UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Robust object tracking using neural-network-based instance segmentation

DM DE FREITAS
18345662

Report submitted in partial fulfilment of the requirements of the module Mechatronic Project 488 for the degree Baccalaureus in Engineering in the Department of Mechanical and Mechatronic Engineering at Stellenbosch University.

Supervisor: Dr C. Van Daalen

October 2020

Executive summary

Title of Project
Robust object tracking using neural network-based instance segmentation
Objectives
Perform instance segmentation on an image to generate measurements to be used in tracking. Incorporate measurement noise and an object's dynamics uncertainty with a principled approach to predict the location of an object. When receiving multiple measurements and tracking multiple objects, solve the data association problem.
What is current practice and what are its limitations?
Current practice in segmentation uses end-to-end neural networks which can be hard to develop and require large amounts of time for training. Other point trackers generally use measurements from radar or Lidar which can be more expensive.
What is new in this project?
The use of probabilistic graphical models applied to an instance segmentation algorithm for the purpose of incorporating uncertainty when tracking objects.
If the project is successful, how will it make a difference?
It will provide a proof of concept for a general tracking system where detection via neural networks is separated from the tracking via probabilistic methods. This allows for detection to be developed and adapted without needing to recreate the tracking algorithm and vice-versa. It also allows for a general tracking algorithm where the detector can be change to suit an application.
What are the risks to the project being a success? Why is it expected to be successful?
The major risk is the inability to complete the project on time due to complexity. This risk is mitigated due to rigorous planning to ensure project is delivered on time.
What contributions have/will other students made/make?
N/A
Which aspects of the project will carry on after completion and why?
The algorithm can be further improved to incorporate additional forms of measurement inaccuracies such as occlusion noise. The feasibility of real time tracking should be determined. Tracking in real time greatly improves the usefulness of tracking
What arrangements have been/will be made to expedite continuation?
N/A

Outcome compliance

ELO	ELO Description	Chapters which assess
ELO 1. Problem solving Demonstrate competence to identify, assess, formulate and solve convergent and divergent engineering problems creatively and innovatively	1. Analyses and defines the problem, identifies the criteria for an acceptable solution; 2. Identifies necessary information and applicable engineering and other knowledge and skills; 3. Generates and formulates possible approaches to solution of problem; 4. Models and analyses possible solution(s); 5. Evaluates possible solutions and selects best solution; 6. Formulates and presents the solution in an appropriate form	6,7
ELO 2. Application of scientific and engineering knowledge: Demonstrate competence to apply knowledge of mathematics, basic science and engineering sciences from first principles to solve engineering problems.	1. Brings mathematical and numerical analysis to bear on engineering problems by using an appropriate mix of: *a) Formal analysis and modelling of engineering components, systems or processes;* *b) Communicating concepts, ideas and theories with the aid of mathematics;* *c) Reasoning about and conceptualising engineering components, systems or processes using mathematical concepts.* 2. Uses physical laws and knowledge of the physical world as a foundation for the engineering sciences and the solution of engineering problems by an appropriate mix of: *a) Formal analysis and modelling of engineering components, systems or processes using principles and knowledge of the basic sciences;* *b) Reasoning about and conceptualising engineering problems, components, systems or processes using principles of the basic sciences.*	2,3,4,5,6,7

