Prof. Johan du Preez for his guidance and support. Prof. du Preez's many excellent explanations and insights were invaluable to this work and my understanding of PGM theory. Finally, I would like to thank Janie Slabber for her unwavering love and support and my parents for their support and encouragement to pursue further studies.

Contents

List of Figures

List of Tables

Abstract	ii
Opsomming	iii
Acknowledgements	iv
Nomenclature	xix
1 Introduction	1
1.1 Motivation	1
1.2 Background	2
1.2.1 Multiple Object Tracking	2
1.2.2 Probabilistic Graphical Models	3
1.3 Literature Synopsis	6
1.4 Objectives	7
1.5 Contributions	7
1.6 Overview	9

<i>CONTENTS</i>	vi
2 Literature Study	17
2.1 Classical Multiple Object Tracking Algorithms	17
2.1.1 The Probability Hypothesis Density Filter	17
2.1.2 The Multiple Hypothesis Tracking Filter	20
2.1.3 The Joint Probabilistic Data Association Filter	24
2.2 Probabilistic Graphical Models for Multiple Object Tracking	24
2.3 Conclusion	26
3 Probabilistic Graphical Models	27
3.1 Bayes Networks	27
3.2 Variable Elimination and Cluster Graphs	28
3.3 Belief Propagation in Cluster Graphs	32
4 The Multivariate Gaussian Distribution	36
4.1 Gaussian Parameterisations and Operations	36
4.1.1 Covariance Form to Canonical Form Conversion	37
4.1.2 Gaussian Operations	37
4.2 Linear Gaussian Distributions	39
4.3 Non-Linear Transformations of Gaussian Random Variables	41
4.4 Conclusion	44
5 The Kalman Filter as a Graphical Model	45
5.1 The Linear Kalman Filter as a PGM	45
5.2 The Kalman Filter Equations	49
5.3 Message Passing Characteristics	51

<i>CONTENTS</i>	vii
5.4 The Unscented Kalman Filter and the Cartesian-Polar Transform	52
5.5 Conclusion	57
6 A PGM for Multiple Object Tracking	58
6.1 A Simplified Example: Two Object Tracking	58
6.2 A Visual Example of Two Object Tracking	63
6.3 Bayesian Model Selection	71
6.4 Object Tracking with Detections from a Scanning Radar	75
6.5 Clutter Classification	77
6.6 Conclusion	86
7 Improving Tracking Efficiency	87
7.1 An Alternative View of Model Selection	87
7.2 Reducing the Number of Hypotheses	91
7.3 Efficient MOT PGM Algorithm	93
7.4 Conclusion	96
8 Gaussian Mixture Utilisation in the LBU Algorithm	97
8.1 Gaussian Mixture Operations	97
8.2 Gaussian Mixture Division	98
8.3 Gaussian Mixture PGM improvement over the Moment Matching Gaussian PGM	101
8.4 Conclusion	103
9 Implementation Overview	105
9.1 Factor Classes	105

<i>CONTENTS</i>	viii
9.2 Model Implementation and Testing	107
9.3 Conclusion	108
10 PGM Performance Analysis	109
10.1 Traditional Filters and PGM Comparison	109
10.2 GM-PHD Filter and PGM Performance Comparison	111
10.3 PGM Performance	115
10.4 Real Radar Data Evaluation	119
10.5 Conclusion	122
11 Conclusion And Future Work	123
11.1 Conclusion	123
11.2 Future Work	125
Appendices	131
A Proofs and Derivations	132
A.1 Derivations of Marginal Expressions from Variable Elimination Example	132
A.2 Multivariate Gaussian Operation Derivations	135
A.2.1 Block Matrix Inversion Lemma	135
A.2.2 Gaussian Covariance Form to Canonical Form Derivation	136
A.2.3 Gaussian Canonical Product and Quotient Derivations	137
A.2.4 Gaussian Marginal Derivation	138
A.2.5 Gaussian Canonical Conditioning Derivation	141
A.3 Cross-Covariance Matrix Derivation	144
A.4 Cartesian-Polar Transformations	144

<i>CONTENTS</i>	ix
A.5 Alternative Model Selection for Multiple Object Tracking - Proof Sketch	145
A.6 Numerical Hessian Calculation	146
B Supplementary Figures	148
B.1 Model Selection Example Hypothesis	148
B.1.1 Gaussian Mixture vs. Gaussian Mixture Moment Matching Performance Example	151
C Causal Clutter Classification Model	153

List of Figures

1.1	Example of a two-dimensional linear Gaussian CPD. Note that, for a given value of x , this distribution reduces to an ordinary Gaussian distribution.	5
1.2	Data association problem represented as a Bayes Network. The measurement (Y) is dependent on the two object states (X_1 and X_2), as well as which state caused the measurement (encoded by the association random variable A).	11
1.3	PGM vs GM-PHD comparison (with simulated data) showing the relatively poor performance (compared to the PGM) of the PHD filter when there are multiple missed detections. The PHD OSPA (as well as that of the detections) also fluctuates erratically with the missed detections, while the performance of the PGM is far more consistent.	14
1.4	Industrial model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)	15
1.5	PGM model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)	15
1.6	A close-up example of one of the tracks from figures 1.4 and 1.5 in (a) and (b) respectively showing the relative similarity between the estimates from the PGM and the industrial model. Although some tracks are only present in one of the outputs, the longest, most prominent track is more or less covered by both models. The detections interpreted as clutter are mostly consistent between the two models.	16
2.1	Example of an MHT hypothesis tree. y_i is the i 'th measurement in the scan and the node numbers in the corresponding row indicate the track with which y_i is associated in the specific hypothesis. The number 0 tracks represent the clutter track. The blank node at the top of the graph represents the initial hypothesis. (Note that this is not a PGM.)	22

LIST OF FIGURES

xi

3.1	Student-grade Bayes network example. The concepts contained in the nodes represent random variables and the edges encode the dependencies between these random variables. (This example is adapted from an example in [1, p. 162]) . . .	28
3.2	Simple example of a Bayes network	29
3.3	Cluster graph corresponding to the Bayes network in Figure 3.2. The elliptical nodes are the cluster nodes, and the square nodes are the separation sets (sepsets). The numbers beneath the cluster nodes indicate their respective indices, corresponding to the subscripts in the variable elimination/message passing equations.	31
3.4	Compiling a Bayes network into a cluster graph (example). The factors represented by the Bayes network in (a) is connected to form the graph in (b). The A and B nodes have been absorbed by their neighbours in the final cluster graph in (c).	33
3.5	Cluster graph message passing example. The messages are indicated by δ_{mn} for a message sent from factor number m to number n . The factor numbers are indicated by the numbers on the outside of the factor nodes.	34
4.1	Linear transformation of Gaussian sigma points. The ellipses indicate the contours of the two distributions. The dots show the sigma points of the respective distributions.	43
5.1	Kalman filter belief updates with fully observed state measurements. The covariance matrices and means of the various distributions are indicated with (1σ) error ellipses and the central dots respectively. The numbers underneath the means indicate the corresponding time step.	46
5.2	Kalman filter belief updates with partially observed state. The distribution covariance matrices and means are indicated with (1σ) error ellipses and the central dots respectively. The position measurement mean and standard deviation are indicated by the thick and thin blue lines respectively. The numbers underneath the means indicate the corresponding time step. Note that the velocity estimates are accurate despite the fact that the object's velocity is not measured.	47
5.3	Kalman filter Bayes (a) network and cluster graph (b) example with four state random variables (X 's), representing the state evolution over four time steps, and four measurement random variables (Y 's).	48

5.4	Cartesian space (x, y) and velocity (v_x, v_y) to polar space (r, θ) and velocity (v_r, v_θ) transformation functions. Note that the v_x and v_y velocities are constant in (c) and (d). Their values do not change the fundamental shape of the functions. The discontinuity in (b) occurs on the negative x ($y = 0$) line because θ increases from 0° to 180° in the anti-clockwise direction and decreases from 0° to -180° in the clockwise direction. These graphs serve to give the reader an idea of the nature of the non-linearities that is inherent in the Cartesian-polar transform.	53
5.5	Sigma point polar velocity example 1, showing poor sigma point approximation when Cartesian position marginals are near the origin. (sp: sigma points)	54
5.6	Sigma point polar velocity example 2, showing improved sigma point approximation (compared to Figure 5.5b) when the Cartesian state marginals are moved away from the origin. Note that the sigma point mean and covariance are virtually identical to that of the samples, as is evident from the two standard deviation error ellipses that are almost exactly aligned. (sp: sigma points)	55
5.7	Sigma point polar spatial example 1 showing poor sigma point approximation when Cartesian position marginals are near the origin. (sp: sigma points)	56
5.8	Sigma point polar spatial example 2 showing improved sigma point approximation (compared to Figure 5.7b) when the Cartesian state marginals are moved away from the origin. (sp: sigma points)	56
6.1	Two time-slice Bayes network (2TBN) for two object tracking, with sing. The partial graphs in the rectangles show the structure for a single time-slice of the larger graph, while the connections between the rectangles indicate how the graph is connected between time-slices	59
6.2	Example of a short section (over only two time steps) of a two object tracking cluster graph.	59
6.3	Prior distributions over the states (one dimensional positions) of two objects. Note that the two distributions are shown on the same axis for comparison, although they have different scopes.	64
6.4	Linear-Gaussian distributions defined by messages from state clusters. Note that the two distributions are shown on the same axes for comparison, although they have different scopes.	64
6.5	Full joint $P(X_1, X_2, Y, a)$ distribution. Note that the two distributions correspond to different assignments of the a variable.	65
6.6	Two views on $P(X_1, X_2, Y, a)$ distribution showing conditional independencies. .	65

6.7	Observing evidence by conditioning the joint distribution $P(Y, X_1, X_2, a)$ on a measurement (indicated by the 2 dimensional plane).	66
6.8	Distributions reduced by evidence: Dominating $P(X_2, Y = \mathbf{y})$ and much smaller $P(X_1, Y = \mathbf{y})$ indicated by the red ellipse.	66
6.9	$\delta_{YX_1}(X_1)$ measurement-state message calculation, showing the marginal distribution in (a) and Gaussian mixture message with one vacuous component in (b).	67
6.10	$\delta_{YX_2}(X_2)$ measurement-state message calculation, showing the marginal distribution in (a) and Gaussian mixture message with one vacuous component in (b).	68
6.11	The Gaussian mixture posterior components for X_1 and X_2 . Note that the two distributions are shown on the same axis for comparison, although they have different scopes. The connotations of the different Gaussian components are shown in the legend. Here we can see that the measurement is most likely associated with object 2 (with state X_2), since the updated X_2 component is much larger than the prior ('not updated') component and the updated component of the X_1 distribution is much smaller than its prior component.	69
6.12	True Gaussian mixture posterior state distributions and corresponding Gaussian approximations. The posterior (and M-projection approximation) of the X_1 distribution is largely unchanged from the prior ('not updated') distribution in (a). The posterior (and M-projection) of the X_2 distribution is significantly moved towards the measurement (at $Y = 10$) and the variance is decreased in (b). These two very different update effects are due to the high probability that the measurement is associated with object 2.	69
6.13	Another example of Gaussian mixture (GM) posterior state distributions and corresponding Gaussian M-projection approximations, showing less accurate approximations when the measurement is ambiguous and the true GM posterior distributions are less Gaussian.	70
6.14	Model selection example showing the most and least likely hypothesis. The separate rectangles indicate different time-slices. The hypothesis in (a) has the highest calculated probability and is also the correct model. The hypothesis in (b) is that one additional object is present at time step three, but this is unlikely and the hypothesis probability is much lower than hypothesis 1 in (a). The hypothesis in (c) is that each measurement corresponds to a new target and is the least likely out of all the hypotheses.	74

6.15	The probability distribution over model hypotheses, showing the correct hypothesis (hypothesis 1 - shown in Figure 6.14a) with the highest probability. Higher model numbers correspond to more complex models and also have decreasing probability. This shows the Occam's razor like effect that is inherent in Bayesian model selection.	75
6.16	Rotating radar view and new object spatial prior. The arrow arc illustrates the rotating nature of the radar beam and spatial prior.	76
6.17	Example of the clutter classification model (Bayes network) connected to a single cluster-slice of the MOT PGM. Whether a detection was generated (g_i^t) by a target i is dependent on the associations (a_i^t), and the 'realness' (r_i^t) of a target (at a specific time) is dependent on the expectation (e_i^t) that the target should have generated a detection and whether it generated (g_i^t) a detection. Note that the time step indices have been omitted here, as all the variables correspond to the same time step.	79
6.18	Example of three target state distributions and visible map area (red window).	80
6.19	Example of three target state distributions and invisible map area (black).	80
6.20	Gaussian equivalent of window (in red) and window complement (in black) functions shown in figures 6.18 and 6.19, respectively. The same three target state distributions are also shown here.	81
6.21	Two time-slice Bayes network of MOT PGM with clutter classification. The e_i^t random variable represents whether or not a detection is expected from target i at time t . The g_i^t variable corresponds to whether or not a detection was generated from target i at time t . The r_i^t random variable corresponds to whether the target is real or not for a specific time step. The r_i nodes consolidate the information from all the applicable r_i^t nodes and correspond to whether or not the target is real in general.	84
6.22	MOT and clutter classification cluster graph, corresponding to the Bayes network in Figure 6.17. This graph corresponds to a sub-graph (of a larger graph, time-unrolled graph) for a single time step, where two measurements have been received, and three objects are being tracked.	85
7.1	An example of a valid ground truth model Bayes network (a) and corresponding cluster graph (b) with three objects (each with a corresponding X node/cluster) and two measurements (each with a corresponding Y node/cluster)). X_2 is not associated with either measurement in this example and therefore has no link to a measurement node/cluster.	88

LIST OF FIGURES

xv

7.2	Densely connected model	89
7.3	Possible valid models corresponding to the assignments and probabilities in the association cluster after message passing.	90
7.4	Example of two targets (X_1 and X_2) with regions of relative certainty about their positions (indicated by ellipses with means at the X_1 and X_2 dots respectively) and four detections (+'s) that are colour coded to indicate which targets they can reasonably be associated with. Detection 'D' is too far from both of the targets and is therefore not associated with either.	92
8.1	Accuracy of Gaussian mixture quotient approximation indicated by the error-bar plots (a), which show the distance between the approximations and the true quotient functions. Random simulations were run for quotients with different numbers of numerator (N) and denominator (D) components, for quotient functions of dimension one to ten. Figure (b) shows an example of the worst approximation for the one dimensional case. The C2 distance for this example is 0.024.	101
8.2	A comparison of tracking accuracy between the MM-PGM and GM-PGM models. All the objects in this example start in the center of the figures and move outward. The scanning radar simulator (without angular velocity measurements) was used in this example. The true tracks in these examples are indicated by solid dark lines, the track estimates by the dashed colour lines, and the detections by the black circles. Note that there are a few errors in the MM-PGM estimates in (a) that are not present in the GM-PGM estimates in (b). The pink estimate in (a), for example, switches between two true tracks, while the correct identities of the tracks are maintained in (b). Also, the grey estimate is only added much later in (a) while the estimates (in pink in (b)) corresponding to the same track is present from the beginning in (b). Lastly, the light green track estimate in (b) is also present throughout the existence of the real track, while the corresponding object is initially not tracked in (a). Note that the track estimates are indicated as dashed lines only to allow the true tracks to be more visible.	102
8.3	Gaussian mixture PGM (GM-PGM) and moment-matching PGM OSPA distance comparison. The OSPA distance on the vertical axis is the OSPA distance between the estimates (and detections) and the ground truth track points at specific time steps. Note that the OSPA distance of the GM-PGM is initially lower than that of the MM-PGM, due to the correct estimation of tracks arising from the more correct Gaussian mixture representation of the state distributions.	103
10.1	single track OSPA comparison, showing almost identical performance between the PGM and PHD for this basic case with one track, no clutter and no missed detections.	112

- 10.2 PGM and PHD two track OSPA comparison, showing the reduced performance of the PHD filter in the presence of missed detections and the good performance of the PGM under the same conditions. The PHD filter OSPA distance fluctuates due to the periodic errors that are made in the cardinality of the estimates with missed detections and the periodic corrections of these errors. Note the increase in the PGM's OSPA in the interval between 8 and 12 seconds. This is due to ambiguous detections that are received when the tracks cross. 114
- 10.3 Consistent performance of the developed PGM with increasing clutter. The standard deviations are indicated by the vertical lines. 115
- 10.4 Consistent performance of the PGM with varying number of targets. The standard deviations are indicated by the vertical lines. (No clutter was introduced for these simulations.) 116
- 10.5 PGM performance over one hundred random simulations, showing relatively consistent tracking performance. The filtered estimates are consistently more accurate than the detections, and the smoothed estimates are consistently more accurate than the filtered estimates. The estimated number of targets is mostly correct (target count error= 0) and, when it is not, the OSPA of the estimates is still low. 117
- 10.6 Computational complexity with number of targets. These graphs were generated from the results of thirty random simulations for each number of targets. The standard deviations are indicated by the error bars. The time axis indicates the computation time per simulation. The function f_1 is fitted to the points that are one standard deviation below the mean values of the time durations. Similarly, f_2 is fitted to the points that are one standard deviation above the means. The f_1 function seems to be close to linear, indicating that one could expect a more or less linear increase in the execution time in the best case scenario. The f_2 function seems to be quadratic, indicating that the longest execution times could be quadratic in the number of targets. These functions could, however, have small higher order components that are not visible in these plots. No false detections were introduced in these simulations. The corresponding average estimate OSPA distances for these simulations are plotted in Figure 10.4. 119
- 10.7 Industrial model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines) 120
- 10.8 PGM model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines) 120

10.9	Real radar detection data and PGM track estimates overlaid on a satellite image. The data was recorded in the Kruger National Park. The radar detections are indicated by black circles and the different colours indicate different tracks. The ‘a’ and ‘c’ circles indicate areas where there are discrepancies between the output of the PGM and the industrial multiple target tracking application. The detections at ‘b’ seem to form a track, although no track is present here in either the industrial model or the PGM outputs.	121
A.1	Simple example of a Bayes Network	132
B.1	GM-PGM vs MM-PGM performance comparison example showing only the detections.	151
B.2	A comparison of tracking accuracy between the MM-PGM and GM-PGM models. All the objects in this example start in the center of the figures and move outward. Note that the track identities is not correct (consistent) in (a), whereas they are in (b). This shows the advantage that the GM-PGM has over the MM-PGM. The scanning radar simulator (without angular velocity measurements) was used in this example. The true tracks in these examples are indicated by solid dark lines, the track estimates by the colour dashed lines and the detections by the black circles.	152
C.1	Two time-slice Bayes network of MOT PGM with clutter classification. The e_i^t and g_i^t random variables represents whether or not a detection is expected and has been generated from a target at a specific time step respectively. The r_i random variable corresponds to whether the target is real or not.	153
C.2	Example of a causal clutter classification cluster graph (attached to the multiple object tracking graph), corresponding to a single time slice graph of the Bayes network in Figure C.1 for three targets and two measurements.	156

List of Tables

1.1	Example of a multivariate categorical distribution	5
2.1	MHT filter hypothesis table (corresponding to the tree in Figure 2.1). The disallowed association hypothesis combinations are marked in red.	22
A.1	Message $\delta(a_0)$ from Y_0 cluster to association cluster	146
A.2	Message $\delta(a_1)$ from Y_1 cluster to association cluster	146
A.3	Association cluster $\psi(a_0, a_1)$ posterior marginal showing the model hypotheses as products of compatible association hypothesis probabilities, where the individual association probabilities were contained in the messages.	146

Nomenclature

Acronyms

CNLG	Conditional Non-linear Gaussian
CPD	Conditional Probability Distribution
DLT	Discrete Log Table
EMDW	Elementary My Dear Watson (Stellenbosch PGM library)
GM	Gaussian Mixture
JPDA	Joint Probabilistic Data Association (Filter)
LBU	Loopy Belief Update
LBP	Loopy Belief Propagation
LTRIP	Layered Trees Running Intersection Property
MM	Moment Matching
OSPA	Optimal Subpattern Assignment
PGM	Probabilistic Graphical Model
PHD	Probability Hypothesis Density (Filter)
RIP	Running Intersection Property
MOT	Multiple Object Tracking
MHT	Multiple Hypothesis Tracking (Filter)

Symbols

a	association random scalar
A	association random vector containing individual a 's
\mathbf{A}	linear transformation matrix
\mathcal{N}	Gaussian (normal) distribution
\mathcal{C}	canonical form of the Gaussian distribution
e	<i>whether or not a target is expected to generate a detection</i> random variable
g	<i>whether or not a target is generated a detection</i> random variable
r	<i>whether or not a target a target is real</i> random variable
ψ	factor/potential function in a cluster graph
ψ'	updated factor/potential: cluster belief
X	target/object state random vector
Y	measurement random vector
X_i^t	object i state random vector at time step t
Y_j^t	measurement j random vector at time step t
\mathbf{x}_i	target/object state sigma point
\mathbf{y}_i	measurement sigma point or observation
$\delta_{X,X}$	message between connected state clusters of the same object
$\delta_{X,Y}$	message from state to measurement cluster
$\delta_{Y,X}$	message from measurement to state cluster
d_X	number of dimensions of the state distribution $P(X)$
d_Y	number of dimensions of the measurement distribution $P(Y)$
$\hat{\Sigma}$	sample covariance matrix
Σ	Gaussian covariance matrix (covariance form parameter)
μ	Gaussian mean (covariance form parameter)
w	Gaussian weight scalar (covariance form parameter)
\mathbf{K}	Gaussian precision matrix (canonical form parameter)
\mathbf{h}	Gaussian h vector (canonical form parameter)
g	Gaussian g scalar (canonical form parameter)
\mathbb{H}	Hessian matrix
Δ_t	short time interval
\mathbf{L}	lower triangular Cholesky factor of a matrix

Mathematical Notation

scalars ¹	lower-case letters
random vectors	upper-case letters
vectors	bold lower-case letters
matrices	bold upper-case letters
$[B, C]$	vertical concatenation of two vectors B and C

¹The distinction between scalar variables and random scalar variables should be clear from context.

Definitions

A/a node	An association node (in the MOT Bayes network).
A/a cluster	An association node (in the MOT cluster graph).
belief	(cluster belief) the cluster potential updated by messages.
cardinality	The number of possible values a discrete random variable can have.
cluster-slice	A collection of clusters that all correspond to the same time-step.
clutter	The same as false detections.
false detections	Detections that do not correspond to objects of interest.
factors	The factors (conditional and prior distributions) of a joint distribution.
measurement	A measurement (or detection) of a unobserved process/state.
measurement cluster	The same as a Y cluster.
objects	A physical object (or target) that can be tracked.
partial measurement	A measurement that contains less information than the true state.
potential	An unnormalised non-negative function (unnormalised distribution).
process dynamics	The function that describe the process/target-/object state evolution.
process noise	Noise that represents uncertainty about the process dynamics.
$P(\alpha \beta)$	The probability of α conditioned on a random variable β .
$P(\alpha; \beta)$	The probability of α conditioned on a constant parameter β .
$P()$	probability mass/density function, also probability distribution.
scope	The set over variables over which a function is defined.
state cluster	The same as an X cluster.
time-slice	A part of a Bayes Network with variables with the same time index.
X cluster	A state cluster (in the MOT graph) containing only X variables.
X node	A state node (in the Bayes network).
Y cluster	A measurement cluster (in the MOT graph) containing a Y variable.
Y node	A measurement node (in the MOT Bayes network).

Chapter 1

Introduction

1.1 Motivation

Probabilistic graphical models (PGMs) provide a framework for representing high-dimensional probability distributions as highly interpretable graphs that are used as a basis for performing efficient inference. PGMs have successfully been applied to problems in a wide variety of fields. Some of these include medical and fault diagnosis, image segmentation and de-noising, noisy channel message decoding, and applications in genetics [1, p. 12]. In this work we set out to apply PGMs to the problem of multiple object tracking in order to confirm that it is possible to do so successfully. We also seek to investigate whether any improvements can be made over current methods. In the course of the study it was found that there have indeed been a small number of previous attempts to cast the multiple object tracking problem in a graphical model framework. This work is discussed in the literature study.

The problem of tracking the state of a single object from noisy measurements is inherently a probabilistic one, and has been solved¹ for over 50 years, following the publication of the seminal paper describing the Kalman Filter in 1960 [2]. Since then there have been many variations to the standard Kalman filter, such as the unscented and extended Kalman filters, which can be used for tracking in systems with non-linear measurement models. The Kalman filter and its variants can be exactly interpreted as graphical models. The problem of tracking multiple objects simultaneously is, however, far more complex than the single object case. Nevertheless, there have, over the years, been many different approaches to solving this problem. The joint probabilistic data association (JPDA), multiple hypothesis tracking (MHT), and probabilistic hypothesis density (PHD) filters are some of the most well-known algorithms in use today. These algorithms are used for various applications, including autonomous vehicles, robotics, and air-traffic control systems [3, p. 1]. Multiple object tracking is therefore an important problem, and effective solutions to this problem have far-reaching applications.

¹Under certain simplifying conditions, such as linear dynamics and linear measurement transforms.

Although the vast majority of well-known multiple object tracking (MOT) algorithms, including those mentioned above, are based on sound probability theory, none of them are formulated as graphical models. This therefore presents an opportunity to develop a PGM for multiple object tracking and to compare it to the current state-of-the-art methods.

1.2 Background

1.2.1 Multiple Object Tracking

The problem of object tracking is to continuously and accurately estimate the state of an object from noisy measurements as the state evolves over time and new measurements are received. The object state typically includes position and velocity variables, but can be any property or collection of properties of an object or system. In this work we will, however, focus on tracking basic moving objects. The object states will therefore comprise position and velocity variables. The single object tracking problem can be solved optimally in the linear Gaussian case with Kalman filter algorithm and approximately in the non-linear Gaussian case with variants of this algorithm, such as the unscented Kalman filter. The state estimates received as the output of object tracking algorithms are the assignments to the state random variables that maximise the posterior state distributions. These estimates are known as the maximum a posteriori (MAP) assignments.

The problem of multiple object tracking (MOT), as the name suggests, extends the single object tracking problem to tracking multiple objects. The main complication in multiple object tracking is the problem of data association, where we do not know which detection should be associated with which object. In certain applications, such as tracking objects in video sequences - where objects have distinct features - this might not be a problem. In other applications such as radar tracking, where the objects appear practically identical, it is complex and crucial to solve the data association problem. The problem of data association can be further complicated by missed detections, where a target does not always generate a detection, and false detections, where received detections do not correspond to any target we wish to track. Missed detections can be due to object occlusions, which can occur when targets move behind hills, or rocks, for example, or due to changing sensor orientation. In this work we will encounter both of these phenomena, with the second type being due to the nature of a scanning radar, which rotates constantly and therefore only periodically sees targets. False detections (also known as clutter), on the other hand, can be due to electrical noise in the sensor, or real-world objects, which we are not interested in tracking, such as trees moving due to wind.

In order to solve multiple object tracking, we can therefore use the theory of single object tracking in combination with a solution to the data association problem. There have been many different approaches to the multiple object tracking problem and some of these are discussed in the literature synopsis below and in the literature study in Chapter 2. Since there are so

many algorithms, it is necessary to have a sensible measure of performance in order to compare them. In single object tracking, comparing the performance of two algorithms is relatively straightforward, and the average squared error between the estimates and the true tracks is often used. In the multiple object tracking, case performance comparison is a problem in its own right. Various metrics have been proposed for this purpose, with the optimal sub-pattern assignment (OSPA) metric [4] being one of the most widely used. The OSPA metric gives us a means of calculating a distance between sets and is therefore appropriate when comparing the estimates from different models. A multiple object tracking model will output a set of estimates, one for each object, at each time step. Since the number of objects that are tracked might differ between models (due to errors in the estimation of the true number of targets), it is important to be able to measure the distance between two sets of different sizes. The OSPA metric allows us to do exactly that and it will yield a larger distance the further the points in the two sets are from each other and/or the greater the size difference between the two sets.

1.2.2 Probabilistic Graphical Models

Probabilistic graphical models give us a general framework for modelling problems involving probability in an interpretable manner and to perform efficient inference in order to solve these problems. There are various types of graphical models, with the Bayes network being one of the most prominent. Bayes networks allow us to model probabilistic problems by using graphs where the nodes represent random variables and the edges describe the dependencies between variables. Bayes networks therefore allow a joint distribution to be represented as a product of conditionally independent factors. Bayes networks are a useful tool to aid in the development of a graph structure, although other graphs and/or algorithms are used for inference.

Probabilistic inference is often concerned with calculating marginal distributions over variables and the operations required for this type of inference are multiplication (for calculating joint probabilities, using the chain rule) and summation or integration (for calculating marginals). The simplest inference algorithm is arguably the variable elimination algorithm. This algorithm allows efficient inference through efficient ordering of the operations that are required for inference. Other, more advanced algorithms, such as the loopy belief update (LBU) and loopy belief propagation (LBP) algorithms, are based on this algorithm. These algorithms are used for inference in cluster graphs and allow the marginal distributions of all the variables in the graphs to be calculated efficiently. The LBP and LBU algorithms belong to a family of algorithms that are known as message passing algorithms. The term ‘message’ here refers to a non-negative function containing information that is not available in other parts of the graph. These graphs are constructed by connecting nodes, which represent initial probability distributions² and are known as potentials. Message passing algorithms therefore allow information to be propagated through graphs in order to update the cluster beliefs (the functions represented by the cluster nodes) with new information. The information contained in these messages is due to the prior distributions specified over certain variables, the relationships specified between variables, and

²These can be prior or conditional distributions, or the product of multiple distributions

the data that is observed. After message passing is complete, the normalised cluster beliefs represent the marginal distributions of the posterior distribution. These marginal distributions therefore correspond to an updated beliefs in relation to the values that the variables can likely have, given the data that was observed.

The theory behind probabilistic graphical models is independent of any specific type of random variables. The type of random variables that are used in a graphical model should be determined by the nature of what the variables are modelling. In practice, however, the choice of random variables and their corresponding distributions that are suitable for inference in graphical models is often limited to a small set of distributions. These distributions must typically have a closed form under the operations required to perform inference. One such type of distribution is the multivariate Gaussian distribution.

An important distribution that is closely related to the Gaussian distribution is the linear Gaussian distribution. This distribution defines a linear relationship between two Gaussian random variables and is widely used, for example, in methods for performing inference within linear dynamic systems, such as the Kalman filter. The linear Gaussian distribution is a conditional probability distribution (CPD) that has an extruded, Gaussian-like shape (see Figure 1.1 below). The product of a linear Gaussian distribution and a Gaussian distribution is a joint distribution that is also Gaussian. This property, together with the other properties of the Gaussian distribution, ensures that all messages that are sent in linear Gaussian networks will have a Gaussian form. The linear Gaussian can be generalised to allow for non-linear relationships between variables, although the resulting joint distributions in this case will not be Gaussian and will therefore need to be approximated as such. This generalisation does not have an official name, but will be referred to in this work as a ‘non-linear Gaussian’. The closed form characteristic of distributions under the operations required for inference is crucial in an automated inference application. If the form of a distribution changed constantly, the necessary software implementations would be required for all possible forms. Furthermore, the integrals of products of different forms might not have analytical solutions. Such an implementation would therefore be infeasible or impossible, and approximations or sampling methods are therefore used to prevent this problem. In this work we will thus use approximations when necessary.

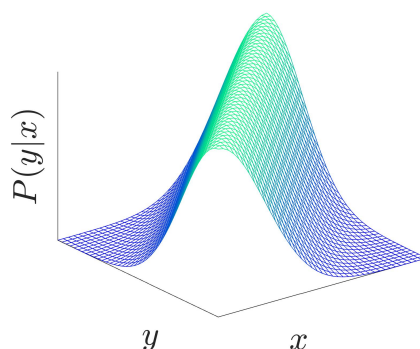


Figure 1.1: Example of a two-dimensional linear Gaussian CPD. Note that, for a given value of x , this distribution reduces to an ordinary Gaussian distribution.

Another widely used type of distribution is the multivariate categorical distribution. This is a discrete distribution that can be represented as a table factor, as shown in the example in Table 1.1 below. In this example, the table factor represents a conditional probability distribution $P(b|a)$. In general, these distributions can have any number of variables and can be either conditional or joint distributions. The random variables can also have any cardinality and are not limited to binary random variables, as in the example below. The multivariate categorical distribution will be referred to simply as a discrete distribution in the rest of this text.

a	b	$P(b a)$
0	0	$P(b = 0 a = 0)$
0	1	$P(b = 1 a = 0)$
0	1	$P(b = 0 a = 1)$
1	1	$P(b = 1 a = 1)$

Table 1.1: Example of a multivariate categorical distribution

The linear Gaussian distributions and the discrete distribution can be combined, in a sense, to form a conditional linear Gaussian distribution. This distribution can also be represented as a table factor, but instead of constant probabilities at every entry, every assignment of the discrete random variables has a corresponding linear Gaussian distribution. When multiplying this distribution with a Gaussian distribution, every table entry is multiplied by the Gaussian distribution and therefore contains a Gaussian distribution after the multiplication. Subsequent marginalisation over the discrete random variables therefore results in a Gaussian mixture distribution. The conditional non-linear Gaussian is a generalisation of the conditional linear Gaussian distribution that defines a non-linear Gaussian distribution for each corresponding assignment to the discrete variables.

1.3 Literature Synopsis

The problem of multiple object tracking has been studied for over 50 years [3, p. 1], with many different solutions proposed over the years. The probability hypothesis density (PHD) and multiple hypothesis tracking (MHT) filters are two of the most prominent solutions in use today. The PHD filter, first proposed by Mahler [5] in 2003, uses random finite sets to approximate the optimal Bayesian multi-target filter. This filter does not solve the data association problem and can therefore only produce point estimates without track identities. The initial PHD filter proposed by Mahler involved integrals to which no closed form solutions exist, and sampling methods have therefore been used in practical implementations of the PHD filter. In 2006, Vo et al. [6] proposed a closed form solution for the linear Gaussian case. This variation is known as the Gaussian mixture PHD (GM-PHD) filter. The GM-PHD has in recent years been widely used in various applications [7]. In contrast to the PHD filter, the MHT filter developed by Reid [8] in 1979 does solve the data association problem and can therefore produce tracks of individual objects. This method is widely regarded as the optimal algorithm for tracking objects in the presence of missed detections [9]. There are, however, many more types (and variations on these types) of algorithms that have been devised for multiple object tracking. In 2005, Pulford [10] presented a summary of multiple object tracking algorithms in which 35 algorithms were considered. There is therefore a significant amount of literature available on multiple object tracking.

In comparison to the literature on traditional multiple object tracking, the literature on graphical model approaches to the multiple object tracking problem seems very limited. One of the works found on this specific topic is a paper published by Segal and Reid [11] in 2013 that frames the multiple object tracking problem as a PGM. The PGM described in this paper solves the data association problem in a different manner than in the work we present here. The authors propose a latent data association model that associates state nodes in adjacent time slices with each other instead of performing state-measurement associations. Another work in which a PGM based multiple object tracker is discussed, is a Master's thesis by Chen [12]. The model developed for this thesis assumes that the number of targets, detection probability and clutter density are known [12, p. 87]. Other work done on PGM based object tracking relates to tracking from video sequences. One such work is the PHD thesis by Schiegg [13] in which a PGM is developed for image segmentation and multiple object tracking and applied to biological cell tracking. Because the models presented in these studies typically involve image segmentation and/or classification and are significantly different from the type of multiple object tracking discussed in this work, these studies will not be discussed here.

A more detailed discussion on the various approaches to multiple object tracking algorithms can be found in the literature study in Chapter 2.

1.4 Objectives

The objectives of this work are to:

- Study and discuss the problem of multiple object tracking and the existing solutions to the problem.
- Establish whether and how probabilistic graphical models can be used for multiple object tracking (although it was later discovered that some work on this has indeed been done previously).
- Give a comprehensive overview of some of the methods and concepts relating to inference in probabilistic graphical models, especially those methods and concepts relevant to solving the multiple object tracking problem.
- Present the findings and results obtained by comparing the developed model to a state of the art method and an industrial application.
- Discuss possible future improvements to the developed model.

1.5 Contributions

In this work a probabilistic graphical model is developed to solve the multiple object tracking problem. While the literature on the MOT problem and traditional (non-PGM) algorithmic solutions is extensive, very few sources are available on probabilistic graphical models based solutions to the problem³. Furthermore, many of these models draw inspiration from the traditional, non-PGM, algorithms. While this might be sensible, it is our view that the model functionality should arise naturally from the structure of a logically constructed PGM. This work therefore strives to develop a logically sound graphical model and to investigate the functionality that arises naturally from it.

Since all the functionality required to implement the proposed MOT PGM was not available in the EMDW library (or any other library to the best of our knowledge), the EMDW library needed to be expanded. The idea with this expansion was to write additional factor classes in a highly object-oriented manner that is consistent with the existing code base. The added classes are therefore suitable for general use in graphical models and not limited to the multiple object tracking problem. The functionality that was added to the EMDW library is as follows⁴:

³Of course, any probabilistic model can be framed as a PGM, but the details of such PGM interpretations are often somewhat obscured by more abstract algorithms and some of the advantages of the PGM approach are lost in the process.

⁴The basic structure of some of the existing classes were used in the creation of these classes. Some of the code is therefore identical to that of the these existing classes. For the classes where a large portion of the code

- Gaussian class (adapted from the existing *gausscanonical* class)
- Non-linear Gaussian class
- Gaussian mixture class
- Conditional non-linear Gaussian class
- Factorised factor class
- Discrete log table class (adapted from the existing *discretetable* class)

The factorised factor class allows for the natural and efficient representation of products of factors. This functionality is especially useful when multiplication of independent factors is required during message passing. In this case it is unnecessary to explicitly multiply the factors, and the factorised factor class allows them to be kept in a factorised form while also supporting any subsequent operations on the (factorised) factor. This class supports Gaussian, Gaussian mixture, or non-linear Gaussian factors, or a combination of these as factorised components.

During this study and the development of the multiple object tracking model, a few valuable insights were gained. These insights are presented in detail in the corresponding sections throughout this work and are summarised as follows:

- A logical, graphical model-based design process for the development of a graphical model for multiple object tracking.
- Interesting message passing characteristics:
 - The effect of the means and variances when multiplying Gaussian distributions (see Section 5.3)
 - How the association variable(s) causes the state posterior to be a superposition of the prior and measurement updated distribution (see sections 6.1 and 6.2). This explanation has not been found in any of the available literature.

Furthermore, some of the methods and concepts presented in this work are believed to be novel. These contributions are as follows:

- An **alternative view on model selection** that allows model selection to be performed efficiently during message passing inference in a single graph was formulated in this work. This effect was later found to be mentioned in the paper by Segal and Reid [11], but is not described in any detail and it has not been found to be mentioned in any other

remains unchanged, this is indicated by "adapted from". The other classes have little functional code in common with the classes that they were adapted from.

literature. In this work, the effect is described in detail. Here it is shown mathematically (see Section 7.1) that inference in graphs, with association type discrete variables, can correspond exactly to classical model selection. Furthermore, this type of model selection was implemented in the MOT PGM and shown to work in practice.

- The **clutter classification** graphical model developed as part of this work does not require the clutter density or detection probability to be known. This is in contrast to the majority of established MOT algorithms that are capable of tracking through clutter, where these parameters are required for accurate tracking. Although some models have been developed in the past that allow the clutter intensity and detection probabilities to be estimated [14], these approaches are very different to the approach presented in this work. The clutter classification model forms part of the larger MOT PGM and was shown to work very well in practice (see Section 10.3).
- An algorithm for performing **approximate Gaussian mixture division** was developed to allow for Gaussian mixtures to be used in the LBU message passing algorithm (where division operations are required). The quotient of Gaussian mixtures is not typically a Gaussian mixture but can often be approximated as one. The developed algorithm allows for quotients of Gaussian mixtures to be accurately approximated by Gaussian mixtures. Such an algorithm has not been found anywhere in the literature. Furthermore it has been shown that the use of Gaussian mixtures (as opposed to the single Gaussian approximation) can increase tracking accuracy under certain conditions (see Chapter 8).

In summary, while the use of PGMs has in the past (in a very small number of studies) been proposed to solve multiple object tracking, we believe that the model developed in this work is at least partially novel. Furthermore, this work contributes to the literature in terms of the detail of explanation and the approach that was taken to develop a multiple object tracking PGM.

1.6 Overview

In this work, Bayesian networks are used as a modelling tool to design logical structures for graphical models relating to the problem of multiple object tracking. A prior belief of the possible values of the object's state random variables is used to define prior distributions, and the relationships between variables are used to construct the conditional probability distributions. These two types of distributions are used to construct the factors that are represented by the nodes of the cluster graph. Throughout this work, the loopy belief update (LBU) algorithm will be used for inference in cluster graphs. The construction of cluster graphs and the operation of the LBU algorithm is discussed in detail in Chapter 3.

Throughout this work, the multivariate Gaussian distribution (which will be referred to simply as the Gaussian distribution) will be used to model the distributions of continuous vari-

ables. The Gaussian distribution has various representational forms. These are the canonical, covariance, and information forms as well as what can be regarded as a fourth form, the sigma point representation. These forms can be converted between one another, and each has specific properties that allow certain operations to be performed more naturally and efficiently. These representations and the multivariate Gaussian operations required for inference are discussed in Chapter 4.

The multivariate Gaussian and the related linear Gaussian CPD are important mathematical tools that make exact inference possible in linear Kalman filter models as well as linear Gaussian networks in general. From a graphical model perspective, the linear Gaussian distributions are the conditional distributions that (together with some prior distributions) define the factors in the graph. The prior over the initial state of an object is multiplied with the corresponding CPD to form a joint distribution over the initial state and the next state (the state at the next time step). This joint distribution can be updated by noisy measurement information (in the form of a Gaussian distribution) to form a posterior distribution. The posterior marginal over the next state can now be used as a prior (before any measurement information about the state at this time is received) and the process can be repeated. This procedure can be viewed as the operation of a Kalman filter or as message passing in a cluster graph. The relationship between these two views is discussed in Chapter 5.

The linear-Gaussian Kalman filter (and its graphical model interpretation) is, however, severely limited in that it can only accommodate linear transformations. The non-linear Gaussian distribution extends the possible transformations to any non-linear transformation. This functionality is crucial in some applications where the relationships between variables are inherently non-linear, as is the case in radar tracking, for example. The polar space in which the radar detections are naturally represented has a non-linear relationship to the Cartesian space in which tracking is performed. One method of performing approximate non-linear random variable transformations is the unscented transform. Using this method, a Gaussian distribution is first represented by a set of deterministic sample points called sigma points. These points can then be transformed by any non-linear function, and the transformed points can be used to calculate a Gaussian approximation of the transformed distribution. The quality of this approximation will depend on the severity of the non-linearity of the transformation. The quality of the approximation can of course also influence inference accuracy, and it is therefore important to ensure that the use of a specific transformation does not result in grossly inaccurate approximations. The accuracy of the approximations resulting from the Cartesian-polar transformation is discussed in Section 4.3.

With the use of the unscented transform in a Kalman filter, or graphical model, we have the necessary mathematical tools to perform single target tracking from radar measurements. Although this model cannot track multiple objects, it does have functionality that will be useful in multiple object tracking. This model is therefore used as a starting point to develop a multiple object tracking PGM. The central problem in multiple object tracking, which is typically not relevant in single object tracking, is the data association problem. We approach this problem, once again, from a graphical model perspective. Here we notice that the state distribution of an

object should be updated by a measurement if the object is associated with the measurement. Whether or not a certain association exists can be modelled by a discrete random variable. The generated detections will therefore be dependent on the states of the various objects that are being tracked and on which objects caused the detection. These dependencies can be intuitively modelled as a Bayes network. An example of such a network, with two object state random variables, a measurement random variable and an association random variable, is shown in Figure 1.2 below.

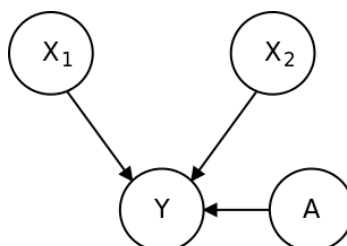


Figure 1.2: Data association problem represented as a Bayes Network. The measurement (Y) is dependent on the two object states (X_1 and X_2), as well as which state caused the measurement (encoded by the association random variable A).

This network is known as a hybrid network, since it contains both discrete and continuous random variables. The measurement factor in Figure 1.2 is an example of a conditional non-linear Gaussian distribution, since it is dependent on both a discrete random variable and continuous Gaussian random variables (through a non-linear transform). The form of the measurement factor causes inference in this model to be much more complex than in the normal Kalman filter model. This is due to the fact that marginalising over the discrete variables of this factor, during message passing, will result in a Gaussian mixture. Furthermore, when Gaussian mixtures are multiplied together the number of components grow exponentially. It will therefore be necessary to periodically reduce the number of mixture components in the Gaussian mixtures generated during message passing. The Gaussian mixture reduction methods and other operations that are required for inference are discussed in Chapter 8.

Since the message passing procedure and the distributions are somewhat complex in the hybrid network, a mathematical example and a corresponding visual example of the process are given in sections 6.1 and 6.2 respectively. In these chapters it is also shown how the association variables have a surprisingly interpretable and logical effect in the graph. These variables, together with the structure of the graph and the measurement factors, cause the posterior over a state of an object to be a superposition of being updated and not updated. In this posterior Gaussian mixture distribution, the weight of the updated component is proportional to the probability that the object is associated with the measurement, and the weight of the non-updated component to the probability that it is not. This is a quite intuitive result that arises naturally from the structure of the graph and the types of distributions therein.

An assumption made in the early stages of development (and in the examples in the aforementioned chapters) is that the number of targets is known and unchanging. These assumptions would typically need to be relaxed for an MOT model to be useful in practice. Therefore, in order to allow the model to accommodate the addition of new objects, Bayesian model selection is used. Using this method, the relative probabilities of the various candidate models, given the measurement data, are compared in order to find the most probable model. This is done by building and performing inference in separate graphs that represent different hypotheses about the number of targets and their associated measurements. This process is discussed in detail in Section 6.3. The PGM developed up to this point is capable of tracking an unknown and changing the number of targets, but has no mechanism for rejecting false detections. To solve this problem, a graphical model is designed in order to allow the PGM to classify the clutter tracks that arise due to false detections. This additional functionality allows the improved model to accurately track an unknown and a changing number of targets, with an unknown detection probability, in the presence of false detections with an unknown density.

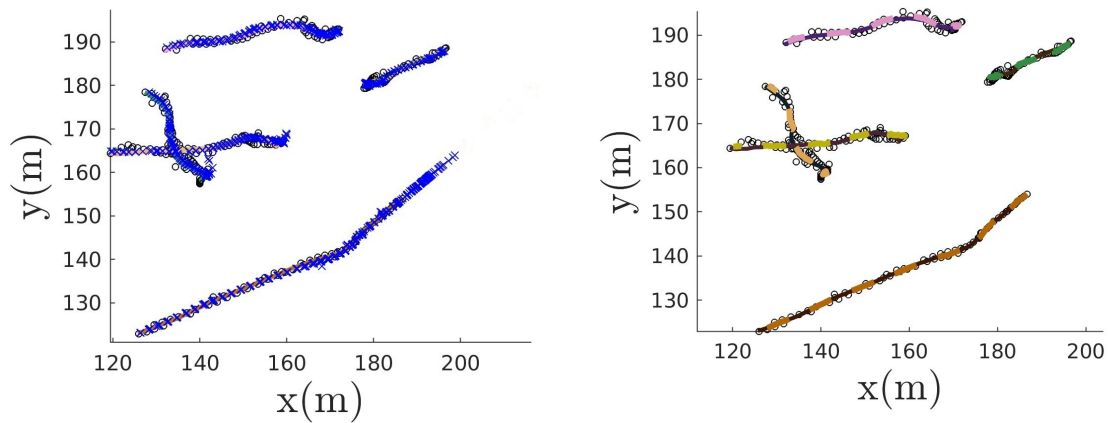
The improved model with clutter classification performs well, but is still very computationally expensive to operate, especially when tracking a large number of targets in clutter. To address this problem, the most computationally expensive sub-routine (model selection) is re-considered. Here the observation is made that the discrete association variables in the graph, which allow the graph to reason about the target-measurement associations, already perform a function that is very similar to model selection. The relationship between model selection and the association variables is investigated in Section 7.1. By taking a closer look at the mathematical expressions for the association cluster belief after message passing, we see that this function is exactly a (unnormalised) distribution over different models. It is therefore shown that the classical Bayesian model selection process can be reformulated in terms of inference in a single graph. The new method of model selection is both simpler and less computationally expensive than the implementation of classic Bayesian model selection.

In order to implement the developed graphical model, the EMDW library (the PGM library belonging to the University of Stellenbosch) was used. Most of the functionality required for the developed model was, however, not available in the library. In order to implement the models described in this work, the EMDW library therefore needed to be expanded. To this end, classes for linear and non-linear Gaussian, conditional non-linear Gaussian, Gaussian mixture and factorised factors were implemented. These classes allow the EMDW library to support hybrid networks that contain both discrete and continuous variables. The new functionality is therefore useful for PGM implementations in general and is not limited to multiple object tracking specifically. The implementation of the additional classes is discussed in Chapter 9.

The developed model can be used with detections from any type of sensor, with minor modifications. Only the relevant state-measurement space transformation and sensor noise will need to be specified in order to allow the model to perform tracking with input from an arbitrary sensor. The model can also in future be expanded, with minimal effort, to allow inputs from multiple identical or different types of sensors. In this work, however, the model is configured to receive detections from a single scanning frequency modulated continuous wave (FMCW)

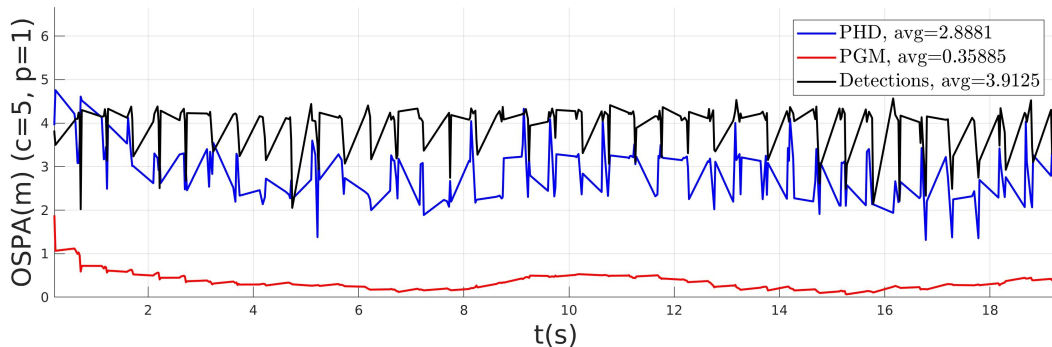
radar. The detections received from the radar contain range, azimuth (horizontal angle) and radial velocity measurements. In order to test the developed model, a scanning radar and object simulator was developed. This simulator allows detections to be generated from any specified number of randomly manoeuvring objects as well as for false detections to be generated at a specific rate. The simulator also accounts for missed detections due to the radar orientation. The simulator, therefore, allows us to compare the model's track estimates to the ground truth tracks of the various targets. The testing of MOT models is a rather difficult process, since it is impossible in practice to obtain the ground truth tracks. The simulator is therefore a valuable tool for quickly and accurately assessing model performance.

The performance of the developed PGM is further compared to a state of the art algorithm, the Gaussian mixture probability hypothesis density (GM-PHD) filter, in Chapter 10. It is noted that the GM-PHD is known to suffer from poor performance under operation with subsequent missed detections (a common occurrence in a sweeping radar application) and this is reflected in the results. The developed PGM, however, is shown not to share this flaw and outperforms the GM-PHD under these conditions. The graphs in Figure 1.3a and Figure 1.3b below show examples of track estimates from the GM-PHD and the developed PGM model, respectively. Figure 1.3c shows the OSPA distances, from the ground truth tracks, for the GM-PHD and the developed PGM respectively and illustrates the advantage that the PGM offers over the PHD when there are multiple missed detections. Another advantage that the PGM (and other methods that solve the data association problem) has over the PHD filter is that it can provide tracks that clearly distinguish different objects, while the PHD filter only provides general point estimates.



(a) PHD estimates. Note that there are only point estimates, without track identities. Estimates (blue x's). True tracks (lines - mostly covered by estimates). Detections (black o's)

(b) PGM estimates. Note that there are estimates for each individual track (indicated with different colours). Estimates (light dashed lines). True tracks (dark lines). Detections (black o's)



(c) PGM vs GM-PHD OSPA distance comparison

Figure 1.3: PGM vs GM-PHD comparison (with simulated data) showing the relatively poor performance (compared to the PGM) of the PHD filter when there are multiple missed detections. The PHD OSPA (as well as that of the detections) also fluctuates erratically with the missed detections, while the performance of the PGM is far more consistent.

Lastly, the developed model is tested on real radar data and the output is compared qualitatively to the output of an industrial MOT filter implementation. The outputs of the two models can, however, not be analysed quantitatively, as there are no ground truth tracks available to accompany the detection data. The track estimates from the industrial model and the developed PGM are shown in Figures 1.4 and 1.5 respectively.

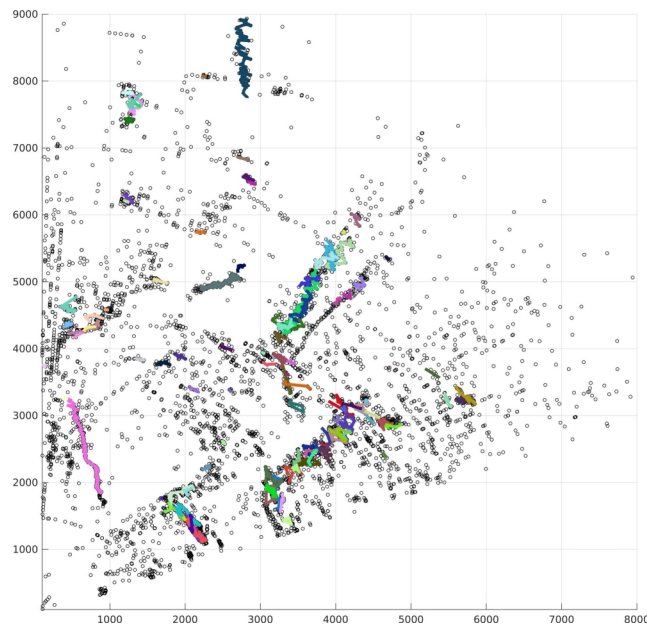


Figure 1.4: Industrial model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)

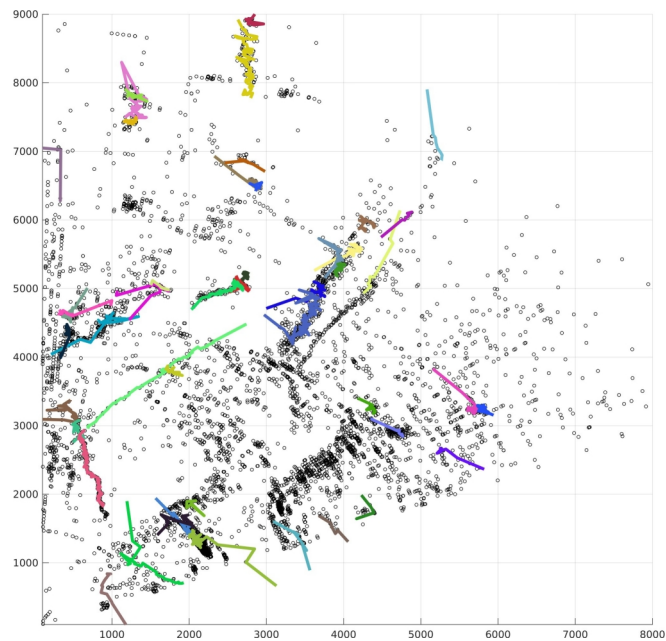


Figure 1.5: PGM model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)

From Figure 1.4 and 1.5 above, we can see that the tracks are quite similar, although there are some tracks that are only present in one of the two outputs. For the tracks that are present in both models, we can be reasonably confident that they are accurate estimates of the tracks of real targets. For the tracks that are present in only one of the two outputs, it is, however, impossible to assert which model's output is correct. Possible explanations for some of the discrepancies are discussed in Section 10.4. Comparing these two figures, and assuming the detections that the industrial model has ignored are really false detections, we can also see that the clutter classification in the developed PGM works well, even in this real world example. Figure 1.6 below shows a close-up view of one of the track estimates for a clearer comparison.

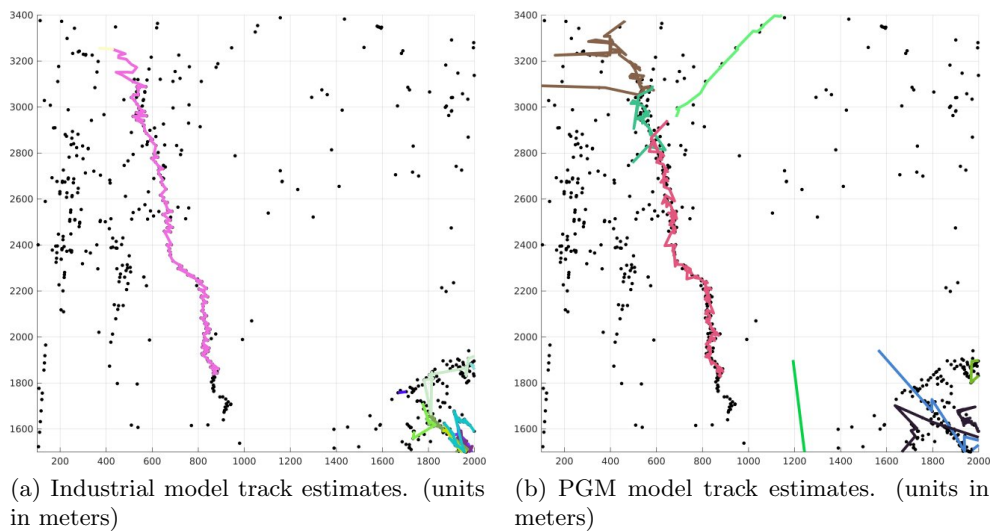


Figure 1.6: A close-up example of one of the tracks from figures 1.4 and 1.5 in (a) and (b) respectively showing the relative similarity between the estimates from the PGM and the industrial model. Although some tracks are only present in one of the outputs, the longest, most prominent track is more or less covered by both models. The detections interpreted as clutter are mostly consistent between the two models.

We conclude that the developed model is capable of accurately and efficiently tracking an unknown and changing number of objects in the presence of an unknown clutter density and unknown detection probability (see Section 10 for the full set of results). We have therefore shown, as we set out to do, that probabilistic graphical models can effectively be used for multiple object tracking, and demonstrated that the developed model gives comparable results to that of an industrial model in a real-world application. Furthermore, we have shown that the developed PGM outperforms the GM-PHD filter under certain conditions. Lastly, since the developed model is framed in terms of a graphical model rather than an abstract algorithm, this model could more easily serve as a foundation on which additional functionality can be added in future work.

Chapter 2

Literature Study

The problem of multiple object tracking (MOT) is concerned with estimating the true positions of multiple targets from unlabelled noisy measurements of the target states. This problem is typically made more challenging by partial measurements, false detections, missed detections, and the true number of targets being unknown. It is therefore a complex problem to solve and one that has been studied for over half a century. Over the years, various solutions have been proposed, each with their own advantages and disadvantages. Today, multiple object tracking is used in many different applications, including air traffic control systems, autonomous vehicles, robotics and biomedical research [3, p. 1]. In this chapter, we will look at some of the most prominent solutions to the multiple object tracking problem, and discuss their relative advantages and disadvantages. We will also discuss the previous work that has been done on PGM-based solutions to the multiple object tracking problem. The mathematical details of the operations and Gaussian distributions which are typically used in these methods are described in Chapter 4. The Kalman filter algorithm, which is closely related to these methods, is discussed in detail in Chapter 5.

2.1 Classical Multiple Object Tracking Algorithms

In this section, some of the most prominent classical (non-PGM-based) solutions to the multiple object tracking problem are discussed. These methods fall into one of two categories, association-based and association-free methods.

2.1.1 The Probability Hypothesis Density Filter

The joint multi-target probability distribution (JMPD) filter is a theoretically optimal solution to the problem of multiple object tracking. This method is, however, computationally infeasible,

even for single object tracking [5, p. 1]. The JMPD filter, as the name implies, models all state distributions together in a single joint probability distribution [15, p. 2]. The JMPD does not solve the data association problem and is therefore known as an association-free method. Since this model is intractable for even relatively simple cases, it must generally be approximated by some means.

The probability hypothesis density filter (PHD) filter proposed by Mahler [5] is an approximation to the JMPD filter that is analogous to the constant gain Kalman filter in that it only propagates the first order moment of the multi-target joint distribution [5]. This first order moment is called the probability hypothesis density or intensity function [16, p. 84]. The underlying theory behind the PHD filter involves random finite sets (RFS) and Mahler's multitarget calculus. Random finite sets are a generalization of random variables in the sense that they represent sets of random variables where the size of the set is itself random [17, p. 32]. A detailed discussion of random finite sets and the multitarget calculus is, however, beyond the scope of this work. Although the general PHD filter updates involve integrals that have no closed form solution [18, p. 1], it was shown by Vo and Ma [19] that a closed form solution can be obtained under linear Gaussian dynamics. This variant of the PHD filter is known as the Gaussian mixture PHD (GM-PHD) filter.

Furthermore, unlike the JMPD filter, the PHD filter operates on a single-target state space [16, p. 96], and the dimensionality of the state distributions therefore does not increase with the number of targets. The PHD filter essentially avoids the association problem by considering all association hypotheses and updating each state with all possible measurements (including no measurement) [20, p. 2]. If there were therefore n states and m measurements, the number of updated states would be $n(m + 1)$. Avoiding the association problem allows the PHD filter to be less computationally expensive than association-based methods such as the MHT filter. This is, however, also the cause of the PHD filter's biggest shortcoming: it cannot produce tracks with target identities. In addition to the process and measurement parameters required by most multiple object tracking methods, the PHD filter requires certain additional parameters to be specified, namely the target detection probability, target survival probability, and clutter intensity. The PHD filter state and measurement update equations are given below.

The PHD Filter Equations [16, p. 96]

State update:

$$\mathcal{V}_{k|k-1}(X) = p_S \int f_{k|k-1}(X|\zeta) \mathcal{V}_{k-1}(\zeta) d\zeta + \gamma_k(X)$$

where

$$\begin{aligned} p_S &= \text{target survival probability,} \\ \gamma_k(X) &= \text{PHD of the birth RFS } \Gamma_k \text{ at time } k, \\ \mathcal{V}_{k-1}(\zeta) &= \text{prior (previous posterior) state PHD,} \\ f_{k|k-1}(X|\zeta) &= \text{single-target transition density at time } k \text{ given previous state } \zeta. \end{aligned}$$

Measurement update:

$$\mathcal{V}_k(X) = \left((1 - p_d) + \sum_{\mathbf{y} \in Z_k} \frac{p_d g_k(\mathbf{y}|X)}{\kappa_k(\mathbf{y}) + p_d \int g_k(\mathbf{y}|X) \mathcal{V}_{k|k-1}(X) dX} \right) \mathcal{V}_{k|k-1}(X)$$

where

$$\begin{aligned} p_d &= \text{target detection probability,} \\ \mathcal{V}_k(X) &= \text{predicted state PHD,} \\ \mathbf{y} &= \text{measurement,} \\ g_k(\mathbf{y}|X) &= \text{likelihood of a measurement given a state } X \text{ at time } k, \\ \kappa_k(\mathbf{y}) &= \text{PHD of the clutter RFS at time } k. \end{aligned}$$

From the equations above we can see that, if the PHD of the previous state $\mathcal{V}_{k-1}(\zeta)$ is a Gaussian mixture and the transition density¹ is a linear Gaussian, the predicted state, $\mathcal{V}_{k|k-1}(X)$ will also be a Gaussian mixture. Similarly, if the measurement likelihood is a linear Gaussian, the posterior intensity $\mathcal{V}_k(X)$ is also a Gaussian mixture. From the above equations we can also see that with every prediction step, the existing state intensities (weighted by the survival probabilities), are propagated forward in time and the new birth intensities ($\gamma_k(X)$) are added. These birth intensities correspond to the prior distributions over the states of possible new targets. During the measurement update, each of the predicted intensities is updated with each of the measurements. The updated mixture associated with each measurement is then divided by the total mass of the entire updated mixture, plus the clutter probability of the measurement.

¹This is the conditional distribution of the next state (given the previous state) that encodes how the states change with time.

This will of course simply correspond to normalisation if the clutter probability is zero. The updated mixtures are also weighted by the detection probability. Furthermore, all the non-updated states will also be present in the posterior due to the $1 - p_{d,k}$ constant term. The posterior intensity is therefore a Gaussian mixture that contains weighted components of all the non-updated intensities and all possible permutations of the measurement-updated intensities. This will of course result in an unmanageable number of components, and methods to reduce the number of components are therefore typically applied. Component merging methods allow similar components to be merged together, while pruning methods are used to discard components with low weights. The detection and clutter weighting, as well as the weight adjustment inherent in Gaussian multiplication, cause the correct components² to be more likely to have larger weights. On the other hand, the incorrect components will typically have very small weights and will therefore be removed by pruning operations.

The PHD filter also allows the number of targets in any specific region to be estimated by integrating the intensity function over the specific region [21]. The total number of targets at any given time can therefore be estimated simply by adding all the weights of the components of the Gaussian Mixture intensity³.

Several methods have been proposed to extract state estimates from the posterior intensity function. One such method is to first estimate n , the number of targets, and then select the n components with the largest weights [22, p. 684]. This method is of course rather sensitive to the accuracy of the target number estimate and is therefore more appropriate in PHD filter variations that have been designed for more accurate cardinality estimates, such as the cardinalised PHD (CPHD) filter. An alternative to the above is to simply choose a threshold (typically around 0.5) and, after each measurement update, select the means of the components with weights higher than this threshold as the estimates [7, p. 766].

This concludes our discussion on the PHD filter. We have given an overview of the operation of the PHD filter and discussed the update equations. In Section 10.2 we will compare the GM-PHD filter to the developed PGM model and discuss the differences between the two.

2.1.2 The Multiple Hypothesis Tracking Filter

The multiple hypothesis tracking filter (MHT) filter, in contrast to the PHD filter, is an example of an association-based method for multiple object tracking. Like most multiple object tracking filters, the MHT filter receives detections in scans, with a scan containing a set of detections received at a given time. The MHT filter assumes that, for any given scan, each detection is associated with at most one object and each object is associated with at most one detection [23, p. 108].

²The ones corresponding to the ground truth measurement updates.

³This is due to the fact that the integral of a Gaussian distribution (over the entire space) is equal to the weight of the Gaussian. The integral of a Gaussian mixture is therefore equal to the sum of the weights of the individual components.

There are two main variations of the MHT filter - the track-oriented MHT (TOMHT), and the hypothesis-oriented MHT (HOMHT) [24, p. 200]. The hypothesis-oriented MHT is more similar to the model that was developed in this work and we will therefore only focus on this variation here. We will be referring to this specific variation when using the ‘MHT’ acronym. The MHT filter, in theory, considers all possible combinations of measurement to track associations, known as association hypotheses, for a given set of measurements and tracks at a specific time. These association hypotheses are used initially as the starting points for more complex hypotheses, spanning multiple time frames. These hypotheses are then extended (in length and number) with new association hypotheses when new measurements are received. The different hypotheses can be represented by a tree.

An example of a hypothesis tree is shown in Figure 2.1 below. For this example we will assume that there is one existing track and that a scan with two measurements has been received. The first node (in the top layer) in this example represents our starting hypothesis (one existing track). The second layer shows that the measurement y_0 can be clutter (associated to the clutter ‘track’ - node 0), or it can be associated with the existing track (node 1), or it can be associated with a new track (node 2). The third layer shows the possible associations for y_1 , conditioned on the associations in the layer above. This process of enumerating all the possible associations is then repeated for each hypothesis when a new scan is received. Each node in the graph therefore corresponds to an association hypothesis. Each descending path through the entire graph corresponds to a complete hypothesis about all measurements received thus far. This will be referred to simply as a ‘hypothesis’. Since all nodes (except for the root node) in the graph will have exactly one parent (neighbour in the layer above it), each node in the bottom layer can also be viewed as representing a complete hypothesis. Node 3 on the right in the last layer, for example, represents the hypothesis that both y_0 and y_1 are associated with new objects. If a new set of measurements is received at a later time, each of the nodes in the bottom layer will serve as starting points from which the individual hypotheses are extended. The process described above is then repeated at each of these nodes, using the new set of measurements to extend the hypotheses. Note that, in this example, a node in the second layer cannot have the same number as its neighbours in the third layer, except if it is clutter. This is due to the fact that y_0 and y_1 are in the same scan and the assumption that there is only one valid association between a track (including the clutter track) and a measurement. Neighbouring nodes in layers that represent measurements from different scans will therefore be allowed to have the same object number.

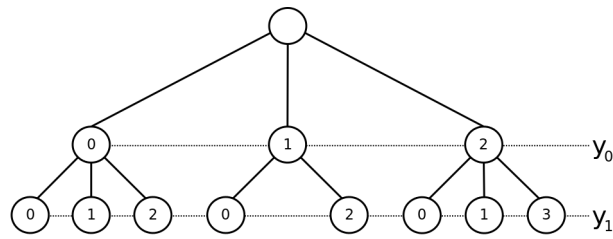


Figure 2.1: Example of an MHT hypothesis tree. y_i is the i 'th measurement in the scan and the node numbers in the corresponding row indicate the track with which y_i is associated in the specific hypothesis. The number 0 tracks represent the clutter track. The blank node at the top of the graph represents the initial hypothesis. (Note that this is not a PGM.)

The tree in Figure 2.1 can be represented by Table 2.1 below. In this table, each column corresponds to an association hypothesis. The entries in the first column indicate which track the measurement y_0 is associated with in each hypothesis, and the second row similarly shows the y_1 track associations. Each column therefore corresponds to a hypothesis. These types of tables are typically used as matrices in implementations of the MHT filter. Note that the invalid association combinations are marked in red. These will not be considered in the hypothesis evaluation and can be removed from the table. We will use this table representation in the rest of this example. This example (including Figure 2.1 and Table 2.1) was adapted from the example in the *Multiple Hypothesis Tracking Implementation* [24] document.

a_0	0	0	0	1	1	1	2	2	2	2
a_1	0	1	2	0	1	2	0	1	2	3

Table 2.1: MHT filter hypothesis table (corresponding to the tree in Figure 2.1). The disallowed association hypothesis combinations are marked in red.

In practice, these tables can of course grow intractably large. In order to avoid an unbounded increase in the number of hypotheses that need to be managed, the MHT typically only considers hypotheses over a fixed number of scans to be formed. This number is known as the window size [24, p. 204]. If the window size is equal to n , we can evaluate hypotheses formed over n time steps and find the most likely one to use as a starting point for forming new hypotheses. Another method for managing the number of hypotheses periodically removes the least likely hypotheses from the set of hypotheses to be considered. In order to evaluate the probability of each hypothesis in any of these methods, the state distributions need to be updated continuously. Since each hypothesis has at most one measurement associated with a certain target at every time step, the normal Kalman filter equations can be used to update the state distributions of the individual targets at each time step [24, p. 204]. The MHT algorithm with Kalman filter updates and subsequent calculation of hypothesis probabilities is shown below.

MHT Recursion [23, pp. 125,129]

Inputs:

$\mathbf{y}_K = \{\mathbf{y}^1, \dots, \mathbf{y}^K\}$: the measurements in the first K scans

$P(X_{n,\kappa}^k)$: prior for track (index) n , at time step k , in hypothesis κ

$H(k-1)$: all hypotheses generated using y^1, \dots, y^{k-1} .

for $k = 0 : K$ **do**

Extend all hypotheses $H(k-1)$ with new association hypotheses Γ^k

(A) Compute Updated Distributions:

for $\gamma \in \Gamma^k$ **do**

with:

- ν : corresponding hypothesis (in $H(k-1)$) being extended by γ
- n, n_T : target index and number of targets
- m_k : number of measurements in scan k
- d_ν : number of associations in ν
- \mathbf{y}_m^k : the measurement associated with $X_{n,\nu}^k$ in γ (if any)
- P_d, P_c : detection and clutter probabilities
- \mathbf{y}^k : the measurements in the k'th scan
- $\alpha(\gamma|\mathbf{y}_K)$: association probabilities of hypothesis γ given \mathbf{y}_K
- d : dimension of the X state distribution.
- $\alpha(\nu)$: association probability of hypothesis ν

for $n = 0 : n_T$ **do**

state prediction:

$$P(X_{n,\nu}^k) = \int_{\mathbb{R}^d} P(X_{n,\nu}^k | X_{n,\nu}^{k-1}) P(X_{n,\nu}^{k-1}) dX_{n,\nu}^{k-1}$$

measurement update (if applicable):

$$P(X_{n,\nu}^k | \mathbf{y}_m^k) = \frac{1}{P(Y_m^k = \mathbf{y}_m^k)} P(Y_m^k = \mathbf{y}_m^k | X_{n,\nu}^k) P(X_{n,\nu}^k)$$

end

(B) Compute Association Probabilities (α):

$$\beta(\gamma) = \alpha(\nu) P(\gamma) (1 - P_d)^{n_T - d_\nu} \prod_m^{m_k} g(\mathbf{y}_m^k)$$

$$\text{with } g(\mathbf{y}_m^k) = \begin{cases} P_c(\mathbf{y}_m^k), & \text{if clutter} \\ P_d \int P(Y_m^k = \mathbf{y}_m^k | X_{n,\nu}^k) P(X_{n,\nu}^k) dX_{n,\nu}^k, & \text{otherwise} \end{cases}$$

$$\alpha(\gamma|\mathbf{y}_K) = \frac{\beta(\gamma)}{\sum_{\gamma' \in H(k)} \beta(\gamma')}$$

end

end

In the above algorithm we can see the standard Kalman filter state prediction and measurement update steps. These steps are, as mentioned earlier, performed for all target states in all hypotheses. Additionally, the algorithm requires the probabilities of each hypothesis to also be updated. An updated hypothesis probability is calculated by multiplying the previous hypothesis probability with the new association hypothesis probability, and then normalising this probability with respect to all updated hypotheses. The above algorithm therefore gives a set of updated hypotheses and their corresponding state distributions, as well as a distribution over the updated hypotheses at each time step. As discussed earlier, the above algorithm will typically yield an intractable number of hypotheses and it is therefore necessary to limit the hypotheses to a manageable number.

The MHT filter can produce individual target tracks because the target identities are explicitly maintained in the various hypotheses. This represents an advantage over the PHD filter, but also requires more computationally expensive calculations.

2.1.3 The Joint Probabilistic Data Association Filter

The joint probabilistic data association (JPDA) filter is an extension of the probabilistic data association (PDA) filter, used for single object tracking. The JPDA filter allows a fixed and known number of targets to be tracked and, in contrast to the MHT filter, allows for soft/uncertain data associations [23, p. 108]. The basic JPDA filter is intractable to implement and many approximations have therefore been proposed. The joint integrated PDA (JIPDA) is an extension of the JPDA that allows a varying number of targets to be tracked [25].

Because the JPDA filter and its variants allow soft associations, it does not form separate hypotheses. Rather, each state distribution is updated by all measurements independently. In the linear Gaussian case, the posterior after a measurement update is therefore a Gaussian mixture. This mixture is typically approximated by a single Gaussian before the subsequent prediction step [23, p. 139]. The JPDA filter therefore effectively tries to summarise all the measurement association possibilities through the weighted measurement update and subsequent approximation. By contrast, the MHT filter keeps all of the hypotheses separate in order to find the most probable one.

2.2 Probabilistic Graphical Models for Multiple Object Tracking

Little work appears to have been done on casting the multiple object tracking problem as a graphical model. The use of PGMs to solve this problem does not seem to be discussed in PGM textbooks, nor in the vast majority of MOT literature. PGMs have, however, been used for multiple object tracking in a small number of instances in the past.

In a master's thesis by Chen [12], the use of Markov random fields to solve the MOT problem is discussed. In this work, emphasis is placed on avoiding loops in graphs by grouping more potentials into single nodes. The use of MRFs instead of cluster graphs, as well as the different graph structure and the use of the LBP algorithm, make the model developed by Chen markedly different from the model developed in our work. Furthermore, the model proposed by Chen requires the number of targets, the detection probability and the clutter density to be known, which makes it somewhat more restricted in its use. The exact mechanism by which the data association problem is solved is also not described in detail. Chen does, however, discuss interesting functionality, such as the use of multiple, different types of sensors and the incorporation of out-of-sequence data. The capability of the developed model to track objects in the presence of clutter, although mentioned, is not clearly demonstrated.

Maskell et al also discuss a graphical model solution for multiple object tracking [26]. The paper is, however, largely focussed on out-of-sequence data. Furthermore the data association mechanism is not discussed in detail.

An interesting approach to the data association problem is taken by Segal and Reid [11] where the focus is on associating latent states over adjacent time slices instead of inferring the state-measurement associations. A key advantage of this approach, as stated in the paper, is that it allows model selection to be performed during inference. However, the paper does not discuss how their model selection compares to the classical model selection method. We will show in Section 7.1 that the alternative state-measurement association model can indeed also be used to perform model selection. Here we will also show that it is mathematically equivalent to classical model selection.

Other work relating to PGM-based multiple object tracking focusses only on tracking in video sequences. Video data offers much richer features for targets compared to radar data, in which the appearance of all targets is practically identical. In this type of application, the data association problem can usually be resolved through segmentation and classification. Although segmentation and classification can be complex in their own right, the data association problem faced in the tracking of indistinguishable targets can be largely avoided by classifying targets first. In this sense, the video tracking problem is less complex (if the classification problem is solved) than multiple identical object tracking. An example of work done on PGM-based tracking in video sequences is the paper by Wang, de La Corce and Paragios [27]. In this work, Markov random fields are used to simultaneously perform image segmentation and tracking using monocular video data. Although interesting, this work is sufficiently different from the work discussed in this text and will therefore not be discussed here.

In summary, whereas PGMs have been employed in some works to solve the multiple object tracking problem, the number of such works is very limited. Furthermore, a PGM that is capable of tracking an unknown number of identical targets with unknown detection probability and clutter intensity has not, to the best of our knowledge, been described in the existing literature.

2.3 Conclusion

The MOT algorithms described in this chapter are, in a sense, all quite similar. This is due to the fact that they are all based on probability theory and all use Gaussian distributions. However, they differ in the assumptions they make and in the specifics of the algorithms. Since the PGM developed in this work is based on sound probability theory and also makes extensive use of Gaussian distributions, one can expect it to also share similarities with the existing algorithms. The developed PGM with the classical model selection possesses elements of the MHT and JIPDA filters, and would be very much like the MHT filter if soft associations were not allowed. The developed PGM with the alternative model selection is very similar to the JIPDA filter, since it does not keep hypotheses completely separate, but rather incorporates the various hypotheses into the state distributions. The developed PGM and the EMDW code bases do, however, offer a more robust framework, especially for the iterative inference. Furthermore, the clutter classification method used in the developed PGM is very different from how clutter is treated in the classical algorithms and in any of the PGM implementations that could be found in the literature. The number of the previous PGM-based implementations is also very limited and the developed model is distinguished from these implementations in that it is capable of tracking an unknown number of targets with unknown detection probability and clutter intensity. Lastly, the developed PGM could, more easily, serve as a foundation on which to build additional functionality.

Chapter 3

Probabilistic Graphical Models

Probabilistic graphical models (PGMs) give us a general framework to efficiently represent high dimensional probability distributions as a graph. Because of their generality and inference efficiency, PGMs have been applied to an extremely wide range of problems in different fields. Some of these include bioinformatics, error-control coding, and language processing [28]. The potential use of PGMs is so widespread because it is effectively applicable to any problem where uncertainty is involved.

This chapter provides a brief overview of some of the theoretical foundations of probabilistic graphical models and cluster graphs specifically. It is therefore aimed at the reader who is not very familiar with graphical models or cluster graphs.

3.1 Bayes Networks

There are various types of graphical models, each with their own characteristics and each being suitable for specific purposes. One of the most prominent graphical models is the Bayes network.

Bayes networks are directed, acyclic graphical models where each node that has parent(s) represents a conditional probability distribution while each parent-less node represents a prior probability distribution. Although the directed edges in Bayes networks do not necessarily correspond to causal dependencies, causal Bayes networks are often more sparsely connected and intuitive and allow for more efficient inference [1, p. 1009]. It is therefore preferable for a Bayes network to have a causal structure. Below is an example of a simple Bayes network.

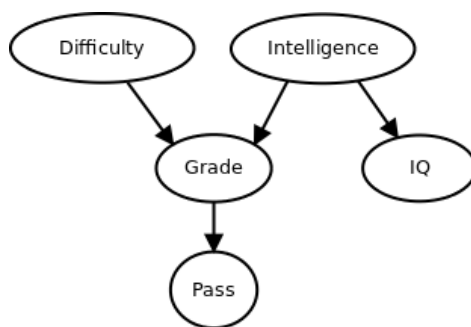


Figure 3.1: Student-grade Bayes network example. The concepts contained in the nodes represent random variables and the edges encode the dependencies between these random variables. (This example is adapted from an example in [1, p. 162])

The network above is a toy model that shows the relationships between some of the factors that might influence whether or not a student passes a test or subject. Two important factors that will determine the grade a student receives for a test is the student's intelligence and the difficulty of the test¹. The IQ test score for the student is also dependent on the student's intelligence. This score can give us information about the student's intelligence and therefore what grade we can expect from the student. If, however, we knew exactly how intelligent the student was, the IQ test results would not give us any new information about possible test scores for the student. Furthermore, whether or not the student passes is independent of everything else if we know the student's grade. The Bayes network above, and Bayes networks in general, encode the dependencies between the random variables in the network as described in this example. These dependencies (and independencies) can be explained logically, as in this example, but also represent mathematical, probabilistic dependencies (and independencies).

The process of constructing a Bayes network serves as a good starting point for setting up a probabilistic inference problem, as it allows us to determine the dependence structure of the larger, joint distribution in an intuitive manner. Bayes networks² can therefore be used for probabilistic reasoning, as discussed in the above example. This reasoning reduces to logical reasoning if deterministic functions are used instead of probability distributions.

3.2 Variable Elimination and Cluster Graphs

In this section, the origin of message passing algorithms and the concept of cluster graphs are discussed. With Bayes networks, one can obtain the joint probability by multiplying together

¹Of course, in reality there may be many other factors, but these factors will suffice for the sake of this example.

²In practice, inference is generally done in other types of graphs, such as cluster graphs, that are slightly different, but related to Bayes networks.

all of the conditional and prior distributions, represented by the nodes in the graph. For the graph in Figure 3.2, below, the expression for the joint probability will therefore be as follows:

$$P(A, B, C, D, E) = P(C|A, B)P(D|B)P(E|C)P(A)P(B).$$

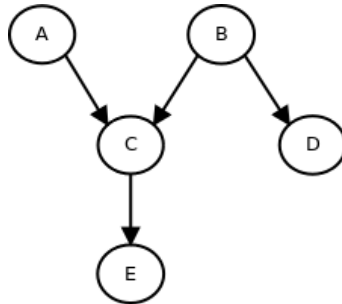


Figure 3.2: Simple example of a Bayes network

In practice, however, we are often less interested in the joint distribution and more interested in the marginal distributions. In order to calculate the marginal distributions for the individual variables, we could calculate the joint distribution and then integrate/sum out all of the variables that we are not interested in. We could repeat this process for all the variables to get all the marginal distributions. In large graphs, however, following this approach can be computationally expensive or even practically impossible. Inference in PGMs can be more efficient with more advanced algorithms used in conjunction with other types of graphical models. The starting point for deriving a specific class of inference algorithms, known as message passing inference, is the variable elimination algorithm. This algorithm will only serve as the means by which these concepts are derived and will not be used for inference in the rest of this work. The variable elimination algorithm essentially uses an efficient ordering of operations to improve efficiency. The example below shows how the variable elimination algorithm is applied to the graph in Figure 3.2. In this example we will show how variable elimination works by calculating the $P(D)$ marginal in a series of steps. These steps are shown below.

$$P(D) = \sum_{A,B,C,E} P(A, B, C, D, E) \quad (3.2.1)$$

$$= \sum_{A,B,C,E} P(C|A, B)P(D|B)P(E|C)P(A)P(B) \quad (3.2.2)$$

$$= \sum_{A,B,C} P(C|A, B)P(D|B)P(A)P(B) \sum_E P(E|C) \quad (3.2.3)$$

$$= \sum_{A,B,C} P(C|A, B)P(D|B)P(A)P(B) \sum_E \psi_1(E, C) \quad (3.2.4)$$

$$= \sum_{A,B,C} P(C|A, B)P(D|B)P(A)P(B)\delta_{12}(C) \quad (3.2.5)$$

$$= \sum_B P(D|B)P(B) \sum_{A,C} P(C|A, B)P(A)\delta_{12}(C) \quad (3.2.6)$$

$$= \sum_B P(D|B)P(B) \sum_{A,C} \psi_2(C, A, B)\delta_{12}(C) \quad (3.2.7)$$

$$= \sum_B P(D|B)P(B)\delta_{23}(B) \quad (3.2.8)$$

$$= \sum_B \psi_3(D, B)\delta_{23}(B). \quad (3.2.9)$$

The calculations above correspond to a run of the variable elimination algorithm in the following order: E, A, C, B, D . Here the E variable is eliminated (marginalised out) first (in equations 3.2.3 to 3.2.5), followed by the A and C variables (in equations 3.2.6 to 3.2.8) and so forth. The key idea with this procedure is to avoid creating and marginalising over high dimensional distributions through an efficient ordering of multiplication and summation operations. A run of variable elimination can also be interpreted as performing inference in a type of graphical model known as a cluster graph [1, pp.346]. In this interpretation, the ψ functions (called potentials) represent the distributions (conditional or prior) of a specific node (or cluster) in the graph and the δ functions are viewed as messages being passed in between the nodes. From the above equations we can see that the messages are calculated by marginalising the potentials (possibly multiplied by received messages) over the variables that are not in the message scope. These messages update the potentials with information from other parts of the graph. The updated functions are known as cluster beliefs and the message information may be related to the prior or conditional distributions, or to evidence that has been observed³. Once message passing is complete, the updated cluster distributions are the posterior marginal distributions over the variables in the cluster's scope. The full list of marginals for this example is given below.

³Observing evidence in a graphical model corresponds to setting the observed variable to the observed value in all clusters that contain it. This is also known as conditioning.

$$\begin{aligned}
P(A) &= \sum_{C,B} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C); & P(B) &= \sum_D \psi_3(D, B) \delta_{23}(B); \\
P(B) &= \sum_{A,C} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C); & P(C) &= \sum_E \psi_1(E, C) \delta_{21}(C); \\
P(C) &= \sum_{A,B} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C); & P(E) &= \sum_C \psi_1(E, C) \delta_{21}(C). \\
P(D) &= \sum_B \psi_3(D, B) \delta_{23}(B);
\end{aligned}$$

Note that the subscripts which have been used for the factors and messages above are defined in order to correspond to the associated cluster graph, as shown in Figure 3.3 below. The factor subscripts indicate the corresponding cluster number in this graph, while the two numbers in the message subscripts indicate from and to which clusters they are sent respectively. The derivations for the marginal expressions above are given in appendix A.1. Note that there is more than one expression for the B and C marginals. From a cluster graph perspective, this is due to the fact that these variables appear in more than one cluster and their marginals can be computed from the posterior distributions in any cluster that contains them.

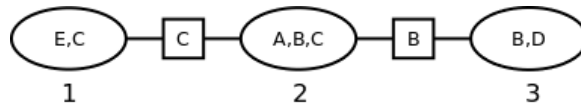


Figure 3.3: Cluster graph corresponding to the Bayes network in Figure 3.2. The elliptical nodes are the cluster nodes, and the square nodes are the separation sets (sepsets). The numbers beneath the cluster nodes indicate their respective indices, corresponding to the subscripts in the variable elimination/message passing equations.

The elliptical nodes in the above cluster graph are the cluster nodes, and the square nodes are known as the separation sets (or sepsets). In tree-structured graphs, the scope of each sepset is equal to the overlap between neighbouring nodes. The messages passed between two nodes will always have the same scope as that of the corresponding sepset. The expressions for the

messages that are passed in the graph are as follows:

$$\begin{aligned}\delta_{12}(C) &= \sum_E \psi_1(E, C); & \delta_{23}(B) &= \sum_{A,C} \psi_2(C, A, B) \delta_{12}(C); \\ \delta_{32}(B) &= \sum_D \psi_3(B, D); & \delta_{21}(B) &= \sum_{A,C} \psi_2(C, A, B) \delta_{32}(B).\end{aligned}$$

In the above expressions, the first number in the message subscript indicates the sending cluster, and the second the receiving cluster (see the cluster indices in Figure 3.3). We can see that each of the messages on the right, which are sent from cluster 2, also incorporates a message received by cluster 2. This allows information from node 1 to be shared with 3 and vice versa. In this manner, information is shared throughout the graph. Note that, in order to calculate any of the posterior marginals, we need to send two messages. We can, however, calculate all the marginals by sending all four messages. In general, message passing in tree-structured graphs allows us to compute all of the marginals for essentially double the computational cost of calculating a single marginal⁴.

3.3 Belief Propagation in Cluster Graphs

The previous section discussed the connection between variable elimination and cluster graphs and the motivation for message passing algorithms. There are, however, more general variations of the basic message passing procedure. The two most prominent variations are known as *loopy belief propagation* (LBP) and *loopy belief update* (LBU). Here, the term ‘loopy’ indicates that the algorithms can in general be applied to graphs that contain loops. Message passing in cluster graphs with loops, as opposed to tree-structured graphs, is an iterative procedure and is not guaranteed to converge to a correct solution, or at all. These algorithms do, however, often yield good results in practice [1, pp.358]. The LBP and LBU algorithms give us a method of inference that is related to (but different from) variable elimination. Since these algorithms are used for inference on graphs, we do not use variable elimination⁵ to first construct the graph when making use of message passing algorithms, rather we can construct a cluster graph first and then perform inference. A cluster graph can be constructed manually⁶ or automatically with the *layered trees running intersection property* (LTRIP) algorithm [29, p. 5]. A basic procedure, for constructing a cluster graph is illustrated by Figure 3.4 below:

⁴This can easily be shown by considering that, for any cluster in a tree-structured graph, messages across all edges need to be sent once to update the factor with information from all clusters. If we then send messages across each edge in the opposite direction, we have calculated all the cluster marginals for double the computational cost.

⁵Although there are algorithms, such as the junction tree algorithm, that does make use of variable elimination (but without computing the distributions) to derive the graph structure.

⁶This can be done with relative ease if the graph is small.

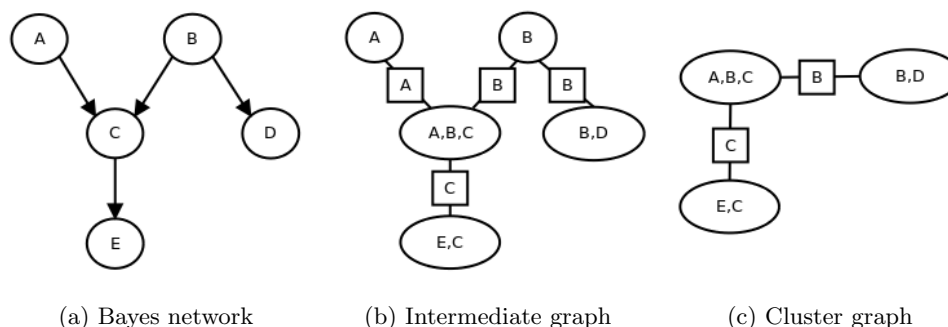


Figure 3.4: Compiling a Bayes network into a cluster graph (example). The factors represented by the Bayes network in (a) is connected to form the graph in (b). The A and B nodes have been absorbed by their neighbours in the final cluster graph in (c).

As shown by Figure 3.4 above, we can construct a cluster graph by first connecting all factors with overlapping scopes (see Figure 3.4b above). Any factor of which the scope is a subset of one of its neighbour's scopes is then absorbed into that neighbour. A final step in compiling a valid cluster graph would be to ensure that the resulting graph satisfies the running intersection property (RIP). This property is satisfied when, for any variable in the graph, there is a unique path between any two clusters containing that variable. Here a path for a specific variable is as defined by a set of adjacent sepsets that all contain the specific variable [1, pp.347].

The running intersection property ensures that there are no feedback loops for individual variables in the graph. Such loops can cause unwarranted and incorrect reinforcement of beliefs and are therefore undesirable. We can ensure that the RIP property is satisfied by removing variables from sepsets in order to break any loops that the graph might have for a specific variable. This step is, however, not necessary in the above example, as the graph in Figure 3.4c already satisfies the running intersection property. This final step can be done manually, but may be infeasible for large graphs. In such cases an algorithm such as the LTRIP algorithm [29, p. 5] would be more suited for constructing the graph. The graphs that we will be dealing with in this work, although large, have a very regular structure that is governed by a small set of rules, and we will therefore be able to construct the necessary cluster graphs with the simple procedure described above.

It should also be noted that there are other popular types of probabilistic graphs that are also used for inference. The factor graph representation is in fact more widely used than the cluster graph. Factor graphs are trivial to construct⁷ from a set of factors and this is probably the reason for their widespread use compared to cluster graphs [29, pp.1]. A study by Streicher and du Preez has, however, shown that cluster graphs often improve on factor graphs in terms of convergence

⁷While the LTRIP algorithm allows cluster graphs to also be automatically constructed, this algorithm is not widely known at present.

time [29]. For this reason, we have chosen to use cluster graphs for our implementation.

In order to perform probabilistic inference in general, one typically needs to perform summation/integration and multiplication operations. In message passing algorithms, these operations are performed on the factors in the graph to compute and absorb messages. The messages from one cluster to another is calculated as the product of the cluster potential (the function which was initially specified by a single factor or product of factors) and the messages received from other clusters, but excluding the message (if any) previously received from the now receiving cluster. The exclusion of this message is quite intuitive if we view these messages as pieces of information that update a distribution or belief of a cluster. We would not expect a cluster to use its previous beliefs to reinforce its updated beliefs, as this would cause a self reinforcement loop of sorts. The LBP and LBU algorithms have a slightly different approach to excluding these messages. In the LBP algorithm an outgoing message is calculated by taking the product of the received messages on the other edges and the cluster potential and marginalising this product. The expression below together with the cluster graph in Figure 3.5 shows an example of this calculation.

$$\delta_{21} = \sum_{C,B} \psi_2(A, B, C) \delta_{32}(B) \delta_{42}(C)$$

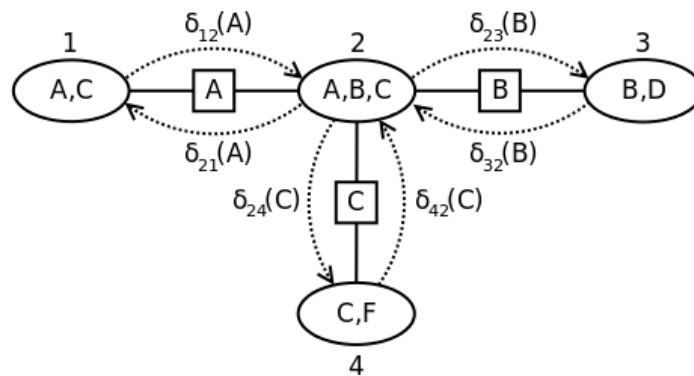


Figure 3.5: Cluster graph message passing example. The messages are indicated by δ_{mn} for a message sent from factor number m to number n . The factor numbers are indicated by the numbers on the outside of the factor nodes.

The LBU algorithm continuously updates the cluster beliefs (the initial cluster potentials updated by messages) with incoming messages. Any outgoing message from a cluster is then calculated by marginalising the updated distribution and then dividing out the message previously received (if any) on the now outgoing edge. In the LBU case, the δ_{21} message will therefore be

calculated as follows:

$$\begin{aligned}\psi'_2(A, B, C) &= \psi_2(A, B, C)\delta_{12}(A)\delta_{32}(B)\delta_{42}(C) \\ \delta_{21} &= \sum_{C,B} \psi'_2(A, B, C)/\delta_{12}(A)\end{aligned}$$

These two examples and the two algorithms in general are of course mathematically identical under most circumstances⁸, but they are computationally different. The LBP algorithm is typically less computationally efficient due to duplicate multiplications. Furthermore, there are cases where the cluster potential/belief functions in the graph are undefined without a conditioning⁹ distribution¹⁰. In such cases, messages will need to be sent over certain edges in the graph, but sending messages in the opposite direction over these edges will require these messages to be omitted from the message calculation. This case therefore presents a problem for the LBP algorithm, since the message needs to be multiplied in to define the distribution, but needs to be omitted in order to calculate the outgoing message over the edge. An example of this is when the unscented transform is used to define conditional distributions. The LBU algorithm allows this problem to be avoided through the division (instead of omission from calculation) of previously received messages.

So far, we have discussed how messages distribute information through a graph, and mentioned that the information can have various sources. However, we still need to discuss a vital source of information, namely the observed data. Data can be incorporated into a graph by simply setting any observed variables to their observed values in the corresponding distributions. In a model where some distributions are not initially defined, we can wait until they are¹¹, and then set the variables. This process is known as ‘observing evidence’ or ‘conditioning’. There are therefore four basic operations that are required to perform message passing in the context of the LBU algorithm:

- summation/integration¹² (for marginalisation)
- multiplication (for forming joint distributions/updating factors)
- division (for preventing self reinforcing feedback)
- conditioning (for observing evidence)

So far we have looked at the basic theory behind inference in graphical models. In the following chapter we will discuss the specifics of the mathematical operations that are needed for such inference.