	3. Uses the techniques, principles and laws of engineering science at a fundamental level and in at least one specialist area to: *a) Identify and solve open-ended engineering problems;* *b) Identify and pursue engineering applications;* *c) Work across engineering disciplinary boundaries through cross disciplinary literacy and shared fundamental knowledge.*	
ELO 3. Engineering Design: Demonstrate competence to perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.	The candidate designs components, systems, engineering works, products or processes as part of the project. The design process and its outcome is documented in the report. The candidate executes an acceptable design process encompassing the following: *1. Plans and manages the design process: focuses on important issues, recognises and deals with constraints;* *2. Acquires and evaluates the requisite knowledge, information and resources: applies correct principles, evaluates and uses design tools;* *3. Performs design tasks including analysis, quantitative modelling and optimisation;* *4. Evaluates alternatives and preferred solution: exercises judgment, tests implementability and performs techno-economic analyses;* *5. Communicates the design logic and information.*	6,7
ELO 5. Engineering methods, skills and tools, including Information Technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	Sufficient demonstration of the critical use of applicable engineering methods, skills and tools at the level of 3rd or 4th year BEng is required. The candidate: 1. Uses method, skill or tool effectively by: *a) Selecting and assessing the applicability and limitations of the method, skill or tool;* *b) Properly applying the method, skill or tool;* *c) Critically testing and assessing the end-results produced by the method, skill or tool.* 2. Creates computer applications as required by the discipline.	2,6,7
ELO 6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	The candidate demonstrated: The communication was clear and understandable; Oral presentations, poster and final report are professionally acceptable; Language usage is as required for technical communication. The candidate executes effective written communication as evidenced by: 1. Uses appropriate structure, style and language for purpose and audience; 2. Uses effective graphical support; 3. Applies methods of providing information for use by others involved in engineering activity; 4. Meets the requirements of the target audience. The candidate executes effective oral communication as evidenced by: 1. Uses appropriate structure, style and language; 2. Uses appropriate visual materials; 3. Delivers fluently; 4. Meets the requirements of the intended audience.	All

Declaration

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 woordelikse 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.

Studentenommer / *Student number*	Handtekening / *Signature*
Voorletters en van / *Initials and surname*	Datum / *Date*

Contents

List of Figures

List of Tables

Nomenclature

General Notation

s	Lowercase italics represent general scalar variable.
X	Uppercase italics represent a random variable.
$\boldsymbol{X}$	Bold uppercase italics represent a random vector.
$\mathbf{v}$	Bold unitalicised lowercase notation represents a vector.
$\mathbf{M}$	Bold unitalicised uppercase notation represents a matrix.

Variables and functions

X	Object state random variable.
$\boldsymbol{X}$	Object state random vector
Y	Measurement random variable.
$\boldsymbol{Y}$	Measurement random vector
$\mathbb{Y}^t$	A set of measurements up until time t, $\mathbb{Y} = [\boldsymbol{Y}^0 = \mathbf{y}^0, \boldsymbol{Y}^1 = \mathbf{y}^1, \ldots, \boldsymbol{Y}^t = \mathbf{y}^t]$
$\mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian PDF in covariant form.
$\boldsymbol{\mu}$	Statistical mean vector.
$\boldsymbol{\Sigma}$	Statistical covariance matrix.
$\mathcal{C}(\boldsymbol{X}; \mathbf{K}, \mathbf{h}, g)$	Multivariate Gaussian PDF in canonical/information form.
$\mathbf{K}$	Information matrix
$\mathbf{h}$	Information vector
g	Information normalization constant
$\mathcal{W}_k(0, \mathbf{Q})$	Zero mean Gaussian with motion noise covariance $\mathbf{Q}$
$\mathbf{Q}$	Motion noise covariance matrix
$\mathcal{V}_k(0, \mathbf{R})$	Zero mean Gaussian with measurement noise covariance $\mathbf{R}$
$\mathbf{R}$	Measurement noise covariance matrix
$\mathbf{K}_g$	Kalman gain
$D(z)$	Normalized distance squared function with respect to measurement z.
z_k^*	The Nearest Neighbour measurement
$p(x)$	PDF with respect to variable x.
$P(A)$	Probability of event A occurring.
$P(fd)$	Probability of a false detection
$\mathbf{i}$	A set of objects
$\mathbf{j}$	A set of measurements
$\mathbf{A}$	An association matrix
$\phi(Y, X)$	A factor of Y and X
$\psi_1(Y, X)$	A cluster of Y and X
$\delta_{1\to2}(Y)$	A message sent from cluster 1 to cluster 2 with information regarding Y

Acronyms and abbreviations

DAG	Directed acyclic graph
fps	Frames per second
MHT	Multiple hypothesis tracker
NDS	Normalized distance squared
NN	Nearest Neighbor
GNN	Global Nearest Neighbor
JPDA	Joint Probabilistic Data Association
PDF	probability density function
PGM	Probabilistic graphical model
R-CNN	Region-based convolutional neural network