⁸The LBP and LBU messages are not exactly identical when some of the distributions contain probabilities that are 0.

⁹This should not to be confused with the conditioning operation that is used when observing evidence.

¹⁰This is a distribution (prior or message function) that is initially multiplied with the conditional distribution to turn it into a joint distribution.

¹¹This will happen during message passing.

¹²Note that the summation operations that have been used thus far will of course be replaced by integration operations in the case when the variables are continuous.

Chapter 4

The Multivariate Gaussian Distribution

In this chapter we will discuss the mathematical details that will allow us to perform inference in the graphical models used in the rest of this work. The equations described in this chapter relate to the basic operations mentioned in the previous chapter, and also include equations for linear and non-linear transformations of Gaussian random variables.

4.1 Gaussian Parameterisations and Operations

In order to use continuous variables in the graphical models that we have described thus far, we need to find a continuous distribution with suitable properties. Specifically, the distribution should have a closed form under integration, multiplication, division and conditioning. The Gaussian distribution has all of these properties. This distribution can be parameterised in various ways, as listed below:

- covariance form
- canonical form
- sigma point form

These various forms are useful for efficiently performing certain operations. Multiplication, for example, can be efficiently performed in the canonical form, while the integration operation can be efficiently computed in the covariance form. The sigma point form is somewhat different from the first two in that it allows a multivariate Gaussian to be represented by a set of weighted points. The applications of this form will be discussed in Section 4.3. In this section we will

show how the canonical and covariance form parameters are used to calculate the results of the various operations required for inference.

4.1.1 Covariance Form to Canonical Form Conversion

The covariance form of the multivariate Gaussian is the most well known and is shown below [1, p. 247]:

$$\mathcal{N}(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = \frac{w}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp(-0.5[X - \boldsymbol{\mu}_X]^T \boldsymbol{\Sigma}^{-1} [X - \boldsymbol{\mu}_X]). \quad (4.1.1)$$

The weight term w is typically not included as a parameter in probability distributions, since they are almost always used in a normalised state (with $w = 1$). It is, however, included here, since we will be working with unnormalised Gaussian distributions in the model developed in this work. These unnormalised distributions will arise when conditioning distributions on data, and when a Gaussian distribution forms part of a Gaussian mixture distribution.

We can rewrite the above expression in equation 4.1.1, with the normalisation constant and weight in the exponent, as follows:

$$\mathcal{N}(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = \exp(-0.5[X - \boldsymbol{\mu}_X]^T \boldsymbol{\Sigma}^{-1} [X - \boldsymbol{\mu}_X] + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)). \quad (4.1.2)$$

From equation 4.1.2, we can derive the canonical form by multiplying out the brackets next to the covariance matrix. The expression for the canonical form is shown below [1, p.609]:

$$\mathcal{N}(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = \mathcal{C}(X; \mathbf{K}, \mathbf{h}, g) = \exp(-0.5[X^T \mathbf{K} X] + X^T \mathbf{h} + g),$$

where the canonical parameters above are defined as follows:

$$\begin{aligned} \mathbf{K} &= \boldsymbol{\Sigma}^{-1}, \\ \mathbf{h} &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}, \\ g &= \frac{-1}{2} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|) \\ &= \frac{-1}{2} \mathbf{h}^T \boldsymbol{\Sigma} \mathbf{h} + \log(w) - 2\log(|2\pi\mathbf{K}^{-1}|). \end{aligned}$$

The derivation of this form is given in appendix A.2.2.

4.1.2 Gaussian Operations

During message passing, operations will often only be performed on a subset of the variables of a multivariate distribution. It is therefore useful to distinguish between the variables that are

involved in an operation and those that are not. The partitioned matrix/vector representation can be used to explicitly express such a distinction. The partitioned representations for the covariance and canonical parameters are given below:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{h}_X \\ \mathbf{h}_Y \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_Y \end{bmatrix},$$

where

$$\Sigma_{YX} = \Sigma_{XY}^T \text{ and } \mathbf{K}_{YX} = \mathbf{K}_{XY}^T.$$

The X and Y subscripts in the above expressions indicate the two sets of variables (X and Y), one set of which will be affected in any given operation. In the case that all variables are affected by the operation, one of the sets can simply be regarded as an empty variable set. Note that the X and Y variables, in this chapter, do not correspond to the state and measurement random variables, as they do in the other chapters. The list of Gaussian operations necessary for inference is given below:

- Gaussian Product [1, p.610]:

$$\mathcal{C}_1([X, Y]; \mathbf{K}_1, \mathbf{h}_1, g_1) \mathcal{C}_2(X; \mathbf{K}_2, \mathbf{h}_2, g_2) = \mathcal{C}_3([X, Y]; \mathbf{K}', \mathbf{h}', g')$$

where the resulting parameters are:

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{1,XX} + \mathbf{K}_2 & \mathbf{K}_{1,XY} \\ \mathbf{K}_{1,YX} & \mathbf{K}_{1,YY} \end{bmatrix}, \quad \mathbf{h}' = \begin{bmatrix} \mathbf{h}_{1,X} + \mathbf{h}_2 \\ \mathbf{h}_Y \end{bmatrix}, \quad g = g_1 + g_2.$$

Note that $\mathbf{K}_{1,XX}$ is the \mathbf{K}_{XX} partition matrix of \mathbf{K}_1 , $\mathbf{h}_{1,X}$ is the \mathbf{h}_X partition vector of \mathbf{h}_1 and $[X, Y]$ is the vertical concatenation of the X and Y column vectors.

- Gaussian Quotient [1, p.611]:

$$\mathcal{C}_1([X, Y]; \mathbf{K}_1, \mathbf{h}_1, g_1) / \mathcal{C}_2(X; \mathbf{K}_2, \mathbf{h}_2, g_2) = \mathcal{C}_3([X, Y]; \mathbf{K}', \mathbf{h}', g')$$

where the resulting parameters are:

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{1,XX} - \mathbf{K}_2 & \mathbf{K}_{1,XY} \\ \mathbf{K}_{1,YX} & \mathbf{K}_{1,YY} \end{bmatrix}, \quad \mathbf{h}' = \begin{bmatrix} \mathbf{h}_{1,X} - \mathbf{h}_2 \\ \mathbf{h}_Y \end{bmatrix}, \quad g = g_1 - g_2.$$

- Gaussian Integration (Marginalisation) [1, p. 250]:

$$\int_{\mathbb{R}^{d_Y}} \mathcal{N}([X, Y]; \boldsymbol{\Sigma}, \boldsymbol{\mu}, w) dY = \mathcal{N}(X; \boldsymbol{\Sigma}_{XX}, \boldsymbol{\mu}_X, w),$$

$$\int_{\mathbb{R}^{d_Y}} \mathcal{C}([X, Y]; \mathbf{K}, \mathbf{h}, g) dY = \mathcal{C}(X; \mathbf{K}', \mathbf{h}', g'),$$

where the resulting parameters are [1, p.611]:

$$\begin{aligned} \mathbf{K}' &= \mathbf{K}_{XX} - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX}, \\ \mathbf{h}' &= \mathbf{h}_X - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{h}_Y, \\ g' &= g + \frac{1}{2} (\mathbf{h}_Y^T \mathbf{K}_{YY}^{-1} \mathbf{h}_Y + \log(|2\pi \mathbf{K}_{YY}^{-1}|)). \end{aligned}$$

(d_Y : dimension of Y)

- Conditioning (Observing Evidence) [1, p.611]:

$$\mathcal{C}([X, Y = \mathbf{y}]^T; \mathbf{K}', \mathbf{h}', g') = \mathcal{C}(X; \mathbf{K}', \mathbf{h}', g'),$$

where the resulting parameters are:

$$\begin{aligned} \mathbf{K}' &= \mathbf{K}_{XX}, \\ \mathbf{h}' &= \mathbf{h}_X - \mathbf{K}_{XY} \mathbf{y}, \\ g' &= g + \mathbf{h}_Y^T \mathbf{y} - \frac{1}{2} \mathbf{y}^T \mathbf{K}_{YY} \mathbf{y}. \end{aligned}$$

The derivations of the above equations are given in appendix A.2. Using these equations, we can perform probabilistic inference in Gaussian graphical models. The type of graphical model that will be required for object tracking, however, requires one more important type of operation to be performed. This operation is the transformation of a random variable. The transformation of Gaussian random variables is discussed in the following sections.

4.2 Linear Gaussian Distributions

In order to track the state of an object as it changes over time, a state transition model and measurement model are needed. These models will be described by transformations of random

variables and additive process/measurement noise. For the model developed in this work, the state transformation will be linear. Non-linear transformations, which will be used for the measurement model, will be described in the following section. The relevant expressions relating to the linear transformation of a Gaussian random variable are given in this section. The following expression shows a linear transformation of a Gaussian random variable X :

$$Y = \mathbf{A}X + \mathbf{c} + N. \quad (4.2.1)$$

Here, \mathbf{A} is the transition matrix, \mathbf{c} is a constant vector and N is a zero-mean Gaussian random variable, representing Gaussian noise. The variable Y resulting from this transformation is also a Gaussian distribution and is distributed as follows [30, p.34]:

$$Y \sim \mathcal{N}(\boldsymbol{\mu}_Y = \mathbf{A}\boldsymbol{\mu}_X + \mathbf{c}, \quad \boldsymbol{\Sigma}_{YY} = \mathbf{A}\boldsymbol{\Sigma}_{XX}\mathbf{A}^T + \boldsymbol{\Sigma}_{NN}, w = w_x).$$

The joint distribution over $[X, Y]$ has the following form [30, p.34]:

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_X \\ \mathbf{A}^T\boldsymbol{\mu}_X + \mathbf{c} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XX}\mathbf{A}^T \\ \mathbf{A}\boldsymbol{\Sigma}_{XX} & \mathbf{A}\boldsymbol{\Sigma}_{XX}\mathbf{A}^T + \boldsymbol{\Sigma}_{NN} \end{bmatrix}, w = w_x\right).$$

In this work, the object state random variable will be a vector in two-dimensional Cartesian space, consisting of two position variables (s_x and s_y) and two velocity variables (v_x and v_y). We will assume that objects tend to move at a constant velocity over short periods of time, and that the object position changes accordingly. This will allow us to make accurate predictions¹ about the next state of an object, over small time steps. The state transition equations are therefore as follows:

$$s'_x = s_x + \Delta_t v_x, \quad s'_y = s_y + \Delta_t v_y, \quad v'_x = v_x, \quad v'_y = v_y.$$

where the s'_x , s'_y , v'_x and v'_y are the new state random variables after a time interval of Δ_t . These equations can be written as a linear matrix equation as follows:

$$\begin{bmatrix} s'_x \\ s'_y \\ v'_x \\ v'_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x \\ s_y \\ v_x \\ v_y \end{bmatrix},$$

which, as we can see, has the following form:

$$Y = \mathbf{A}X.$$

¹These predictions can be corrected by measurements, as will be discussed in Section 5.1.

We can therefore implement this state evolution as a linear transformation of a Gaussian random variable. We will also add process noise to the transformed variable to encode the fact that the model is not perfect and that objects might move in a much less predictable manner. We can also allow the covariance of the noise to be proportional to the time interval to encode the increasing uncertainty about the state, with longer time intervals. With the addition of the process noise, the final equation therefore has the following form:

$$Y = \mathbf{A}X + N,$$

which is the same as equation 4.2.1, except for the constant, c , which will not be used in this work.

4.3 Non-Linear Transformations of Gaussian Random Variables

In this section we will consider a generalisation of the linear Gaussian distribution that allows approximate non-linear transformations of Gaussian random variables. We will refer to such a conditional distribution as a ‘non-linear Gaussian’ distribution.

Unlike the linear transformation of a Gaussian distribution, a non-linear transformation does not result in another Gaussian distribution. Furthermore, deriving the transformed distribution can be quite complex and practically useless if it does not have a suitable form for inference. One approach to solve this problem would be to sample from the distribution, transform the samples, and calculate the moments of the transformed sample distribution. This is essentially the idea behind the use of sigma points [31, p. 3], except that the samples are not random but deterministic, and the number of sigma points are orders of magnitude less than the number of samples typically required in sampling methods. The sigma point parameterisation allows us to fully specify a d -dimensional multivariate Gaussian distribution with $2d + 1$, d -dimensional points. These points are therefore chosen so as to have the same mean and covariance as the Gaussian they are parameterising.

We can apply a linear or non-linear transformation to these points and estimate the transformed distribution as a Gaussian distribution from the transformed points. If the transformation is linear, this estimate will be exact. If the transformation is non-linear, the mean and covariance estimates will be accurate to the third order of the Taylor series expansion of the transformation function [31, p. 1].

Below, we will discuss the calculation of the sigma points of a Gaussian distribution. The derivation and equations in this section are extracted from [30, pp. 48-51]. We will start the derivation with the definition of the sample covariance matrix:

$$\hat{\Sigma} = \frac{\sum_{i=1}^{d_s} w_i (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T}{\sum_i w_i}. \quad (4.3.1)$$

Here, x_i is the i 'th sigma point, w_i is its weight of the point, d_s is the number of points, and $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are the mean and covariance of the points, respectively. We can write the covariance of a Gaussian distribution using the Cholesky decomposition [32, p. 43] and the outer product method for matrix multiplication [33, p. 9], as follows:

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T = \sum_{i=1}^d L_i L_i^T \quad (4.3.2)$$

where d is the dimension of the covariance matrix, L is the lower triangular Cholesky factor of $\boldsymbol{\Sigma}$, L_i is the i 'th column of L and L_i^T is the i 'th row of L^T . We can ensure that the mean of the points is the same as that of the distribution by choosing the points to be symmetric around the mean of the distribution ($\boldsymbol{\mu}$), as follows:

$$\mathbf{x}_{i\pm} = \boldsymbol{\mu} \pm kL_i,$$

where k is the constant scalar that we need to find in this derivation. In the standard sigma point formulation, there is also a point at the mean of the distribution:

$$\mathbf{x}_0 = \boldsymbol{\mu}.$$

We choose all the weights to be the same (w) and rewrite equation (4.3.1) as follows:

$$\begin{aligned} \hat{\boldsymbol{\Sigma}} &= \frac{w \sum_{i=1}^d (\boldsymbol{\mu} + kL_i - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu} + kL_i - \hat{\boldsymbol{\mu}})^T + w \sum_{i=1}^d (\boldsymbol{\mu} - kL_i - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu} - kL_i - \hat{\boldsymbol{\mu}})^T}{(2d+1)w} \\ &+ \frac{w \sum_{i=1}^d (\boldsymbol{\mu} - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}})^T}{(2d+1)w} \\ &= \frac{w \sum_{i=1}^d (\boldsymbol{\mu} + kL_i - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu} + kL_i - \hat{\boldsymbol{\mu}})^T + w \sum_{i=1}^d (\boldsymbol{\mu} - kL_i - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu} - kL_i - \hat{\boldsymbol{\mu}})^T}{(2d+1)w}. \end{aligned} \quad (4.3.3)$$

Simplifying the above expression, we get:

$$\hat{\boldsymbol{\Sigma}} = \frac{wk^2 \sum_{i=1}^d L_i L_i^T + w(-k)^2 \sum_{i=1}^d L_i L_i^T}{2(d+1)w} = \frac{2wk^2 \sum_{i=1}^d L_i L_i^T}{(2d+1)w}. \quad (4.3.4)$$

Now, using the equality in equation (4.3.2), we can rewrite equation (4.3.4) above as follows:

$$\hat{\boldsymbol{\Sigma}} = \frac{2wk^2 \mathbf{L}\mathbf{L}^T}{(2d+1)w} = \frac{2k^2 \boldsymbol{\Sigma}}{2d+1}.$$

Thus, if $k = \sqrt{0.5(2d+1)}$, then the sigma point sample covariance will be equal to the covariance of the distribution. We can now use this value of k , the Cholesky factor of the distribution's covariance matrix and the equations below to calculate the sigma points of any Gaussian distribution.

$$\mathbf{x}_0 = \boldsymbol{\mu}, \quad \mathbf{x}_{i\pm} = \boldsymbol{\mu} \pm kL_i.$$

Once we have calculated the sigma points, we can apply any transformation to the points and calculate the (approximate) mean vector and covariance matrix of the transformed distribution. Figure 4.1 below shows an example of a linear sigma point transformation. Examples of the non-linear transformation of sigma points are given in Section 5.4.

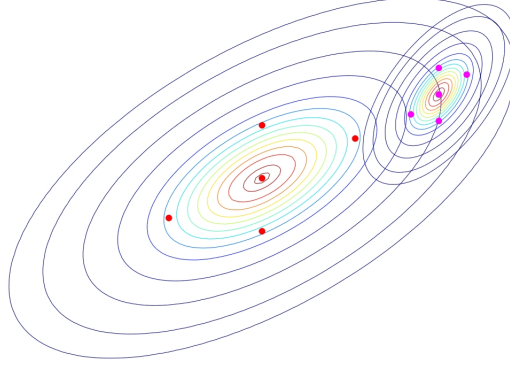


Figure 4.1: Linear transformation of Gaussian sigma points. The ellipses indicate the contours of the two distributions. The dots show the sigma points of the respective distributions.

Typically, we would also add (zero-mean) Gaussian noise to the transformed distribution. Here the noise could, for instance, represent the measurement noise associated with a specific sensor. The equations below can be used to calculate the approximate parameters of the transformed distribution $P(Y)$ and the joint distribution $P(X, Y)$ from the transformed points. The Σ_{NN} term is the covariance of the zero-mean Gaussian noise.

$$\begin{aligned}\boldsymbol{\mu}_Y &= \frac{1}{s_w} \sum_{i=0}^{2d_X} w_i \mathbf{y}_i, \\ \Sigma_{YY} &= \frac{1}{s_w} \sum_{i=0}^{2d_X} w_i (\mathbf{y}_i - \boldsymbol{\mu}_Y)(\mathbf{y}_i - \boldsymbol{\mu}_Y)^T + \Sigma_{NN} = \frac{1}{s_w} \left(\sum_{i=0}^{2d_X} w_i \mathbf{y}_i \mathbf{y}_i^T \right) - \boldsymbol{\mu}_Y \boldsymbol{\mu}_Y^T + \Sigma_{NN}, \\ \Sigma_{XY} &= \frac{1}{s_w} \sum_{i=0}^{2d_X} w_i (\mathbf{x}_i - \boldsymbol{\mu}_X)(\mathbf{y}_i - \boldsymbol{\mu}_Y)^T = \frac{1}{s_w} \left(\sum_{i=0}^{2d_X} w_i \mathbf{x}_i \mathbf{y}_i^T \right) - \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^T,\end{aligned}$$

with

$$s_w = \sum_{i=0}^{2d_X} w_i \quad (\text{and } d_X \text{ being the dimension of } X).$$

The joint distribution² $P(X, Y)$ can be expressed in terms of the above parameters as follows:

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_Y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}_{XY}^T & \boldsymbol{\Sigma}_{YY} \end{bmatrix} \right).$$

The sigma point parameterisation, together with the equations for calculating the mean and covariance from samples, therefore allow us to use any transformation in a non-linear Gaussian distribution. The method of using transformed sigma points to approximate distributions is known as the unscented transform. The sigma point formulation and unscented transform are very useful for inference in networks with non-linear dependencies.

4.4 Conclusion

In this chapter, we discussed the basic Gaussian operations and Gaussian random variable transformations required for inference in the graphical models that will be discussed in the rest of this text. We have seen how the different Gaussian parametric forms can be used for natural and efficient calculation of the parameters resulting from the various operations. Furthermore, we have shown that the linear transform of a Gaussian random variable is also Gaussian, and how to approximate the non-linear transform of a Gaussian random variable using sigma points and the unscented transform. In the next chapter we will examine how these operations and transformations are used in a Kalman filter graphical model. The specific non-linear transformation used in radar tracking will be discussed in Section 5.4.

²Approximate joint distribution, if the transformation is non-linear.

Chapter 5

The Kalman Filter as a Graphical Model

The seminal paper for the Kalman filter algorithm was published in 1960 [2], and the algorithm has been widely used ever since. It remains the provably optimal method for filtering in linear dynamic systems [34, p. 20] and, since the Kalman filter is based on principled probability theory, it can be described exactly within the framework of probabilistic graphical models.

In order to better understand the Kalman filter, we will first provide a basic overview of the algorithm and consider two visual examples. We will then show how the same algorithm can be viewed as message passing in a cluster graph and how the Kalman filter equations can be derived from the PGM perspective. Lastly, we will discuss the need for the unscented transform and investigate its accuracy.

5.1 The Linear Kalman Filter as a PGM

The Kalman filtering process is started with a prior distribution over the initial state of the system at time step 1. A measurement is then used to update the prior distribution to yield a posterior distribution over the state. The state corresponding to the maximum of this posterior is taken as the best estimate of the state, given past measurement observations. This is known as the filtered estimate. The state at time step 2 is predicted by transforming the posterior distribution using the state transformation (with additive noise). This distribution serves a prior distribution over the state at time step 2. The process is then repeated, starting again with a measurement update. This process is known as filtering. Once filtering is complete, the previous state distributions can also be refined by future measurement information. This process is known as smoothing and allows more accurate state estimates to be determined. These estimates are known as the smoothed estimates. An example of the distributions that

arise during filtering and smoothing is shown in Figure 5.1 below.

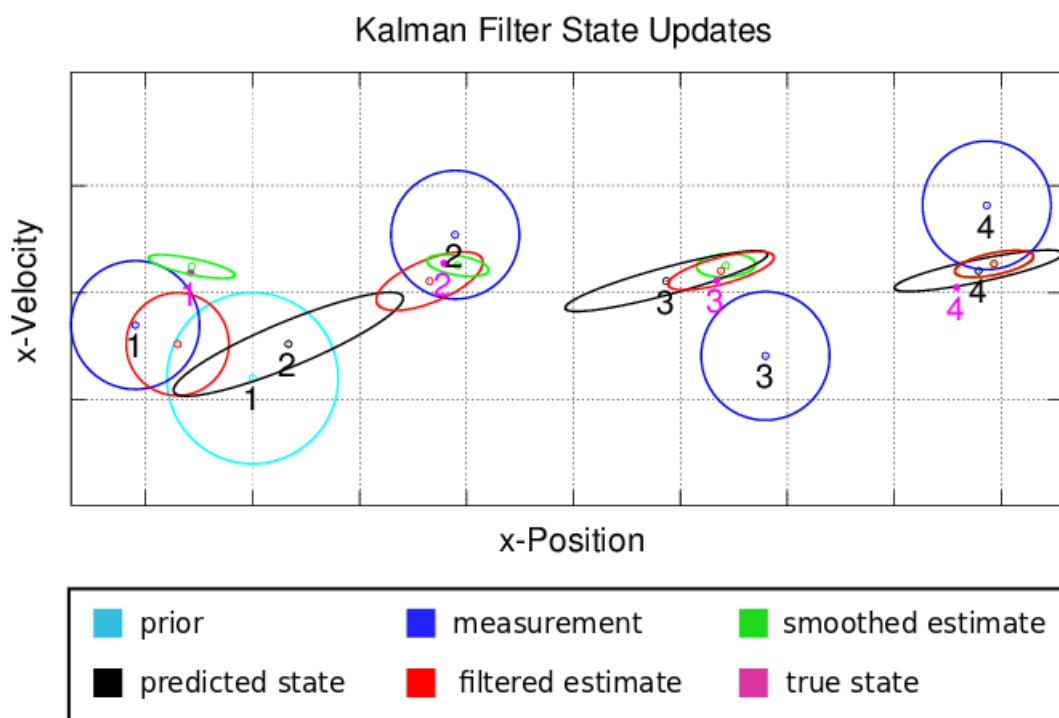


Figure 5.1: Kalman filter belief updates with fully observed state measurements. The covariance matrices and means of the various distributions are indicated with (1σ) error ellipses and the central dots respectively. The numbers underneath the means indicate the corresponding time step.

In Figure 5.1 we can see that the filtered estimates (red dots) are on average much closer to the true state (magenta dots and numbers) than the measurements. Furthermore, the smoothed estimates are even more accurate. From this example, we can also gain some intuition about the characteristics of Gaussian multiplication. If a prior distribution with a large variance is updated (multiplied) by a distribution with a relatively small variance, the distribution will be ‘pulled’ towards the mean of the updating distribution, and the updated distribution will have a much smaller variance. If the updating distribution has a large variance compared to the prior distribution, the prior will be largely unchanged. This effect also makes intuitive sense since a precise measurement should be trusted much more than an imprecise measurement. We will discuss this effect in more detail in Section 5.3. We can see this effect in the above figure, for example, where the predicted state (for time step 2) has a large variance in the direction of measurement 2. The updated distribution (filtered state 2) is therefore moved very far in this direction. The large correlation between the position and velocity, which is especially noticeable

in the predicted distributions, is also interesting. This correlation is due to the system dynamics that encode the relationships between the position and velocity across time steps. If the velocity is higher, the object would move a greater distance, and the position and velocity variables are therefore correlated.

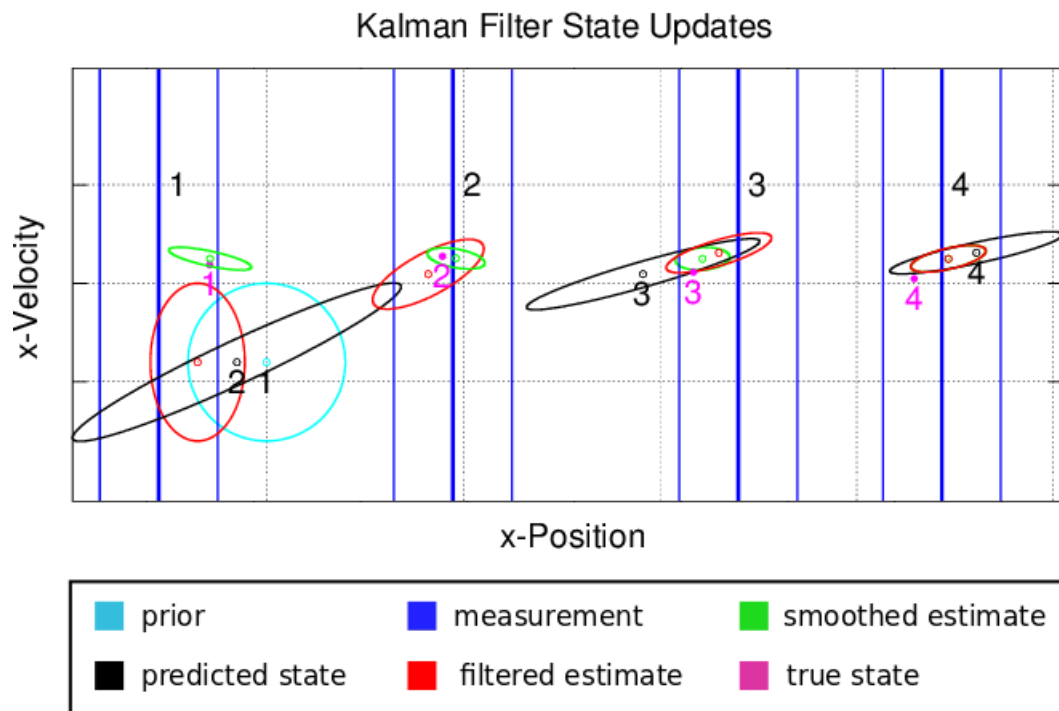


Figure 5.2: Kalman filter belief updates with partially observed state. The distribution covariance matrices and means are indicated with (1σ) error ellipses and the central dots respectively. The position measurement mean and standard deviation are indicated by the thick and thin blue lines respectively. The numbers underneath the means indicate the corresponding time step. Note that the velocity estimates are accurate despite the fact that the object's velocity is not measured.

In certain applications it might be the case that only partial measurements are available and that the dimension of the state that we wish to track is greater than the dimension of the measurement. In such cases the measurements will typically contain less information (even if we ignore the noise) than is necessary to fully describe the state. Surprisingly, accurate and full-state estimates can still be obtained even with such partial measurements. This is due to the correlation that exists between the state random variables because of the defined process dynamics. The model is therefore able to infer the probable object velocity by combining position

information over different time steps¹. An example of this is shown in Figure 5.2 above, where the velocity of the object is not measured. The blue lines therefore show only the mean and the standard deviation for the position measurement². Despite this missing measurement, we can, however, still obtain good velocity estimates. This characteristic of the Kalman filter allows it to be very useful in applications such as radar tracking, for example, where we cannot directly measure the angular velocity of an object.

Now that we have a better understanding of the operation of the Kalman filter, we will discuss how this algorithm can be viewed in terms of a graphical model. A Kalman filter can be described by a Bayes network, where each state is dependent on the previous state and each measurement is dependent on the state at the corresponding time step. An example of a Kalman filter Bayes network and corresponding cluster graph is shown in Figure 5.3 below.

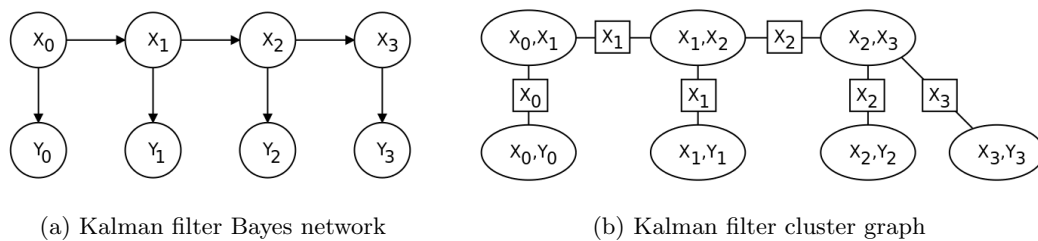


Figure 5.3: Kalman filter Bayes (a) network and cluster graph (b) example with four state random variables (X 's), representing the state evolution over four time steps, and four measurement random variables (Y 's).

In order to perform message passing in the above cluster graph, we need to determine an appropriate message-passing schedule. We will choose the message-passing order to correspond to the temporal order represented in the graph, and start at the first state cluster (the $\psi(X_0, X_1)$ cluster). From this cluster we will send a message to the connected measurement cluster $\psi(X_0, Y_0)$. The message that is sent is the state prior $P(X_0)$. This message will change the linear Gaussian distribution in the $\psi(X_0, Y_0)$ cluster into a Gaussian distribution. Next, we can condition the Gaussian distribution on the observed variable Y_0 . An upwards message back to the $\psi(X_0, X_1)$ cluster can then be calculated from the conditioned distribution. This message will update the prior distribution $P(X_0)$ to form the posterior $P(X_0|Y_0 = \mathbf{y}_0)$ ³. This is also the filtered distribution from which the filtered estimate can be extracted. The updated distribution $P(X_0|Y_0 = \mathbf{y}_0)$ can then be transformed through the linear state transformation to yield the predicted distribution. This distribution is sent as a message to the next state cluster.

¹This type of inference will, however, not be possible if the target state variables are independent.

²An unobserved velocity is effectively the same as having infinite variance in the velocity dimension of the measurement update distribution, and these distributions are therefore represented as parallel lines in Figure 5.2.

³Of course, this distribution forms part of the larger joint distribution $P(X_0, X_1|Y_0 = \mathbf{y}_0)$. We can, however, consider the $P(X_0|Y_0 = \mathbf{y}_0)$ and $P(X_1|Y_0 = \mathbf{y}_0)$ marginals separately, as we are not concerned with the joint distribution at this stage. The separate marginal representation is also more aligned with the Kalman filter view.

ter, $\psi(X_1, X_2)$. Here the message is used as a prior and the process can be repeated, starting with the measurement-cluster message pass. When the end of the graph is reached, messages can be propagated backwards through the state clusters⁴ to yield the smoothed distributions from which the smoothed estimates can be obtained. The filtered distributions can therefore be seen as the posterior marginals given all previous observations, while the smoothed distributions represent the true posterior marginals, given all observations. In the following section, we show how the Kalman filter equations relate to message-passing inference.

5.2 The Kalman Filter Equations

From the algorithm described in Section 5.1 above and using the Gaussian operations discussed in 4.1 and the linear Gaussian theory in Section 4.2, we can derive the equations that are often used in standard implementations of Kalman Filters. The linear state and measurement transformations that will be used in this derivation are given below:

$$X_{t+1} = \mathbf{A}X_t + \mathbf{c}_x + N_x, \quad Y_t = \mathbf{B}X_t + \mathbf{c}_y + N_y.$$

The ‘ N ’ terms in the above expressions are zero-mean Gaussian random variables that represent the additive process (state) and measurement noise. From the classical Kalman filter perspective, the filtering process involves two steps that need to be performed iteratively. The first step is a measurement update and the second the state prediction step. These two steps are described in detail below and the cluster graph interpretation is given at each step. Here, we will define messages from the state to measurement clusters as $\delta_{X,Y}$, and, similarly, the measurement to state and state to next state messages as $\delta_{Y,X}$ and $\delta_{X,X}$ respectively.

1. Measurement update - calculate $P(X_0|Y_0 = \mathbf{y})$.

This step is equivalent to sending the $\delta_{X,Y}$ message, observing evidence, and sending the $\delta_{Y,X}$ message back to the state cluster. The $\delta_{Y,X}$ message pass will involve the division of the state message that was received, but this distribution will immediately be multiplied in again when the message is received by the state cluster. The measurement update can therefore be calculated as simply the $\delta_{X,Y}$ message pass and evidence observation. Furthermore, the $\delta_{X,Y}$ message pass is equivalent to calculating the joint distribution $P(X_0, Y_0)$. The calculations for these two steps are shown below:

(a) Calculate the joint distribution:

$$P(X_0, Y_0) = \mathcal{N}([X_0, Y_0]; \boldsymbol{\Sigma}_m, \boldsymbol{\mu}_m),$$

⁴This corresponds to the smoothing process.

where (using the linear Gaussian equations from Section 4.2):

$$\begin{aligned}\boldsymbol{\Sigma}_m &= \begin{bmatrix} \boldsymbol{\Sigma}_{X_0X_0} & \boldsymbol{\Sigma}_{X_0Y_0} \\ \boldsymbol{\Sigma}_{Y_0X_0} & \boldsymbol{\Sigma}_{Y_0Y_0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{X_0} & \boldsymbol{\Sigma}_{X_0}\mathbf{B}^T \\ \mathbf{B}\boldsymbol{\Sigma}_{X_0}^T & \mathbf{B}\boldsymbol{\Sigma}_{X_0}\mathbf{B}^T + \boldsymbol{\Sigma}_{N_y} \end{bmatrix}, \\ \boldsymbol{\mu}_m &= \begin{bmatrix} \boldsymbol{\mu}_{X_0} \\ \boldsymbol{\mu}_{Y_0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{X_0} \\ \mathbf{B}\boldsymbol{\mu}_{X_0} + \mathbf{c}_y \end{bmatrix}.\end{aligned}$$

(b) Observe evidence:

$$P(X_0, Y_0 = \mathbf{y}_0) \propto \mathcal{N}(X_0; \boldsymbol{\Sigma}'_{X_0}, \boldsymbol{\mu}'_{X_0}),$$

where (using the joint distribution in (a) and equations from Section 4.1):

$$\begin{aligned}\boldsymbol{\Sigma}'_{X_0} &= \boldsymbol{\Sigma}_{X_0X_0} - \boldsymbol{\Sigma}_{X_0Y_0}\boldsymbol{\Sigma}_{Y_0Y_0}^{-1}\boldsymbol{\Sigma}_{Y_0X_0} \\ &= \boldsymbol{\Sigma}_{X_0} - \boldsymbol{\Sigma}_{X_0}\mathbf{B}^T(\mathbf{B}\boldsymbol{\Sigma}_{X_0}\mathbf{B}^T + \boldsymbol{\Sigma}_{N_y})^{-1}\mathbf{B}\boldsymbol{\Sigma}_{X_0}^T, \\ \boldsymbol{\mu}'_{X_0} &= \boldsymbol{\mu}_{X_0} + \boldsymbol{\Sigma}_{X_0Y_0}\boldsymbol{\Sigma}_{Y_0Y_0}^{-1}(\mathbf{y}_0 - \boldsymbol{\mu}_{y_0}) \\ &= \boldsymbol{\mu}_{X_0} + \boldsymbol{\Sigma}_{X_0}\mathbf{B}^T(\mathbf{B}\boldsymbol{\Sigma}_{X_0}\mathbf{B}^T + \boldsymbol{\Sigma}_{N_y})^{-1}(\mathbf{y} - (\mathbf{B}\boldsymbol{\mu}_{X_0} + \mathbf{c}_y)).\end{aligned}$$

By normalising the $P(X_0, Y_0 = \mathbf{y}_0)$ distribution, we can get the posterior distribution $P(X_0|Y_0 = \mathbf{y}_0)$. The mean and covariance parameters are of course not affected by normalisation, and the posterior distribution, in terms of the above parameters, is therefore :

$$P(X_0|Y_0 = \mathbf{y}_0) = \mathcal{N}(X_0; \boldsymbol{\Sigma}'_{X_0}, \boldsymbol{\mu}'_{X_0}).$$

2. **State prediction** - calculate $P(X_1|Y_0 = \mathbf{y}_0)$:

The state prediction step corresponds to calculating the marginal of the next state, which is equivalent to calculating the $\delta_{X,X}$ message to the next state cluster in a cluster graph. For this step, we start by calculating the joint state distribution $P(X_0, X_1|Y_0 = \mathbf{y}_0)$ using the linear Gaussian equations.

$$P(X_0, X_1|Y_0 = \mathbf{y}_0) = \mathcal{N}(X_0; \boldsymbol{\Sigma}'_s, \boldsymbol{\mu}'_s),$$

where:

$$\begin{aligned}\boldsymbol{\Sigma}'_s &= \begin{bmatrix} \boldsymbol{\Sigma}_{X_0X_0} & \boldsymbol{\Sigma}_{X_0X_1} \\ \boldsymbol{\Sigma}_{X_1X_0} & \boldsymbol{\Sigma}_{X_1X_1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{X'_0} & \boldsymbol{\Sigma}_{X'_0}\mathbf{A}^T \\ \mathbf{A}\boldsymbol{\Sigma}_{X'_0} & \mathbf{A}\boldsymbol{\Sigma}_{X'_0}\mathbf{A}^T + \boldsymbol{\Sigma}_{N_x} \end{bmatrix}, \\ \boldsymbol{\mu}'_s &= \begin{bmatrix} \boldsymbol{\mu}_{X'_0} \\ \mathbf{A}\boldsymbol{\mu}_{X'_0} + \mathbf{c}_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{X'_0} \\ \boldsymbol{\mu}_{X_1} \end{bmatrix}.\end{aligned}$$

We know from the covariance form marginalisation equations in Section 4.1 that we can marginalise a distribution in covariance form simply by extracting the relevant covariance matrix and mean vector partitions. The predicted state distribution (and message to the next state cluster), in terms of the above parameters, is therefore:

$$P(X_1|Y_0 = \mathbf{y}_0) = \mathcal{N}(X_0; \boldsymbol{\Sigma}_{X_1X_1}, \boldsymbol{\mu}_{X_1}).$$

We can now use the posterior distribution above as a new prior and repeat the process, starting with the next measurement update. In summary, then, we have seen how message passing in a PGM corresponds to the Kalman Filter equations. These equations are summarised below.

The Kalman Filter Equations [23, p. 85]

Measurement Update:

$$\begin{aligned}\Sigma'_{X_0} &= \Sigma_{X_0} - \mathbf{K}_G \mathbf{B} \Sigma_{X_0}, \\ \mu'_{X_0} &= \mu_{X_0} + \mathbf{K}_G (\mathbf{y} - (\mathbf{B} \mu_X + \mathbf{c}_y)).\end{aligned}$$

State Prediction:

$$\begin{aligned}\Sigma_{X_1} &= \mathbf{A} \Sigma'_{X_0} \mathbf{A}^T + \Sigma_{N_x}, \\ \mu_{X_1} &= \mathbf{A} \mu'_{X_0} + \mathbf{c}_x.\end{aligned}$$

With:

$$\mathbf{K}_G = \Sigma_{X_0} \mathbf{B}^T (\mathbf{B} \Sigma_{X_0} \mathbf{B}^T + \Sigma_{N_y})^{-1} \quad (\text{Kalman Gain}).$$

5.3 Message Passing Characteristics

In this section we take a closer look at how Gaussian messages update distributions through Gaussian multiplication. In order to investigate this process, we consider a simple example where a one-dimensional state distribution is updated by a one-dimensional measurement message. This process is described by the equations below:

$$\psi'(x) = \psi(x) \delta(x),$$

with:

$$\psi'(x) = \mathcal{N}_{us}(x; \sigma_{us}^2, \mu_{us}), \quad \psi(x) = \mathcal{N}_s(x; \sigma_s^2, \mu_s), \quad \delta(x) = \mathcal{N}_m(x; \sigma_m^2, \mu_m)$$

being the updated distribution, the original distribution and the measurement message, respectively. The parameters of the updated factor $\psi'(X)$ can be shown (using the Gaussian multiplication equations in Section 4.1.2) to be as follows:

$$\sigma_{us}^2 = \frac{\sigma_s^2 \sigma_m^2}{\sigma_s^2 + \sigma_m^2}, \quad \mu_{us} = (1 - k) \mu_s + k \mu_m, \quad \text{with} \quad k = \frac{\sigma_s^2}{\sigma_m^2 + \sigma_s^2}.$$

From the updated variance (σ_{us}^2) above, we can see that, if the variance of the measurement message (σ_m^2) is much larger than the state variance σ_s^2 , then the state variance stays

approximately the same. This is an intuitive result, since a very noisy measurement should not have a large effect on the certainty about the state. Furthermore, the new state variance will always be smaller than the prior state variance and the amount by which it is decreased is inversely proportional to the measurement variance. We can therefore naturally expect a precise measurement to increase the certainty about the state significantly⁵.

Furthermore, looking at the expression for k , we can see that $k \approx 1$ if $\sigma_m^2 \approx 0$ and that this will result in the new state mean being approximately equal to the mean of the measurement. If, however, σ_m^2 is much larger than the state variance σ_s^2 , then $k \approx 0$ and the new mean will be approximately equal to the state mean before the update. This makes sense, since we expect a precise measurement to change the mean of the state much more than an imprecise measurement, and we would expect the updated state mean to be equal to the measurement mean if the measurement is perfect.

5.4 The Unscented Kalman Filter and the Cartesian-Polar Transform

The PGM for the unscented Kalman filter (UKF) has exactly the same structure as the Kalman filter described in Section 5.1. The UKF, however, allows for non-linear transformations by making use of the sigma point representation and the unscented transform. This functionality is critical for certain applications such as radar tracking, where the measurement transformation is non-linear.

In the previous sections we discussed the theory behind Kalman filters from a PGM perspective. In this section we will investigate the quality of the approximation that is provided by the unscented transform (discussed in Section 4.3). For this purpose, we will specifically consider the Cartesian-polar transformation. The Cartesian-polar transformation is used in radar tracking as it relates the Cartesian space, in which we intuitively think and movement is more naturally described in⁶, to the polar space in which the radar measurements are represented. The equations for this transformation are given below. The derivations for the velocity equations are given in appendix A.4.

⁵Here we assume that the certainty of the state was previously much lower than the accuracy of the measurement. In other words, the variance of the state prior is much larger than that of the measurement message.

⁶The ground based objects that we will be tracking in this work tend to move in straight lines due to momentum. Of course tracking in polar space could be more sensible and convenient if we were tracking objects moving in circles or spirals, such as objects orbiting in space.

Cartesian to polar transformations

$$r = \sqrt{x^2 + y^2}, \quad \theta = \arctan(y/x),$$

$$v_r = \frac{xv_x + yv_y}{r}, \quad v_\theta = \frac{xv_y - yv_x}{r},$$

with: x : x position, v_x : x velocity,
 y : y position, v_y : y velocity,
 r : radius, v_r : radial velocity,
 θ : angle, v_θ : angular velocity.

The transformation functions described by the above equations are plotted below:

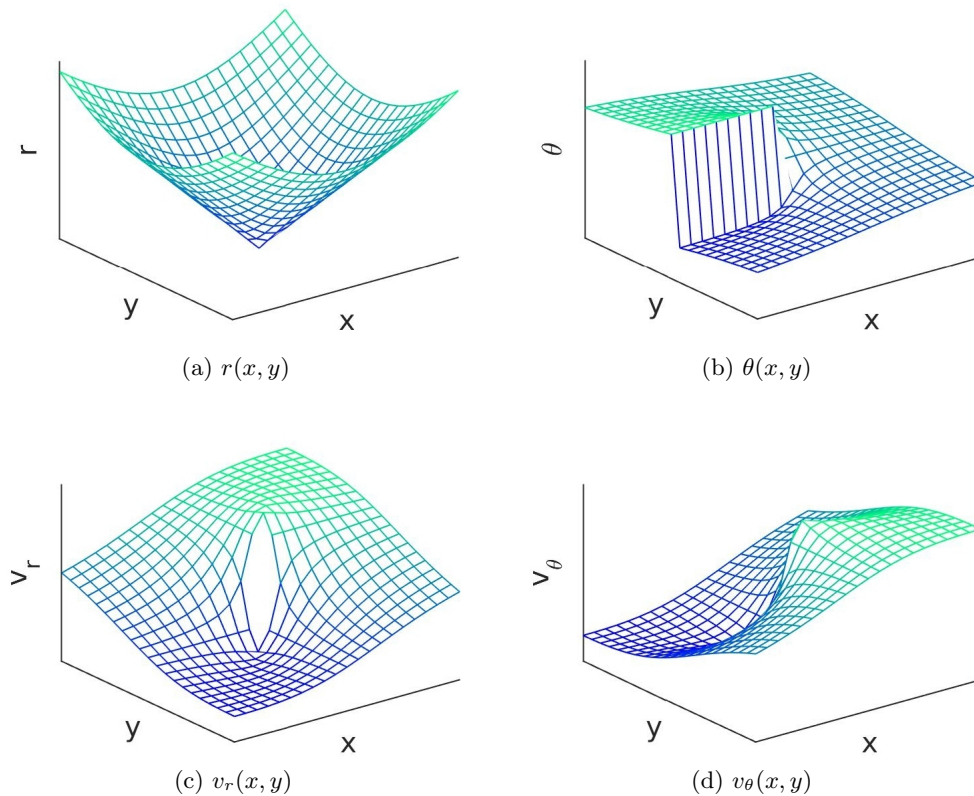


Figure 5.4: Cartesian space (x, y) and velocity (v_x, v_y) to polar space (r, θ) and velocity (v_r, v_θ) transformation functions. Note that the v_x and v_y velocities are constant in (c) and (d). Their values do not change the fundamental shape of the functions. The discontinuity in (b) occurs on the negative x ($y = 0$) line because θ increases from 0° to 180° in the anti-clockwise direction and decreases from 0° to -180° in the clockwise direction. These graphs serve to give the reader an idea of the nature of the non-linearities that is inherent in the Cartesian-polar transform.

From Figure 5.4, above, we can see that, although the functions are highly non-linear, they are at least relatively smooth (except for the discontinuity in the $\theta(x, y)$ function) and therefore approximately linear over most small regions. We also note that most non-linear areas are close to the origin.

We now consider how a Gaussian distribution is changed when these transformations are applied. This can be done through sampling from the Cartesian state distribution and transforming the sampled points. We will, however, not be able to visualise the four-dimensional polar distribution and therefore only consider the two-dimensional marginals of the transformed distribution. It should also be noted that low-dimensional marginals of high-dimensional distributions can be misleading. It is possible, for instance, for the marginal of a highly non-Gaussian distribution to have a Gaussian shape. Bearing these limitations in mind, however, we should still be able to gain some insight by investigating the sampled marginals. We start with an example where the unscented transform results in a particularly inaccurate approximation.

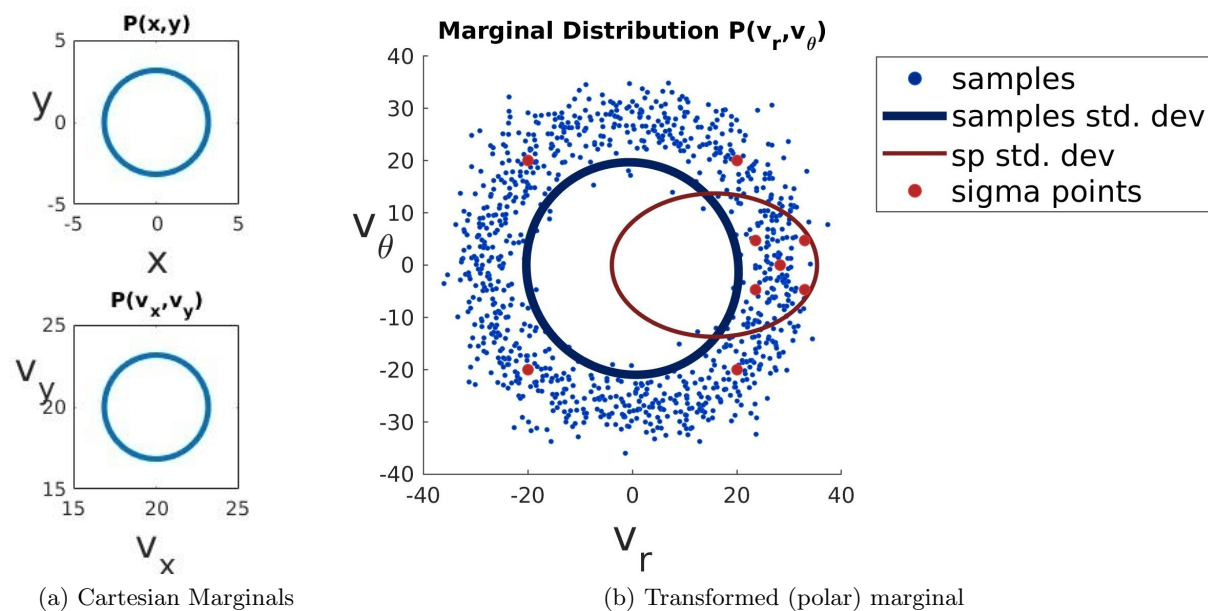


Figure 5.5: Sigma point polar velocity example 1, showing poor sigma point approximation when Cartesian position marginals are near the origin. (sp: sigma points)

The marginal distribution shown by the samples in Figure 5.5b above arises when the Cartesian spatial marginal (see the top plot in Figure 5.5a) has a mean very close to the origin. Note that this plot also reflects our earlier observation that the polar transforms have the greatest non-linearities close to the origin. This is evident from the fact that the transformed distribution should be Gaussian for a linear transformation, and the shape of the above distribution is very different from a Gaussian. In Figure 5.5b above we see that the polar transformed distribution,

or at least the marginal of the distribution, is ring-shaped. The reason for this shape can be understood by first fixing the Cartesian velocity to a single point. If the size of the Cartesian velocity is fixed to a certain value, the size of the polar velocity must be fixed to the same value, since it is just a different representation of the same velocity. The valid combinations of radial and angular velocities must therefore lie on a circle. If the Cartesian position is distributed symmetrically around the origin, each point on the polar velocity circle will be equally probable, and the polar velocity will therefore be distributed uniformly over the circle. If the Cartesian velocity is a distribution around a specific velocity, instead of a fixed value, this will cause the circle to have a non-zero width and become ring-shaped. Approximating such a distribution as a Gaussian will of course result in a very poor approximation. In such a case, a solution could be to use sampling methods for inference, or to somehow split the Gaussian distribution into a Gaussian mixture before transforming the distribution. As can be seen in Figure 5.6 below, however, this specific example should not present a problem in our case - this figure shows that moving the Cartesian spatial mean even a relatively small distance away from the origin results in a much more Gaussian-shaped transformed distribution. When the Cartesian position is moved away from the centre, the polar velocity distribution becomes concentrated on a small part of the ring. This allows for a more accurate Gaussian approximation.

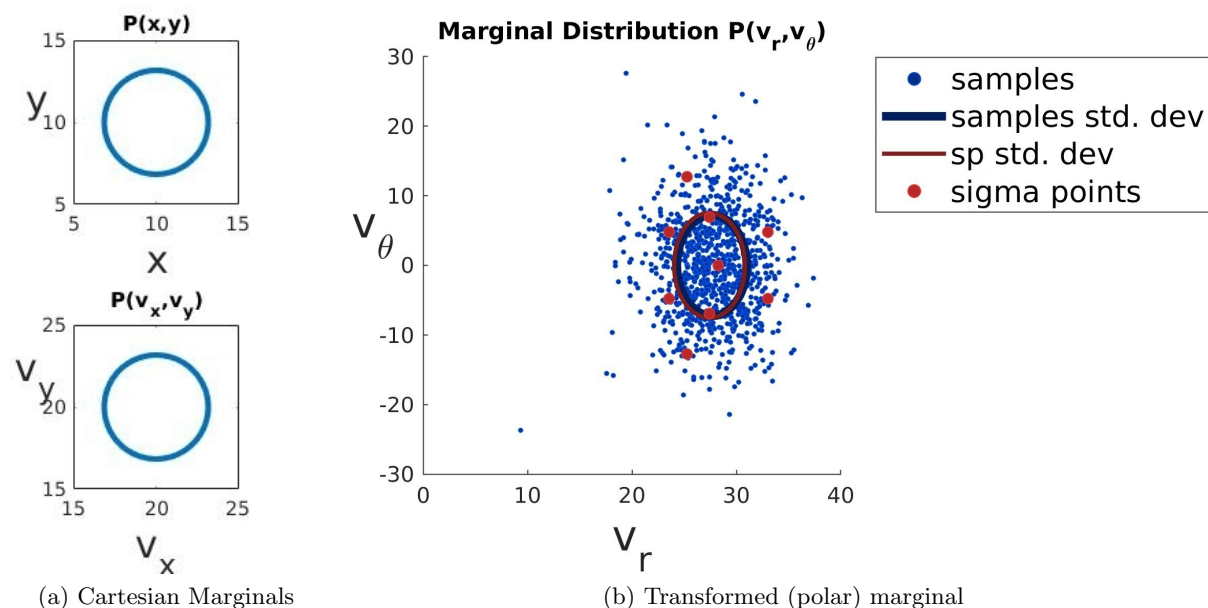


Figure 5.6: Sigma point polar velocity example 2, showing improved sigma point approximation (compared to Figure 5.5b) when the Cartesian state marginals are moved away from the origin. Note that the sigma point mean and covariance are virtually identical to that of the samples, as is evident from the two standard deviation error ellipses that are almost exactly aligned. (sp: sigma points)

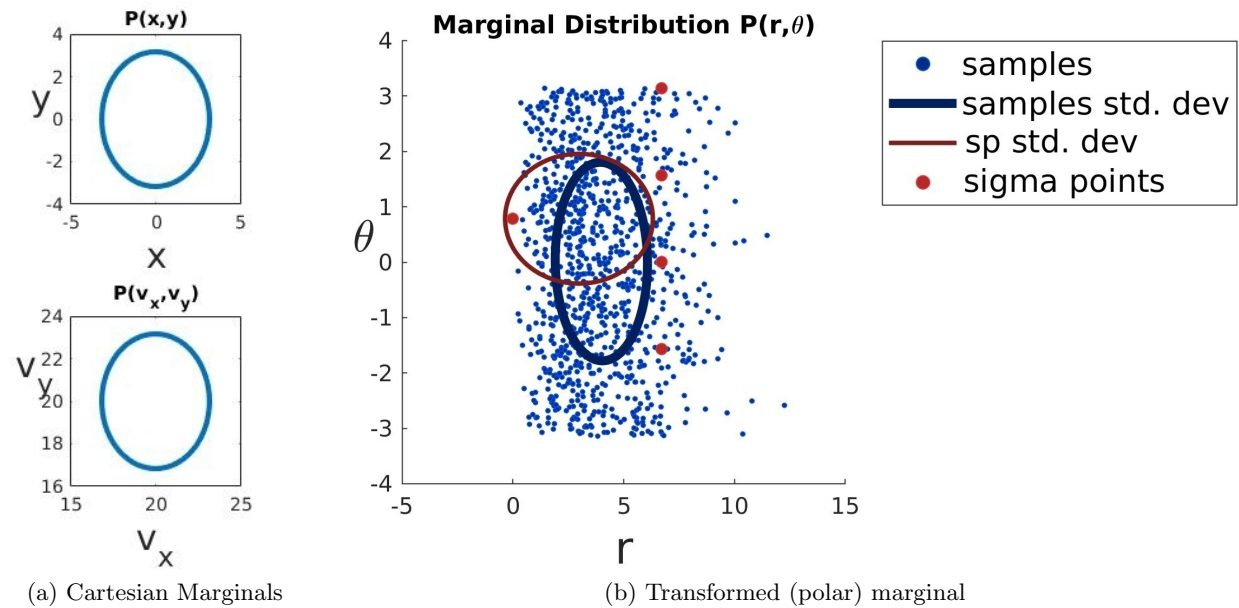


Figure 5.7: Sigma point polar spatial example 1 showing poor sigma point approximation when Cartesian position marginals are near the origin. (sp: sigma points)

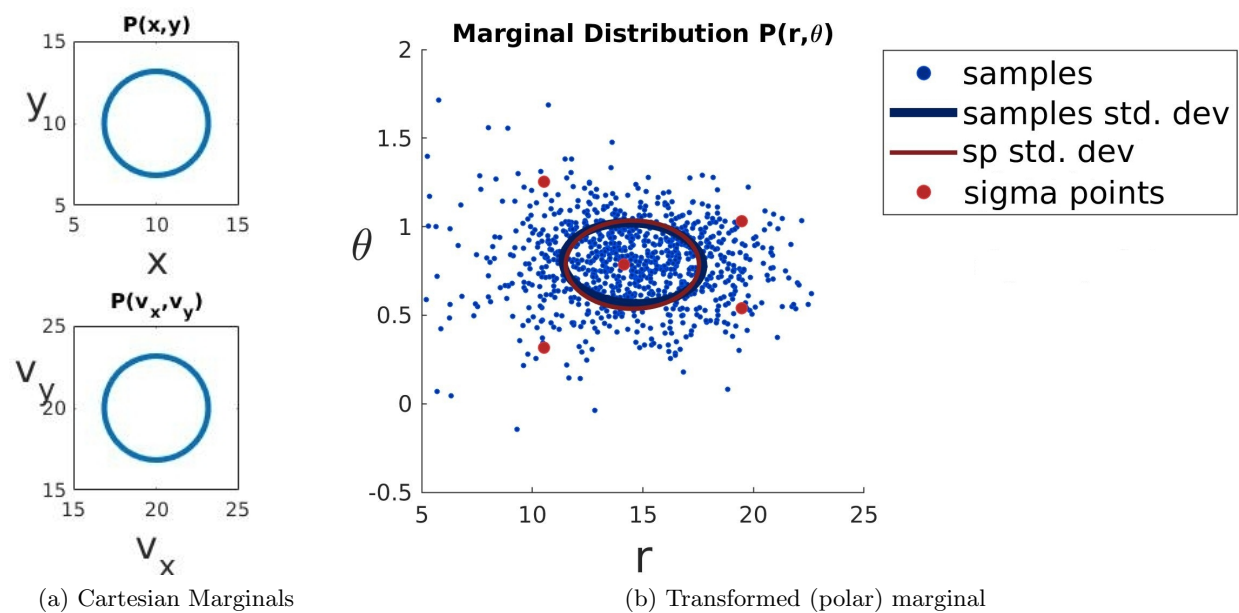


Figure 5.8: Sigma point polar spatial example 2 showing improved sigma point approximation (compared to Figure 5.7b) when the Cartesian state marginals are moved away from the origin. (sp: sigma points)

We see a similar result, although not quite as extreme, when looking at the polar spatial (r, θ) marginal. Figure 5.7 above shows the transformed marginal when the Cartesian spatial marginal is near the origin. Figure 5.8 shows an improved result when the Cartesian spatial mean is further away from the origin. Here we can see that the sigma point mean and covariance are virtually identical to that of the samples. This is evident from the two standard deviation error ellipses that are almost exactly aligned.

From this investigation, we can therefore conclude that using the unscented transform approximation with the Cartesian-polar transformation can yield both very accurate and very inaccurate results. Conveniently, however, the inaccurate results seem to arise only when the Cartesian distribution is close to the origin. When the distribution is further away from the origin, as it would typically be in tracking applications, the approximations can be very accurate. Lastly, the discontinuity in the angle function will of course also result in very poor approximations. It will, however, often be possible to avoid this region when tracking does not need to be performed over the full 360 degrees.

5.5 Conclusion

In this chapter we discussed how the operations and transformations from the previous chapter can be used for inference in a Kalman filter PGM. We showed how the state distributions are updated with the measurement information and how predictions about the next state are made through message passing. Lastly, we noted that the unscented transform is required for inference when the measurement transform is non-linear, as it is in radar tracking applications. We have found that this transform and the subsequent Gaussian approximation can yield both very accurate and very inaccurate approximations. Fortunately, it seems that the cases where poor approximations arise will rarely occur in our application, if at all. In the following chapter we will see how the basic Kalman filter graphical model can be extended to solve the multiple object tracking problem.

Chapter 6

A PGM for Multiple Object Tracking

The Kalman filters described thus far can only be used to track the state of a single object. We could use multiple parallel Kalman filters for multiple object tracking if we knew the number of targets and which detections corresponded to which target. This scenario, however, seems unlikely to occur in practice. It is therefore critical for multiple object tracking (MOT) systems to solve the data association problem in order to automatically determine the probability of associations between objects and measurements. In this chapter, we discuss the development of a PGM that solves the data association problem and incorporates the tracking capabilities of the Kalman filter in order to solve the multiple object tracking problem. We will first consider a mathematical example and a corresponding visual example where the number of targets is known and constant. We will then show how Bayesian model selection can be used to infer the number of targets automatically from the detections. Finally, in order to allow the model to perform tracking in the presence of false detections, a clutter classification PGM is developed and incorporated into the object tracking PGM.

6.1 A Simplified Example: Two Object Tracking

We will first consider a simplified problem in order to gain an intuition about the inference that is performed by a multiple object tracking PGM. In this example, two objects are tracked simultaneously, with a single detection that could be from either object at each time step. A more general model that can track any number of targets and process multiple measurements at each time step will be discussed in Section 6.3 and onwards. Figure 6.1 below shows the two time-slice Bayes network (2TBN¹) for this simplified problem. In this graph we can still see the

¹2TBNs allow graphs with repeating structure and variable length to be represented compactly.

basic Kalman filter structure with the X state nodes and Y measurement nodes. As with the standard Kalman filter, the Y variables are the only variables that are ever observed. The a nodes are association nodes. These nodes represent discrete variables and the distributions of the Y nodes are therefore dependent on both continuous and discrete variables. After message passing, the a nodes will contain the probabilities of each measurement being associated with each object (X node). The subscripts of the variables in this section indicate which object they correspond to, while the superscripts are the time step indices. The X_1^1 node, for example, is the random state vector of object one at the first time step, and X_1^2 is the random state vector of the same object at the second time step. The subscripts of the a and Y symbols will be used in later sections to distinguish them from other Y 's and a 's in the same time slice.

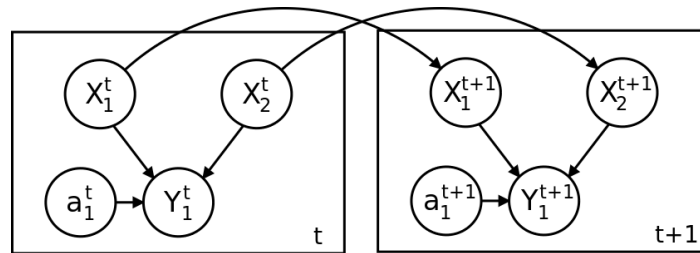


Figure 6.1: Two time-slice Bayes network (2TBN) for two object tracking, with sing. The partial graphs in the rectangles show the structure for a single time-slice of the larger graph, while the connections between the rectangles indicate how the graph is connected between time-slices

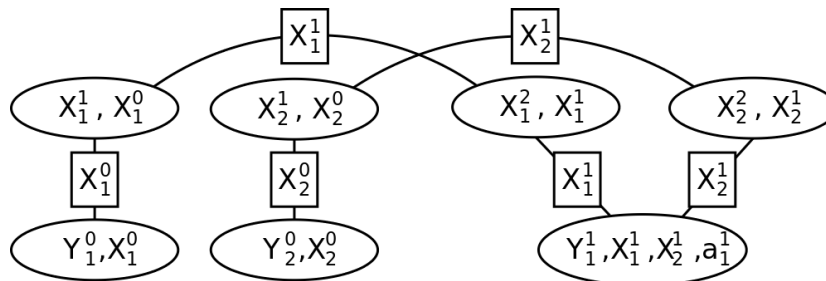


Figure 6.2: Example of a short section (over only two time steps) of a two object tracking cluster graph.

In order to explain the operation of this model in more detail, we will first discuss how the belief update algorithm (discussed in Section 3.3) is applied to the cluster graph in Figure 6.2 and further investigate the characteristics of the different factors and messages that are sent. In this section we will describe the graph mathematically, and a visual example is given in Section 6.2. Looking at the cluster graph in Figure 6.2, we can see that each state cluster ($\psi(X_i^t, X_i^{t+1})$)

in the first cluster-slice² is connected to only one measurement cluster ($\psi(Y_j^t, X_i^t)$). Since the object state priors are identical, and the object labels in the graph are arbitrary, we can assign any measurement to any object in the first cluster-slice of the graph. After that, however, the track identities need to remain consistent. We will therefore focus this discussion on the three clusters in the second cluster-slice, where the measurement cluster is connected to both state clusters. Note that we will neglect the superscripts for a cleaner notation in the rest of this discussion; this should not cause confusion since we will only be focussing on variables for $t = 1$. We will also drop the subscripts for a and Y , since there is only one of each in this example. We will start by discussing the most complex cluster, the measurement cluster. The measurement cluster potential is an instantiation of a conditional non-linear Gaussian factor. This conditional distribution is shown below:

a	$P(Y X_1, X_2, a, \Gamma)$
1	$P(Y X_1)$
2	$P(Y X_2)$

where Γ is the full set of measurements that have been received. Note that in the table above, $P(Y|X_1, X_2, a = i, \Gamma)$ is only a function of X_i and Y . If a message with the scope X_i is received, it will therefore affect the conditional distribution through the specified relationship between Y and X_i . If, however, the incoming message has a different scope, the variables in the message scope are independent of those at that table entry. The distribution in the first table entry ($a = 1$) is therefore independent of X_2 and the distribution in the second entry ($a = 2$) is independent of X_1 . The measurement cluster's potential, after messages from all neighbouring clusters have been received, is as follows:

$$\begin{array}{|c|c|} \hline a & P(Y, X_1, X_2, a|\Gamma) \\ \hline 1 & P(Y|X_1, \Gamma)P(X_1|\Gamma)P(X_2|\Gamma)P(a = 1) \\ \hline 2 & P(Y|X_2, \Gamma)P(X_1|\Gamma)P(X_2|\Gamma)P(a = 2) \\ \hline \end{array} = \begin{array}{|c|c|} \hline a & P(Y, X_1, X_2, a|\Gamma) \\ \hline 1 & P(a = 1)\mathcal{N}_1([X_1, X_2, Y]; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \hline 2 & P(a = 2)\mathcal{N}_2([X_1, X_2, Y]; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ \hline \end{array}$$

where: Γ is the set of all previous measurements and

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \boldsymbol{\Sigma}_{X_1X_1} & \mathbf{0} & \boldsymbol{\Sigma}_{X_1Y} \\ \mathbf{0} & \boldsymbol{\Sigma}_{X_2X_2} & \mathbf{0} \\ \boldsymbol{\Sigma}_{YX_1} & \mathbf{0} & \boldsymbol{\Sigma}_{YY} \end{bmatrix}, \quad \boldsymbol{\mu}_1 = [\boldsymbol{\mu}_{X_1}, \boldsymbol{\mu}_{X_2}, \boldsymbol{\mu}_Y]$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} \boldsymbol{\Sigma}_{X_1X_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{X_2X_2} & \boldsymbol{\Sigma}_{X_2Y} \\ \mathbf{0} & \boldsymbol{\Sigma}_{YX_2} & \boldsymbol{\Sigma}_{YY} \end{bmatrix}, \quad \boldsymbol{\mu}_2 = [\boldsymbol{\mu}_{X_1}, \boldsymbol{\mu}_{X_2}, \boldsymbol{\mu}_Y].$$

Note that the zeros in the covariance matrices above show the independence of one of the objects from the other and the measurement. Note that we do not have to explicitly multiply

²Here the term 'cluster-slice' refers to a group of clusters where the prior variables all correspond to the same time step. Note that the term time-slice is not used, only because the conditional state variables (X^t in $P(X^t|X^{t+1})$) in the state clusters correspond to the following time step.

the independent distributions and these distributions are therefore kept separate as shown in the table below. The $P(a)$ probabilities will typically be uniformly distributed, unless we have additional information about the measurement associations. Once the measurement cluster has received all messages, the measurement data can be used in the conditioning operation. The $\psi(Y, X_1, X_2, a)$ cluster's potential, conditioned by the evidence, can be expressed as follows:

a	$P(Y = \mathbf{y}, X, a \Gamma)$
1	$P(Y = \mathbf{y}, X_1 \Gamma)P(X_2 \Gamma)P(a = 1)$
2	$P(Y = \mathbf{y}, X_2 \Gamma)P(X_1 \Gamma)P(a = 2)$

Note that each entry in the above table is now a weighted multivariate Gaussian, where each weight is the product of the probability of observing the evidence and the prior probability that the measurement is associated with the specific object. In order to send a message back to the $\psi(X_1, X_2)$ state cluster, we first have to cancel the effect of the message received from the $\psi(X_1, X_2)$ cluster, as per the LBU message passing procedure. After this operation, the Y cluster potential is:

a	$P'(Y = \mathbf{y}, X, a \Gamma)$
1	$P(Y = \mathbf{y}, X_1 \Gamma)P(X_2 \Gamma)P(a = 1)/P(X_1 \Gamma)$
2	$P(Y = \mathbf{y}, X_2 \Gamma)P(X_1 \Gamma)P(a = 2)/P(X_1 \Gamma)$

We now marginalise out all the variables, except for X_1 , in order to calculate the message to be sent to the $\psi(X_1, X_2)$ cluster. Since the Y variable has been observed and the distribution is no longer a function of Y , we only need to marginalise over the X_2 and a variables. The message distribution is therefore:

$$\begin{aligned}
\delta_{Y, X_1}(X_1) &= \frac{1}{P(X_1|\Gamma)} \int_{\mathbb{R}^d} \sum_a P'(Y = \mathbf{y}, X_1, X_2, a|\Gamma) dX_2 \\
&= \frac{1}{P(X_1|\Gamma)} \int_{\mathbb{R}^d} (P(Y = \mathbf{y}, X_1|\Gamma)P(X_2|\Gamma)P(a = 1) + P(Y = \mathbf{y}, X_2|\Gamma)P(X_1|\Gamma)P(a = 2)) dX_2 \\
&= \frac{P(Y = \mathbf{y}, X_1|\Gamma)P(a = 1)}{P(X_1|\Gamma)} \int_{\mathbb{R}^d} P(X_2|\Gamma) dX_2 + P(a = 2) \int_{\mathbb{R}^d} P(Y = \mathbf{y}, X_2|\Gamma) dX_2 \\
&= \frac{P(Y = \mathbf{y}, X_1|\Gamma)}{P(X_1|\Gamma)} P(a = 1) + P(a = 2) \int_{\mathbb{R}^d} P(Y = \mathbf{y}, X_2|\Gamma) dX_2 \\
&= P(Y = \mathbf{y}|X_1, \Gamma)P(a = 1) + P(a = 2) \int_{\mathbb{R}^d} P(Y = \mathbf{y}, X_2|\Gamma) dX_2. \tag{6.1.1}
\end{aligned}$$

The first term equation 6.1.1 above will update the X_1 potential as in the normal Kalman filter, but with the additional weighting component $P(a = 1)$, due to the association uncertainty.

The second term is a constant of which the value is the relative probability³ that the measurement was caused by object two. We can view this constant as an unnormalisable, vacuous Gaussian⁴. This representation will be useful for implementation purposes. In this work we will therefore also see message functions, such as the one in equation 6.1.1, as a Gaussian mixture and will refer to it as such. The effect of this message is that both the prior distribution over X_1 as well as the measurement updated distribution will be present in the posterior distribution over X_1 . The prior component will, however, be scaled by the probability that the measurement was caused by the other objects, and the updated component by the probability that the measurement is indeed associated with the object. This is a very intuitive result - if there is a high probability that the measurement was generated by one of the other objects, then the prior component will have a large weight and the measurement updated component will have a small weight in the posterior. The distribution in this case will therefore be largely unchanged, as it should be. If, on the other hand, it is very probable that the measurement is associated with the object, the distribution will be updated in much the same way as in a Kalman filter⁵. When the message is received by the object state cluster, the state distribution is updated as follows:

$$\begin{aligned} P'(X_1) &= P(X_1, Y = \mathbf{y}|\Gamma) = P(X_1|\Gamma)\delta_{Y,X_1}(X_1) \\ &= P(Y = \mathbf{y}|X_1, \Gamma)P(a = 1)P(X_1|\Gamma) + \left(P(a = 2) \int_{\mathbb{R}^d} P(Y = \mathbf{y}, X_2|\Gamma) dX_2 \right) P(X_1|\Gamma). \end{aligned}$$

Note that, although the state clusters represent a distribution over the current and the next states, we keep this distribution factorised here and focus only on the distribution over the current state⁶ (denoted here as X_1). From the above equation it is evident that the first term, as mentioned earlier, is just the normal updated distribution, as in a normal Kalman filter, but it is weighted by the probability that the measurement is associated with the object. The second term is a non-updated (prior) distribution weighted by the probability that the measurement was caused by another object.

We could now, as usual, use this updated distribution to perform a prediction step and repeat the process. This would, however, result in an exponential increase in the number of Gaussian mixtures in the network and therefore an exponential increase in computation time. At this stage we will therefore approximate the Gaussian mixture as a single Gaussian. For this purpose, we use the method of moment matching (also known as M-projection). Moment matching⁷ is a method by which a distribution of a certain type can be approximated by a distribution of a different type by setting the relevant moments of the approximating distribution equal to that of the original. In this case, this corresponds to calculating the mean and covariance of the

³These ‘probabilities’ of a measurement being associated with a certain object and being associated with any of the other objects will, however, not sum to one, but can be interpreted by comparing the one relative to the other.

⁴A vacuous Gaussian is a Gaussian with infinite variance, or a zero precision matrix.

⁵This will in fact be exactly the same if the probability of association with the object is one.

⁶This is mainly to increase the clarity of the example.

⁷Approximating a function through moment matching is equivalent to minimizing the Kullback-Leibler divergence, $\mathbb{D}(p||q)$, between the true (p) and approximate (q) distributions [1, pp. 620].

Gaussian mixture and approximating the mixture by a Gaussian with this mean and covariance. The equations for approximating a Gaussian mixture as a Gaussian through moment matching is given below.

Moment Matching for approximating a Gaussian mixture as a Gaussian [1, pp. 620]

$$\mathcal{M} \left(\sum_{i=1}^k \mathcal{N}(X; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right) = \mathcal{N}(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

with:

$$\boldsymbol{\mu} = \frac{1}{\sum_{i=1}^k (w_i)} \sum_{i=1}^k (w_i \boldsymbol{\mu}_i),$$

$$\boldsymbol{\Sigma} = \frac{1}{\sum_{i=1}^k (w_i)} \left(\sum_{i=1}^k (w_i \boldsymbol{\Sigma}_i) + \sum_{i=1}^k (w_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T) \right)$$

and k being the number of components in the mixture.

After approximating the Gaussian mixture posterior distribution as a Gaussian, we can perform prediction steps for each object and repeat the steps described above to continue the filtering process. In the next section we will consider a more visual example in order to make the algorithm described here less abstract.

6.2 A Visual Example of Two Object Tracking

The various distributions and their interactions in the example in the previous section may be difficult to visualise. In order to provide the reader with a better intuition about these distributions and operations, a visual example is given here. In this example, each of the two objects only has a single parameter (i.e. a one dimensional position) and the measurement transformation is linear, with some measurement noise. Here we visualize the types of distributions that are formed during the message passing process, as described mathematically in the previous section. Firstly, the prior distributions from each of the state clusters are sent as messages to the measurement cluster. An example of such priors is shown in Figure 6.3 below.

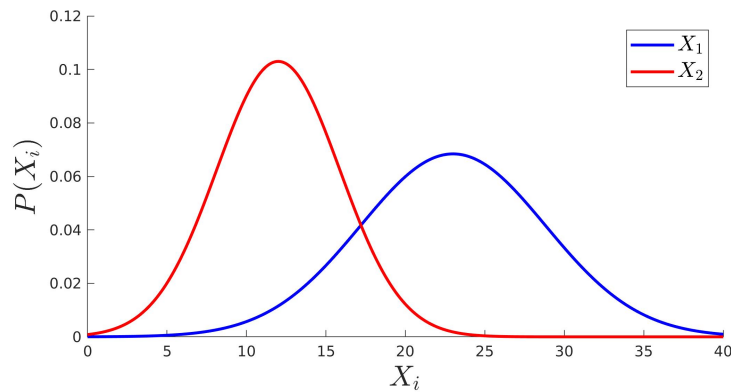


Figure 6.3: Prior distributions over the states (one dimensional positions) of two objects. Note that the two distributions are shown on the same axis for comparison, although they have different scopes.

These prior state distributions are multiplied with the linear Gaussian distributions in each of the table entries in the conditional linear Gaussian measurement factor. Each of these linear Gaussian distributions are therefore changed into Gaussian distributions. These two distributions are shown in Figure 6.4 below. Here it should be noted that this is not a Gaussian mixture; the distributions are merely plotted in the same space for comparison. Furthermore, the independent components that are also multiplied with each respective entry are not shown here. These distributions are shown in the full joint distribution plots in Figures 6.5 and 6.6.

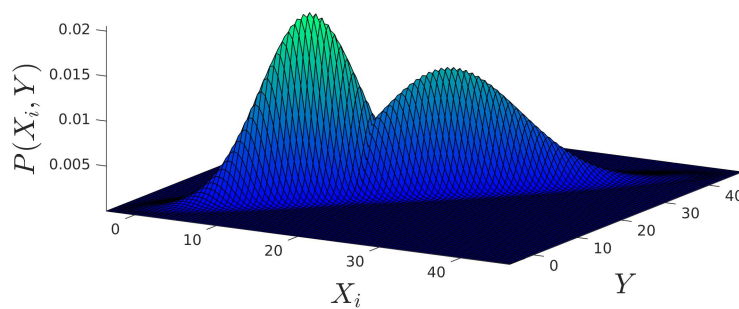


Figure 6.4: Linear-Gaussian distributions defined by messages from state clusters. Note that the two distributions are shown on the same axes for comparison, although they have different scopes.

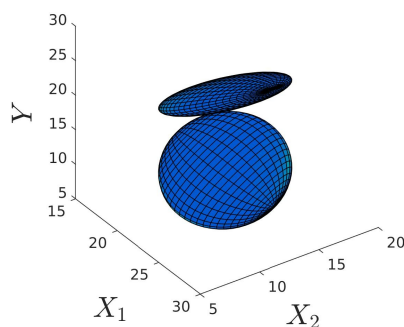


Figure 6.5: Full joint $P(X_1, X_2, Y, a)$ distribution. Note that the two distributions correspond to different assignments of the a variable.

Figure 6.5 above shows the full conditional Gaussian joint distribution. Here, standard deviation ellipsoids are used to depict the three dimensional distributions⁸. Each of these ellipsoids corresponds to a specific assignment of the a variable. Figure 6.6a below shows the plot in Figure 6.5 as seen in the X_1 direction. From this perspective we can see that Y and X_2 are independent in the top distribution (with $a = 1$). This distribution corresponds to the first table entry that contains the $P(Y|X_1)$ linear Gaussian. Similarly, if we align the graph to look in the X_2 direction as depicted in Figure 6.6b, we can see that Y and X_2 are independent in the bottom distribution (with $a = 2$). This distribution corresponds to the second table entry in the conditional linear Gaussian factor. These independencies are due to the fact a certain object's state is independent of the measurement if the measurement was generated by the other object. Furthermore, the objects are in general independent of one another.

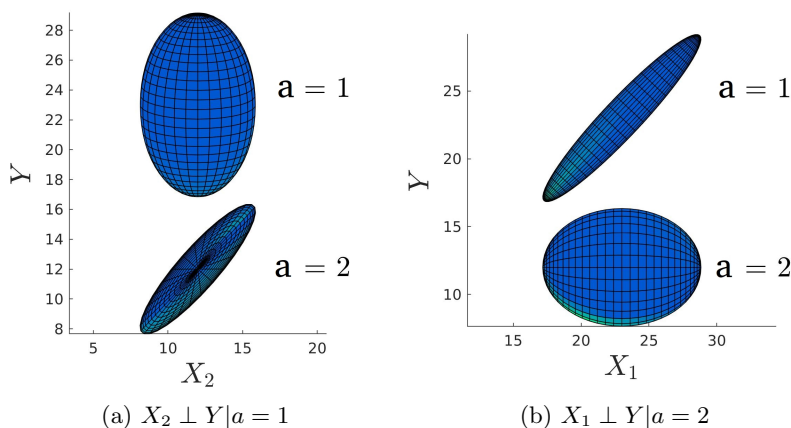


Figure 6.6: Two views on $P(X_1, X_2, Y, a)$ distribution showing conditional independencies.

⁸The actual distributions do of course have an infinite extent in all three dimensions.

Now that we have a full joint distribution over the state and measurement variables, we can observe evidence before calculating the messages to update the state clusters. An example of this process is shown in Figure 6.7 below. This graph shows the joint distribution with an intersecting surface that represents a certain measurement value being observed.

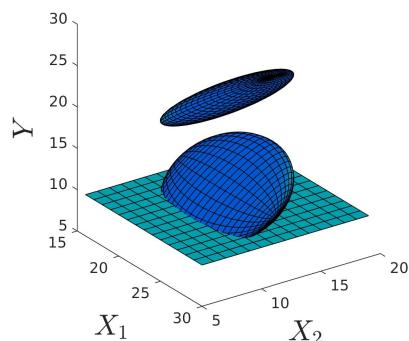


Figure 6.7: Observing evidence by conditioning the joint distribution $P(Y, X_1, X_2, a)$ on a measurement (indicated by the 2 dimensional plane).

After observing a certain value of Y , the Y variable is eliminated from the distribution and the dimensionality of the distribution is reduced. The resulting distribution after the evidence observation is shown in Figure 6.8 below. Although the measurement is much closer to the mean of the $P(Y, X_1, X_2, a = 2)$ distribution, the reduced component of the $P(Y, X_1, X_2, a = 1)$ distribution can also be seen (indicated by the red ellipse) in the reduced distribution below, although its weight is very small.

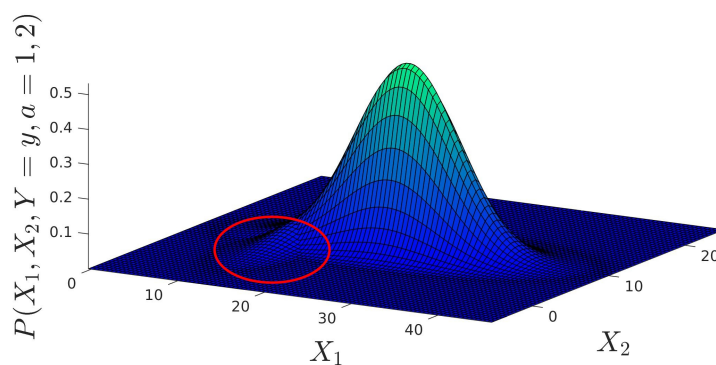


Figure 6.8: Distributions reduced by evidence: Dominating $P(X_2, Y = \mathbf{y})$ and much smaller $P(X_1, Y = \mathbf{y})$ indicated by the red ellipse.

We now need to calculate the messages to be sent back to the state clusters. We will start with the message to the X_1 state cluster. In order to calculate this message, we need first to marginalise out all other variables. The distribution resulting from the marginalisation operation is shown in Figure 6.9a below. This distribution has the form of a Gaussian mixture with weighted Gaussian components. The blue component corresponds to the case where the measurement is associated with X_1 ($a = 1$), and the magenta distribution corresponds to the case where the measurement is associated with the other object (with state X_2).

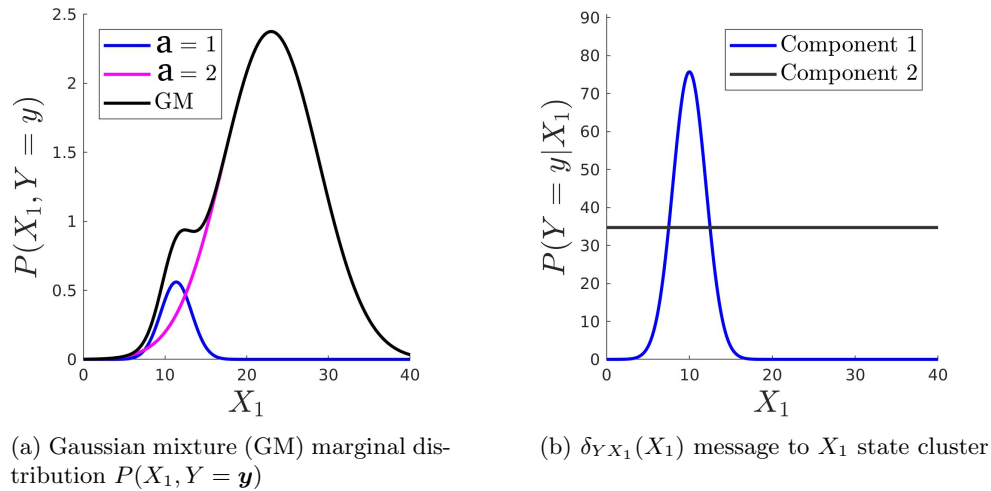


Figure 6.9: $\delta_{Y X_1}(X_1)$ measurement-state message calculation, showing the marginal distribution in (a) and Gaussian mixture message with one vacuous component in (b).

The final step needed to calculate the message to the X_1 cluster is to divide out the message that was originally received from the X_1 cluster. This division results in a Gaussian mixture with one vacuous component, as discussed in the previous section. The message to be sent back to the X_1 cluster is shown in Figure 6.9b. Multiplication of this message with the prior state distribution in the X_1 state cluster gives us the (unnormalised) posterior⁹ over X_1 . We can now follow the same process to compute the message for the X_2 cluster. The marginalised distribution is shown in Figure 6.10a and the message is shown in Figure 6.10b. Here we again have a Gaussian mixture of a well-defined and a vacuous Gaussian.

⁹The posterior distribution, in this case, is the same as the distribution shown in Figure 6.9a. This will, however, not always be the case. If there were more than one measurement and a specific state cluster was connected to more than one measurement cluster, for example, the posterior would differ from the reduced and marginalised distributions in the measurement clusters. For this reason, the general case is discussed here.

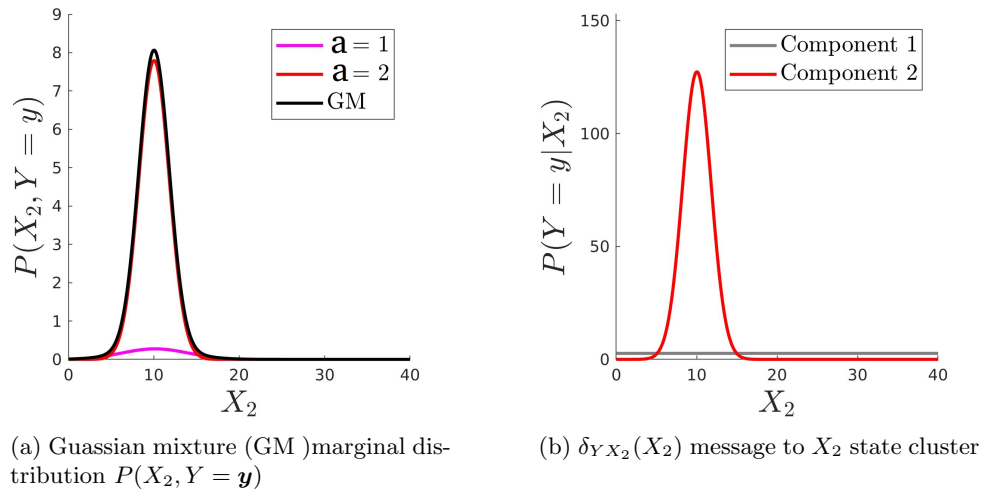


Figure 6.10: $\delta_{Y X_2}(X_2)$ measurement-state message calculation, showing the marginal distribution in (a) and Gaussian mixture message with one vacuous component in (b).

Finally, we can update the two prior distributions by multiplying with the X_1 and X_2 messages respectively. Each posterior distribution will, as discussed previously, effectively be a superposition of the updated and non-updated states, with the weights of the components being equal to the relative probabilities that the specific distribution should be updated or not. The updated state Gaussian mixture distributions (as well as the M-projection approximations) are shown in Figure 6.12. The same distributions are shown on the same axis, with the measurement for comparison in Figure 6.11. From these figures we can clearly see that the measurement is most likely associated with the second object (with state X_2). The measurement update therefore has a large effect on this distribution, while barely changing the distribution over X_1 . From these figures we can also see that the M-projection approximations are reasonably accurate (since the Gaussian mixtures are already quite close to being Gaussian). There will, however, be cases where the M-projection results in less accurate approximations. An example of this is given in Figure 6.13.

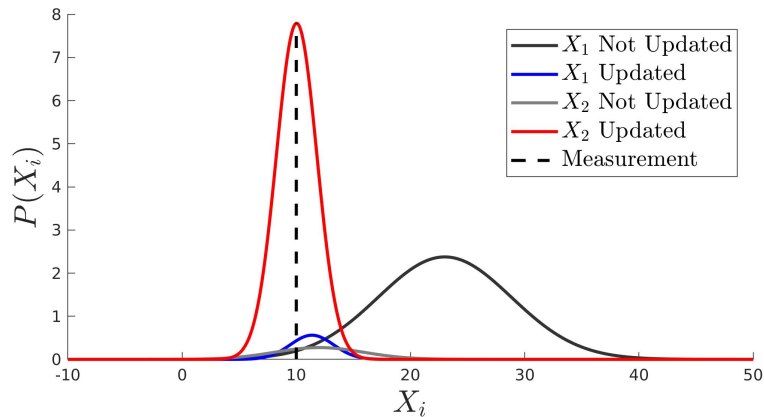
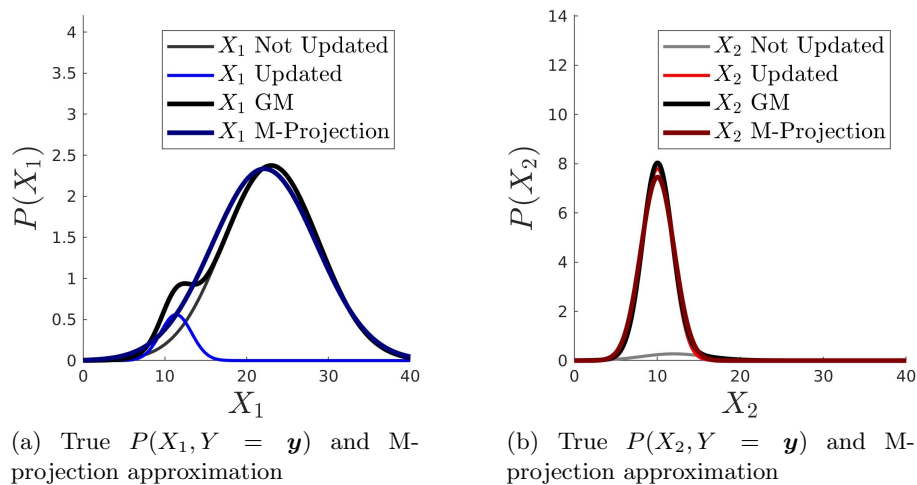


Figure 6.11: The Gaussian mixture posterior components for X_1 and X_2 . Note that the two distributions are shown on the same axis for comparison, although they have different scopes. The connotations of the different Gaussian components are shown in the legend. Here we can see that the measurement is most likely associated with object 2 (with state X_2), since the updated X_2 component is much larger than the prior ('not updated') component and the updated component of the X_1 distribution is much smaller than its prior component.



(a) True $P(X_1, Y = \mathbf{y})$ and M-projection approximation

(b) True $P(X_2, Y = \mathbf{y})$ and M-projection approximation

Figure 6.12: True Gaussian mixture posterior state distributions and corresponding Gaussian approximations. The posterior (and M-projection approximation) of the X_1 distribution is largely unchanged from the prior ('not updated') distribution in (a). The posterior (and M-projection) of the X_2 distribution is significantly moved towards the measurement (at $Y = 10$) and the variance is decreased in (b). These two very different update effects are due to the high probability that the measurement is associated with object 2.

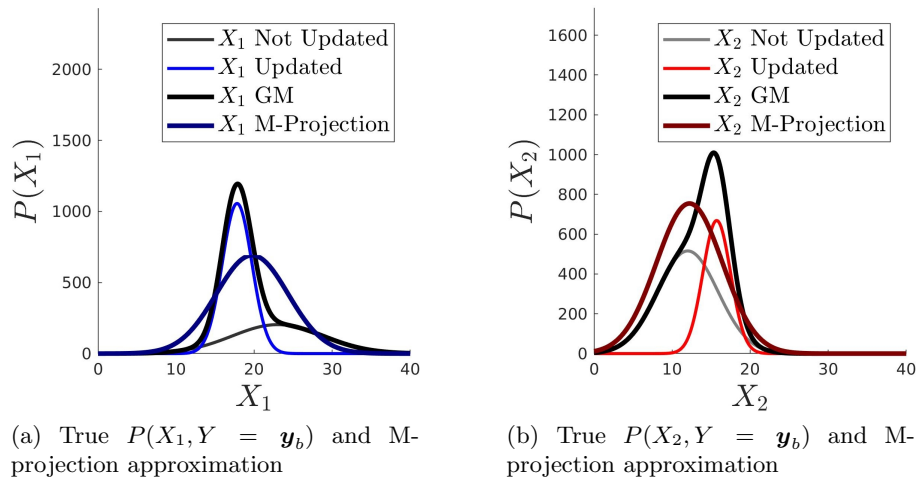


Figure 6.13: Another example of Gaussian mixture (GM) posterior state distributions and corresponding Gaussian M-projection approximations, showing less accurate approximations when the measurement is ambiguous and the true GM posterior distributions are less Gaussian.

Looking at the approximations in Figure 6.13 above, one might wonder how inaccurate the approximations can be in general. Since the approximating distribution is uni-modal, we can expect the approximation to be especially inaccurate when the Gaussian mixture has prominent modes that are separated by a large distance. Typically, however, if the targets have sufficiently different states¹⁰ and the measurement noise is low enough, the Gaussian mixtures will only have a single dominant mode. This is because of the fact that the mixture component corresponding to the measurement update will have a weight that is inversely proportional to how far it is from the prior mean (if the prior was a single Gaussian). The result is that the further the posterior modes are separated, the smaller one of the components will be. This will allow a reasonable approximation of the mixture as a Gaussian distribution. On the other hand, if the targets have very similar states, the measurements will also be close together and measurement updates will also not cause strongly multi-modal posteriors. So, in summary, if the target states are very different or very similar, the true posteriors will be well approximated by single Gaussians. In the case where they are very similar, each target's state distribution will become a kind of average of the targets' states if the measurement is not accurate enough to distinguish between the targets. Since the targets states are very similar, such an average distributions will still allow accurate state estimates to be inferred. The above observations explain why accurate multiple object tracking in this model is possible even with the seemingly extreme M-projection approximation.

Somewhere between the extremes of the objects being very far apart and very close together, the approximation will, however, be maximally inaccurate. In such scenarios, the use of Gaussian

¹⁰If the true target states are reasonably far from each other in the state space.

mixtures for the state posteriors can indeed improve tracking efficiency. An example of such a scenario is presented in Section 8.3. The use of Gaussian mixtures with the LBU message passing algorithm is, however, not trivial. The main difficulties that arise with the use of Gaussian mixtures are: limiting the number of mixture components, and performing Gaussian mixture division, which has no closed form solution. These problems and possible solutions are discussed in Chapter 8.

6.3 Bayesian Model Selection

In sections 6.1 and 6.2, the operation of the multiple object tracking (MOT) PGM was described using an example with two targets. In a general multiple object tracking problem, however, the number of targets will neither be fixed nor known. In order to allow multiple objects to be tracked under these conditions, the model needs to be able to infer, with each new set of detections, whether or not any of the detections are from new objects. In this section we will discuss a Bayesian mechanism for this type of inference. Here we will be discussing the evaluation of different hypotheses, corresponding to different graphs, models, or distributions. Note that the terms ‘hypothesis’, ‘model’ and ‘model distribution’ will therefore have very similar meanings in this section and will be, to some extent, used interchangeably here. In Section 7.1 we discuss an alternative view of the classical model selection method which is described here.

We have previously discussed probabilistic inference concerning random variables in graphs. One can, however, also use similar methods to reason about the probability of entire graphs given the available data. This process is known as model selection and can be explained through the use of Bayes’ theorem. Bayes’ theorem, as it is applied to the model selection problem, is given below:

Bayes’ Theorem for Model Selection [35, p. 347]

$$P(\mathcal{H}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i)}{P(\mathcal{D})},$$

\mathcal{H}_i = hypothesis (specific model/graph),
 \mathcal{D} = data.