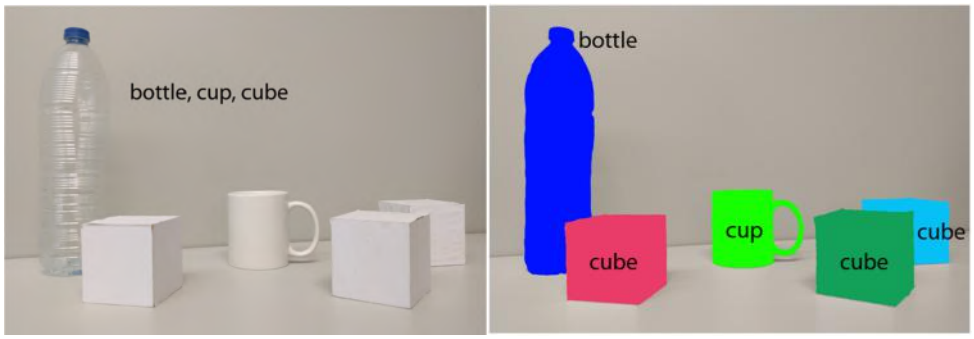
Chapter 1

Introduction

1.1. Background

Sight provides humanity with a tool to reason about the world, perform tasks and interact with the environment. Similarly, mechatronic systems can benefit from computer vision. Computer vision aims to describe the world and gain an understanding of the environment based on an image or sequence of images. One task is that of object tracking which has been defined by Yilmaz et al. [2, p. 2] as estimating the trajectory of an object in an image plane as it moves around a scene. This project aims to perform multiple object tracking which, requires each object in a sequence of images to be assigned a consistent label and its path determined.

The input into the tracking application is a sequence of images that have had instance segmentation performed. An image taken from a review on segmentation by Garcia-Garcia et al. [3], shown in Figure 1.1, shows that instance segmentation is the task of generating a pixel-wise mask around each individual object. Recent advances in neural-networks have resulted in algorithms such as Mask R-CNN being able to perform this task [4]. However, errors such as missed detections, false detections and mask inaccuracies lead to measurement noise. The dynamics of each object is unknown and unpredictable, which



(a) Original image **(b)** Instance segmentation

Figure 1.1: (a) Original image used where objects are not assigned labels and their position are unknown (b) Instance segmentation providing a mask and label for each object.

creates additional uncertainty. The result of our single object tracker from Chapter 6

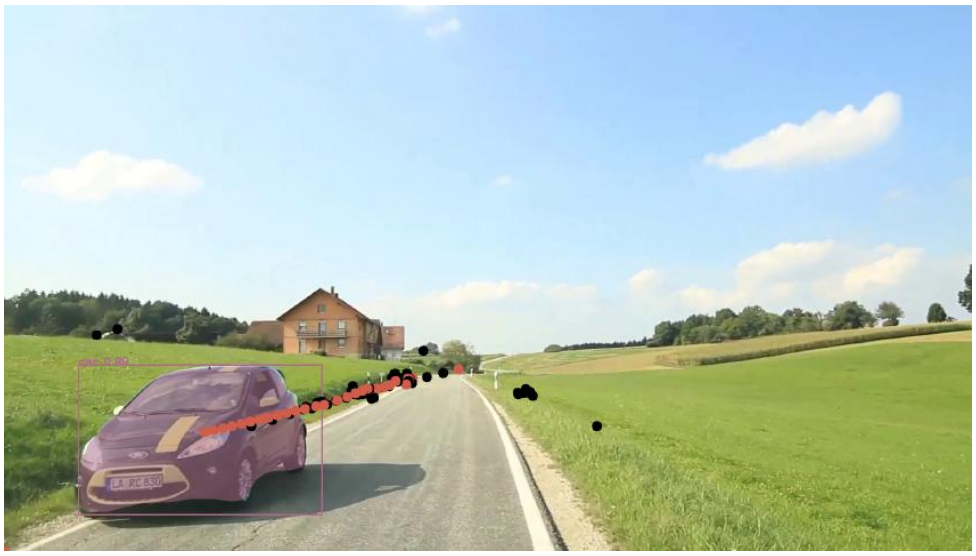


Figure 1.2: Tracking a single object: The output from the detector is the mask shown on the car. The centroids of all previous masks are plotted in black, while the predicted object location at each time step is given in orange.