In the above expression, $P(\mathcal{H}_i|\mathcal{D})$ is the posterior probability of the hypothesis, $P(\mathcal{D}|\mathcal{H}_i)$ is the data likelihood, and $P(\mathcal{D})$ is the normalising constant (also known as the ‘evidence’). In order to get the posterior probability of the model given the data, we need the three terms on the right of the equation. In order to calculate $P(\mathcal{D})$, however, we will need to sum over all

possible models (M):

$$P(\mathcal{D}) = \sum_{i \in M} P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i).$$

This is obviously not possible in general and we will therefore only seek to find the relative probability between different hypotheses. This will allow us to find the most likely model in the subset of models that we will evaluate. The expression for this relative probability is given below.

$$\frac{P(\mathcal{H}_i|\mathcal{D})}{P(\mathcal{H}_j|\mathcal{D})} = \frac{P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i)/P(\mathcal{D})}{P(\mathcal{D}|\mathcal{H}_j)P(\mathcal{H}_j)/P(\mathcal{D})} \quad (6.3.1)$$

$$= \frac{P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i)}{P(\mathcal{D}|\mathcal{H}_j)P(\mathcal{H}_j)} \quad (6.3.2)$$

The likelihood term in the above expression is the probability of the data, given the respective models. We can calculate the likelihood term by calculating the normalising constant of the joint distribution represented by a graph after the data has been observed. In a tree-structured graph, this probability could be exactly calculated by calculating the normalisation constant of any of the marginals after passing all messages once. The multiple object tracking PGM is, however, not tree-structured and messages have to, in theory, be iteratively passed until convergence. In practice, however, for this particular graph structure, passing each message once has been found to be sufficient, with very little change in the distributions with additional message passing. We will therefore calculate the data likelihood by finding the normalisation constant of one of the cluster distributions after a single message passing iteration.

We now have a method to calculate the data likelihood and turn our attention to the last term necessary to compute the probability ratio of two models, namely the model prior $P(\mathcal{H})$. If we have no reason to think that any model is inherently more probable than another, we can simply specify an uninformative (flat) prior over the model space (this is the approach that will be taken in this implementation). If, on the other hand, we do have some idea about the prior probability of a model, we can encode this knowledge into the model prior. Here we must note that it is not necessary to incorporate a preference for simpler models into this prior. Such an Occam's razor-like effect occurs naturally when conditioning different model distributions on data. A more complex model, with more parameters, needs to spread its probability mass more thinly over more parameters to fit a variety of observations. Conversely, a simpler model with a more concentrated¹¹ probability mass will make more precise predictions. Data that is consistent with the simpler model will therefore result in a higher data likelihood for that model, compared to the more complex models¹² [35, p. 344].

¹¹This can be concentrated relative to another distribution with more variance in the same space, or concentrated because of decreased dimensionality.

¹²Note that the original principle stated by the philosopher William of Ockham was *pluralitas non est ponenda sine necessitate, plurality should not be posited without necessity* [36] and not "the simplest theory is the most likely theory". This is an important distinction, since it allows for models to be sufficiently complex if they are supported by sufficiently complex evidence. This mechanism is also present in Bayesian model selection.

Now that we have a method for evaluating models we will look at how model selection is used in the context of the MOT PGM. Here, we must again make one more approximation. Because the number of hypotheses will grow exponentially with the length (in time-slices) of the models that we wish to compare, we will typically only consider a small subset the full set of hypotheses. As new measurements are received, all the hypotheses (graphs) will be expanded (both in length and in number), and when the specified length is reached we will find the most probable hypothesis (graph) and add it to the existing graph. This process will then be repeated iteratively.

We will now consider a simple example to illustrate the model selection process as it is applied in the MOT PGM. In this example, detections are received from two targets and a detection from each target is received at each time step (although the detection associations are still unknown). At each time step for each existing hypothesis, the model selection algorithm will consider models with a minimum number of targets equal to the number of targets tracked in the previous time step, and a maximum equal to this number plus the number of detections received (at the current time step). The minimum corresponds to all detections being associated with existing targets, and the maximum to all associations being from new targets. A subset of the considered hypotheses is shown in Figure 6.14, and the probability distribution over all the considered hypotheses is shown in 6.15. The full set of hypothesis graphs can be found in appendix B.1.

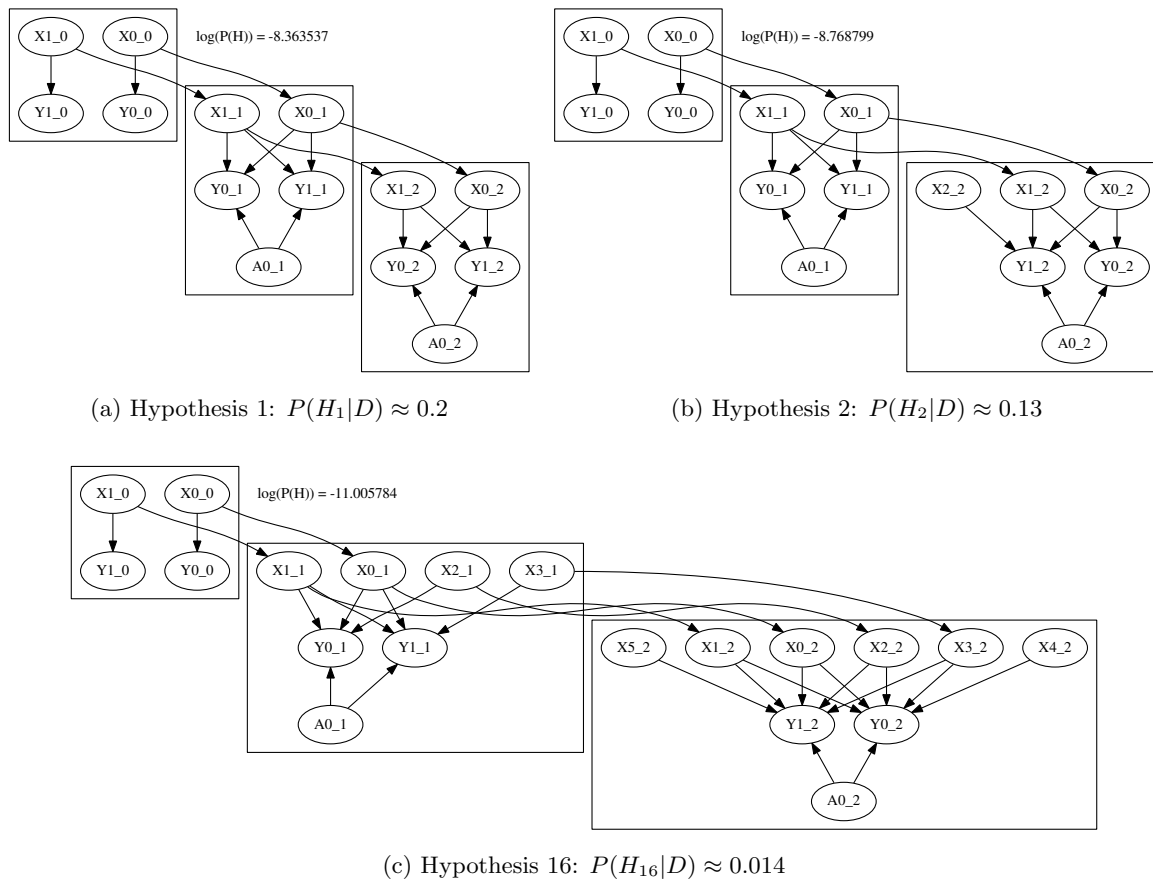


Figure 6.14: Model selection example showing the most and least likely hypothesis. The separate rectangles indicate different time-slices. The hypothesis in (a) has the highest calculated probability and is also the correct model. The hypothesis in (b) is that one additional object is present at time step three, but this is unlikely and the hypothesis probability is much lower than hypothesis 1 in (a). The hypothesis in (c) is that each measurement corresponds to a new target and is the least likely out of all the hypotheses.

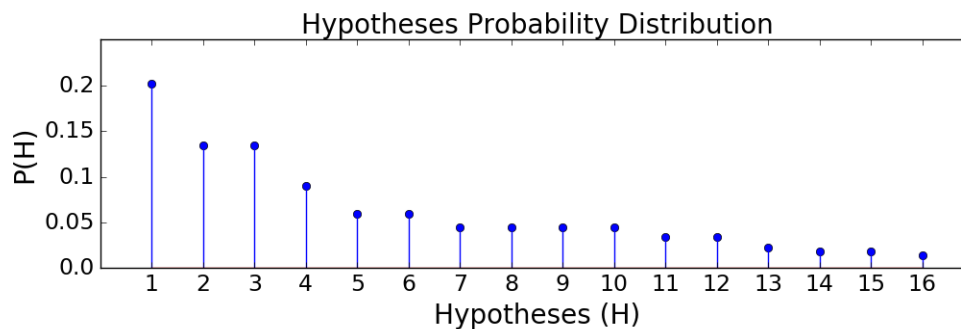


Figure 6.15: The probability distribution over model hypotheses, showing the correct hypothesis (hypothesis 1 - shown in Figure 6.14a) with the highest probability. Higher model numbers correspond to more complex models and also have decreasing probability. This shows the Occam's razor like effect that is inherent in Bayesian model selection.

As we can see from the above graphs, the evidence for the correct model is higher than any of the other models and the model selection algorithm therefore selects the correct model. We can also see that more complex models have lower probability and that models with similar complexity have similar probabilities. Note that for any given model, each object state cluster (except for new objects) is connected to all measurement clusters. The new objects are only connected to a single measurement cluster in order to avoid multiple new targets with the same priors from being updated by the same set of measurements. This would cause the new target posterior distributions to be the same and prevent effective updates to the new target distributions. The densely connected structure between the established object clusters and measurement clusters allows the model to maintain some flexibility, even after a specific model has been selected.

In this section, we have discussed the theory of classical Bayesian model selection and shown how it can be effectively used for inferring the number of targets in conjunction with the developed PGM. In Section 7.1 we discuss an alternative view to the classic Bayesian model selection discussed here.

6.4 Object Tracking with Detections from a Scanning Radar

The model that has been discussed thus far is quite general in the sense that it does not make any assumptions about the type of sensor from which detections are received. We have, however, discussed the Cartesian-polar transform and how it can be incorporated into a model for radar tracking. In the rest of this document we will be focussing specifically on object tracking with

detections being generated by a scanning radar system. Although the developed model remains unchanged with this detail, the use of such a radar has some subtle implications. It is therefore necessary to provide some of the radar-specific details here. The type of radar used in the collection of the real data on which the model was tested is a frequency modulated continuous wave (FMCW) radar. While a technical discussion of the operation of such a radar is beyond the scope of this work, the main implications arising from its use are discussed here. The radar system processes raw signal data in order to produce detections that can be used by an object tracking model. The radar measures the range, horizontal angle (or azimuth) and radial velocity of real world objects. The radar transmission frequency changes according to a saw-tooth function and the range and radial velocity is calculated using the transmitted and received signal. The range is calculated using the time delay between the peaks of the transmitted and received signals and the radial velocity is calculated using the doppler shift (frequency offset) caused when moving objects reflect the radar signal. The horizontal angle is measured by a sensor which measures the physical orientation of the radar's antenna. The raw signals measured by the radar are discretised and further refined by post processing steps in order to remove duplicate detections, for instance. The measurement noise of the radar is therefore determined by the combined effect of the radar's range, doppler, and azimuth resolution, as well as the effects of the subsequent post processing and discretisation. This noise was modelled as (zero mean) Gaussian noise in order to be able to incorporate its effects into the developed model. An important characteristic of this radar is that it rotates (or 'sweeps') constantly, so the part of the map from which detections can be generated is constantly changing. A simplified, simulated version of such a sweeping radar is implemented in software in order to test the developed model. The effect that this rotating characteristic will have on the developed PGM is that the prior distribution¹³ for new targets will constantly need to be calculated with regard to the radar's orientation. The rotating radar view as well as the prior are illustrated in Figure 6.16 below.

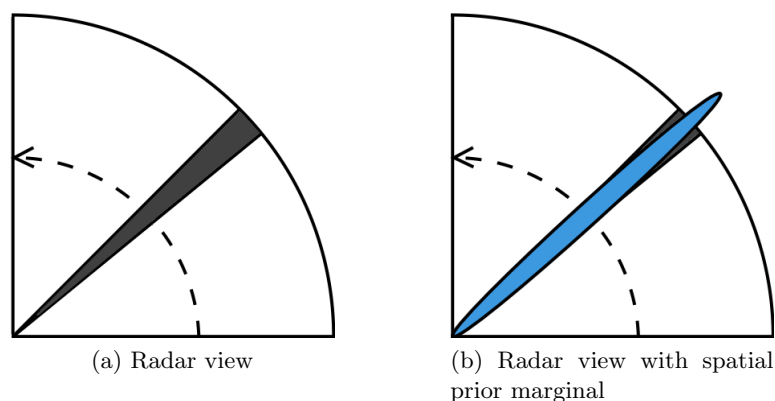


Figure 6.16: Rotating radar view and new object spatial prior. The arrow arc illustrates the rotating nature of the radar beam and spatial prior.

¹³Or rather, the spatial component (marginal) of the prior.

The new object prior distribution is chosen so that the spatial component approximates the radar view. The prior therefore also rotates as the radar view rotates, or ‘sweeps’. This prior will therefore be referred to here as a ‘sweeping prior’. The velocity components of the prior, unlike the spatial components, will be time invariant. The spatial prior distribution is also independent of the velocity component, so the joint prior is simply the product of the spatial and velocity components. The fact that we can define a prior that only covers a small part of the map has advantages when transforming the prior random variables with the Cartesian-polar unscented transform. If the prior’s probability mass is more concentrated, and not situated at the origin¹⁴, it will typically result in a more Gaussian shaped transformed distribution and thus produce a more accurate Gaussian approximation (see Section 5.4 for details).

6.5 Clutter Classification

Thus far we have discussed all the necessary parts and characteristics of a PGM for multiple object tracking. If we were to use this PGM with data that contains false detections (clutter), however, the PGM will continuously add new tracks due to detections that cannot be associated with existing objects. This will result in an increase in the computation that the PGM needs to perform, and also result in false tracks in the output. In this section we will present a solution to the clutter problem that can be incorporated into the existing model.

In order to get an idea of what kind of model might help solve the clutter problem, let us consider how a person looking at a screen would determine whether detections are clutter¹⁵. The person would notice that some of the detections follow a pattern that can be used to predict where the future detections might be. The other type of detections will seem to be completely random (and they would be), and contain no information about future detections. This thought experiment was used as a starting point for the development of our clutter classification network. The network developed thus far already uses the existing track distributions to predict where objects will be in the next time step. The sweeping prior, which is calculated to correspond to the radar view at a given time step, can be thought of as a distribution that describes where detections are possible. The ‘overlap’ of the target distributions and the prior distribution therefore gives us the areas that we expect to get detections from. (Exactly what is meant with the term ‘overlap’ is described mathematically later in this section.) If detections are consistently generated in the area that we expect an object to be, this will enforce our belief that the object is real. If, on the other hand, detections are not received when we expect them, this will increase our belief that the object is actually not there and that the track was created due to clutter.

As with the model developed up to this point, we will still allow any detection to initialise

¹⁴This is the case with the sweeping prior in Figure 6.16a, as can be seen from the fact that the variance is relatively small in the angular direction and that its mean is located away from the origin.

¹⁵A person might of course not be able to classify clutter if the amount of clutter is too extreme and the true detections are too noisy and/or have a low sample rate, but we will assume an easy example here.

a track if it is unlikely that the detection is associated with any of the existing tracks. Here we note that this is not a design choice but rather a necessity, since there is typically no way to distinguish a real detection from clutter without further context. Although both real tracks and clutter tracks will still be tracked with the additional clutter classification model, this model will allow false tracks to be identified as clutter tracks. These tracks and the corresponding clusters can then be removed from the graph.

In order to perform inference in the clutter classification model, we will need to calculate the posterior distribution $P(e_i^t|\Gamma)$ (with Γ being the set of all measurements received thus far) over whether or not we can expect a detection from the target i (at time step t). We will also calculate the posterior $P(g_i^t|\Gamma)$ over whether or not the target generated a detection. We will then use these probabilities to calculate the posterior probability distribution $P(r_i^t|\Gamma)$ over the ‘realness’ of the target at every time step. Here a real target will be a real world object that changes its state in a way that is consistent with the defined motion model and consistently generates detections when it is expected to. A false target can correspond to a real world object that changes its state in a manner that is different to the defined motion model, such as a tree moving erratically in the wind. Alternatively a false target can also correspond to a false track that was initialised by electrical noise and does not correspond to a real world object. For each target there will be a set of e_i^t , g_i^t and r_i^t variables in each cluster-slice. Furthermore, all the $P(r_i^t|\Gamma)$ distributions for a given track will be used to calculate an overall probability $P(r_i|\Gamma)$ that the target is real. Figure 6.17 below shows an example of a single cluster-slice of the MOT PGM (with three object nodes and two measurement nodes) with the connected clutter classification model.

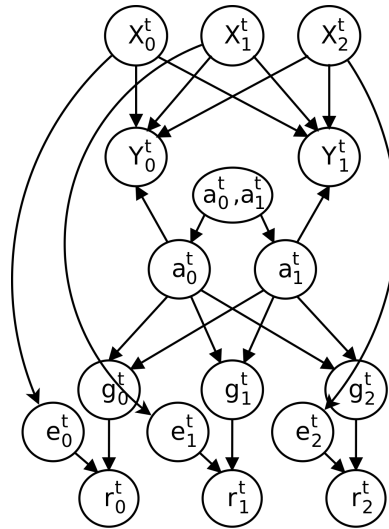


Figure 6.17: Example of the clutter classification model (Bayes network) connected to a single cluster-slice of the MOT PGM. Whether a detection was generated (g_i^t) by a target i is dependent on the associations (a_i^t), and the ‘realness’ (r_i^t) of a target (at a specific time) is dependent on the expectation (e_i^t) that the target should have generated a detection and whether it generated (g_i^t) a detection. Note that the time step indices have been omitted here, as all the variables correspond to the same time step.

With the general structure of the model established, we can now consider the specifics of the different types of nodes and the distributions that they represent. We start with the ‘expectation of detection’ (e) node. This node represents the random variable corresponding to whether or not we expect a detection from a target. Here we therefore have to ask the following question: what is the probability that a target in its current state can cause a detection if the radar is looking in a specific area? If we know in which direction the radar is pointing, along with the field of view and range of the radar, we could construct a function that defines where objects are visible to the radar. We could use this function, together with the target distributions, to calculate the probabilities that a target will be seen by the radar. To illustrate this process, we will consider a simplified one dimensional example. The following figure shows three target position distributions and a window function that represents a hypothetical one dimensional radar’s field of view.

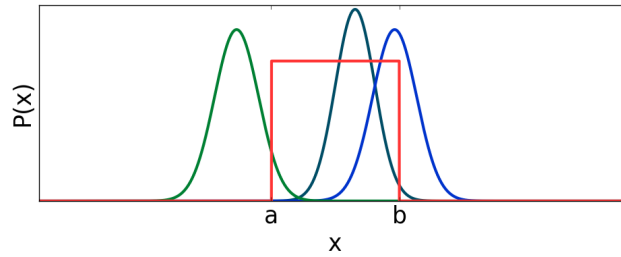


Figure 6.18: Example of three target state distributions and visible map area (red window).

If the visible area of the map is defined by $P(e_i = 1|x_i)$ and the distribution over a target position is $P(x_i)$, the probability of the target being seen by the radar is

$$P(e_i = 1) = \int_{-\infty}^{\infty} P(e_i = 1|x_i)P(x_i)dx_i = \int_a^b P(x_i)dx_i.$$

The above expression therefore states that the probability of detection (of a certain target) is the fraction of the mass of the target's distribution within the visible area of the map. In order to calculate the probability that a target is not seen by the radar, we can use the complement of the $P(e_i = 1|x_i)$ distribution, shown by the window function above. The function is shown in Figure 6.19, below.

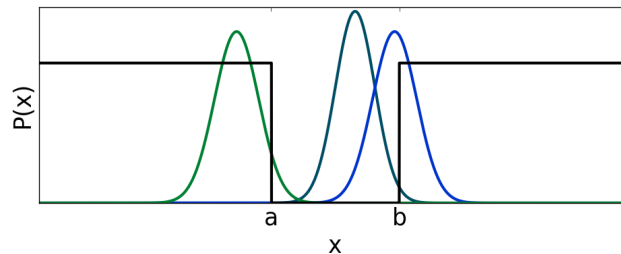


Figure 6.19: Example of three target state distributions and invisible map area (black).

Although the above example gives us a starting point for calculating the required e probabilities, the window function is somewhat unrealistic, since the radar field of view does not have hard boundaries, but has a continuous transition band at its edges. In order to make this function more realistic, we will use a Gaussian distribution for the visible area distribution. Another motivation for choosing this function is of course the convenience of working with Gaussian distributions. The Gaussian equivalents of the window and window complement functions used in the previous example are shown in Figure 6.20 below.

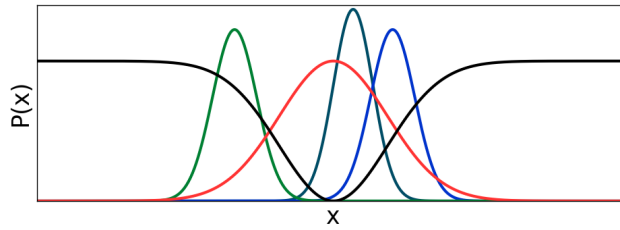


Figure 6.20: Gaussian equivalent of window (in red) and window complement (in black) functions shown in figures 6.18 and 6.19, respectively. The same three target state distributions are also shown here.

We can now use the following equations to calculate the posterior e probabilities:

$$\begin{aligned}
 P(e_i^t = 1|\Gamma) &= \int_{-\infty}^{-\infty} P(e_i^t = 1|x_i^t, \Gamma)P(x_i^t|\Gamma)dx_i^t, \\
 P(e_i^t = 0|\Gamma) &= \int_{-\infty}^{-\infty} P(e_i^t = 0|x_i^t, \Gamma)P(x_i^t|\Gamma)dx_i^t \\
 &= \int_{-\infty}^{-\infty} (P_{max} - P(e_i^t = 1|x_i^t, \Gamma))P(x_i^t|\Gamma)dx_i^t \\
 &= \int_{-\infty}^{-\infty} P_{max}P(x_i^t|\Gamma)dx_i^t - \int_{-\infty}^{-\infty} P(e_i^t = 1|x_i^t, \Gamma)P(x_i^t|\Gamma)dx_i^t \\
 &= P_{max} - P(e_i^t = 1|\Gamma) \quad (\text{if } P(x_i^t|\Gamma) \text{ is normalised})
 \end{aligned}$$

with:

- $P(e_i^t = 1|x_i^t, \Gamma)$: Probability that we can expect a detection from target i , given that it is at position x_i^t and given the previously received measurements Γ ,
- $P(x_i^t|\Gamma)$: Posterior distribution over the position of target i ,
- P_{max} : The maximum of the $P(e_i^t = 1|x_i^t, \Gamma)$ function.

The above method can easily be extended to the multivariate case that will be needed for implementing clutter classification in the multiple object tracking model.

Since the spatial marginal of the new object prior (sweeping prior) already describes where new detections can come from, we will use this distribution as $P(e_i^t = 1|x_i, \Gamma)$. We also have $P(x_i|\Gamma)$, as this is simply the spatial marginal of the target's state distribution. So we can calculate the probability that any given target should generate a detection.

We will now focus on the other variables that are involved in the clutter classification inference procedure. The conditional distribution ($P(g_i|a_0, \dots, a_{m-1})$) that a target generated a detection, given the associations, can be derived through some simple logical reasoning. If at least one measurement is associated with a target, then by definition that target caused a detection. If, on the other hand, no measurement is associated with a target, then the target did not cause a detection. This logic is encoded in the following equations:

$$P(g_i^t = 1 | a_0^t = \alpha_0^t, \dots, a_{m-1}^t = \alpha_{m-1}^t) = \begin{cases} 1 & i \in \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \\ 0 & i \notin \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \end{cases},$$

$$P(g_i^t = 0 | a_0^t = \alpha_0^t, \dots, a_{m-1}^t = \alpha_{m-1}^t) = \begin{cases} 1 & i \notin \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \\ 0 & i \in \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \end{cases}.$$

The α 's in the above equations correspond to the assignments of values to the a variables. These equations can be used to construct a conditional probability distribution (CPD), with any number of association variables. An example of such a conditional distribution, with two association variables, is shown below:

a_0^t	a_1^t	g_0^t	$P(g_0^t a_0^t, a_1^t)$
0	0	0	0.0
0	0	1	1.0
0	1	0	0.0
0	1	1	1.0
0	2	0	0.0
0	2	1	1.0
1	0	0	0.0
1	0	1	1.0
1	1	0	1.0
1	1	1	0.0
1	2	0	1.0
1	2	1	0.0
2	0	0	0.0
2	0	1	1.0
2	1	0	1.0
2	1	1	0.0
2	2	0	1.0
2	2	1	0.0

We now have a way to calculate distributions over the e variables and to construct the table CPDs for the g nodes. Lastly, we need to construct a CPD for the r variables to reason about whether or not a target is real given the expected detections and the generated detections. Here we can apply our intuition from earlier that a real target should generate detections when we

expect it to, while a false target will probably not generate consistent detections. An example of a conditional distribution that encodes this logic is presented in the table below¹⁶. The exact probability values shown in this example could be adjusted if necessary.

e_i^t	g_i^t	r_i^t	$P(r_i^t e_i^t, g_i^t)$
0	0	0	0.5
0	0	1	0.5
0	1	0	0.5
0	1	1	0.5
1	0	0	0.9
1	0	1	0.1
1	1	0	0.1
1	1	1	0.9

In the above table we can see that the distribution is uniform when $e_i^t = 0$. The scenario where a target is not expected to cause a detection but does, should not be possible in the context of the multiple object tracking model. The main reason (apart from occlusions) that a target would not be expected to cause a detection would be that the radar is looking at a part of the map where the target is unlikely to be. If this is the case, then a detection in this area would not be associated with the target. The probability that a target generated a detection, if no detection is expected, should therefore always be close to zero. If, however, we consider the conditional probability $P(r_i^t|e_i^t = 0, g_i^t = 1)$, we will not be able to gain any information about the ‘realness’ of the target - since the condition is somewhat paradoxical. We therefore set this part of the CPD to be uniform. In the case where no detection is expected, nor received, we also cannot gain any information about whether the target is real (and not clutter), and this part of the distribution is therefore also uninformative. If a detection is expected from the target and not received, then we have reasonable certainty that the target is not real. This probability is set to 0.9 in the example table above. If, however, we know that occlusions are very likely, we can lower this probability. If a target detection is expected and one is received, we similarly have a high (0.9 in the example) probability that the target is real. If there is a very high level of clutter in the detections, we could lower this probability. Although the values in this example are, in a sense, arbitrarily chosen, they are at least logically motivated. Furthermore, the clutter classification model with the values in this example has been shown to work very well in practice over a large variation in clutter and missed detections.

As mentioned earlier, the variable r_i in the above table only gives an estimate of the legitimacy of a target at a single time step. One would, however, expect real targets to consistently be real, and false tracks to consistently be false. We will therefore add one additional random variable per track to allow information from all time-slices to be combined in order to give a more robust and coherent estimate of the nature of the target. This additional node is shown in the two time-slice representation of the MOT PGM with clutter classification model below.

¹⁶The probabilities in this table was found to work well in practice over a wide range of tracking scenarios.

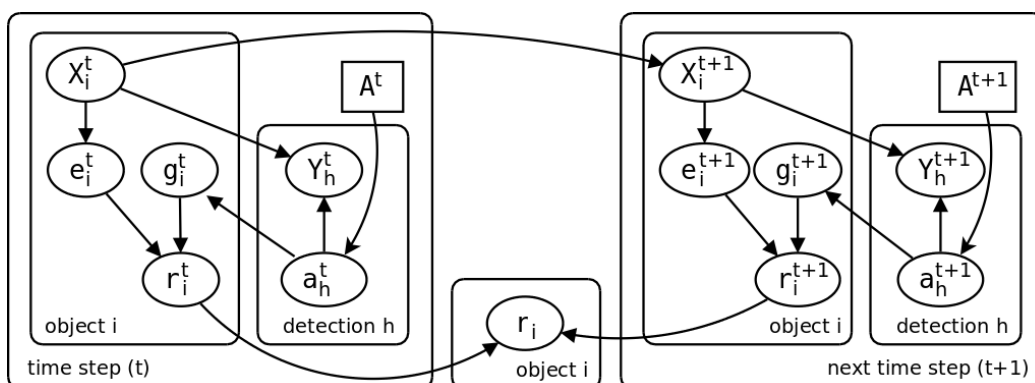


Figure 6.21: Two time-slice Bayes network of MOT PGM with clutter classification. The e_i^t random variable represents whether or not a detection is expected from target i at time t . The g_i^t variable corresponds to whether or not a detection was generated from target i at time t . The r_i^t random variable corresponds to whether the target is real or not for a specific time step. The r_i nodes consolidate the information from all the applicable r_i^t nodes and correspond to whether or not the target is real in general.

The R_i variable in the above graph is the variable that will incorporate the target legitimacy information from the r_i^t variables in all time-slices. The equations that can be used to construct the $P(R_i|r_i^0, \dots, r_i^T)$ CPD are shown below:

$$P(r_i = 1|r_i^0 = \rho^0, \dots, r_i^T = \rho^T) = \begin{cases} 1 & 0 \notin \{\rho^0, \dots, \rho^T\} \\ 0 & 1 \notin \{\rho^0, \dots, \rho^T\} \end{cases},$$

$$P(r_i = 0|r_i^0 = \rho^0, \dots, r_i^T = \rho^T) = \begin{cases} 1 & 1 \notin \{\rho^0, \dots, \rho^T\} \\ 0 & 0 \notin \{\rho^0, \dots, \rho^T\} \end{cases}.$$

An example of a conditional distribution, corresponding to the above equations, is shown below:

r_i^0	r_i^1	r_i^2	r_i^3	r_i	$P(r_i r_i^0, r_i^1, r_i^2, r_i^3)$
0	0	0	0	0	1.0
1	1	1	1	1	1.0

The above table CPD is an example for a graph with five time-slices. Note that only the entries with non-zero probabilities are shown in the table. The effect of having a table like the one above is identical to having a prior over the permutations of the r_i^t assignments ($\{\rho^0, \dots, \rho^T\}$), which is zero if all assignments are not identical (either 0 or 1). This encodes the fact that a track must either always correspond to a real target or always correspond to clutter. In a practical

implementation, we can therefore simply compute the overall probability that the track is real as follows:

$$P(r_i = 1) = \frac{\prod_t^T P(r_i^t = 1)}{\prod_t^T P(r_i^t = 1) + \prod_t^T (1 - P(r_i^t = 1))}.$$

In order to give a clear idea of the cluster graphs that instantiate the clutter classification model described in this section, an example is given in Figure 6.22 below. This cluster graph corresponds to the Bayes network in Figure 6.17. Note that the graph only shows the model for a single time-slice and does not show the inter-time-slice connections.

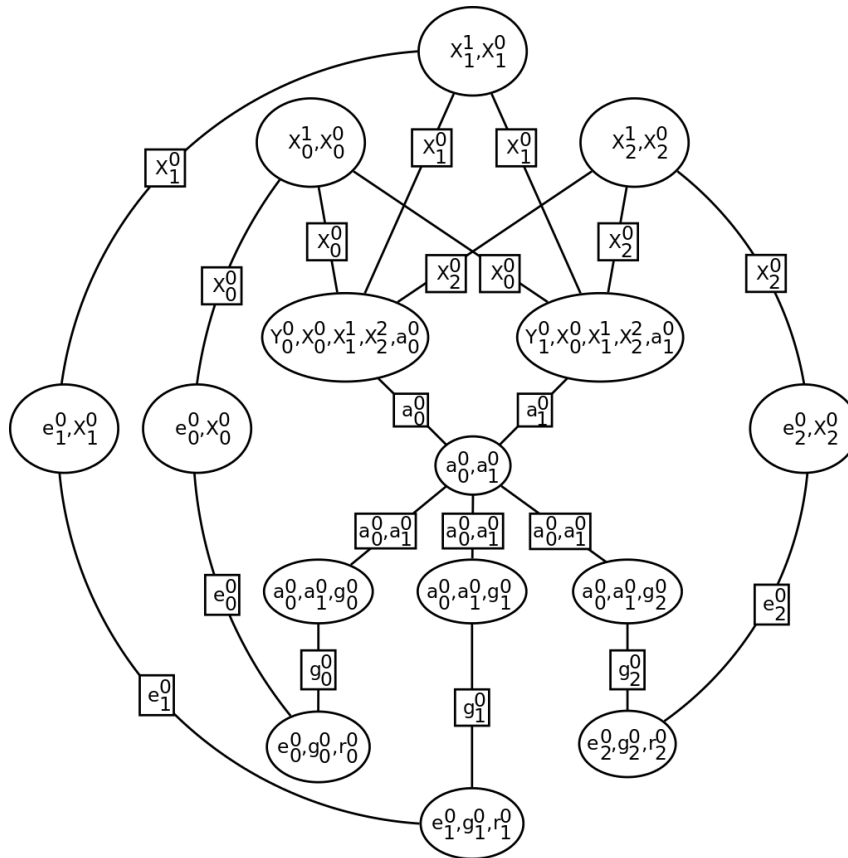


Figure 6.22: MOT and clutter classification cluster graph, corresponding to the Bayes network in Figure 6.17. This graph corresponds to a sub-graph (of a larger graph, time-unrolled graph) for a single time step, where two measurements have been received, and three objects are being tracked.

We have now fully specified the model for clutter classification. This model can be used to identify false tracks and remove the corresponding clusters from the graph. With the clutter classification model discussed in this section, the developed PGM has all the capabilities necessary to track multiple targets in realistic situations. The performance of the developed PGM with the clutter classification network is evaluated in Section 10.3. Although this model performs very well in practice, it was noticed upon further consideration that the model does not have a causal structure. The fact that the model performs well despite not being causal is interesting and highlights the fact that probabilistic models do not need to be causal¹⁷ in order to work in practice [1, p. 1009]. There can, however, be differences in dependencies between causal and non-causal models, which could influence the correctness of inference. Furthermore, causal graphs tend to allow for more efficient inference [1, p. 1009]. For these reasons, an alternative and causal graph structure for the clutter classification model is given in appendix C.

6.6 Conclusion

In this chapter we have shown how the data association problem is formulated mathematically with the use of a multiple object tracking PGM. We also gave a visual example to make the message passing operations less abstract and to give the reader a better intuition of how the model works. With these examples, we showed how the association variables cause superposition-like states to arise with the measurement update messages. The model in these examples could, however, not infer the number of targets automatically. We discussed how Bayesian model selection can be used for this purpose and showed an example of how this method works in practice. In order to allow the model to track targets in the presence of false detections, we developed a PGM for classifying clutter tracks. The PGM, with the model selection and clutter classification methods discussed in this section, should be capable of tracking targets accurately in realistic situations. The model is, however, fairly complex and perhaps unnecessarily computationally expensive. In the next section we will focus on increasing the efficiency of the model developed thus far.

¹⁷Unless of course they are required specifically for causal inference.

Chapter 7

Improving Tracking Efficiency

The model discussed thus far performs well (as indicated by the model performance evaluation in Section 10.3), but is computationally expensive to operate. In this section we therefore investigate a few methods for increasing the efficiency of the developed model.

7.1 An Alternative View of Model Selection

In Section 6.3 we discussed the classical view of Bayesian model selection. This method can be computationally expensive, since message passing is performed in multiple graphs in order to find the most probable model. Many of the message passing calculations will effectively be duplicated in the various graphs, and all of the graphs, except the most probable, are eventually discarded. Furthermore, it can be noted that there is already a mechanism inherent in the graph structure that seems to facilitate a function very similar to model selection. The association variables indicate which states are most probably associated with which measurements. If the associations were observed, there would be no reason for the unassociated state and measurement clusters to be connected. In this sense, the association cluster already provides a mechanism for model selection [37]. In this section we will investigate the details of this mechanism.

We will start with a simple example of a ground truth network, with three state clusters and two measurement clusters. Figure 7.1 below shows an example of a valid model for this case. We will assume that each measurement will have exactly one object associated with it¹. There can, however, be objects that are not associated with any measurement, due to missed detections. False detections are not considered here, as clutter tracks will be identified by the clutter classification model.

¹This assumption will typically hold for the real radar data. Also, if it does not, any duplicate tracks that were added due to duplicate measurements will be removed by the clutter classifier.

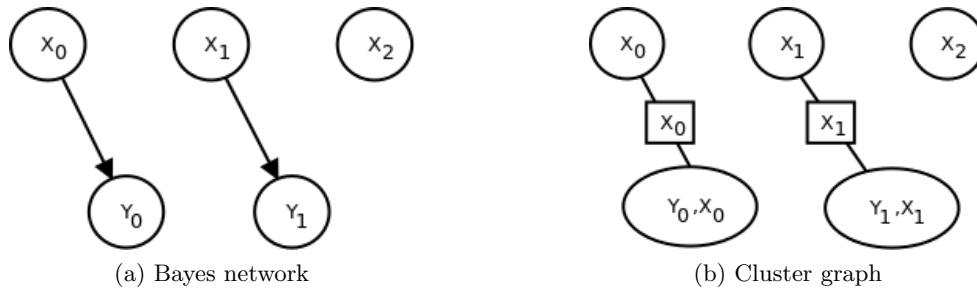


Figure 7.1: An example of a valid ground truth model Bayes network (a) and corresponding cluster graph (b) with three objects (each with a corresponding X node/cluster) and two measurements (each with a corresponding Y node/cluster)). X_2 is not associated with either measurement in this example and therefore has no link to a measurement node/cluster.

The above graph has three disconnected parts and the model evidence will be given by the following expression:

$$P(D|H) = \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_0) dX_0 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_1) dX_1 \right)$$

In general, for any valid network, the model evidence can be computed as follows:

$$P(D|H) = \prod_i^N \left(\int_{\mathbb{R}^d} P(Y_i = \mathbf{y}_i, X_j) dX_j \right) \quad (7.1.1)$$

where X_j is the state variable of the node connected to the Y_i variable node and N is the number of measurement nodes. Note that the state distributions of the nodes that are not connected to a measurement node are not included in these expressions as they will be normalised priors² and will not affect the evidence. We now consider a more general model that can be used if the association variables were not observed. This model is shown in Figure 7.2 below.

²In the case of new targets, these priors will be the sweeping priors described in Section 6.4, and in the case of existing targets the messages sent from the previous state clusters will be normalised, as is typically done in message passing algorithms.

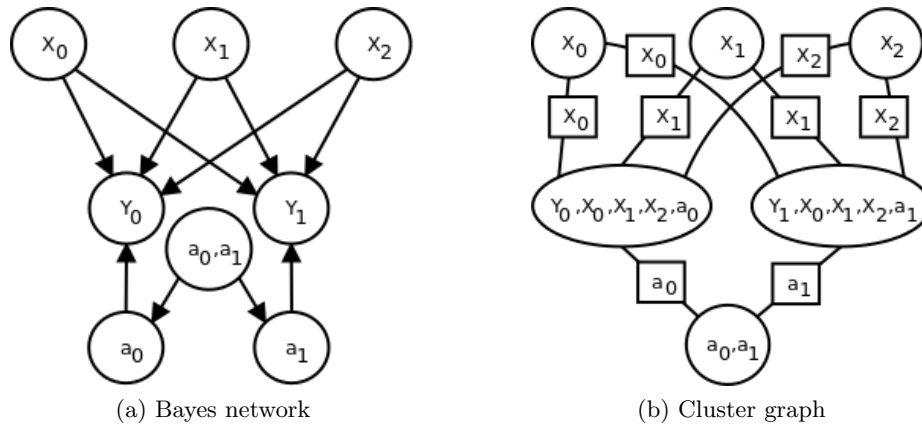


Figure 7.2: Densely connected model

If we perform message passing in this model and condition the measurement potentials on the detections, the resulting measurement potentials will be:

a_0	$P(Y_0 = \mathbf{y}_0, X_0, X_1, X_2, a_0)$
0	$P(Y_0 = \mathbf{y}_0, X_0)P(X_1)P(X_2)$
1	$P(Y_0 = \mathbf{y}_0, X_1)P(X_0)P(X_2)$
2	$P(Y_0 = \mathbf{y}_0, X_2)P(X_0)P(X_1)$

a_1	$P(Y_1 = \mathbf{y}_1, X_0, X_1, X_2, a_1)$
0	$P(Y_1 = \mathbf{y}_1, X_0)P(X_1)P(X_2)$
1	$P(Y_1 = \mathbf{y}_1, X_1)P(X_0)P(X_2)$
2	$P(Y_1 = \mathbf{y}_1, X_2)P(X_0)P(X_1)$

The association cluster's prior distribution is shown in the table below. Here we have encoded our assumption that each measurement must be associated with exactly one state and that any two measurements cannot be associated. This is done by assigning zero probabilities to any assignments that do not reflect this assumption. The valid association assignments are all (a priori) equally likely and are shown in the table below.

a_0	a_1	$P(a_0, a_1)$
0	1	1/6
0	2	1/6
1	0	1/6
1	2	1/6
2	0	1/6
2	1	1/6

If we now send messages from the measurement clusters to the association cluster, the potential of the association cluster changes to:

a_0	a_1	$P(a_0, a_1)$	Model
0	1	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_0) dX_0 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_1) dX_1 \right)$	(a)
0	2	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_0) dX_0 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_2) dX_2 \right)$	(b)
1	0	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_1) dX_1 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_0) dX_0 \right)$	(c)
1	2	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_1) dX_1 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_2) dX_2 \right)$	(d)
2	0	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_2) dX_2 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_0) dX_0 \right)$	(e)
2	1	$(1/6) \left(\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_2) dX_2 \right) \left(\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_1) dX_1 \right)$	(f)

Finally, we note that the expressions in the table entries above have exactly the same form (except for the normalising constant from the association prior) as the expression for model evidence shown in equation 7.1.1. In fact, the expressions in the table entries above correspond exactly to the model probabilities of the full list of valid models for this example. These models are shown in Figure 7.3 below. The association cluster’s potential, after message passing, is therefore a probability distribution³ over all valid models that we have specified.

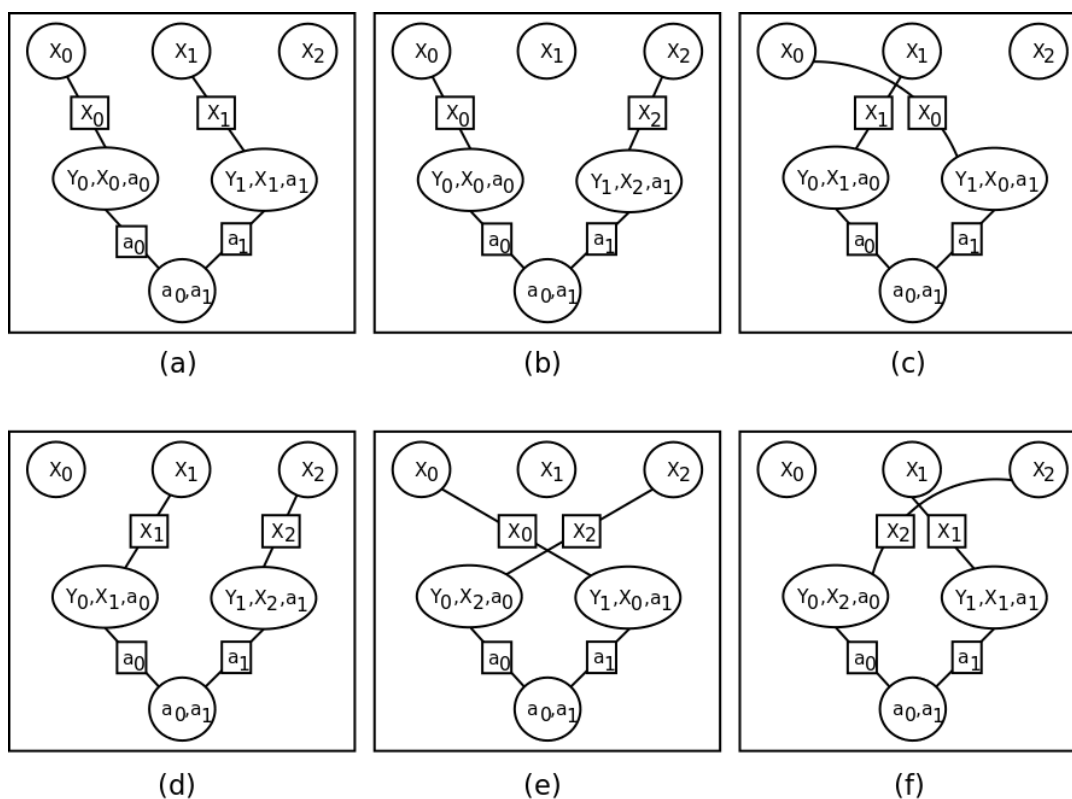


Figure 7.3: Possible valid models corresponding to the assignments and probabilities in the association cluster after message passing.

³If it is normalised.

This result is quite elegant, as it encapsulates the characteristics of classical model selection in a single graph by using assignments to discrete variables to represent different models. This allows us to perform model selection through message passing in a single graph. This process is much more computationally efficient than the classical model selection procedure described in Section 6.3. This effect is not unique to this example and will arise in general with any number of X clusters connected to any number of Y clusters. A sketch of a proof for this statement is given in appendix A.5.

Note that the model selection described in this section is only for a single time slice, with prior state distributions. In this case, the alternative model selection method corresponds exactly to that of classical Bayesian model selection. Although instructive this, however, does not prove general equivalence with Bayesian model selection. This should be considered for further research. This method can, however, be interpreted as forming a distribution over edges (the edges between the state and measurement clusters in this case) in the graph. With this interpretation it is clear that the method described here will indeed perform a function that is very similar to, if not exactly equivalent to, classical Bayesian model selection. Furthermore, this alternative method has been shown to work very well in practice as is evident from the results in Chapter 10.

7.2 Reducing the Number of Hypotheses

The main cause of the complexity of multiple object tracking is the large number of association hypotheses that need to be considered. In this section we discuss two methods for reducing the number of these hypotheses. In the model discussed thus far, including the variation that uses the alternative model selection, the model will consider all state-measurement association hypotheses, at least for a single time-step. Often, however, only a small number of these hypotheses will be remotely plausible. If we can therefore find a computationally inexpensive method of ruling out some of the very unlikely hypotheses, we could significantly improve the efficiency of the model. Figure 7.4 below shows an example with two objects and four detections. Here the ellipses indicate areas of reasonable certainty for the object locations⁴. From this figure we see that detections A and B can likely be associated with object 1, while detections B and C can likely be associated with object 2. It is very unlikely that detection D is associated with either of the objects.

⁴These ellipses could correspond to standard deviation ellipses, for instance.

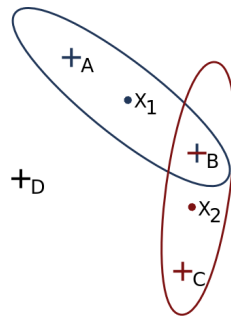


Figure 7.4: Example of two targets (X_1 and X_2) with regions of relative certainty about their positions (indicated by ellipses with means at the X_1 and X_2 dots respectively) and four detections (+’s) that are colour coded to indicate which targets they can reasonably be associated with. Detection ‘D’ is too far from both of the targets and is therefore not associated with either.

In this case, and in general, we can therefore disregard the very unlikely association hypotheses. In order to do this, however, it would be preferable to circumvent the computationally expensive calculation of the association probabilities. Calculating these probabilities in this manner involves calculating the $2d_X + 1$ sigma points, applying the transformation to the points, calculating the parameters of the joint distribution $P(X, Y)$, conditioning the joint distribution on the observed evidence, and marginalising the conditioned distribution. Another method that might be used to estimate the association probabilities more efficiently, is alluded to in Figure 7.4. Here we have simply transformed the detection points to the Cartesian space. We can then consider the association between a detection and an object if the detection is within a few (specified number of) standard deviations from the mean of the Gaussian state distribution. If it is outside this region, the association will not be considered possible. A convenient method of determining whether a point c_j lies in such a region is to compare the ratio $P(X = c_j)/\max(P(X))$ to a specific threshold. Note that this process of eliminating association hypotheses, although logical, is somewhat ad hoc and should therefore only be used to eliminate extremely unlikely associations. This method is known as ‘measurement gating’ in the multiple object tracking literature [38, p. 3].

Another method that can be used to reduce the number of hypotheses is simply to discard the least likely hypotheses. This method is commonly used in the MHT filter. In the developed PGM, with alternative model selection, a similar effect can be obtained through an approximation of the association posterior⁵. In order to perform such an approximation, we can evaluate the probabilities of the assignments relative to the maximum probability and set these probabilities to zero if they are below a certain (specified) threshold. This step will be performed after the association messages have been received and before any further messages are sent. If many of the probabilities are much lower than the maximum, as will often be the case, then perform-

⁵The potential of the association cluster after it has received messages from all measurement clusters.

ing this step will result in a significant increase in the inference efficiency while also yielding a good approximation of the association potential. This approximation and subsequent message passing will often result in zero probabilities⁶ in the measurement potentials. This will, in turn, result in completely uninformative messages⁷ being sent from the measurement clusters to the state clusters. When such a message is sent, it has no effect on the state distribution and is effectively an indication that the measurement is not associated with the corresponding object. In such cases the edge over which the message is sent can therefore be removed from the graph. The alternative model selection method with the described association cluster approximation can therefore allow the structure of the graph to be directly determined through message passing in the graph. This mechanism for model selection should also be useful in general PGMs.

7.3 Efficient MOT PGM Algorithm

In the previous chapters we discussed the necessary theory and operation of the various components of the developed PGM. Because of the dynamic nature of the MOT graph and the fact that new detections will be received continuously, inference in this PGM is somewhat different from how PGMs are typically used. With the developed model the graph will change constantly and the process will involve iterative graph expansion, inference and graph structure refinement steps. We therefore need an algorithm to clearly define how these steps should be performed. This algorithm is presented below.

⁶Gaussian distributions with weights equal to zero.

⁷These messages will have a vacuous Gaussian form, which is equivalent to an unnormalisable uniform distribution.

MOT PGM Inference Algorithm

- t_{a1} : Cartesian detection threshold (for measurement gating)
 t_{a2} : Association hypothesis threshold (for association potential approximation)
 t_r : Real target probability threshold (for clutter classification)
 t_t : Clutter track time threshold (for clutter classification)
 \mathbf{c}_j : Cartesian space detection corresponding to \mathbf{y}_j (for measurement gating)

if *detection set (with size m) is received at time-step t* **then**

1. Extend the graph with X^t clusters for each X^{t-1} cluster.
Add m new X^t clusters, with prior distributions.
Add m new Y^t clusters and 1 new A cluster
2. Send all $\delta_{X,X}$ messages to the new state clusters
3. Eliminate association hypotheses if
 $P(X_i^t = \mathbf{c}_j) / \max(P(X_i^t)) < t_{a1}$
4. Connect the clusters as per the remaining hypotheses
5. Send all $\delta_{X,Y}$ and $\delta_{Y,A}$ messages
6. Approximate the A cluster distribution by setting:
 $P(A = \alpha) = 0$, if $P(A = \alpha) / \max(P(A)) < t_{a2}$
7. Pass all $\delta_{A,Y}$ and $\delta_{Y,X}$ messages
8. Remove any edges over which vacuous messages were sent.
Remove any X clusters which has no connected edges.
9. Add and connect the clutter classification sub-graph.
10. Perform message passing in the added sub-graph
(including messages to R_i clusters)
11. If, for any object i $P(R_i = 1) < t_r$ and it has been for $t > t_t$,
terminated the track by removing the last X_i cluster.

Note that the above algorithm is implicitly iterative in the sense that the entire algorithm will be performed whenever new detections are received. When the first detections are received (at the first time-step), a number of state priors equal to the number of detections are used as the state cluster potentials. The state clusters are each connected to a single unique measurement cluster. This can be done because the exact labels of the targets are arbitrary and there is therefore no data association problem with the first set of detections. If previous detections have been received and the graph is not empty, the graph needs to be extended with new state clusters for each previous state cluster. The messages that will be sent between the new and previous state clusters will correspond to the state prediction step (step 2).

To allow for the possibility that some or all of the m detections are from new targets, m additional state clusters (with appropriate prior potentials) are added to the graph. The m measurement (Y) clusters and single association (A) cluster is also added to the graph (step 1). Any unlikely hypotheses are then ruled out (step 3) using the method described in Section 7.2. The graph is then connected according to the remaining plausible hypotheses (step 4). Once the messages from the previous state clusters to the new state clusters have been sent (step 2), the messages in the new cluster-slice can be passed (step 5). When all messages have been received by the association cluster, the cluster belief can be approximated by setting low probabilities to zero (step 6). This will result in vacuous messages being sent to state clusters that are unlikely to be associated with the corresponding measurements. These messages essentially mean that the measurements are not dependent on the specific object (or its state) and we can therefore remove the edge over which the message was sent (step 8). After this step, if a state cluster has no edges connected to it, it can be removed from the graph. This is the mechanism that allows the new target state clusters added in step 1 to be removed again if the detections do not indicate the existence of previously untracked targets.

The clutter classification inference process for the new cluster slice is done in steps 9 and 10. In step 11 the probabilities of each track being real are evaluated. If this probability is lower than the specified threshold t_r for longer than a specified time t_t , the track can be terminated. We found that completely removing clutter tracks sometimes resulted in real tracks being removed. This could happen when an object is lost (due to occlusions or unlikely target dynamics) and the target does not generate any further detections when it is expected. The track that once corresponded to a real target in such a scenario will be classified, after a certain period of time, as a clutter track. It is therefore advisable to only remove the part of a specific track in which its probability of being a clutter track is high. Removing a part of a track corresponds to removing all clusters (X_i , r_i , g_i and e_i clusters) relating to object i in each applicable cluster-slice. After such a removal, the real track probabilities are also recalculated.

The threshold parameters (t 's), which are required to be specified by this model, can be used as a means of adjusting the trade-off between computational requirements and tracking accuracy. Setting the parameters to $[t_{a1}, t_{a2}, t_r, t_t] = [0, 0, 0, t_T]$ (where t_T is the total tracking time) will result in the most accurate tracking performance and also the highest computational requirements. Allowing the $[t_{a1}, t_{a2}, t_r]$ thresholds to be closer to one and the t_t threshold to be less than the total tracking time, will result in more approximate tracking and reduced

computational requirements. Because the measurement gating step is somewhat ad-hoc, it is suggested that the t_{a1} threshold is set to a relatively small value. Typical values that have been found to work well (and have been used as the settings for the model that was evaluated in Section 10.3) are as follows: $[t_{a1}, t_{a2}, t_r, t_t] = [10^{-10}, 0.2, 0.5, 8t_S]$ (where t_S is the radar scan period). These values can, however, be adjusted to either ensure more accurate tracking or to reduce computational requirements in order to meet the requirements of a specific application.

When more accurate estimates of past states are required, state messages can be passed backwards through the graph. This step will typically be performed after the filtering process, but can be performed at any time, or multiple times as the graph is extended. The smoothed estimates can then be retrieved from the updated state cluster beliefs. As a final, post processing step (after tracking is finished), any tracks that still have a low probability of being real can be removed.

Note that each message in the graph is not sent more than once. Although message passing in this graph should in theory be iterative (since the graph can contain loops), it was found that passing each message only once yields very good results⁸ and that additional passes very rarely have an effect on the tracking accuracy. Sending every message twice would of course also double the required computation.

7.4 Conclusion

In this chapter we showed how the model selection problem can be interpreted exactly as message passing in a single graph with discrete association variables to represent the various hypotheses. We also discussed how the number of hypotheses can be reduced through an initial ad-hoc step and with a more principled association belief approximation step. Compared to the classical model selection implementation, the incorporation of the alternative model selection and the hypothesis reduction methods significantly increase the efficiency of the developed PGM. In the following chapter we will investigate if the performance of the model can be increased through more accurate approximations of the Gaussian mixtures that arise during message passing.

⁸This is evident from the results in Section 10.3.

Chapter 8

Gaussian Mixture Utilisation in the LBU Algorithm

In Section 6.2 we discussed how Gaussian mixtures arise during message passing in the developed PGM and how they can be approximated as single Gaussians. We also gave theoretical justifications for why this approximation should be accurate in the majority of cases that will arise in multiple object tracking. It is, however, suspected that there could be scenarios in which this approximation can adversely affect tracking accuracy. In this chapter, we will investigate the possibility of using Gaussian mixtures with less extreme approximations.

8.1 Gaussian Mixture Operations

Most Gaussian mixture (GM) operations that are required for performing inference in hybrid Bayesian networks have closed-form solutions. Marginalisation and conditioning are carried out by performing the relevant operation on the individual Gaussian components. These operations therefore result in GMs and meet the closed-form requirement for parametric inference. Similarly, multiplication can be done by performing multiple Gaussian multiplications and this results in another GM. Because of the exponential increase in the number of components, these products often need to be approximated by a GM with fewer components in practice. Unlike the approximation in Section 6.1, this approximation does not have to be a single Gaussian.

Various approaches can be taken in order to reduce the number of components in a GM. Typically, after GM multiplication, especially in an MOT application, most of the components have a very low weight¹. These components do not contribute significantly to the shape of the distribution and can be discarded in order to obtain an approximation of the true distribution.

¹Relative to the largest weights.

A method that has been found to work well in practice is to keep a certain percentage (typically around 90%) of the GM's mass by discarding the smallest components. The types of methods where components are discarded are known as pruning methods. Other types of methods focus on the merging of similar components. This can be done through moment matching, as discussed in Section 6.1. For merging operations, a method for determining the similarity between two Gaussian components is required. Typically, the Kullback-Leibler (KL) divergence [39, p. 55] is used for this purpose [40, p. 4]. Combinations of merging and pruning methods can of course also be used and thus there are many conceivable and established algorithms for GM reduction. Some of these methods are discussed in [40, p. 4].

In contrast with the multiplication, marginalisation and conditioning operations, the division operation with GMs, does not, in general, result in a function with a GM form. In the following section we will discuss a possible solution to this problem.

8.2 Gaussian Mixture Division

In order to allow Gaussian mixtures to be fully utilised with the LBU message passing algorithm, we developed a method for approximating a GM quotient as a GM. The idea behind this method is to split the GM quotient function into a sum of functions and to approximate each of these functions as a Gaussian. We will explain this method through a simple example of a scenario that will typically arise during message passing. If a cluster has the following potential:

$$\psi(X) = \mathcal{N}_1(X) + \mathcal{N}_2(X)$$

and it receives the following message:

$$\delta(X) = \mathcal{N}_3(X) + \mathcal{N}_4(X),$$

the updated distribution is

$$\begin{aligned} \psi'(X) &= \psi(X)\delta(X) \\ &= (\mathcal{N}_1(X) + \mathcal{N}_2(X))(\mathcal{N}_3(X) + \mathcal{N}_4(X)) \\ &= \mathcal{N}_1(X)\mathcal{N}_3(X) + \mathcal{N}_2(X)\mathcal{N}_3(X) + \mathcal{N}_1(X)\mathcal{N}_4(X) + \mathcal{N}_2(X)\mathcal{N}_4(X). \end{aligned}$$

If we could keep a distribution such as the one above in a factorised form and we only needed to divide out a GM that is already in the factorised product, we could simply remove the appropriate factor in order to perform division. Although such scenarios do arise in practice, this will often not be possible for a number of reasons. Firstly, in order to calculate transformed and joint distributions with linear and non-linear Gaussian distributions, we have to first multiply out all factors in the conditioning distribution. Secondly, if we want to condition a distribution on evidence, the factors must also be multiplied out. We therefore need to work with the updated distribution in its unfactorised form. Let us now consider the effect of dividing the updated

cluster belief $\psi'(X)$ by the previously received message $\delta(X)$. The expression for this quotient is:

$$\frac{\psi'(X)}{\delta(X)} = \mathcal{Q}(X) = \frac{\mathcal{N}_1(X)\mathcal{N}_3(X) + \mathcal{N}_2(X)\mathcal{N}_3(X) + \mathcal{N}_1(X)\mathcal{N}_4(X) + \mathcal{N}_2(X)\mathcal{N}_4(X)}{\mathcal{N}_3(X) + \mathcal{N}_4(X)}.$$

We can rewrite and manipulate the above expression as follows:

$$\begin{aligned} \mathcal{Q}(X) &= \frac{\mathcal{N}_1(X)\mathcal{N}_3(X)}{\mathcal{N}_3(X) + \mathcal{N}_4(X)} + \frac{\mathcal{N}_2(X)\mathcal{N}_3(X)}{\mathcal{N}_3(X) + \mathcal{N}_4(X)} + \frac{\mathcal{N}_1(X)\mathcal{N}_4(X)}{\mathcal{N}_3(X) + \mathcal{N}_4(X)} + \frac{\mathcal{N}_2(X)\mathcal{N}_4(X)}{\mathcal{N}_3(X) + \mathcal{N}_4(X)} \\ &= \left(\frac{1}{\mathcal{N}_1(X)} + \frac{\mathcal{N}_4(X)}{\mathcal{N}_1(X)\mathcal{N}_3(X)} \right)^{-1} + \left(\frac{1}{\mathcal{N}_2(X)} + \frac{\mathcal{N}_4(X)}{\mathcal{N}_2(X)\mathcal{N}_3(X)} \right)^{-1} \end{aligned} \quad (8.2.1)$$

$$+ \left(\frac{\mathcal{N}_3(X)}{\mathcal{N}_1(X)\mathcal{N}_4(X)} + \frac{1}{\mathcal{N}_1(X)} \right)^{-1} + \left(\frac{\mathcal{N}_3(X)}{\mathcal{N}_2(X)\mathcal{N}_4(X)} + \frac{1}{\mathcal{N}_2(X)} \right)^{-1} \quad (8.2.2)$$

$$= (\mathcal{F}_1(X) + \mathcal{F}_2(X))^{-1} + (\mathcal{F}_3(X) + \mathcal{F}_4(X))^{-1} \quad (8.2.3)$$

$$+ (\mathcal{F}_5(X) + \mathcal{F}_6(X))^{-1} + (\mathcal{F}_7(X) + \mathcal{F}_8(X))^{-1} \quad (8.2.4)$$

$$= \mathcal{G}_1(X) + \mathcal{G}_2(X) + \mathcal{G}_3(X) + \mathcal{G}_4(X) \quad (8.2.5)$$

The $\mathcal{G}_i(X)$ terms in equation 8.2.5 will be uni-modal if all the corresponding $\mathcal{F}_i(X)$ functions in equation 8.2.4 are convex functions. This is due to the fact that the sum of non-negative convex functions is also convex [41, p. 79] and the fact that the inverse of a convex function has a single optimum. Since the $\mathcal{F}_i(X)$ functions have a Gaussian form, they will be convex when the covariance matrices are negative definite. Even when only a single $\mathcal{F}_i(X)$ function is convex, the corresponding $\mathcal{G}_i(X)$ function will have one dominant mode in most cases². From equation 8.2.2 we can see that this is the case for all four of the terms, since each bracketed sum contains the inverse of a proper Gaussian distribution. Although there is no theoretical guarantee that these terms will be uni-modal in the general case³, it is more likely that they will be. This has often been found to be the case in practice. We will therefore assume that all of the $\mathcal{G}_i(X)$ terms in equation 8.2.5 and in the equivalent equations, in other examples, are uni-modal. Furthermore, it was found through simulations that the $\mathcal{G}_i(X)$ terms often have a shape that is quite similar to a Gaussian distribution. Thus we will approximate these terms as Gaussian distributions. We will do this by finding the mode of each $\mathcal{G}_i(X)$ term through numerical optimisation and calculating the Hessian matrix of the function at this point. We will use the mode as the mean $\boldsymbol{\mu}_i$ of the Gaussian approximation and we will use the Hessian to calculate its covariance matrix $\boldsymbol{\Sigma}_i$. The Hessian matrix (\mathbb{H}) of a Gaussian distribution is related to its covariance matrix through the following expression [42, p. 257]:

$$\mathbb{H}(-\log(N(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}))) = \boldsymbol{\Sigma}^{-1}.$$

²This will be the case unless the proper Gaussian function(s) has a large weight, small covariance and mean that is close to the minimum (minima) of the concave component(s).

³With additional complicating factors such as non-linear transformations and data conditioning.

Using this expression and assuming that the $\mathcal{G}_i(X)$ functions have a Gaussian shape, we can therefore calculate the covariance matrix of the Gaussian approximation of each $\mathcal{G}_i(X)$ term (in equation 8.2.5) as follows:

$$\mathbf{\Sigma}_i = [\mathbb{H}(-\log(\mathcal{G}_i(X)))]^{-1} \Big|_{X=\boldsymbol{\mu}_i}. \quad (8.2.6)$$

The exact equations for numerically calculating the Hessian of a function can be found in appendix A.6. With equation 8.2.6 and the optimisation procedure we can therefore find the means and the covariance matrices of Gaussians that approximate the $\mathcal{G}_i(X)$ terms in equation 8.2.5. The Gaussian approximation of each of these terms is therefore:

$$\mathcal{N}(X; \boldsymbol{\mu}_i, \mathbf{\Sigma}_i, w_i) = \frac{w_i}{\sqrt{|2\pi\mathbf{\Sigma}_i|}} \exp(-0.5[X - \boldsymbol{\mu}_i]^T \mathbf{\Sigma}_i^{-1} [X - \boldsymbol{\mu}_i]),$$

with the weight w_i still unknown. We can calculate the weight of the Gaussian component by evaluating $\mathcal{G}_i(X)$ at $\boldsymbol{\mu}_i$ as follows:

$$\mathcal{N}(X = \boldsymbol{\mu}_i; \boldsymbol{\mu}_i, \mathbf{\Sigma}_i, w_i) = \frac{w_i}{\sqrt{|2\pi\mathbf{\Sigma}_i|}} = \mathcal{G}_i(X = \boldsymbol{\mu}_i).$$

From the above expression, we get:

$$w_i = \sqrt{|2\pi\mathbf{\Sigma}_i|} \mathcal{G}_i(X = \boldsymbol{\mu}_i).$$

The GM approximation of the GM quotient function in this example is therefore:

$$\mathcal{Q}(X) \approx \mathcal{N}(X; \boldsymbol{\mu}_1, \mathbf{\Sigma}_1, w_1) + \mathcal{N}(X; \boldsymbol{\mu}_2, \mathbf{\Sigma}_2, w_2) + \mathcal{N}(X; \boldsymbol{\mu}_3, \mathbf{\Sigma}_3, w_3) + \mathcal{N}(X; \boldsymbol{\mu}_4, \mathbf{\Sigma}_4, w_4),$$

with the parameters calculated as described above. The above method can, of course, be applied to any GM quotient with any number of terms in the numerator or denominator. In order to test the accuracy of this method, a large number of simulations were run for quotient functions with different numbers of numerator and denominator components and with dimensions from one to ten. The accuracy of the approximations was measured by the C2 distance [43], which is a metric that can be used to measure the similarity between GMs⁴. Figure 8.1a below shows the results of these simulations. From this figure, we can see that the C2 distance is consistently small⁵ over all the simulations. Figure 8.1b shows an example of one of the worst approximations for the one dimensional case.

⁴We can use this metric in these test cases since the true quotient functions are also GMs.

⁵Small here means smaller than that of the C2 distance of the example in Figure 8.1b. This example serves to give an idea of the meaning of specific C2 values.

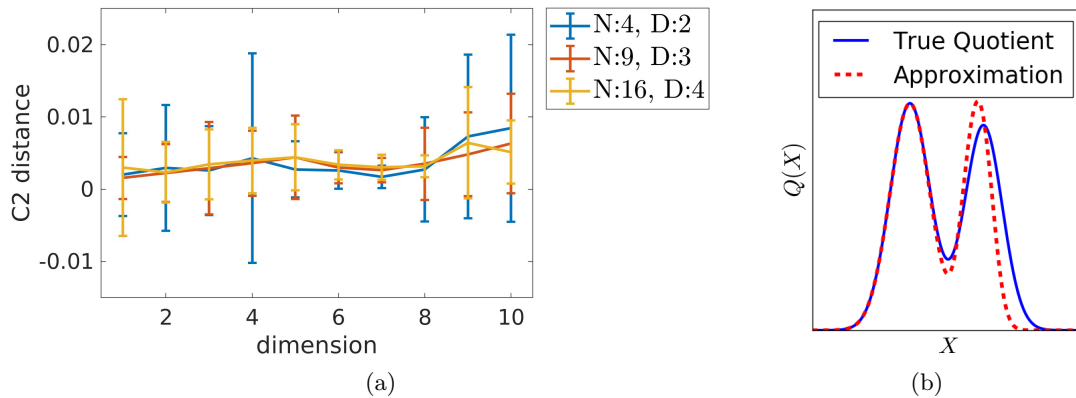


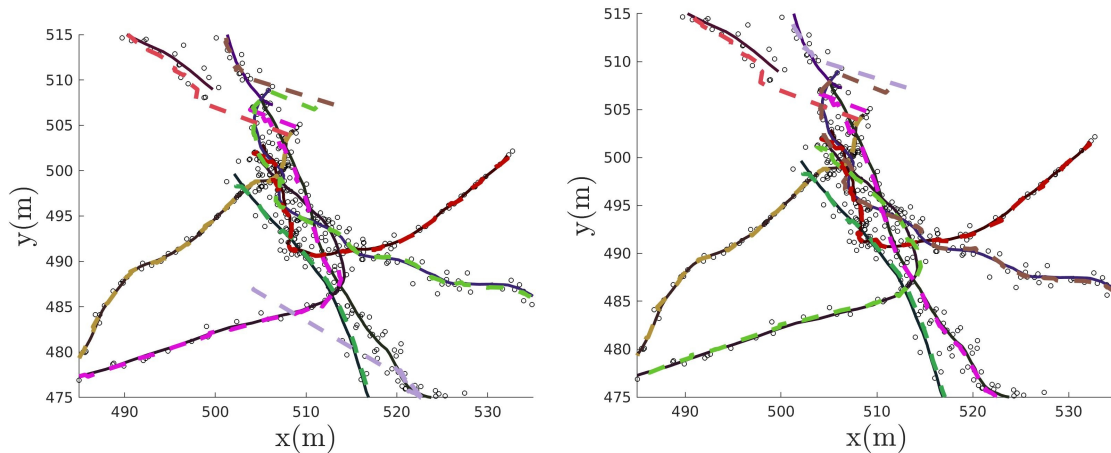
Figure 8.1: Accuracy of Gaussian mixture quotient approximation indicated by the error-bar plots (a), which show the distance between the approximations and the true quotient functions. Random simulations were run for quotients with different numbers of numerator (N) and denominator (D) components, for quotient functions of dimension one to ten. Figure (b) shows an example of the worst approximation for the one dimensional case. The C2 distance for this example is 0.024.

The results in Figure 8.1a therefore seem to indicate that the GM quotient approximation (GMQA) method can accurately estimate the true GMs when the true quotient function is a GM. This will of course not typically be the case with inference in the multiple object tracking PGM. In this setting the use of GM approximations, such as pruning and merging, as well as the use of non-linear transformations will typically cause the true quotient function to not be a GM. The approximation method described in this section should, however, be able to approximate general multi-modal distributions accurately when they are reasonably close to GMs. It has also been observed that the use of GMs, which is enabled by this approximation method, can indeed increase the tracking accuracy of the multiple object tracking PGM in some challenging scenarios. An example of such a scenario and the performance gain that arises from the use of GMs is discussed in the following section.

8.3 Gaussian Mixture PGM improvement over the Moment Matching Gaussian PGM

In the previous section, we described a method for approximating Gaussian mixture quotients that will allow us to use GMs in the LBU algorithm. In this section we will consider an example of the difference that the use of GMs can make compared to the single Gaussian approximation described in Section 6.1. In Section 6.2 we discussed the accuracy of the single Gaussian approximation of a GM in the context of the multiple object tracking PGM. Here we

stated that this approximation should be accurate when the targets are very far apart and in scenarios they are so close together that it is fundamentally impossible to tell them apart in any case. We further noted that the approximation should yield the worst results somewhere between these extremes. We therefore expect the more correct use of GMs to make the biggest difference in tracking accuracy when the targets are close together, but not too close to be fundamentally indistinguishable. Figure 8.2 below shows an example of such a scenario. In this example we will refer to the PGM that approximates GMs as single Gaussians as the moment-matching PGM (MM-PGM), and the PGM that makes use of GMs and the GMQA method as the GM-PGM.



(a) Multiple object tracking with moment matching Gaussian approximations of state distributions (MM-PGM model)

(b) Multiple object tracking with Gaussian mixture state distributions and Gaussian mixture quotient approximation (GM-PGM model).

Figure 8.2: A comparison of tracking accuracy between the MM-PGM and GM-PGM models. All the objects in this example start in the center of the figures and move outward. The scanning radar simulator (without angular velocity measurements) was used in this example. The true tracks in these examples are indicated by solid dark lines, the track estimates by the dashed colour lines, and the detections by the black circles. Note that there are a few errors in the MM-PGM estimates in (a) that are not present in the GM-PGM estimates in (b). The pink estimate in (a), for example, switches between two true tracks, while the correct identities of the tracks are maintained in (b). Also, the grey estimate is only added much later in (a) while the estimates (in pink in (b)) corresponding to the same track is present from the beginning in (b). Lastly, the light green track estimate in (b) is also present throughout the existence of the real track, while the corresponding object is initially not tracked in (a). Note that the track estimates are indicated as dashed lines only to allow the true tracks to be more visible.

From Figure 8.2, we can see that the more correct use of Gaussian mixtures and the GMQA method significantly increases the tracking accuracy of the developed PGM in this challenging scenario. We can see that the tracks in 8.2a do not have the correct (consistent) identities in all instances, whereas in 8.2b the identities are consistent. Specifically, the pink track in 8.2a

changes between two true tracks and the grey track is only added later, because its corresponding true track was initially associated with the pink track estimate. The use of Gaussian mixtures and the GMQA method allows the model to maintain consistent track identities and estimate the correct number of targets throughout the simulation. Larger versions of the images in Figure 8.2 can be found in appendix B.1.1 for easier comparison.

Figure 8.3 below shows the difference in tracking accuracy between the two models in terms of the optimal sub-pattern assignment (OSPA) distance from the ground truth tracks. Here we can see the adverse effect that the incorrect track estimates have on the OSPA distance of the MM-PGM track estimates. We can also see that the GM-PGM has increased accuracy during the first part of the simulation, where it is able to correctly track the closely spaced targets. We can see that the OSPA of the two models converges with time as the targets move further away from each other and the MM-PGM eventually manages to track the targets correctly.

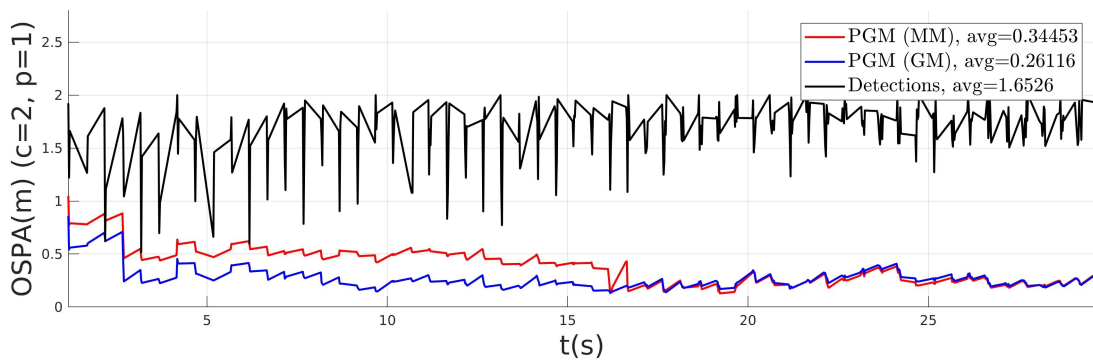


Figure 8.3: Gaussian mixture PGM (GM-PGM) and moment-matching PGM OSPA distance comparison. The OSPA distance on the vertical axis is the OSPA distance between the estimates (and detections) and the ground truth track points at specific time steps. Note that the OSPA distance of the GM-PGM is initially lower than that of the MM-PGM, due to the correct estimation of tracks arising from the more correct Gaussian mixture representation of the state distributions.

This example therefore illustrates the advantages that a better approximation of the state distributions can have in challenging tracking scenarios. It also indicates that the GMQA method can be of use in multiple object tracking inference with the LBU algorithm even when there is no guarantee that the Gaussian mixture quotients have a Gaussian mixture form.

8.4 Conclusion

In this chapter we investigated if an improvement can be made to the moment-matching approximation of a Gaussian mixture in the context of multiple object tracking. We noted that,

in order to make proper use of Gaussian mixtures with the LBU algorithm, we need to be able to approximate the quotient of Gaussian mixtures as a Gaussian mixture. To this end we developed a method for performing such an approximation. We showed that this approximation can be used to accurately retrieve the Gaussian mixture factors when the true quotient function is a Gaussian mixture. Lastly, we showed that this algorithm can be of practical use even when the quotient is not guaranteed to be a Gaussian mixture and that it can allow for significantly improved tracking accuracy in challenging scenarios. In the following section we will discuss the C++ implementation of the developed PGM.

Chapter 9

Implementation Overview

The implementation of the model described in the previous chapters required the expansion of the EMDW library. This section gives a brief overview of the classes that were added to the library and the code that was written to implement and test the developed model.

9.1 Factor Classes

Implementations of some of the distributions required for inference in the MOT PGM were not available in the existing EMDW library. In order to allow the MOT PGM to be implemented, the following factor classes were therefore developed¹:

- Gaussian factor
- non-linear Gaussian (NLG) factor
- factorised factor
- conditional non-linear Gaussian (CNLG) factor
- Gaussian mixture (GM) factor
- discrete log table (DLT) factor²

¹The basic structure, generic code, and some basic functionality were reused from the existing classes in the development of the new classes. The existing Gaussian canonical class was adapted in order to create the new Gaussian class. This class was also initially used as the foundation for the non-linear Gaussian classes, but it was subsequently changed and expanded to such an extent that the two have very little functional code in common.

²This class was created by way of only slight modifications to the Discrete Table class

These classes were implemented in an object-oriented manner in line with the established code base of the EMDW library. The factor classes are derived from the abstract base class ‘Factor’, which serves as a common interface to any specific class. This allows general inference functions or classes to use the abstract factor class and its member functions (factor operations) without requiring the specification of factor types. All of the necessary factor operations for each type of factor were implemented as member functions of the respective classes.

The Gaussian class allows the parameters to be expressed in either covariance form or canonical form and will update the appropriate set of parameters with any specific operation for optimal efficiency³. This class is also used to instantiate the Gaussian components of the Gaussian mixture class.

The non-linear Gaussian class maintains separate conditioning, conditional and joint distributions. These distributions are updated only when necessary, for increased efficiency. The relationship between the conditioning and conditional distributions is defined by a transformation, which is included in the class as an abstract transformation member variable. The conditional and joint distribution parameters are calculated with an implementation of the unscented transformation. The conditioning distribution is allowed to be a Gaussian or a Gaussian mixture in this class, and the joint and conditional distributions can therefore also be Gaussian mixtures in general.

The factorised factor class allows products of Gaussian, NLG and/or GM factors to be kept in a factorised form. This functionality is useful for natural and efficient representation of the independent continuous distributions that arise in the CNLG factors in the MOT PGM. The factorised form also allows GM quotients to be calculated exactly when the denominator GM exists as a component in the factorised factor.

The discrete log table class was adapted from the existing discrete table class to allow the table assignment probabilities to be represented as log probabilities. This was done for consistency with the Gaussian class (and the other classes that use the Gaussian class), where the weight is represented in log form. The log representation was chosen to allow the weight of a Gaussian Factor to be much smaller (without being rounded to zero) than is possible with a linear representation of double precision variables.

The CNLG factor class uses the discrete log table class for the discrete part of the factor and factorised factors for the continuous distributions corresponding to each discrete log table entry. The continuous distributions can therefore have any form allowed by the factorised factor. This allows the CNLG class to instantiate conditional Gaussian, conditional linear Gaussian and conditional non-linear Gaussian factors as well as the more general Gaussian mixture equivalents of these classes. Marginalising out all discrete variables from a CNLG factor, when in a conditional Gaussian or conditional Gaussian mixture form, results in a GM factor.

³This functionality was already available in the existing Gaussian canonical class from which the Gaussian class was adapted.

The GM factor also uses Gaussian factors for the mixture components. Merging and pruning methods were implemented in this class to allow for the automatic reduction of the number of components (if necessary) after multiplication. The maximum number of components that a GM factor is allowed to have can be specified. The actual number of components could, however, remain lower than this number if the mixtures can consistently be approximated well by fewer components. In order to allow (approximate) Gaussian mixture division, the GMQA method described in the previous chapter was implemented in this class. The DLIB library [44] was used for the optimisation functionality required for this implementation.

9.2 Model Implementation and Testing

In order to implement the multiple object tracking filter described in this work, an MOT filter class was created. The algorithm described in Section 7.3, as well as the classical model selection version of the model, is implemented in this class. Here it is worth noting that the EMDW library does contain classes for automatically constructing and performing inference in cluster graphs. However, this functionality is somewhat more suited for non-dynamic PGMs and is especially useful for graphs in which a sensible message passing schedule⁴ is not obvious or constant. Because of the repeating structure and dynamic nature of the MOT graph, an appropriate message passing schedule is quite obvious and will be constant (as described in Section 7.3). For this reason, and to allow for more precise control over the model, new cluster graph related classes were developed. These classes are the ‘cluster node’ and ‘message’ classes. The cluster node class was created to allow the implicit instantiation of a cluster graph as a collection of clusters that can generate and receive message type objects. The cluster node objects contain the local distribution information (in the form of a factor object), information about their neighbours in the graph (ID numbers and sepsets) and also store evidence (if any) that has been observed about the relevant variables. The cluster node objects also store messages that are received. The message objects contain the message distribution and the ID numbers of the sender and receiver nodes. A set of cluster node objects with a specific message passing schedule can therefore be used to perform message passing inference in the cluster graph that is defined by the cluster set. This is how message passing is performed in the developed model.

A random radar simulation test class was implemented to test the developed model. This class allows an arbitrary number of randomly moving objects to be simulated with specified process noise. The effect of a sweeping radar is simulated in order to generate detections from the simulated objects. This simulation class also allows the clutter intensity and basic radar system parameters to be specified.

⁴This is simply the order in which messages are passed in the graph.

9.3 Conclusion

The development of the above factor classes allows for the implementation of the multiple object tracking PGM, as well as the implementation of hybrid probabilistic graphical models⁵ in general. The use of these classes is therefore not limited to the multiple object tracking problem; they could be useful for solving many different problems with PGMs. The implementation of the developed model uses the factor classes that were developed and allows the model described in this work to be instantiated in software. The radar test class allows for effective testing of the developed model. In the following chapter we will evaluate the performance of the developed model using this class and compared the model with other multiple object tracking methods.

⁵Hybrid models are models that contain both continuous and discrete variables.

Chapter 10

PGM Performance Analysis

The aim of this chapter is to put the developed model into context by comparing it to traditional algorithms and to evaluate its performance through various tests. We will discuss the similarities between the existing algorithms and the developed PGM and compare the performance of the PGM to that of the Gaussian mixture PHD (GM-PHD) filter. In order to provide statistically significant results, a large number of random simulation tests have been conducted. The results of these simulations are presented and discussed in Section 10.3. Finally, in order to provide evidence that the PGM can be used in real world applications, the PGM output was also compared to that of an industrial application, with real radar data as input. The output of the PGM for this test is discussed in Section 10.4.

10.1 Traditional Filters and PGM Comparison

In Section 2.1 we discussed some of the most widely used multiple object tracking algorithms. All of these algorithms solve the multiple object tracking problem by applying the same probabilistic principles and are, therefore, similar to some extent. Since the developed PGM is based on the same principles, one can expect it to also share some similarities with the traditional methods. In this section we will discuss these similarities and some of the differences between the models.

The classical model selection version of the PGM allows models to be considered as separate graphs before incorporating the most probable into a final graph. This is very much like the operation of the multiple hypothesis tracking (MHT) filter. Unlike the MHT filter, however, the PGM also allows for soft associations in the hypotheses that are considered. In this sense, the PGM is more like the joint probabilistic data association (JPDA) filter, except that the PGM uses model selection to determine the number of targets, whereas the JPDA can only track a fixed and known number of targets. The PGM with the alternative model selection process allows all

the probable¹ hypotheses to be encoded by the association variables and the structure of a single graph. This differs from the MHT filter, where the various hypotheses are kept separate from each other. Furthermore, the developed PGM differs from the traditional models in its ability to use Gaussian mixtures for the state distributions and in its mechanism for classifying false tracks. These capabilities allow the PGM to accurately track multiple targets under extremely difficult conditions and with unknown clutter and detection probability parameters, which are required to be known by most variations of the traditional algorithms. Here it should be noted that the parameters of the clutter classification model are similar to the detection and clutter parameters of the other models. The PGM model, however, allows the relative probability that each target is real, to be inferred. If all targets have similar detection probabilities, these relative probabilities should give an accurate estimate of which objects are real. This is believed to be the reason that the model is so insensitive to the values of the clutter classification parameters (as is evident from the results in 10.3).

The GM-PHD filter, being an association free filter, is perhaps the most different from the developed PGM. It should, however, be possible to cast the GM-PHD filter algorithm as a graphical model, and the process of doing so might also help clarify the connections between the GM-PHD filter and the developed PGM. Although we will not provide a detailed study of a PGM based GM-PHD filter here, we do offer a brief speculation of how one might construct a PGM that is similar to the GM-PHD filter. Since the the PHD filter does not keep track of target identities, it would make sense that the equivalent cluster graph should have a single state cluster for each time-step. Each measurement cluster (which will be connected to a single state cluster) will then be a conditional linear (or non-linear) Gaussian, with two possible assignments to the discrete variable. The one assignment will correspond to the hypothesis that the state should be updated by the measurement, while the other will be that the measurement is false. This will result in all prior² components being updated by each measurement and by no measurement, allowing the posterior to be a combination of all possible updated components and a scaled version of the prior components. The number of mixture components can then be reduced through standard merging and pruning algorithms. The next state, for all components, will then be predicted and the next measurement update can then be performed. Such a model will therefore represent the distributions over all targets with a single Gaussian mixture distribution. The modes of this distribution should then correspond to individual targets, although this cannot be guaranteed. The GM-PHD filter would, therefore, be similar to the developed PGM, if the PGM only tracked a single state, where the state corresponded to the states of all the objects and the state distribution was a Gaussian mixture.

The model described above, however, does not account for all the intricacies of the PHD filter and an implementation of such a model will undoubtedly require a thorough study. The speculative model therefore serves only to broadly illustrate some of the connections between the operation of the GM-PHD filter and the developed graphical model.

Although the developed PGM shares similarities with the traditional algorithms, the graphi-

¹Approximations in the association clusters can be made that will eliminate improbable hypotheses.

²The prior, here, will typically be a Gaussian mixture.

cal model framework allows for a more flexible and expandable model. The developed PGM can therefore be made to function in a manner similar to the MHT filter, JPDA filter, or possibly even the PHD filter, but can also function in ways that these algorithms cannot. Examples of this include the functionality of the clutter classification graph and the Gaussian mixture utilisation, which allow for more accurate state distributions³. The developed PGM is therefore, in a sense, more general and more interpretable than the existing algorithms that have been discussed here.

10.2 GM-PHD Filter and PGM Performance Comparison

In this section we will compare the developed PGM and the GM-PHD filter and see how their differences manifest in tracking performance. Here we will refer specifically to the GM-PHD filter when using the PHD acronym.

The task of comparing the performance of two multiple object trackers is a problem in its own right. For single target tracking such a comparison can be relatively simple, with the mean squared error often being used as a performance measure. In the multiple object tracking case, especially with clutter and missed detections, a reasonable measure of performance is less obvious. In this case, two models might not estimate the same number of tracks. In the case of association free tracking, different models might have a different number of estimates at the same time-steps. Such complications make direct comparison between two models difficult. However, various methods have been used to measure and compare the performance of multiple object tracking algorithms. One widely used method is the optimal subpattern assignment (OSPA) metric [4]. The OSPA metric is a metric for calculating the distance between sets. The metric considers the distance between the points in the sets, as well as the difference in the size of the sets, to determine the distance between them. The OSPA metric requires the specification of two parameters. These parameters are the cut-off parameter c , an order parameter p . The c parameter determines how heavily cardinality errors are weighted, while the p parameter influences how sensitive the metric is to outliers that are far from any ground truth points [4, pp. 4-5]. In this chapter, we will use $c = 10$ and $p = 1$. The specific choice of these parameters should, however, not affect the outcome of the experiments, since we will compare the model estimate OSPA distances to other OSPA distances using the same parameters.

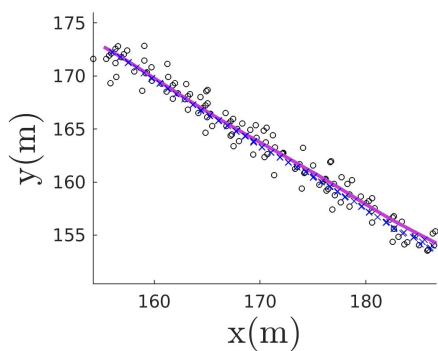
In this section, we will use the OSPA metric to compare the point estimates of the PHD and PGM models at individual time-steps⁴. Here, the OSPA distance (or just OSPA for short) of a certain set of points will refer to the OSPA distance between that set and the ground truth set.

³Of course this functionality could be reduced to an algorithm, without the concept of a graph, and possibly incorporate some of the traditional algorithms. At some point, however, the repeated addition of functionality to abstract algorithms makes them unintelligible and makes further expansion challenging.

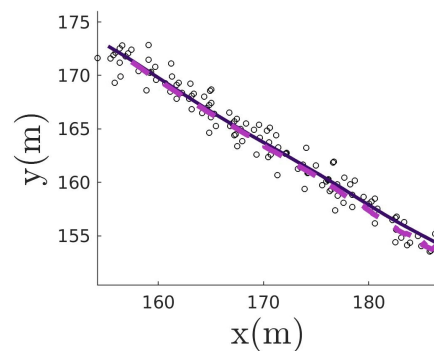
⁴The reason why we only consider point estimates, without the context of track identities, is that the PHD filter does not supply track identities in its estimates.

The PGM OSPA, for example, will refer to the distance between the PGM estimates and the ground truth target states at a specific time. Because the OSPA is calculated relative to the ground truth states, a lower OSPA will indicate a more accurate estimate. In order to provide a performance baseline, the OSPA of the detections will also be given in the results.

The software application of the GM-PHD filter⁵ that was used for the following comparison was adapted from a software implementation by Stuart Robertson [45]. We start the comparison with an example where one object was tracked. The PHD and PGM estimates can be seen in Figures 10.1a and 10.1b, respectively. The OSPA distance of the estimates of the two models as well as that of the measurements is shown in Figure 10.1c. We can see that the PHD and PGM have almost identical performance and the estimates of both are, on average, much better than that of the measurements.



(a) PHD estimates. Note that there are only point estimates, without track identities. Estimates (blue x's). True tracks (lines - mostly covered by estimates). Detections (black o's).



(b) PGM estimates. (Filtered) Estimates (magenta dashed line). True tracks (purple line). Detections (black o's).

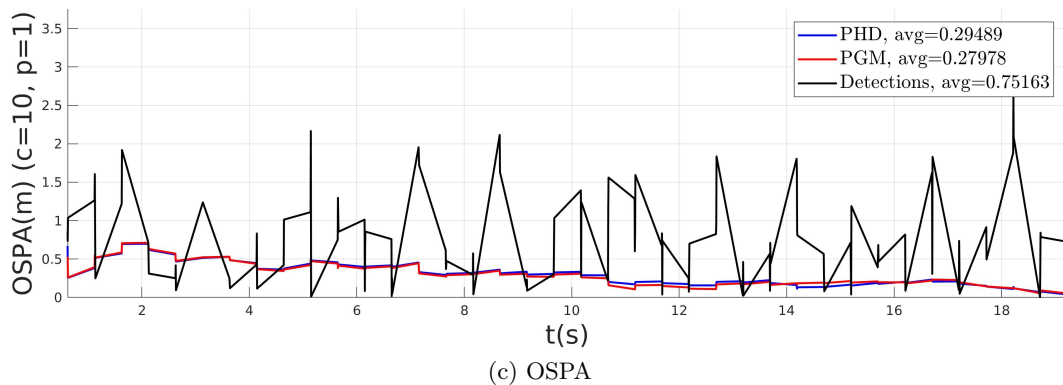


Figure 10.1: single track OSPA comparison, showing almost identical performance between the PGM and PHD for this basic case with one track, no clutter and no missed detections.

⁵Including the OSPA calculation and other supporting functions.

We will now consider a simulation with two tracks and a scanning radar. The true tracks and estimates of the PHD filter and the PGM are shown in Figures 10.2a and 10.2b, respectively. From Figure 10.2b we can see that the PGM estimates again improve on the measurements. The PHD OSPA distance, however, fluctuates between the PGM's OSPA and that of the measurements. This is most likely due to the PHD filter discarding Gaussian components when only certain components have measurements nearby⁶. These missed detections cause the weights of certain Gaussian components to be reduced and these components are then likely to be discarded by a subsequent mixture pruning step. When an object of which the corresponding Gaussian component was previously removed, is detected again, a new Gaussian component is added. However, because no past information⁷ is available to incorporate the measurement data with, the best estimate can be no more accurate than the measurement itself. The example above is therefore not a special case and the PHD filter generally suffers from poor performance when detections from all objects are not included at all time steps. Furthermore, the fact that the PHD filter is incapable of accurately tracking objects in the presence of missed detections is also well known in the literature [7] [46]. The developed PGM, as is indicated in this example, does not suffer from poor performance under these conditions. This is due to the fact that the PGM has separate distributions for each individual target and does not discard these distributions due to missed detections alone. The PGM will, however, identify tracks as clutter and remove the appropriate clusters from the cluster graph if detections are not consistently received when they are expected.

⁶Here, 'nearby' means near the mode of the distribution, or in an area of high probability.

⁷This past information is the previous state information that would have been provided by the component that was discarded

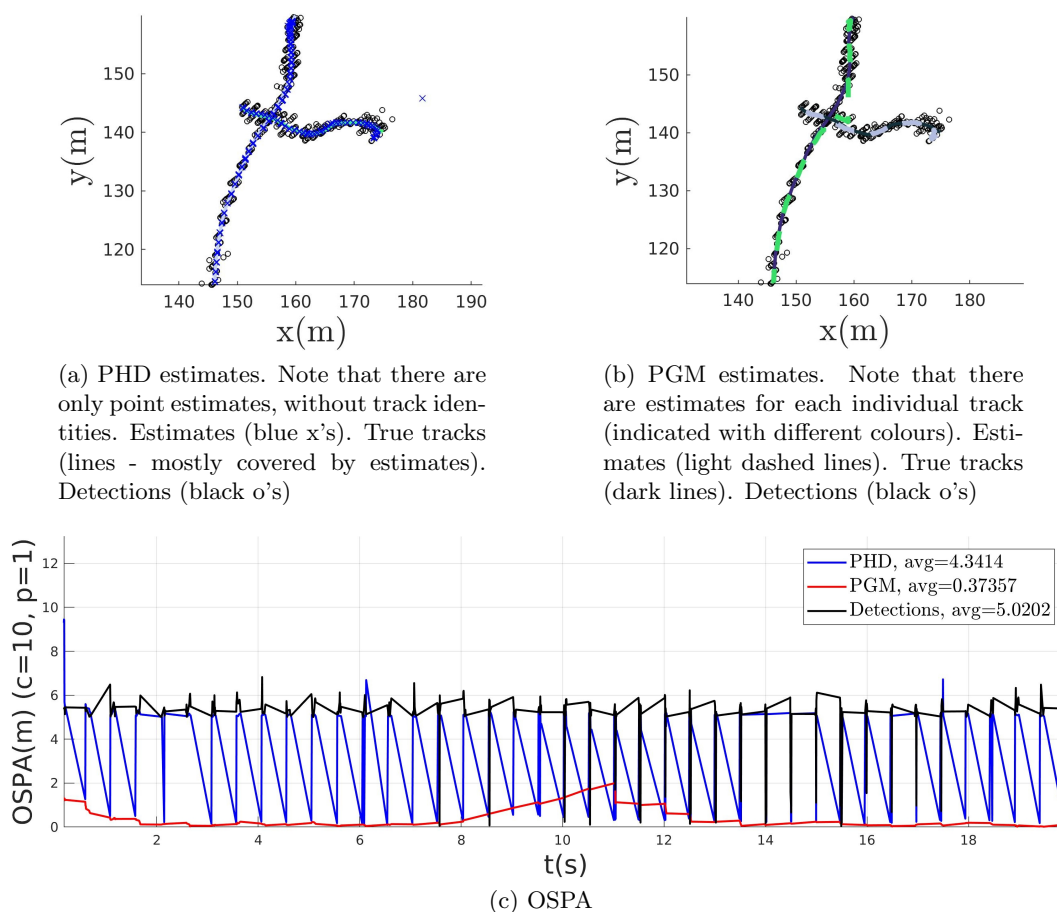


Figure 10.2: PGM and PHD two track OSPA comparison, showing the reduced performance of the PHD filter in the presence of missed detections and the good performance of the PGM under the same conditions. The PHD filter OSPA distance fluctuates due to the periodic errors that are made in the cardinality of the estimates with missed detections and the periodic corrections of these errors. Note the increase in the PGM's OSPA in the interval between 8 and 12 seconds. This is due to ambiguous detections that are received when the tracks cross.