shows how one must take into account all the noisy measurements, shown by black dots, to find the belief of the object state, shown by the coloured dots. When there are multiple measurements and objects, an association of this data needs to be performed for multiple object tracking. A probabilistic approach, where sources of uncertainty are incorporated in a principled manner must be used to improve the reliability of tracking.

1.2. Problem statement

The aim of this project is to demonstrate how robust object tracking can be achieved using neural-network based instance segmentation. Tracking involves updating an objects state when it is measured. However, measurements will be noisy and an objects motion is unknown, so predicting the path that the object has taken will need to incorporate uncertainty. Additionally, it is unknown which measurement was caused by which object and so this needs determine.

We limit this work to using a pre-trained neural network, as training requires extensive resources. The tracking application should be capable of tracking multiple objects but does not need to perform in real-time.

1.3. Objectives

In achieving the overall goal of object tracking we seek to complete three primary objectives. The first is to select a pre-trained neural network, and perform instance segmentation on an image. The centroid of the mask of each object from instance segmentation, can be

seen as a measurement similarly to LiDAR or Radar. The first objective is to generate these measurements to be used for tracking objects. Secondly, we must take into account the noise in these measurements and uncertain object dynamics, to track the object's state. This needs to be done using a robust probabilistic framework for dealing with uncertainty such as probabilistic graphical models (PGMs). The last objective is to solve the data association problem where a measurement must be associated with the correct object when tracking multiple objects.

1.4. Motivation

Object tracking is an active research area with applications such as pedestrian and vehicle tracking as well as robot vision and video indexing [5]. Surveillance and autonomous navigation are industries that can benefit greatly. Currently there are many different object tracking techniques, each with their own advantages and disadvantages. End-to-end neural network trackers can become complex and difficult to reason about, update and change. Simpler models make many assumptions, such as smooth motion, and cannot work practically in many environments.

Investigating the use of a tracker based off of a probabilistic approach, where a model can include uncertainties, and inference can be performed is beneficial. The application of PGMs to object tracking is not mainstream and being able to separate neural-network-based detection from a probabilistic approach to tracking has the following benefits; If the detection is retrained, or replaced, the tracking model can be changed without needing to retrain an entire end-to-end neural network. Uncertainties are also taken into account in a principled manner so it is also easier to investigate a PGM and understand the relationship between variables, than with an end-to-end neaural network.

1.5. Report layout

Chapter 2 describes the measurement system used and the selection of a neural-network. A literature review on object tracking is provided in Chapter 3. Probabilistic Graphical Models and the relevant graphs and inference strategies are explained in Chapter 4. Random variables and Gaussian Distributions are used extensively throughout this work and Chapter 5 provides the reader with the required background theory. Chapter 6 uses the Kalman filter as a PGM to track single objects. This is extended in Chapter 7 to multiple objects. A summary and conclusion of this work is then presented in the final chapter.

Chapter 2

Measurement system

An object can be detected via various means, for example Radar or LiDAR. However, an increase in computation power, attainability of high-quality low-cost cameras, and a need for automated analysis of a large database of videos has resulted in swift progression within visual object tracking [6]. This work uses images from a camera as its sensor and is therefore categorized as visual object tracking. In Section 2.1 the process of taking a measurement is discussed, the required neural network is selected in the next section and then discussed in the last section.

2.1. Instance segmentation via pre-trained neural networks

Instance segmentation can be seen as a form of signal processing on an image which identifies the location of an object via a mask. The centroid of this mask can then be used in object tracking as one would use the blip of a radar detection. A pre-trained neural network needs to be implemented as resources for training are not available.

2.2. Neural network selection

The neural network selected needs to satisfy two conditions. Firstly, it must perform instance segmentation succsesfully, and secondly it must be able to be implemented with ease on a standard office laptop. Additionally, the tracking application will be developed in python and so a neural network that is compatible and simple to implement should be selected.