We realise that the single example above cannot be used as statistical proof of the poor performance of the GM-PHD filter with missed detections, or of the superior performance of the developed PGM. Theoretical explanations have, however, been given for the performance difference between the two models. Furthermore, these examples illustrate a known problem with the PHD filter.

10.3 PGM Performance

Thus far we have demonstrated the performance of the developed PGM through a comparison to the GM-PHD filter in a few examples. In this section we will consider a large number of random simulations so as to provide a more statistically robust indication of the developed model's performance.

We start by investigating how the clutter density affects the performance of the model. The plot in Figure 10.3 below shows how the OSPA distance between the true tracks and the smoothed estimates changes with increasing clutter probability. Here, ten random simulations (with four targets) were run for each level of clutter intensity. The clutter intensity is the probability that a false detection will be generated at a single time-step. With a clutter intensity of 0.3, for example, false detections will be generated at almost a third of the time-steps. Since the objects are typically only detected a small percentage of the time with a sweeping radar, the number of false detections can be much higher than the true detections in these simulations. The detections OSPA is calculated using only the true detections. In Figure 10.3, we can see that the PGM performance is relatively stable with increasing clutter levels and that the estimates remain much more accurate than the true detections even with high clutter densities.

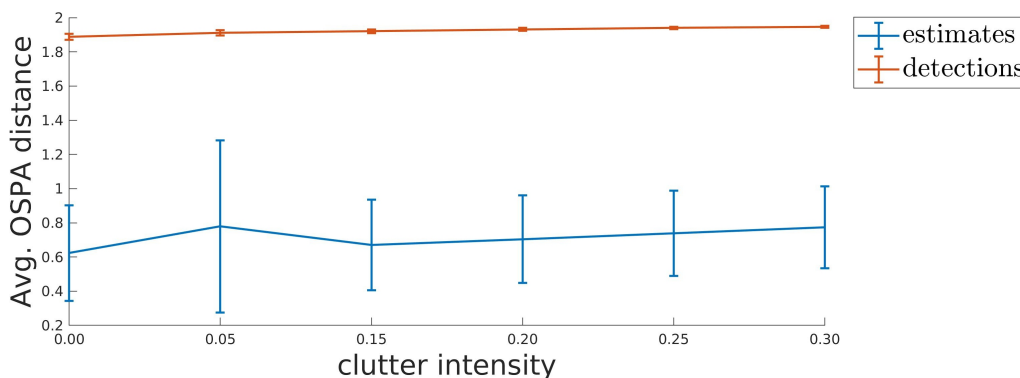


Figure 10.3: Consistent performance of the developed PGM with increasing clutter. The standard deviations are indicated by the vertical lines.

We now investigate the performance of the developed model with changes in the number of targets. Figure 10.4 below shows that the performance is very consistent with variation in the number of targets being tracked. Note that the detections OSPA increases significantly between the one and two target simulations. This is due to the missed detections that arise when more than one object is introduced when tracking with a scanning radar⁸. This figure was generated

⁸'Scanning radar', here, refers to the simulated scanning radar.

by taking the average⁹ OSPA distances of thirty random simulations for each number of targets.

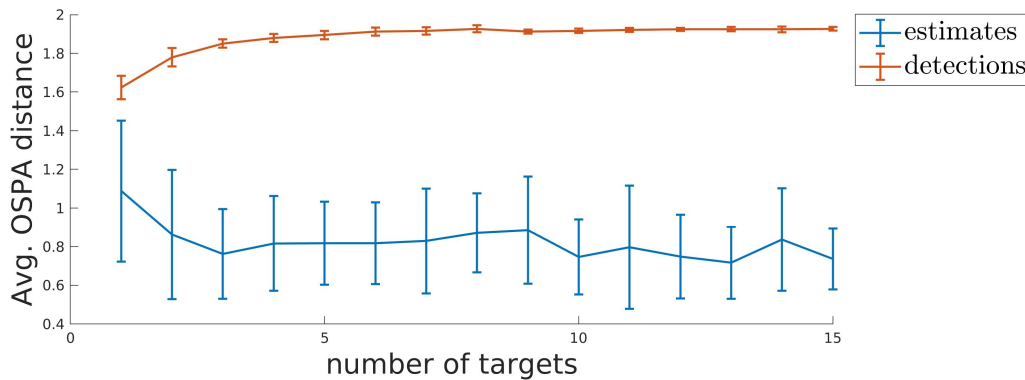


Figure 10.4: Consistent performance of the PGM with varying number of targets. The standard deviations are indicated by the vertical lines. (No clutter was introduced for these simulations.)

We will now further evaluate the average performance of the PGM by considering the performance over a hundred different simulations. The average OSPA for the filtered and smoothed estimates for each of these simulations is shown in Figure 10.5. Here we can see that the filtered estimates are consistently more accurate than the detections and that the smoothed estimates show a further improvement on the filtered estimates.

In Figure 10.5, the target number error is also shown. This error is the difference between the estimated number of targets and the true number of targets. We can see that the PGM infers the correct number of targets in the vast majority of simulations. There are, however, a few instances where there are errors in the target number. These errors can occur for a couple of reasons. Firstly, the Gaussian measurement noise can occasionally result in very unlikely detections that are far from the mean of an object's state distribution. These detections are unlikely to be associated with the objects that generated them and will result in a false track being added to the model. The clutter classification model will eventually classify this track as a clutter track, but if the track was added near the end of the simulation, this might not be possible. Furthermore, a single true track, could have more than one track estimate associated with it over its length. This could be due to unlikely track dynamics that, together with a low detection rate (due to the operation of the scanning radar), cause the track to seemingly disappear and reappear in an area that it is not expected. This results in a new track being added, while the old track will eventually be stopped by the clutter classifier. Lastly, targets that continuously move very close to each other might be indistinguishable and only have a single track associated with them. Alternatively, the underestimation of the number of targets could be due to some objects moving off the visible map before they are detected. Note that none of these scenarios cause a significant decrease in tracking performance from an OSPA perspective.

⁹These OSPA distances were first averaged over single simulations and then over the set of simulations for each number of targets.

In the case of the clutter track not being removed, the track is only present for a small amount of time and so does not effect the average tracking accuracy over the timespan of the simulation by much. This track would also likely be removed if the simulation time were to be increased. In the unlikely process dynamics case, the entire true track is still covered by accurate estimates and although the track is fragmented, this does not have a significant effect on the OSPA distance either.

In summary, we can see that the correct number of targets are tracked in the overwhelming majority of cases and, when there is an error in the estimated number of objects, this could be logically explained. Furthermore, even in the error cases, the average performance is not significantly affected from the perspective of the OSPA metric. The causes of most errors also point out fundamentally ambiguous situations and, therefore, do not necessarily indicate weaknesses in the developed model.

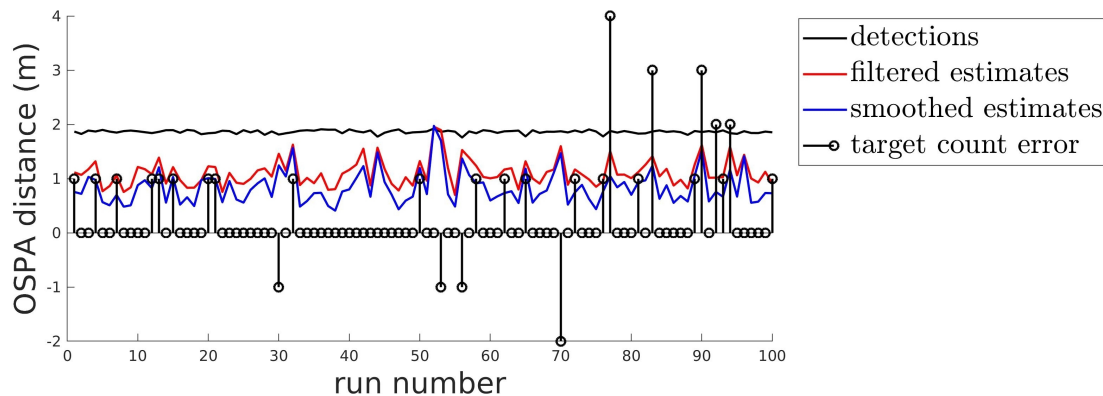


Figure 10.5: PGM performance over one hundred random simulations, showing relatively consistent tracking performance. The filtered estimates are consistently more accurate than the detections, and the smoothed estimates are consistently more accurate than the filtered estimates. The estimated number of targets is mostly correct (target count error=0) and, when it is not, the OSPA of the estimates is still low.

A simple way to evaluate the the statistical significance of the above results is to consider two hypotheses. The first hypothesis will be that the developed model provides estimates with an average accuracy that is no better than that of the detections and that the results in the above graph is a statistical fluke. The second hypothesis will be that the model gives estimates that are more accurate than the detections 98% of the time¹⁰. The probabilities of these types of hypotheses are distributed according to the binomial distribution [47, p. 868]:

$$P(n) = B(n; s, p_e) = p_e^n (1 - p_e)^{s-n} \binom{s}{n},$$

¹⁰There are two cases out of the one hundred cases where the OSPA of the filtered estimates were more or less the same as the detections.

where p_e is the probability that the model estimates are better than the detections, s is the number of simulations, and n is the number of times that the estimates are more accurate than the detections. For the first hypothesis, we stated that the probability p_e is 0.5 and for the second hypothesis, that p_e is 0.98. If we assume that the priors for the hypotheses are equal, we can calculate the relative probability as follows:

$$\frac{P(H_2|D)}{P(H_1|D)} = \frac{P(D|H_2)}{P(D|H_1)} = \frac{B(n; s, p_e = 0.98)}{B(n; s, p_e = 0.5)} \approx 6.75(10)^{25}.$$

The hypothesis that the model will improve on the accuracy of the detections in 98% of simulations is therefore vastly more probable than the first hypothesis and we can therefore state, with reasonable certainty, that the developed model consistently gives better estimates than the noisy detections.

Now that we have investigated the tracking accuracy of the PGM more thoroughly, we will turn our attention to the computational complexity of the model. Here we will investigate how the target number affects the execution time of the model. This is an important performance measure, since high-order complexities with the number of targets can severely limit the number of targets that an MOT implementation can practically track in real time. Figure 10.6 below shows the execution time of the PGM for an increasing number of targets. This figure was generated by taking the average execution times of thirty random simulations for each number of targets. These are the same simulations used to generate Figure 10.4, and the corresponding average OSPA distances can therefore be seen in this figure. The mean and standard deviations of the execution times are indicated by the error-bar plots. These simulations were run on a standard laptop with an Intel core i7 (i7-5500U CPU, 2.40 GHz) processor, utilising a single (sequential) thread. In the top figure, we can see from the lower and upper envelopes (denoted by f_1 and f_2) that the shorter execution times seem to be more or less linear with the number of targets, while the longer execution times seem to follow a quadratic function. This makes sense, since the gating approximation should allow the graph to be completely disconnected when targets are far from each other, which would result in almost¹¹ linear time complexity. When the targets are close together, the graph becomes more densely connected between the measurement and state clusters and the complexity should therefore be proportional to nm , where n is the number of targets and m is the average number of measurements at each time-step. Although the number of measurements will not be constant¹² over the different time-steps, we can still expect a greater average number of measurements with more targets if the targets are closely spaced. The expected average number of measurements is therefore proportional to the number of targets when the targets are close together. The complexity of the model under these conditions should therefore be approximately proportional to n^2 .

¹¹For the gating approximation, each state distribution still needs to be evaluated at a number of points equal to the number of measurements. When the targets are far apart, however, the average number of measurements will be much lower than the number of targets (when tracking with a scanning radar) and the operations involved with the gating approximation are much less computationally expensive than the message passing operations.

¹²This is due to the operation of the scanning radar.

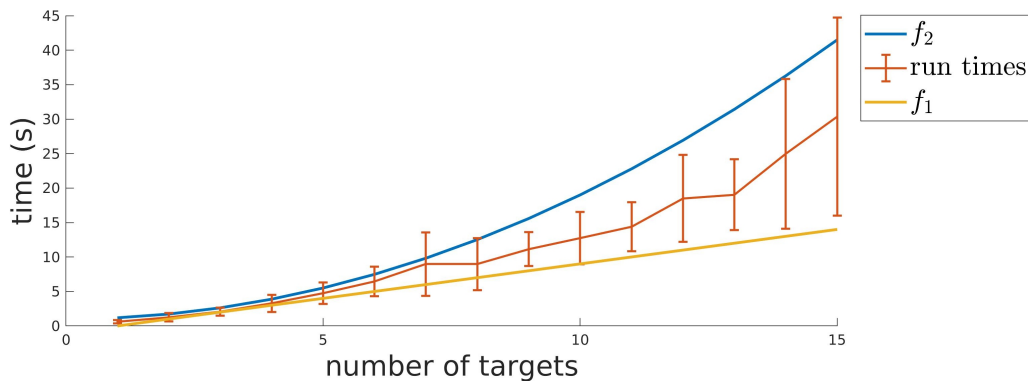


Figure 10.6: Computational complexity with number of targets. These graphs were generated from the results of thirty random simulations for each number of targets. The standard deviations are indicated by the error bars. The time axis indicates the computation time per simulation. The function f_1 is fitted to the points that are one standard deviation below the mean values of the time durations. Similarly, f_2 is fitted to the points that are one standard deviation above the means. The f_1 function seems to be close to linear, indicating that one could expect a more or less linear increase in the execution time in the best case scenario. The f_2 function seems to be quadratic, indicating that the longest execution times could be quadratic in the number of targets. These functions could, however, have small higher order components that are not visible in these plots. No false detections were introduced in these simulations. The corresponding average estimate OSPA distances for these simulations are plotted in Figure 10.4.

The plot in Figure 10.6 therefore shows how the model can operate more efficiently when the targets are spaced further apart and how the amount of computation is necessarily and automatically increased when the situation requires it. The corresponding accuracy plot in Figure 10.4 shows how the model indeed manages to maintain good performance over all the random simulations and over a wide range of target numbers.

10.4 Real Radar Data Evaluation

In Section 1.6 we presented the track outputs of the industrial object tracking application and the developed PGM, and briefly discussed the results. We noted that whereas the majority of tracks were plausible and consistent between the two models, there were also cases where there were discrepancies and seemingly incorrect results. In this section we will discuss some of these cases. Figure 10.9 below shows the real radar detections overlaid on a satellite image. The outputs of the developed model and the industrial model are given again here (see Figures 10.7 and 10.8 below) for convenience.

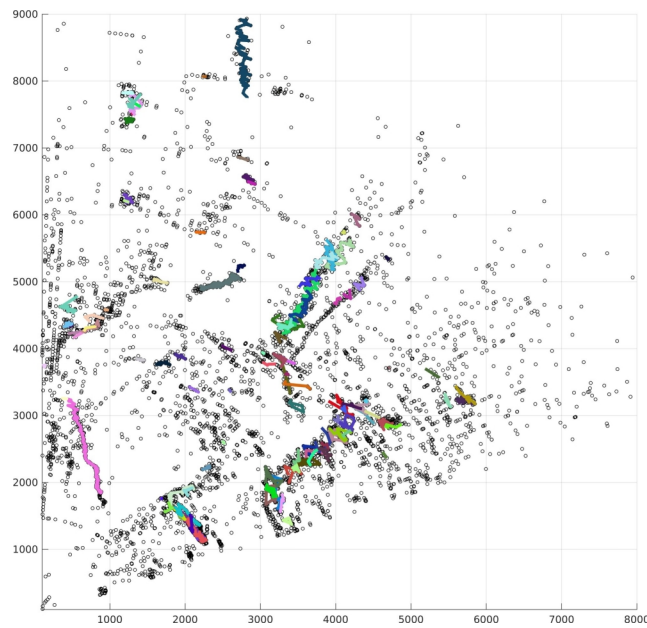


Figure 10.7: Industrial model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)

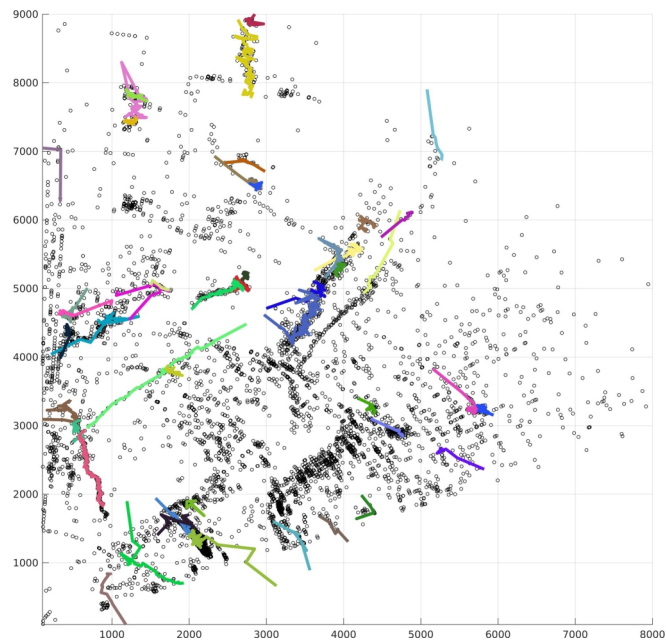


Figure 10.8: PGM model track estimates (units in meters). Note the relatively dense clutter. Detections (black o's), tracks (colour lines)

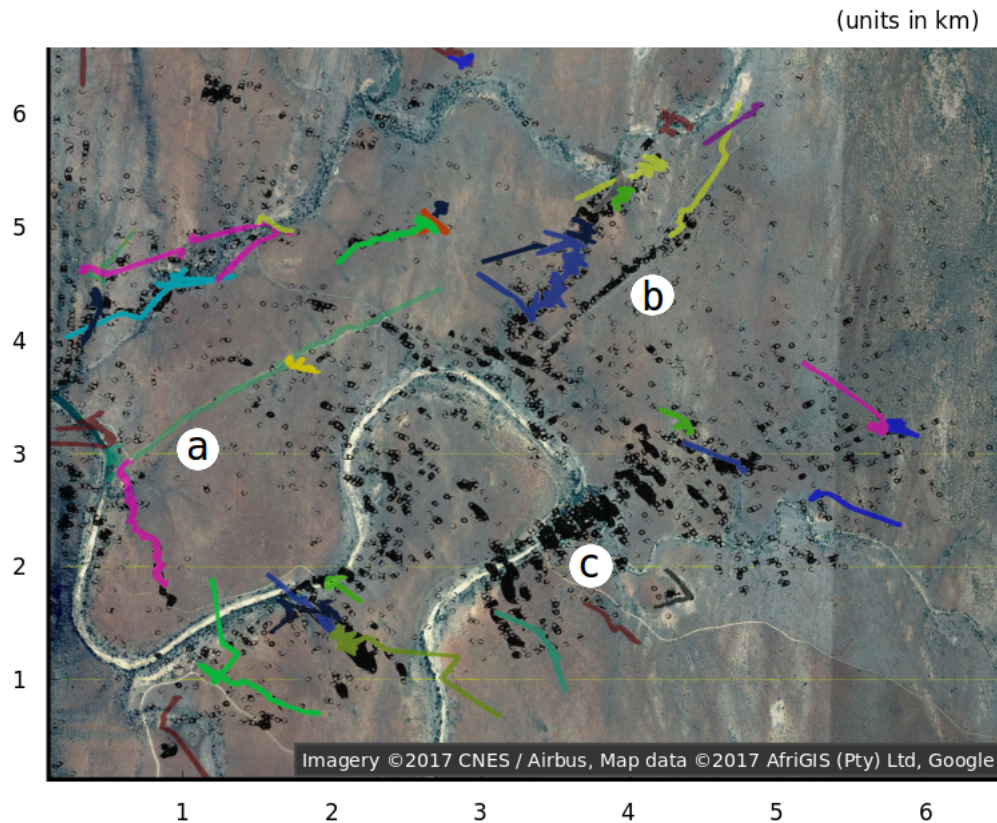


Figure 10.9: Real radar detection data and PGM track estimates overlaid on a satellite image. The data was recorded in the Kruger National Park. The radar detections are indicated by black circles and the different colours indicate different tracks. The ‘a’ and ‘c’ circles indicate areas where there are discrepancies between the output of the PGM and the industrial multiple target tracking application. The detections at ‘b’ seem to form a track, although no track is present here in either the industrial model or the PGM outputs.

The green track at ‘a’ shows one of the tracks that are not present in the industrial model track estimates. This is believed to be a motor vehicle, since the track aligns almost perfectly with a (dirt) road. The industrial application was set up to track relatively slow moving objects and this track has an average velocity of about 60 km/h. This could be the reason that it is not present in the industrial application’s output. The detections that seem to form a track but have no associated track at ‘b’ are also aligned with a road, although this is a two-lane, paved road that should have more bi-directional traffic than the dirt road aligned with the track at ‘a’. The absence of tracks here could therefore be due to seemingly inconsistent dynamics due to bidirectional traffic and the relatively high speed of the traffic with respect to the effective detection frequency. It should be noted that the output of the industrial model also does not

contain tracks in this area. The large number of detections without tracks at ‘c’ are thought to be due to trees moving in the wind, as there are many trees in this area. Lastly, there are a few additional, short tracks that are present only in the PGM’s output. These tracks are believed to be clutter tracks, where a few false detections¹³ that were, by chance, generated in such a way that they seemed to form tracks and were then stopped by the clutter classification model when they generated no further detections. Some of these tracks could also have been added near the end of the filter operation and could therefore not have been removed by the clutter classification procedure. An example of these tracks can be seen in the area below ‘c’.

The plausibility of the above explanations notwithstanding, they are speculative and there is no hard evidence that they are indeed correct. Aside from these few explainable discrepancies, however, most of the tracks are consistent between the outputs of the two models. The track estimate plots of the two respective models can be seen in Section 1.6.

In the previous section, we have shown through simulations that the developed PGM can consistently and accurately track multiple targets in the presence of clutter and missed detections. In this section we have presented an additional example that shows that the PGM is indeed capable of tracking real objects from real radar data.

10.5 Conclusion

In this chapter we discussed the similarities between the developed PGM and the traditional methods. Here we noted that, while the PGM is similar to the traditional methods in some low-level aspects, it is different from the traditional methods at a high level. Furthermore, it was shown that the PGM does not suffer from poor performance as the PHD filter does when there are multiple missed detections. Additional performance tests show that the PGM consistently offers high performance under many different and challenging circumstances. From these tests we have also seen that the computational complexity of the model is low in simple problems and automatically increases in order to maintain good performance when required. Lastly, the results of the real radar data test indicate that the PGM does not only work with simulated data, but can be applied to real-world problems.

This chapter concludes the discussion about the developed PGM. In the next and final chapter we will look back on the work that has been done and consider possible future work that can be done in order to further expand and improve the developed model.

¹³Which could have been caused due to trees moving in the wind.

Chapter 11

Conclusion And Future Work

11.1 Conclusion

The objective of this work was to develop a probabilistic graphical model for effective multiple object tracking. To this end, we have studied and discussed the necessary probability and PGM theory and shown how the theory can be applied to solve the multiple object tracking problem. Most of the theory that we have discussed in this work is, however, applicable to PGMs in general and is not limited to multiple object tracking applications. Similarly, the factor classes that were implemented in software are also very general and can be used to construct and perform inference in general, hybrid graphical models. These classes therefore also contribute important functionality to the EMDW library and could allow for faster development of future PGM applications.

Multiple object tracking is a complex problem and finding a solution to this problem is challenging. The approach taken to solve this problem was to start with an oversimplified graphical model and to incrementally increase functionality and complexity. To this end, we used the basic Kalman filter as the foundation for the developed model. We first investigated the connections between the Kalman filter algorithm and the corresponding graphical model. We then extended this model to allow multiple objects to be tracked by solving the data-association problem with the use of discrete association variables. Here we noticed that interesting and logical data-association characteristics arise automatically from the graph structure. This model could, however, only track a fixed and known number of targets and we therefore needed to extend its functionality further. Bayesian model selection was used to allow the model to automatically determine the number of targets. The model that was developed up to this point still had no mechanism for identifying false detections, and we concluded that this would be impossible on a single detection basis. We therefore used inspiration from the logical reasoning that a person might apply to solve the clutter problem and implemented similar probabilistic logic in a PGM. This model was integrated into the object-tracking PGM, allowing the model to

infer the probability that a track is real by considering all the possible detections in the context of the larger track instead of focussing on single detections. The improved model finally had all the necessary functionality for tracking multiple targets in a realistic setting.

The model developed thus far was, however, found to be very computationally expensive and would therefore be practically unusable for tracking a large number of targets. In order to increase efficiency, we first focussed on the most computationally expensive aspect of the model - the model selection subroutine. Here we discovered that model selection can be performed much more efficiently through inference in a single graph. This was a surprisingly elegant solution, as it allowed both variable inference and structure learning to be performed through message passing. With the more efficient model selection method successfully implemented, we turned our attention to a seemingly unnecessary and computationally expensive characteristic of the developed model. This was the fact that the model considered the possibility that any detection could be associated with any target on the map no matter how far away the detection was from the target. In order to improve efficiency on this aspect of the model, a detection gating subroutine was implemented in order to efficiently rule out unlikely detection associations and allow for more sparsely connected graphs to be constructed. These two improvements vastly increased the efficiency of the model.

Next we turned our attention to the tracking accuracy of the model. Here we focussed on one of the approximations that is made by the model - the moment matching approximation of Gaussian mixtures. The reason that we could not make proper use of Gaussian mixtures in the model was that Gaussian mixture quotients do not generally have a Gaussian mixture form. This conflicts with our closed-form requirement for operations on distributions, and no solution to this problem could be found in the literature. We therefore sought to find a method for approximating a Gaussian mixture quotient as a Gaussian mixture. We designed an algorithm to perform this approximation and tested the performance of the algorithm. It was found that the algorithm can determine almost the exact Gaussian mixture when the true quotient function is a Gaussian mixture, and that it often yields good approximations even when it is not. The extent to which the use of this algorithm would make a difference in an MOT application was still unclear, since there are good reasons why the moment matching approximation should be accurate in the majority of situations that will arise in such an application (discussed in Section 6.2). It was, however, shown that the proper use of Gaussian mixtures, which is enabled by the developed algorithm, can increase performance in certain challenging scenarios. It was also shown that the use of the algorithm can make the difference between correct and incorrect track identifications (see Section 8.3).

Finally, the completed model was compared to the existing, traditional multiple object tracking algorithms. Here it was found that the PGM shares some low-level aspects with the traditional models, but that it is more general at a high level. The developed PGM with the clutter track classification functionality also has the advantage that it does not require the clutter density or detection probability to be known. Furthermore, it was shown that the PGM does not suffer from poor performance when there are frequent missed detections, as the PHD filter does.

In order to assess the performance of the developed PGM in a statistically meaningful manner, a large number of random tests were performed. Here it was found that the PGM can consistently and accurately track multiple targets in challenging scenarios and under a wide range of circumstances. Finally, the model was tested on real radar data through a qualitative comparison with an industrial model. The results of the outputs were largely similar and possible explanations for the few discrepancies were given. The results of this test indicate that the model not only works in a simulated environment, but can also be used in real-world applications.

In this work we have therefore shown that probabilistic graphical models can be successfully used for multiple object tracking, and that some improvements can be made to some of the existing algorithms.

11.2 Future Work

Although the model developed in this work performs well, there are a few interesting ideas for additional improvements that can possibly be investigated in future work. Some of these ideas can be implemented in the existing model and some will require the development of new PGMs. Due to the interpretable and modular nature of the PGM framework, these PGMs could easily be incorporated into the existing model.

One possible extension to the developed model could be to construct and incorporate a PGM for inferring the distribution of clutter over the tracking area. This could possibly allow more efficient tracking through immediate rejection of detections in regions that are known to have high clutter density. A similar model could possibly be developed for learning where on a map object occlusions are likely to occur. Such a model could prevent tracks from being classified as clutter when they are not detected due to occlusions.

Target classification is also an interesting problem and one that has received significant attention in video tracking applications. This problem seems to have received significantly less attention in radar tracking applications. Some radar systems can provide micro-Doppler data that can contain information about the characteristics of a certain object. This information could possibly be used for target identification. This problem was investigated during the course of this work, although it is not discussed in this text¹. It was reasoned that the micro-Doppler measurement distribution should not be used for classification directly, but should be transformed to be relative to the object's movement. This created the problem that the variables that should be used for classification will not be observed, and information about their true states can only be obtained indirectly and through the observed micro-Doppler data. The process of learning the parameters of a Gaussian distribution from data is widely discussed in the literature

¹Some progress was made during this investigation and some work was done towards the realisation of a MOT PGM that could also classify targets from micro-Doppler signals, but a working model could not be completed due to fundamental mathematical limitations and time constraints.

and can be done without resorting to approximations. The process of learning such parameters indirectly through only probabilistic relationships with other, observed variables, however, does not seem to be discussed in any of the literature. This is also believed to be mathematically impossible without resorting to approximations. Object classification from micro-Doppler data in a PGM framework therefore seems to be a very challenging problem and the implementation of such a system would require an in-depth investigation, probably requiring the development of new approximation methods.

Lastly, the use of detections from multiple sensors could also be a useful extension to the developed model. This extension should require relatively little modification to the existing model and should only require the joint association priors to be changed and the relevant state-measurement transformations and sensor noise covariance matrices to be incorporated into the correct measurement clusters.

In summary, although the developed model performs well and is capable of efficiently tracking multiple targets, there are many interesting improvements that can possibly be introduced in future work. These improvements would have the potential to significantly increase the functionality of the model. The feasibility and specifics of these improvements will, however, require thorough investigation.

Bibliography

- [1] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, ser. Adaptive computation and machine learning. MIT Press, 2009, ISBN: 9780262013192.
- [2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME—Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [3] M. Mallick, B.-N. Vo, T. Kirubarajan, and S. Arulampalam, "Introduction to the issue on multitarget tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 373–375, 2013.
- [4] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, 2008.
- [5] R. P. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003.
- [6] B.-N. Vo and W.-K. Ma, "The Gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.
- [7] M. Yazdian-Dehkordi and Z. Azimifar, "An improvement on GM-PHD filter for target tracking in presence of subsequent miss-detection," in *Electrical Engineering (ICEE), 2015 23rd Iranian Conference on*, IEEE, 2015, pp. 765–769.
- [8] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.
- [9] T. L. Song, D. Musicki, H. W. Kim, and F. Govaers, "Gaussian mixture tracking: MHT and ITS comparison," in *Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2013 Workshop on*, IEEE, 2013, pp. 1–6.
- [10] G. Pulford, "Taxonomy of multiple target tracking methods," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 291–304, 2005.
- [11] A. V. Segal and I. Reid, "Latent data association: Bayesian model selection for multi-target tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2904–2911.
- [12] Z. Chen, "Efficient multi-target tracking using graphical models," PhD thesis, Massachusetts Institute of Technology, 2008.

- [13] M. J. Schiegg, “Multi-target tracking with probabilistic graphical models,” PhD thesis, 2015.
- [14] S. H. Rezatofghi, S. Gould, B. T. Vo, B.-N. Vo, K. Mele, and R. Hartley, “Multi-target tracking with time-varying clutter rate and detection profile: application to time-lapse cell microscopy sequences,” *IEEE Transactions on Medical Imaging*, vol. 34, no. 6, pp. 1336–1348, 2015.
- [15] C. Kreucher, K. Kastella, and A. O. Hero, “Multitarget tracking using the joint multitarget probability density,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1396–1414, 2005.
- [16] M. Mallick, V. Krishnamurthy, and B.-N. Vo, *Integrated tracking, classification, and sensor management: theory and applications*. John Wiley & Sons, 2012.
- [17] K. Panta, *Multi-target tracking using 1st moment of random finite sets*. University of Melbourne, Department of Electrical and Electronic Engineering, 2008.
- [18] K. Panta, B.-N. Vo, and S. Singh, “Novel data association schemes for the probability hypothesis density filter,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 2, 2007.
- [19] B.-N. Vo and W.-K. Ma, “A closed-form solution for the probability hypothesis density filter,” in *Information Fusion, 2005 8th International Conference on*, IEEE, vol. 2, 2005.
- [20] K. Y. Leung, F. Inostroza, and M. Adams, “Relating random vector and random finite set estimation in navigation, mapping and tracking,” *IEEE Transactions on Signal Processing*, 2017.
- [21] N. L. Baisa and A. M. Wallace, “Development of a n-type GM-PHD filter for multiple target, multiple type visual tracking,” *CoRR*, vol. abs/1706.00672, 2017. arXiv: 1706.00672. [Online]. Available: <http://arxiv.org/abs/1706.00672>.
- [22] B.-T. Vo, B.-N. Vo, and A. Cantoni, “The cardinalized probability hypothesis density filter for linear Gaussian multi-target models,” in *Information Sciences and Systems, 2006 40th Annual Conference on*, IEEE, 2006, pp. 681–686.
- [23] L. D. Stone, R. L. Streit, T. L. Corwin, and K. L. Bell, *Bayesian multiple target tracking*. Artech House, 2013.
- [24] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis, and G. Karaseitanidis, “Multiple hypothesis tracking implementation,” in *Laser Scanner Technology*, InTech, 2012.
- [25] B. T. Vo, “Random finite sets in multi-object filtering,” PhD thesis, University of Western Australia, 2008.
- [26] S. R. Maskell, R. G. Everitt, R. Wright, and M. Briers, “Multi-target out-of-sequence data association: Tracking using graphical models,” vol. 7, pp. 434–447, 2006. DOI: 10.1016/j.inffus.2005.07.001.
- [27] C. Wang, M. de La Gorce, and N. Paragios, “Segmentation, ordering and multi-object tracking using graphical models,” in *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 2009, pp. 747–754.

- [28] M. I. Jordan, “Graphical models,” vol. 19, 2003. [Online]. Available: https://projecteuclid.org/download/pdfview_1/euclid.ss/1089808279.
- [29] S. Streicher and J. A. du Preez, “Graph coloring: comparing cluster graphs to factor graphs,” in *Proceedings of the ACM Multimedia 2017 Workshop on South African Academic Participation*, ser. SAWACMMM ’17, Mountain View, California, USA: ACM, 2017, pp. 35–42, ISBN: 978-1-4503-5505-6. DOI: 10.1145/3132711.3132717. [Online]. Available: <http://doi.acm.org/10.1145/3132711.3132717>.
- [30] J. A. du Preez, *EMDW: A Pragmatic Approach to PGMs*.
- [31] 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*, Ieee, 2000, pp. 153–158.
- [32] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C*. Cambridge university press Cambridge, 1996, vol. 2.
- [33] I. C. Ipsen, *Numerical matrix analysis: Linear systems and least squares*. SIAM, 2009.
- [34] M. Montemerlo and S. Thrun, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*. Springer, 2007, vol. 27.
- [35] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [36] E. Britannica, *Occam’s razor*, <https://www.britannica.com/topic/Occams-razor>, (Accessed on 04/11/2017), Jun. 2015.
- [37] J. A. du Preez, *Discrete variable factors for model selection*, personal communication, May 2017.
- [38] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple hypothesis tracking revisited,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.
- [39] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
- [40] D. F. Crouse, P. Willett, K. R. Pattipati, and L. Svensson, “A look at Gaussian mixture reduction algorithms,” *14th International Conference on Information Fusion*, pp. 1–8, 2011.
- [41] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [42] K.-V. Yuen, “Appendix a: relationship between the hessian and covariance matrix for Gaussian random variables,” *Bayesian Methods for Structural Dynamics and Civil Engineering*, pp. 257–262, 2010.
- [43] G. Sfikas, C. Constantinopoulos, A. Likas, and N Galatsanos, “An analytic distance metric for Gaussian mixture models with application in image retrieval,” *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pp. 755–755, 2005.
- [44] D. E. King, “Dlib-ml: a machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

- [45] S Robertson, *GM-PHD filter implementation*, personal communication, May 2017.
- [46] Z. Liu, Q. Zhang, and Y. Zou, “Sequential measurement-driven multi-target Bayesian filter for nonlinear multi-target models,” in *Signal Processing (ICSP), 2016 IEEE 13th International Conference on*, IEEE, 2016, pp. 1524–1528.
- [47] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [48] C. B. Do, “More on multivariate Gaussians,” 2008.
- [49] T. Tao, *Matrix identities as derivatives of determinant identities*, <https://terrytao.wordpress.com/2013/01/13/matrix-identities-as-derivatives-of-determinant-identities/>, Accessed: 2017-12-15, 2013.
- [50] C. Valente, *Hessian matrix*, <http://planetmath.org/hessianmatrix>, Accessed: 2017-09-30, 2002.

Appendices

Appendix A

Proofs and Derivations

A.1 Derivations of Marginal Expressions from Variable Elimination Example

In this section, the derivations for all the marginals in the variable elimination example in Section 3.2 are given. The Bayes network is shown again here for convenience.

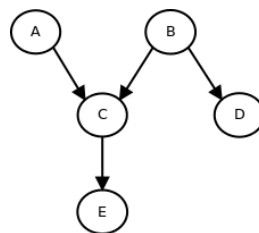


Figure A.1: Simple example of a Bayes Network

$$P(A) = \sum_{C,B} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C)$$

$$P(B) = \sum_{A,C} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C)$$

$$P(C) = \sum_{A,B} \psi_2(C, A, B) \delta_{32}(B) \delta_{12}(C)$$

$$P(B) = \sum_D \psi_3(D, B) \delta_{23}(B)$$

$$P(C) = \sum_E \psi_1(E, C) \delta_{21}(C)$$

$$P(D) = \sum_B \psi_3(D, B) \delta_{23}(B)$$

$$P(E) = \sum_C \psi_1(E, C) \delta_{21}(C)$$

$$\begin{aligned} P(E) &= \sum_{A,B,C,D} P(A, B, C, D, E) \\ &= \sum_{A,B,C,D} P(C|A, B) P(D|B) P(E|C) P(A) P(B) \\ &= \sum_{A,B,C} P(C|A, B) P(A) P(E|C) \sum_D P(D|B) P(B) \\ &= \sum_{A,B,C} P(C|A, B) P(A) P(E|C) \sum_D \psi_3(D, B) \\ &= \sum_{A,B,C} P(C|A, B) P(A) P(E|C) \delta_{32}(B) \\ &= \sum_C P(E|C) \sum_{A,B} P(C|A, B) P(A) \delta_{32}(B) \\ &= \sum_C P(E|C) \sum_{A,B} \psi_2(C, A, B) \delta_{32}(B) \\ &= \sum_C \psi_1(E, C) \delta_{21}(C) \\ P(C) &= \sum_E \psi_1(E, C) \delta_{21}(C) \end{aligned}$$

$$\begin{aligned}
P(C) &= \sum_{A,B,D,E} P(A, B, C, D, E) \\
&= \sum_{A,B,E} P(C|A, B)P(E|C)P(A) \sum_D P(D|B)P(B) \\
&= \sum_{A,B,E} P(C|A, B)P(E|C)P(A) \sum_D \psi_3(D, B) \\
&= \sum_{A,B,E} P(C|A, B)P(E|C)P(A)\delta_{32}(B) \\
&= \sum_{A,B} P(C|A, B)P(A)\delta_{32}(B) \sum_E P(E|C) \\
&= \sum_{A,B} P(C|A, B)P(A)\delta_{32}(B) \sum_E \psi_3(E, C) \\
&= \sum_{A,B} \psi_2(C, A, B)\delta_{32}(B)\delta_{12}(C) \\
P(A) &= \sum_{C,B} \psi_2(C, A, B)\delta_{32}(B)\delta_{12}(C) \\
P(B) &= \sum_{A,C} \psi_2(C, A, B)\delta_{32}(B)\delta_{12}(C)
\end{aligned}$$

A.2 Multivariate Gaussian Operation Derivations

The derivations for the multivariate Gaussian operations presented in Section 4.1.2 are given in this appendix. A few useful equalities and conventions that are used in these derivations are discussed below.

Note that the cross covariance Σ_{XY} between X and Y is the transpose of the cross covariance Σ_{YX} between Y and X . This is also true for the precision and the following equalities are used in the derivations in this appendix:

$$\Sigma_{YX} = \Sigma_{XY}^T \text{ and } \mathbf{K}_{YX} = \mathbf{K}_{XY}^T.$$

Furthermore, the covariance and precision matrices Σ , Σ_{XX} , Σ_{YY} , \mathbf{K} , \mathbf{K}_{XX} , \mathbf{K}_{YY} are all symmetric and assumed to be non-singular.

With \mathbf{a} and \mathbf{c} being $n \times 1$ column vectors and \mathbf{B} being an $n \times n$ matrix, the following equality holds

$$\mathbf{a}^T \mathbf{B} \mathbf{c} = (\mathbf{a}^T \mathbf{B} \mathbf{c})^T = \mathbf{c}^T \mathbf{B}^T \mathbf{a} \quad (\text{A.2.1})$$

The first equality above holds because the quadratic form reduces to a scalar and the transpose of a scalar is the scalar itself.

In these derivations, the zero matrix (or vector) $\mathbf{0}$ will implicitly have the correct dimensionality as per the context in which it is used.

The vertical concatenation of two matrices or vectors A and B will be indicated by: $[A, B]$.

A.2.1 Block Matrix Inversion Lemma

The block matrix inversion lemma can be expressed as follows [39, p. 87]:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{M} & -\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{M} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}$$

where $\mathbf{M} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$.

This lemma is useful for describing the relationships between the sub-matrices in the covariance and precision matrices of Gaussian distributions. This lemma will therefore be used in some of the derivations in this appendix. The partitioned matrix inverses for the covariance and precision matrices are shown below.

$$\begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}_{YX} & \boldsymbol{\Sigma}_{YY} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{M} & -\mathbf{M}\boldsymbol{\Sigma}_{XY}\boldsymbol{\Sigma}_{YY}^{-1} \\ -\boldsymbol{\Sigma}_{YY}^{-1}\boldsymbol{\Sigma}_{YX}\mathbf{M} & \boldsymbol{\Sigma}_{YY}^{-1} + \boldsymbol{\Sigma}_{YY}^{-1}\boldsymbol{\Sigma}_{YX}\mathbf{M}\boldsymbol{\Sigma}_{XY}\boldsymbol{\Sigma}_{YY}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \quad (\text{A.2.2})$$

where $\mathbf{M} = (\boldsymbol{\Sigma}_{XX} - \boldsymbol{\Sigma}_{XY}\boldsymbol{\Sigma}_{YY}^{-1}\boldsymbol{\Sigma}_{YX})^{-1}$.

$$\begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{M} & -\mathbf{M}\mathbf{K}_{XY}\mathbf{K}_{YY}^{-1} \\ -\mathbf{K}_{YY}^{-1}\mathbf{K}_{YX}\mathbf{M} & \mathbf{K}_{YY}^{-1} + \mathbf{K}_{YY}^{-1}\mathbf{K}_{YX}\mathbf{M}\mathbf{K}_{XY}\mathbf{K}_{YY}^{-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}_{YX} & \boldsymbol{\Sigma}_{YY} \end{bmatrix} \quad (\text{A.2.3})$$

where $\mathbf{M} = (\mathbf{K}_{XX} - \mathbf{K}_{XY}\mathbf{K}_{YY}^{-1}\mathbf{K}_{YX})^{-1}$

A.2.2 Gaussian Covariance Form to Canonical Form Derivation

The covariance form can be converted into the canonical form as shown in the derivation below:

$$\begin{aligned} \mathcal{N}(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) &= \exp(-0.5[X - \boldsymbol{\mu}]^T \boldsymbol{\Sigma}^{-1}[X - \boldsymbol{\mu}] + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)) \\ &= \exp(-0.5(X^T \boldsymbol{\Sigma}^{-1} X - \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} X - X^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)) \\ &= \exp(-0.5(X^T \boldsymbol{\Sigma}^{-1} X - 2X^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)) \\ &= \exp(-0.5X^T \boldsymbol{\Sigma}^{-1} X + X^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - 0.5\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)) \\ \mathcal{N}(X; \mathbf{K}, \mathbf{h}, g) &= \exp(-0.5X^T \mathbf{K} X + X^T \mathbf{h} + g), \end{aligned}$$

where the canonical parameters are defined as

$$\begin{aligned} \mathbf{K} &= \boldsymbol{\Sigma}^{-1} \\ \mathbf{h} &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \\ g &= -0.5\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|) \\ &= -0.5\mathbf{h}^T \boldsymbol{\Sigma} \mathbf{h} + \log(w) - 2\log(|2\pi\mathbf{K}^{-1}|). \end{aligned}$$

A.2.3 Gaussian Canonical Product and Quotient Derivations

A product of two Gaussian distributions with overlapping variable scope can be expressed, in canonical form, as follows:

$$\mathcal{C}_1([X, Y]; \mathbf{K}_1, \mathbf{h}_1, g_1) \mathcal{C}_2(X; \mathbf{K}_2, \mathbf{h}_2, g_2) = \mathcal{C}_3([X, Y]; \mathbf{K}', \mathbf{h}', g').$$

Note that the two distributions not have the same scope. We can however increase the scope of the second distribution to match that of the first, by augmenting the parameters as follows:

$$X^T \mathbf{K}_2 X = \begin{bmatrix} X \\ Y \end{bmatrix}^T \mathbf{K}'_2 \begin{bmatrix} X \\ Y \end{bmatrix} \quad X^T \mathbf{h}_2 = [X, Y]^T \mathbf{h}'_2,$$

with

$$\mathbf{K}'_2 = \begin{bmatrix} \mathbf{K}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{h}'_2 = \begin{bmatrix} \mathbf{h}_2 \\ \mathbf{0} \end{bmatrix}.$$

We can now rewrite and compute the product as follows:

$$\begin{aligned} & \mathcal{C}_1([X, Y]; \mathbf{K}_1, \mathbf{h}_1, g_1) \mathcal{C}_2(X; \mathbf{K}'_2, \mathbf{h}'_2, g_2) \\ &= \exp(-0.5[X, Y]^T \mathbf{K}_1 [X, Y] + [X, Y]^T \mathbf{h}_1 + g_1) \exp(-0.5[X, Y]^T \mathbf{K}'_2 [X, Y] + [X, Y]^T \mathbf{h}'_2 + g_2) \\ &= \exp(-0.5[X, Y]^T \mathbf{K}_1 [X, Y] + [X, Y]^T \mathbf{h}_1 + g_1 - 0.5[X, Y]^T \mathbf{K}'_2 [X, Y] + [X, Y]^T \mathbf{h}'_2 + g_2) \\ &= \exp(-0.5[X, Y]^T (\mathbf{K}_1 + \mathbf{K}'_2) [X, Y] + [X, Y]^T (\mathbf{h}_1 + \mathbf{h}_2) + g_1 + g_2) \\ &= \exp(-0.5[X, Y]^T \mathbf{K}' [X, Y] + [X, Y]^T \mathbf{h}' + g') \\ &= \mathcal{C}_3([X, Y]; \mathbf{K}', \mathbf{h}', g'), \end{aligned}$$

where

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{1,XX} + \mathbf{K}_2 & \mathbf{K}_{1,XY} \\ \mathbf{K}_{1,YX} & \mathbf{K}_{1,YY} \end{bmatrix}, \quad \mathbf{h}' = \begin{bmatrix} h_{1,X} + h_2 \\ \mathbf{h}_Y \end{bmatrix}, \quad g' = g_1 + g_2.$$

Similarly, it can be shown that the quotient of two Gaussian distributions is

$$\frac{\mathcal{C}_1([X, Y]; \mathbf{K}_1, \mathbf{h}_1, g_1)}{\mathcal{C}_2(X; \mathbf{K}_2, \mathbf{h}_2, g_2)} = \mathcal{C}_3([X, Y]; \mathbf{K}', \mathbf{h}', g'),$$

with

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{1,XX} - \mathbf{K}_2 & \mathbf{K}_{1,XY} \\ \mathbf{K}_{1,YX} & \mathbf{K}_{1,YY} \end{bmatrix}, \quad \mathbf{h}' = \begin{bmatrix} h_{1,X} - h_2 \\ \mathbf{h}_Y \end{bmatrix}, \quad g' = g_1 - g_2.$$

A.2.4 Gaussian Marginal Derivation

The derivation for multivariate Gaussian marginalisation is given in this section. This derivation is adapted from a derivation in [48, pp. 6-7]. The multivariate Gaussian can be written in a partitioned matrix form as:

$$\mathcal{N}([X, Y]; \boldsymbol{\mu}, \mathbf{K}) = \exp(-0.5 \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix} - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})). \quad (\text{A.2.4})$$

We can expand the partitioned quadratic form in the exponent in the above expression to give:

$$\begin{aligned} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix} &= (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) + (Y - \boldsymbol{\mu}_Y)^T \mathbf{K}_{XY} (X - \boldsymbol{\mu}_X) \\ &\quad + (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{YX} (Y - \boldsymbol{\mu}_Y) + (Y - \boldsymbol{\mu}_Y)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y) \end{aligned}$$

Now, since $\mathbf{K}_{YX} = \mathbf{K}_{XY}$ and the transpose of a scalar is the scalar itself, the following equality holds:

$$(X - \boldsymbol{\mu}_X)^T \mathbf{K}_{YX} (Y - \boldsymbol{\mu}_Y) = (Y - \boldsymbol{\mu}_Y)^T \mathbf{K}_{XY} (X - \boldsymbol{\mu}_X)$$

and we can simplify the quadratic form expansion to:

$$\begin{aligned} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix} &= (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) + 2(Y - \boldsymbol{\mu}_Y)^T \mathbf{K}_{XY} (X - \boldsymbol{\mu}_X) \\ &\quad + (Y - \boldsymbol{\mu}_Y)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y). \end{aligned} \quad (\text{A.2.5})$$

We can now use the method of ‘completion of squares’ to rewrite the above expression in a different form. The ‘completion of squares’ method for a quadratic matrix function is given below.

$$Z^T \mathbf{A} Z + 2B^T Z + 2c = (Z + \mathbf{A}^{-1} B)^T \mathbf{A} (Z + \mathbf{A}^{-1} B) - B^T \mathbf{A}^{-1} B + 2c, \quad (\text{A.2.6})$$

Using equation A.2.6 above, together with the following substitutions:

$$\begin{aligned} Z &= Y - \boldsymbol{\mu}_Y, \\ \mathbf{A} &= \mathbf{K}_{YY}, \\ B &= \mathbf{K}_{YX} (X - \boldsymbol{\mu}_X), \\ c &= 0.5 (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X). \end{aligned}$$

we can rewrite the expression in A.2.5, as follows:

$$\begin{aligned}
& (Y^T - \boldsymbol{\mu}_Y^T) \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y) \\
& + (X^T - \boldsymbol{\mu}_X^T) \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) \\
& + 2(X^T - \boldsymbol{\mu}_X^T) \mathbf{K}_{YX} (Y - \boldsymbol{\mu}_Y) \\
& = Z^T \mathbf{A} Z + B^T Z + 2c \\
& = (Z + \mathbf{A}^{-1} B)^T \mathbf{A} (Z + \mathbf{A}^{-1} B) - B^T \mathbf{A}^{-1} B + 2c \\
& = (Y - \boldsymbol{\mu}_Y + \mathbf{K}_{YY}^{-1} (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X)))^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y + \mathbf{K}_{YY}^{-1} (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X))) \\
& \quad - (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X))^T (\mathbf{K}_{YY})^{-1} (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X)) + (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) \\
& = (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B) \\
& \quad - (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X))^T (\mathbf{K}_{YY})^{-1} (\mathbf{K}_{YX} (X - \boldsymbol{\mu}_X)) + (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) \\
& = (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B) \\
& \quad - (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX} (X - \boldsymbol{\mu}_X) + (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) \\
& = (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B) \\
& \quad + (X - \boldsymbol{\mu}_X)^T (\mathbf{K}_{XX} - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX}) (X - \boldsymbol{\mu}_X).
\end{aligned}$$

From the block matrix inversion lemma in appendix A.2.1 we can see that the above expression can be re-written as follows:

$$\begin{aligned}
& (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B)^T \mathbf{K}_{YY} (Y - \boldsymbol{\mu}_Y + \mathbf{A}^{-1} B) + (X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1}) (X - \boldsymbol{\mu}_X) \\
& = (Y - D)^T \mathbf{K}_{YY} (Y - D) + (X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1}) (X - \boldsymbol{\mu}_X),
\end{aligned}$$

where we have defined $D = \boldsymbol{\mu}_Y - \mathbf{A}^{-1} B$ for compactness of representation. We can now reincorporate the above expression into equation A.2.4 to give the following expression:

$$\begin{aligned}
& \mathcal{N}([X, Y]; \boldsymbol{\mu}, \mathbf{K}) \\
& = \exp(-0.5((Y - D)^T \mathbf{K}_{YY} (Y - D) - (X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1}) (X - \boldsymbol{\mu}_X)) - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}))
\end{aligned}$$

and we can rewrite the this expression as the product of two Gaussian functions as follows:

$$\begin{aligned}
& \mathcal{N}([X, Y]; \boldsymbol{\mu}, \mathbf{K}) \\
& = \exp(-0.5(Y - D)^T \mathbf{K}_{YY} (Y - D)) \exp\left(-0.5(X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1}) (X - \boldsymbol{\mu}_X) - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})\right)
\end{aligned}$$

We can now finally integrate over the Y variable:

$$\begin{aligned}
& \int_{\mathbb{R}^d} \mathcal{N}([X, Y]; \boldsymbol{\mu}, \mathbf{K}) dY \\
&= \exp\left(-0.5(X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1})(X - \boldsymbol{\mu}_X) - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})\right) \int_{\mathbb{R}^d} \exp(-0.5(Y - D)^T \mathbf{K}_{YY}(Y - D)) dY \\
&= (2\pi)^{d_Y/2} |\mathbf{K}_{YY}^{-1}|^{-1/2} \exp\left(-0.5(X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1})(X - \boldsymbol{\mu}_X) - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})\right) \\
&= \left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{YY}^{-1}|}{(2\pi)^d |\boldsymbol{\Sigma}|}\right)^{1/2} \exp(-0.5(X - \boldsymbol{\mu}_X)^T (\boldsymbol{\Sigma}_{XX}^{-1})(X - \boldsymbol{\mu}_X))
\end{aligned}$$

We can manipulate the normalising factor as follows:

$$\left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{YY}^{-1}|}{(2\pi)^d |\boldsymbol{\Sigma}|}\right)^{1/2} = \left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{YY}^{-1}| |\mathbf{K}|}{(2\pi)^d}\right)^{1/2} \quad (\text{A.2.7})$$

In order to simplify the above expression further, we can use Shur's determinant identity [49]:

$$\begin{vmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{vmatrix} = |\mathbf{A}| |\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}|.$$

This identity can be derived from the following decomposition [49]:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \end{bmatrix}$$

With the determinant identity above, we can write the determinant of the precision as follows:

$$\mathbf{K} = \begin{vmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{vmatrix} = |\mathbf{K}_{YY}| |\mathbf{K}_{XX} - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX}|$$

With the above expression and the block matrix inversion identities we can rewrite the normalising constant in A.2.7 as:

$$\begin{aligned}
\left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{YY}^{-1}| |\mathbf{K}|}{(2\pi)^d}\right)^{1/2} &= \left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{YY}^{-1}| |\mathbf{K}_{YY}| |\mathbf{K}_{XX} - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX}|}{(2\pi)^d}\right)^{1/2} \\
&= \left(\frac{(2\pi)^{d_Y} |\mathbf{K}_{XX} - \mathbf{K}_{XY} \mathbf{K}_{YY}^{-1} \mathbf{K}_{YX}|}{(2\pi)^d}\right)^{1/2} \\
&= \left(\frac{|\boldsymbol{\Sigma}_{XX}^{-1}|}{(2\pi)^{d-d_Y}}\right)^{1/2} \\
&= \left(\frac{1}{(2\pi)^{d_X} |\boldsymbol{\Sigma}_{XX}|}\right)^{1/2}
\end{aligned}$$

Finally, we can write the result of the marginalisation over Y as:

$$\int_{\mathbb{R}_Y^d} \mathcal{N}([X, Y]; \boldsymbol{\mu}, \mathbf{K}) dY = \left(\frac{1}{(2\pi)^{d_X} |\boldsymbol{\Sigma}_{XX}|} \right)^{1/2} \exp(-0.5(X - \boldsymbol{\mu}_X)^T \boldsymbol{\Sigma}_{XX}^{-1} (X - \boldsymbol{\mu}_X))$$

If the above joint distribution was not normalised the weight of the marginal would therefore be the same as the weight of the joint distribution. The mean and covariance parameters are simply the partitioned matrix and vector of the covariance parameters.

A.2.5 Gaussian Canonical Conditioning Derivation

In this section a derivation the conditioning operation on multivariate Gaussians is given. We start the derivation by distinguishing between the observed variable (Y) and unobserved variable (X) and conditioning the Y variable on the observation y :

$$N([X, Y]; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp(-0.5 \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ Y - \boldsymbol{\mu}_Y \end{bmatrix} - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})), \quad (\text{A.2.8})$$

$$N([X, Y = \mathbf{y}]; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp(-0.5 \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix} - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})). \quad (\text{A.2.9})$$

We can use the partitioned representation of the precision matrix to rewrite the above expression as follows:

$$N(X; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp(-0.5 \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix} - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})).$$

We can now use the simplified expression for the expansion of the quadratic form above (given by equation A.2.5) and condition Y on \mathbf{y} by setting $Y = \mathbf{y}$ to obtain the expression below:

$$\begin{aligned} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} X - \boldsymbol{\mu}_X \\ \mathbf{y} - \boldsymbol{\mu}_Y \end{bmatrix} &= (X - \boldsymbol{\mu}_X)^T \mathbf{K}_{XX} (X - \boldsymbol{\mu}_X) + 2(\mathbf{y} - \boldsymbol{\mu}_Y)^T \mathbf{K}_{XY} (X - \boldsymbol{\mu}_X) \\ &\quad + (\mathbf{y} - \boldsymbol{\mu}_Y)^T \mathbf{K}_{YY} (\mathbf{y} - \boldsymbol{\mu}_Y). \end{aligned}$$

We can now expand the above expression and manipulate it as follows:

$$\begin{aligned}
&= X^T \mathbf{K}_{XX} X - 2X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X + \boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X \\
&\quad + 2X^T \mathbf{K}_{XY} \mathbf{y} - 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} - 2X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y + 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&\quad + \mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} - 2\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y \\
&= X^T \mathbf{K}_{XX} X \\
&\quad - 2X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X + 2X^T \mathbf{K}_{XY} \mathbf{y} - 2X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&\quad + \boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&\quad + \mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} - 2\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y \\
&= X^T \mathbf{K}_{XX} X \\
&\quad - 2X^T (\mathbf{K}_{XX} \boldsymbol{\mu}_X - \mathbf{K}_{XY} \mathbf{y} + \mathbf{K}_{XY} \boldsymbol{\mu}_Y) \\
&\quad + \boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&\quad + \mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} - 2\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y \\
&= X^T \mathbf{K}' X - 2X^T \mathbf{h}' + \gamma \tag{A.2.10}
\end{aligned}$$

where we have defined

$$\begin{aligned}
\mathbf{K}' &= \mathbf{K}_{XX}, \\
\mathbf{h}' &= \mathbf{K}_{XX} \boldsymbol{\mu}_X - \mathbf{K}_{XY} \mathbf{y} + \mathbf{K}_{XY} \boldsymbol{\mu}_Y, \\
\gamma &= \boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + 2\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y + \mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} - 2\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y.
\end{aligned}$$

We can now reincorporate the expression in A.2.10 into the exponent of the conditioned distribution in equation A.2.9. This substitution results in the following expression:

$$N([X, Y = \mathbf{y}]; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp(-0.5(X^T \mathbf{K}' X - 2X^T \mathbf{h}' + \gamma) - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})) \tag{A.2.11}$$

$$= \exp(-0.5X^T \mathbf{K}' X + X^T \mathbf{h}' - 0.5\gamma - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2})) \tag{A.2.12}$$

Note that the above expression is a Gaussian function in canonical form and the \mathbf{K}' and \mathbf{h}' parameters are therefore the new parameters of the conditioned distribution. We can simplify the \mathbf{h}' parameter as follows:

$$\begin{aligned}
\mathbf{h}' &= \mathbf{K}_{XX} \boldsymbol{\mu}_X - \mathbf{K}_{XY} \mathbf{y} + \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&= \mathbf{K}_{XX} \boldsymbol{\mu}_X + \mathbf{K}_{XY} \boldsymbol{\mu}_Y - \mathbf{K}_{XY} \mathbf{y} \\
&= \mathbf{h}_X - \mathbf{K}_{XY} \mathbf{y}
\end{aligned}$$

Here we have used the following identity:

$$\mathbf{K}_{XX} \boldsymbol{\mu}_X = \mathbf{h}_X - \mathbf{K}_{XY} \boldsymbol{\mu}_Y$$

which can be derived from the partitioned representation of the canonical parameters as follows:

$$\begin{aligned}
\mathbf{h} &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \\
&= \mathbf{K} \boldsymbol{\mu} \\
&= \begin{bmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{YX} & \mathbf{K}_{YY} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_Y \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{K}_{XX} \boldsymbol{\mu}_X + \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\ \mathbf{K}_{YX} \boldsymbol{\mu}_X + \mathbf{K}_{YY} \boldsymbol{\mu}_Y \end{bmatrix} = \begin{bmatrix} \mathbf{h}_X \\ \mathbf{h}_Y \end{bmatrix}.
\end{aligned} \tag{A.2.13}$$

From the expression in A.2.12, we can see that the new g parameter is

$$\begin{aligned}
g' &= -0.5\gamma - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}) = -0.5\boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 0.5\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y - \boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y \\
&\quad + \boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} \\
&\quad - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}) + \log(w) \\
&= -0.5\boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 0.5\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y - 0.5\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y - 0.5\boldsymbol{\mu}_Y^T \mathbf{K}_{YX} \boldsymbol{\mu}_X \\
&\quad + \boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} \\
&\quad - \log((2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}) + \log(w)
\end{aligned}$$

Here we can make use of the following equality:

$$\boldsymbol{\mu}^T \mathbf{K} \boldsymbol{\mu} = \boldsymbol{\mu}_X^T \mathbf{K}_{XX} \boldsymbol{\mu}_X - 0.5\boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \boldsymbol{\mu}_Y - 0.5\boldsymbol{\mu}_X^T \mathbf{K}_{XY} \boldsymbol{\mu}_Y - 0.5\boldsymbol{\mu}_Y^T \mathbf{K}_{YX} \boldsymbol{\mu}_X$$

(which can be derived by multiplying out the block matrix representation of $\boldsymbol{\mu}^T \mathbf{K} \boldsymbol{\mu}$) and the definition of g :

$$g = -0.5\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \log(w) - 2\log(|2\pi\boldsymbol{\Sigma}|)$$

to simplify the above expression to:

$$\begin{aligned}
g' &= g + \boldsymbol{\mu}_X^T \mathbf{K}_{XY} \mathbf{y} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY} \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} \\
&= g + (\boldsymbol{\mu}_X^T \mathbf{K}_{XY} + \boldsymbol{\mu}_Y^T \mathbf{K}_{YY}) \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y} \\
&= g + \mathbf{h}_Y^T \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y}.
\end{aligned}$$

Where we have used the \mathbf{h}_Y identity in A.2.13 for the last step above. Finally, we can write the conditioned distribution as follows:

$$\mathcal{N}([X, Y = \mathbf{y}]; \mathbf{K}', \mathbf{h}', g') = \mathcal{N}(X; \mathbf{K}', \mathbf{h}', g')$$

with:

$$\mathbf{K}' = \mathbf{K}_{XX}, \quad \mathbf{h}' = \mathbf{h}_X - \mathbf{K}_{XY} \mathbf{y}, \quad g' = g + \mathbf{h}_Y^T \mathbf{y} - 0.5\mathbf{y}^T \mathbf{K}_{YY} \mathbf{y}.$$

A.3 Cross-Covariance Matrix Derivation

The following is a derivation for the cross-covariance matrix between two sets of samples x_m and y_n , with a total of n_p samples in each set. This derivation was extracted from [30].

$$\begin{aligned}
\Sigma_{XY} &= \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^T] \\
&= \mathbb{E}[XY^T - X\mathbb{E}[Y]^T - \mathbb{E}[X]Y^T + \mathbb{E}[X]\mathbb{E}[Y]^T] \\
&= \mathbb{E}[XY^T - X(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T - (\frac{1}{n_p} \sum_n \mathbf{x}_n)Y^T + (\frac{1}{n_p} \sum_n \mathbf{x}_n)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T] \\
&= \frac{1}{n_p} \sum_m [\mathbf{x}_m \mathbf{y}_m^T - \mathbf{x}_m (\frac{1}{n_p} \sum_n \mathbf{y}_n)^T - (\frac{1}{n_p} \sum_n \mathbf{x}_n) \mathbf{y}_m^T + (\frac{1}{n_p} \sum_n \mathbf{x}_n)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T] \\
&= \frac{1}{n_p} \sum_m \mathbf{x}_m \mathbf{y}_m^T - \frac{1}{n_p} \sum_m \mathbf{x}_m (\frac{1}{n_p} \sum_n \mathbf{y}_n)^T - \frac{1}{n_p} \sum_m (\frac{1}{n_p} \sum_n \mathbf{x}_n) \mathbf{y}_m^T + \frac{1}{n_p} \sum_m (\frac{1}{n_p} \sum_n \mathbf{x}_n)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T \\
&= \frac{1}{n_p} \sum_m \mathbf{x}_m \mathbf{y}_m^T - (\frac{1}{n_p} \sum_m \mathbf{x}_m)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T - (\frac{1}{n_p} \sum_n \mathbf{x}_n)(\frac{1}{n_p} \sum_m \mathbf{y}_m)^T + (\frac{1}{n_p} \sum_n \mathbf{x}_n)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T \\
&= \frac{1}{n_p} \sum_m \mathbf{x}_m \mathbf{y}_m^T - (\frac{1}{n_p} \sum_m \mathbf{x}_m)(\frac{1}{n_p} \sum_n \mathbf{y}_n)^T \\
&= \frac{1}{n_p} \sum_m (\mathbf{x}_m \mathbf{y}_m^T) - \boldsymbol{\mu}_X \boldsymbol{\mu}_Y^T
\end{aligned}$$

A.4 Cartesian-Polar Transformations

The derivations for converting between Cartesian and polar parameters are given in this section. The parameters are defined as follows:

x	: x position,	v_x	: x velocity,
y	: y position,	v_y	: y velocity,
r	: radius,	v_r	: radial velocity,
θ	: angle,	v_θ	: angular velocity.

The derivations for transforming polar position and velocity parameters to the equivalent Cartesian parameters are given below:

$$\begin{aligned}
 x &= r \cos(\theta), & y &= r \sin(\theta), \\
 v_x &= \frac{dx}{dt} = \frac{d}{dt}(r \cos \theta) & v_y &= \frac{dy}{dt} = \frac{d}{dt}(r \sin \theta), \\
 &= \frac{dr}{dt} \cos \theta - r \sin \theta \frac{d\theta}{dt} & &= \frac{dr}{dt} \sin \theta + r \cos \theta \frac{d\theta}{dt} \\
 &= v_r \cos \theta - v_\theta \sin \theta, & &= v_r \sin \theta + v_\theta \cos \theta.
 \end{aligned}$$

The derivations for transforming Cartesian position and velocity parameters to the equivalent polar parameters are given below:

$$\begin{aligned}
 r &= \sqrt{x^2 + y^2}, & \theta &= \arctan\left(\frac{y}{x}\right), \\
 v_r &= \frac{dr}{dt} = \frac{d}{dt} \sqrt{x^2 + y^2} & v_\theta &= \frac{d\theta}{dt} r = r \frac{d\theta}{dt} \arctan \frac{y}{x}, \\
 &= \frac{1}{2\sqrt{x^2 + y^2}} (2xv_x + 2yv_y) & &= \frac{r}{1 + \left(\frac{y}{x}\right)^2} \frac{d}{dt} \left(\frac{y}{x}\right) \\
 &= \frac{xv_x + yv_y}{r}, & &= \frac{rx^2}{x^2 + y^2} \left(\frac{dy}{dt} \frac{x}{x^2} - y \frac{dx}{dt} \frac{1}{x^2} \right) \\
 & & &= \frac{xv_y - yv_x}{r}.
 \end{aligned}$$

A.5 Alternative Model Selection for Multiple Object Tracking - Proof Sketch

In Section 7.1 the claim is made that message passing in a cluster-slice of the MOT PGM (with all X clusters connected to all Y clusters and an appropriate association prior) will always result in the association cluster containing a distribution over the different model hypotheses. A sketch of a proof for this statement is given here.

The statement is true for the following reasons:

1. For each valid model, the total model evidence is the product of compatible $Y_i - X_j$ association probabilities $(\int_{\mathbb{R}^d} P(Y_i = \mathbf{y}_i, X_i) dX_i)$.
2. A message, from the Y_i cluster to the association cluster, contains the probability of each association hypothesis involving Y_i in each corresponding table entry.
3. The association prior only has non-zero probabilities for valid association combinations.

4. After the messages from all Y clusters have been received, the probability of each valid assignment combination, in the association cluster, is the product of the individual assignment probabilities, which, as stated in (1), is the model hypothesis probability.

The above reasoning is perhaps better conveyed through the colour coded message passing representation below.

a_0	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_0, X_1, X_2, a_0) dX_{0,1,2}$
0	$\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_0) dX_0$
1	$\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_1) dX_1$
2	$\int_{\mathbb{R}^d} P(Y_0 = \mathbf{y}_0, X_2) dX_2$

Table A.1: Message $\delta(a_0)$ from Y_0 cluster to association cluster

a_1	$\int_{\mathbb{R}^d} P(\mathbf{y}_1, X_0, X_1, X_2, a_0) dX_{0,1,2}$
0	$\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_0) dX_0$
1	$\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_1) dX_1$
2	$\int_{\mathbb{R}^d} P(Y_1 = \mathbf{y}_1, X_2) dX_2$

Table A.2: Message $\delta(a_1)$ from Y_1 cluster to association cluster

a_0	a_1	$P(a_0, a_1)$	Model
0	1	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_0) dX_0 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_1) dX_1$	(a)
0	2	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_0) dX_0 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_2) dX_2$	(b)
1	0	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_1) dX_1 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_0) dX_0$	(c)
1	2	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_1) dX_1 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_2) dX_2$	(d)
2	0	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_2) dX_2 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_0) dX_0$	(e)
2	1	$\int_{\mathbb{R}^d} P(\mathbf{y}_0, X_2) dX_2 \int_{\mathbb{R}^d} P(\mathbf{y}_1, X_1) dX_1$	(f)

Table A.3: Association cluster $\psi(a_0, a_1)$ posterior marginal showing the model hypotheses as products of compatible association hypothesis probabilities, where the individual association probabilities were contained in the messages.

A.6 Numerical Hessian Calculation

The definition of the Hessian matrix and useful equations for its numerical calculation is given below [50].

$$\mathbb{H}(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (\text{A.6.1})$$

The equations in A.6.2 and A.6.2 can be used to numerically calculate the entries of the Hessian matrix (at a specific point $X = [x_1, \dots, x_n]^T$) of unknown function if the function can be evaluated around the point of interest. In the derivation of these equations (given below) the following definition of a derivative is used:

$$f_{x_i} = \frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_i + h, x_j) - f(x_i, x_j)}{h}.$$

Using the above definition, we can calculate the diagonal terms of the Hessian matrix as follows:

$$\begin{aligned} \frac{\partial^2 f}{\partial x_i^2} &= \frac{\partial f_{x_i}}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f_{x_i}(x_i + h, x_j) - f_{x_i}(x_i, x_j)}{h} \\ &= \lim_{h \rightarrow 0} \frac{(f(x_i + 2h, x_j) - f(x_i, x_j + h)) - (f(x_i + h, x_j) - f(x_i, x_j))}{h^2} \\ &= \lim_{h \rightarrow 0} \frac{f(x_i + 2h, x_j) - f(x_i, x_j + h) - f(x_i + h, x_j) + f(x_i, x_j)}{h^2} \\ &= \lim_{h \rightarrow 0} \frac{f(x_i + 2h, x_j) - 2f(x_i, x_j + h) + f(x_i, x_j)}{h^2} \\ &\approx \frac{f(x_i + 2\Delta, x_j) - 2f(x_i, x_j + \Delta) + f(x_i, x_j)}{\Delta^2}. \end{aligned} \tag{A.6.2}$$

The off-diagonal terms are:

$$\begin{aligned} \frac{\partial^2 f}{\partial x_i \partial x_j} &= \frac{\partial f_{x_i}}{\partial x_j} = \lim_{h \rightarrow 0} \frac{f_{x_i}(x_i, x_j + h) - f_{x_i}(x_i, x_j)}{h} \\ &= \lim_{h \rightarrow 0} \frac{(f(x_i + h, x_j + h) - f(x_i, x_j + h)) - (f(x_i + h, x_j) - f(x_i, x_j))}{h^2} \\ &= \lim_{h \rightarrow 0} \frac{f(x_i + h, x_j + h) - f(x_i, x_j + h) - f(x_i + h, x_j) + f(x_i, x_j)}{h^2} \\ &\approx \frac{f(x_i + \Delta, x_j + h) - f(x_i, x_j + \Delta) - f(x_i + \Delta, x_j) + f(x_i, x_j)}{\Delta^2}, \end{aligned} \tag{A.6.3}$$

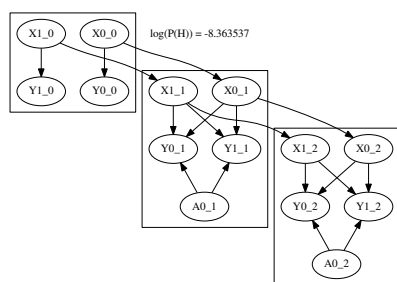
where Δ is a very small positive number.

Appendix B

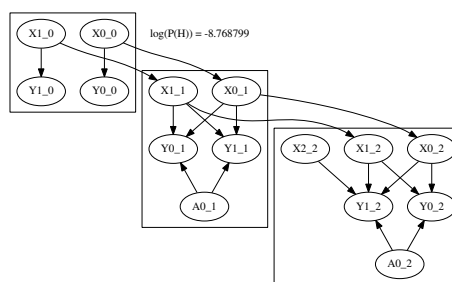
Supplementary Figures

B.1 Model Selection Example Hypothesis

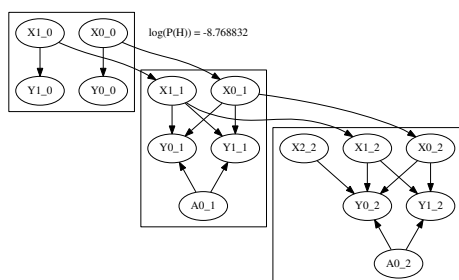
Bellow is the full set of hypotheses for the model selection example in Section 6.3.



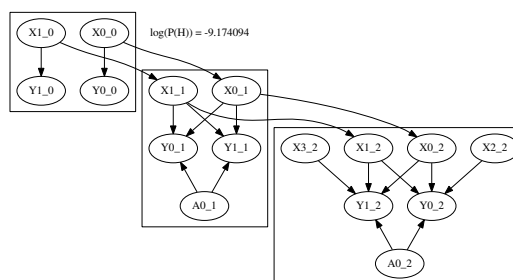
(a) Hypothesis 1



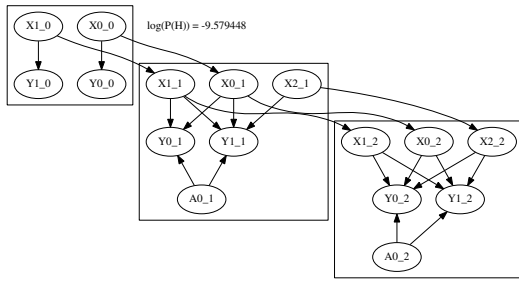
(b) Hypothesis 1



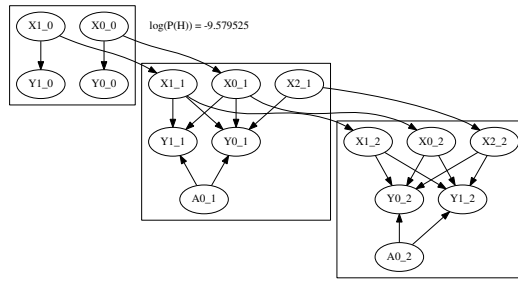
(c) Hypothesis 1



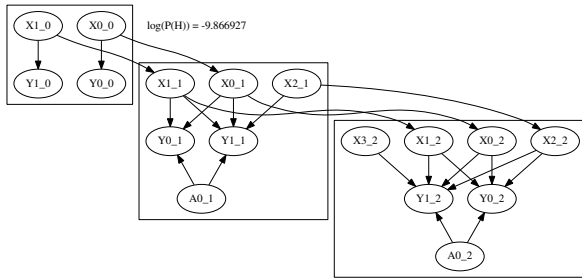
(d) Hypothesis 1



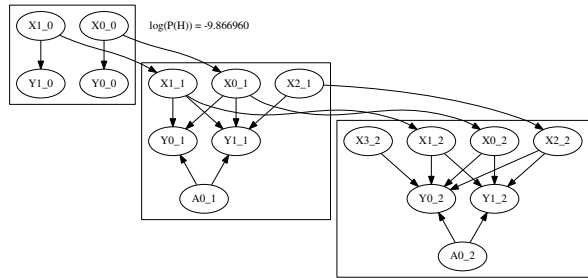
(e) Hypothesis 1



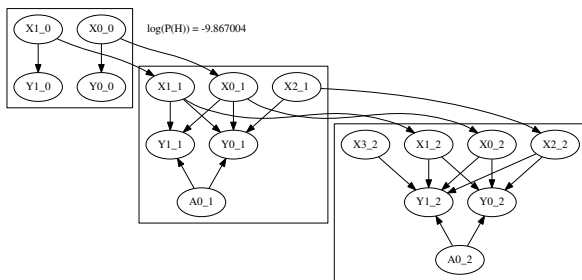
(f) Hypothesis 1



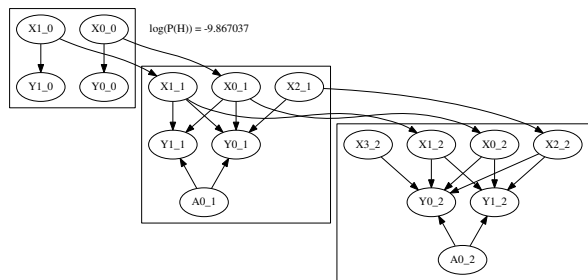
(g) Hypothesis 1



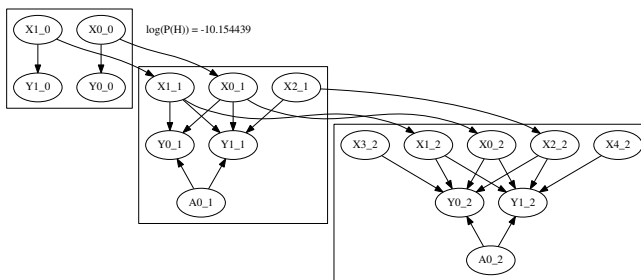
(h) Hypothesis 1



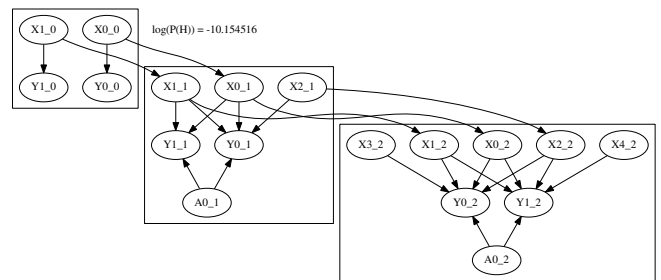
(i) Hypothesis 1



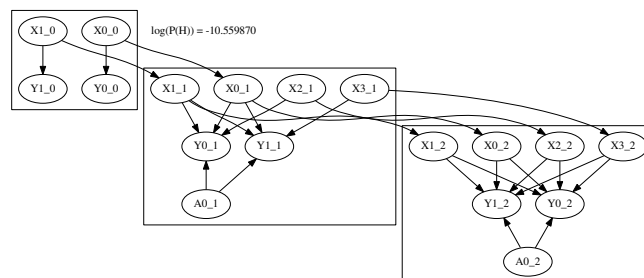
(j) Hypothesis 1



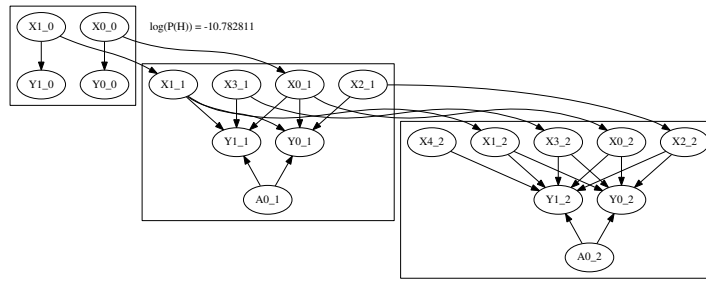
(k) Hypothesis 1



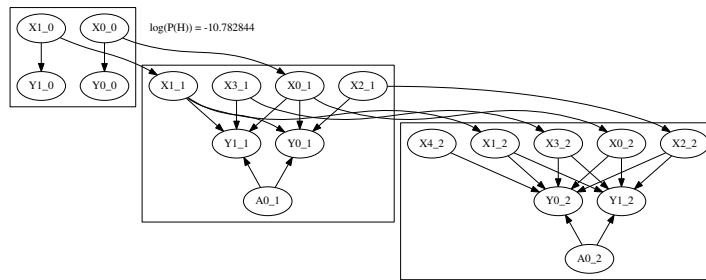
(l) Hypothesis 1



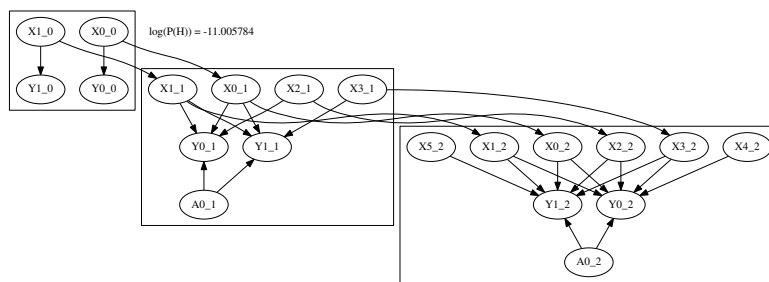
(m) Hypothesis 1



(n) Hypothesis 1



(o) Hypothesis 1



(p) Hypothesis 1

B.1.1 Gaussian Mixture vs. Gaussian Mixture Moment Matching Performance Example

The images in figures B.2a and B.2b are larger versions of the images found in Section 8.3. These figures show an example of the difference in performance when the state distributions are allowed to have Gaussian mixture forms (in the GM-PGM) as opposed to the single Gaussian approximations made by the moment matching PGM (MM-PGM). The detections for these examples are shown in Figure B.1 below to give an idea of the difficulty of the tracking task in this example.

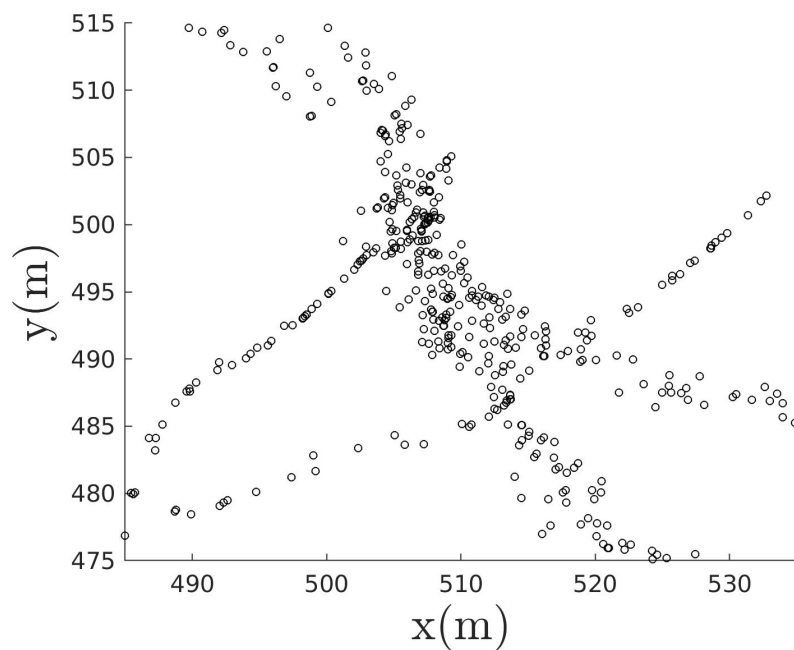
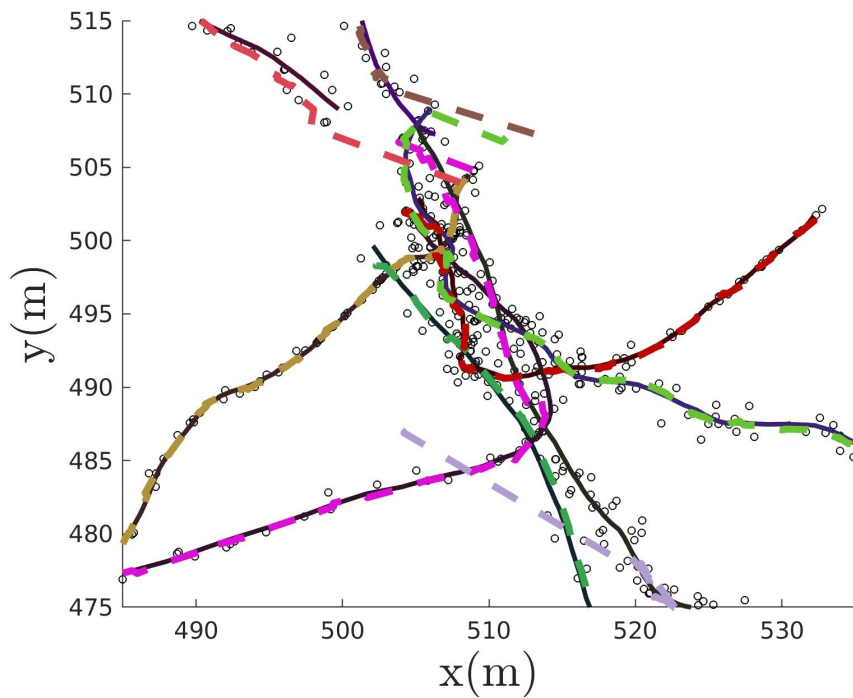
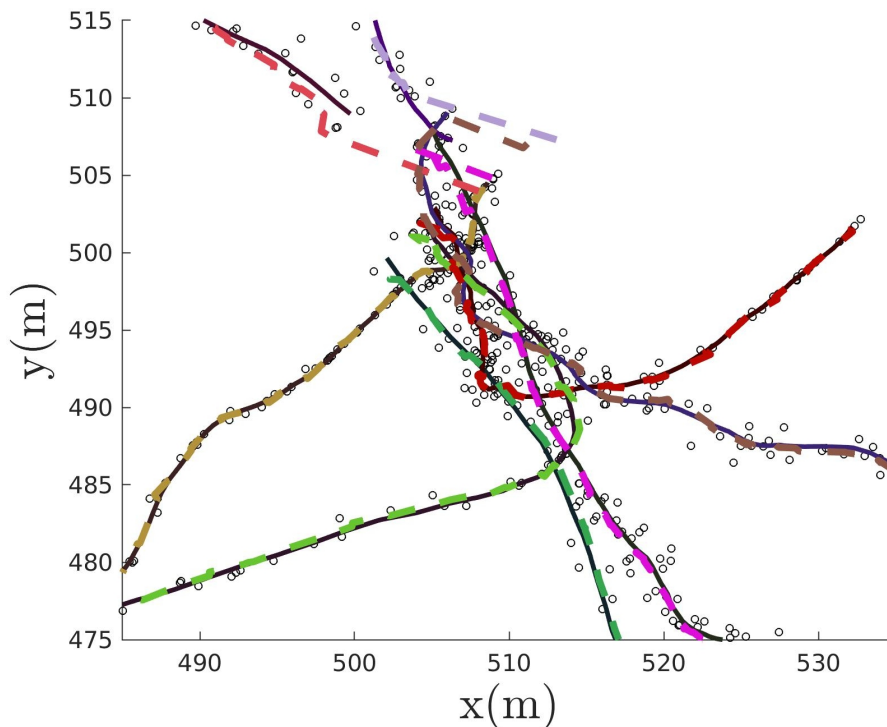


Figure B.1: GM-PGM vs MM-PGM performance comparison example showing only the detections.



(a) Multiple object tracking with moment matching Gaussian approximations of state distributions (MM-PGM model)



(b) Multiple object tracking with Gaussian mixture state distributions and Gaussian mixture quotient approximation (GM-PGM model).

Figure B.2: A comparison of tracking accuracy between the MM-PGM and GM-PGM models. All the objects in this example start in the center of the figures and move outward. Note that the track identities is not correct (consistent) in (a), whereas they are in (b). This shows the advantage that the GM-PGM has over the MM-PGM. The scanning radar simulator (without angular velocity measurements) was used in this example. The true tracks in these examples are indicated by solid dark lines, the track estimates by the colour dashed lines and the detections by the black circles.

Appendix C

Causal Clutter Classification Model

In this section an alternative and causal model is given to the model in 6.5. The two time-slice Bayes network for this model is shown below.

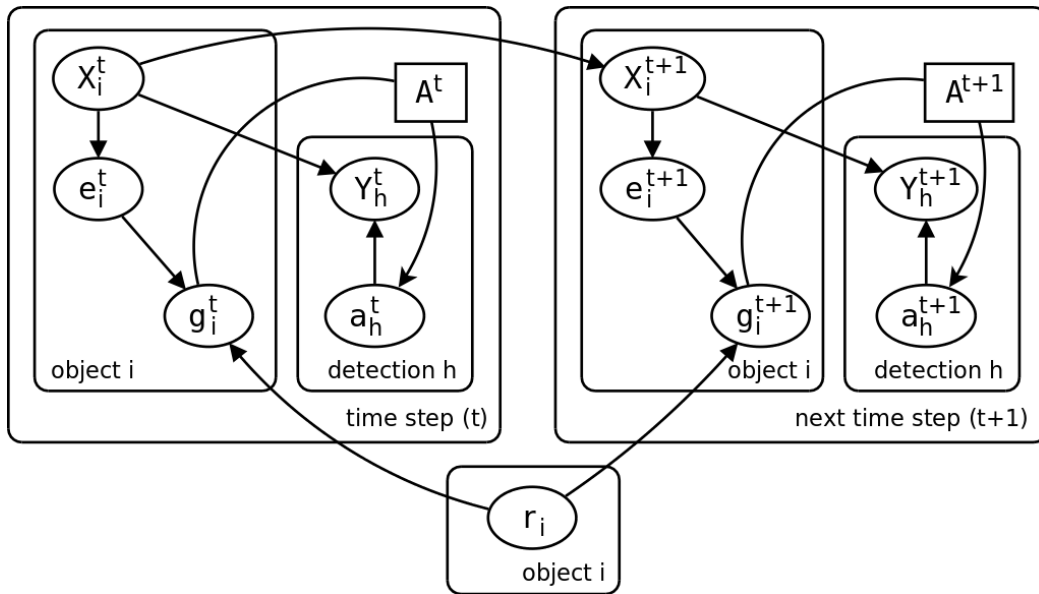


Figure C.1: Two time-slice Bayes network of MOT PGM with clutter classification. The e_i^t and g_i^t random variables represents whether or not a detection is expected and has been generated from a target at a specific time step respectively. The r_i random variable corresponds to whether the target is real or not.

An example of a sensible conditional probability distribution for the $P(g_i^t | e_i^t, r_i^t)$ factor is

given below. The probabilities in this example could be changed if necessary for a specific application.

e_i^t	r_i	g_i^t	$P(g_i^t r_i, e_i^t)$
0	0	0	0.9
0	0	1	0.1
0	1	0	0.9
0	1	1	0.1
1	0	0	0.9
1	0	1	0.1
1	1	0	0.1
1	1	1	0.9

The above table encodes the logic that detections are probable from real targets when they are expected (when a sensor is capable of receiving detections from a target) and that detections from false targets are unlikely.

Note that there is an undirected edge between the g_i^t and A^t nodes in Figure C.1. This encodes the fact that there is no causal link between the associations and whether a detection was generated. Rather, this is a definitional or relational link and the corresponding logic that will be enforced by this link is given below.

$$\phi(g_i^t = 1, a_0^t = \alpha_0^t, \dots, a_{m-1}^t = \alpha_{m-1}^t) = \begin{cases} 1 & i \in \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \\ 0 & i \notin \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \end{cases},$$

$$\phi(g_i^t = 0, a_0^t = \alpha_0^t, \dots, a_{m-1}^t = \alpha_{m-1}^t) = \begin{cases} 1 & i \notin \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \\ 0 & i \in \{\alpha_0^t, \dots, \alpha_{m-1}^t\} \end{cases}.$$

Note that A^t is the vector random variable containing all the a^t random variables ($A^t = [a_0^t, \dots, a_m^t]$) and that the α^t values above correspond to specific values of the a^t random variables. The above logic can be used to create an edge potential corresponding to the edge between the g^t and A^t variables. An example of such an edge potential is given below.

a_0^t	a_1^t	g_0^t	$\phi(g_0^t, a_0^t, a_1^t)$
0	0	0	0.0
0	0	1	1.0
0	1	0	0.0
0	1	1	1.0
0	2	0	0.0
0	2	1	1.0
1	0	0	0.0
1	0	1	1.0
1	1	0	1.0
1	1	1	0.0
1	2	0	1.0
1	2	1	0.0
2	0	0	0.0
2	0	1	1.0
2	1	0	1.0
2	1	1	0.0
2	2	0	1.0
2	2	1	0.0

An example of the cluster graph corresponding to the two time-slice Bayes network in Figure C.1 is shown in Figure C.2. Here there are three targets being tracked (as indicated by the three state clusters) and two measurements have been received, each with their own measurement cluster.

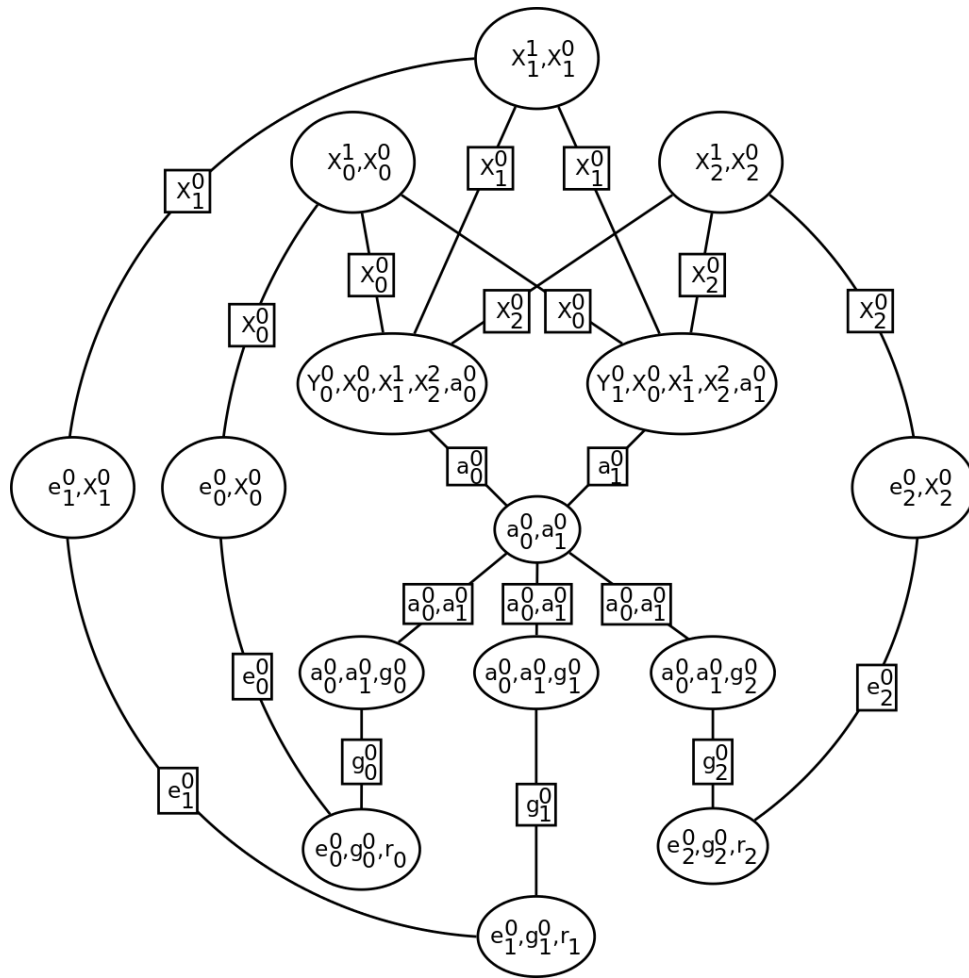


Figure C.2: Example of a causal clutter classification cluster graph (attached to the multiple object tracking graph), corresponding to a single time slice graph of the Bayes network in Figure C.1 for three targets and two measurements.

Note that the above graph structure is exactly the same as that of the non-causal graph in Figure 6.22. In this graph, however, there is only a single r_i random variable per object. In the larger, time unrolled graph, the real target clusters (clusters with $[e_i^t, g_i^t, r_i]$ scope) will simply be connected to each other through sepsets with scope r_i . This is in contrast to the non-causal model where all these clusters are connected to a single time independent r_i cluster.