Mask R-CNN [4] is often used a baseline when comparing instance segmentation methods, such as in the survey on instance segmentation performed by Hafiz et al. [7]. Mask R-CNN achieved first position in the 2016 COCO challenge[1] for instance segmentation [7]. There are also many tutorials, such as one from Kumar [8], explaining how one can perform instance segmentation using python and a pre-trained weights.

[1]COCO is a large-scale object detection, segmentation, and captioning dataset. Every year there are tasks to perform on these datasets called challenges.

Figure 2.1: Three frames from a video with instance segmentation performed using Mask R-CNN pre-trained with the Coco dataset. The first image shows a false detection, the second shows a missed detection, and the third shows a mask which does not accurately cover the car.

YOLACT [9] can perform instance segmentation at real time, with a marginal loss in precision when compared to Mask R-CNN [7]. This neural network may be an option for implementation if this project was adapted to perform real time tracking. A tutorial on how to implement a pre-trained YOLACT detector in python was not easily found.

Hybrid task cascade [10] was released in 2019 and is a more recent neural network capable of outperforming Mask R-CNN [7]. However, its ease of implementation was lacking. This Neural network may be an option for implementation if accuracy is the primary concern in a practical application.

Ultimately, it was decided that Mask R-CNN would be used on the basis of its ease of implementation shown in the tutorial by Kumar [8]. The weights used for this detector have been pre-trained on the COCO dataset [11] which contains 328000 general images with 80 object categories annotated.

2.3. Conclusion

Instance segmentation has been performed on images and the results shown in Figure 2.1. Here we can see the errors that exist, false detections, general mask inaccuracy and missed detection. The ability to use different detectors as an input into this tracking application allows for greater application potential. In this case we select Mask R-CNN for its ease of use, and obtain pre-trained weights on the Coco dataset as we lack the resources required for training. However, if one trained the neural network, it would be capable of detecting objects at an astronomical level [12] and single cell level [13]. This lead to an increase in the applicability of tracking via detection.

Chapter 3

Object tracking reviewed

Chen [14], highlights the main two tasks of multiple object tracking: to solve the data association problem where each measurement must be associated with the correct object, and to filter noisy measurements and predict the true state of an object. These two problems are discussed further in this Chapter. The Kalman filter, which takes into account uncertainty when tracking an object, is then discussed in Section 3.1. Methods for solving the data association problem using the nearest neighbours (NN) approach or by multiple hypothesis tracking (MHT) are presented in Sections 3.2 and 3.3.

Tracking with uncertainty

The measurements used usually contain noise and do not necessarily represent the true state of an object. The dynamics of the object may be unknown and it is common to perform simplifications such as a linear assumption when modelling object behaviour. This results in uncertainty from two sources, a measurement model and a state transition model. The Kalman filter [15] discussed in the next section contains these uncertainties and an estimation of the true state is provided. It tracks using a linear assumption but can be extended for non-linear motion using an extended Kalman filter (EKF). Alternatively, particle filters are described by Thrun in Chapter 4 of Probabilistic robotics [16], and use mixtures of Gaussian's in the measurement model so a tracker can be implemented.

Data association problem

For multiple objects, one needs to determine which measurements belong to which objects. In static data association, measurements are assigned to an object at each frame using certain constraints. A well known approach is the nearest neighbours (NN) algorithm discussed in Section 3.2. This requires a hard decision at each time-step and is likely to deteriorate in performance when an incorrect decision is made. Methods such as multiple hypothesis tracking (MHT) attempt to solve this using deferred logic and is discussed in Section 3.3.

3.1. Kalman filter

Kalman published a paper in 1960 on the topic of linear filtering and prediction [15]. Since then the tracking community has used the Kalman filter extensively [17]. A motion model or state model is defined as:

$$\boldsymbol{X}^k = \mathbf{A}\boldsymbol{X}^{k-1} + \mathbf{B}\boldsymbol{u}^{k-1} + \boldsymbol{W}^k \tag{3.1}$$

where $\boldsymbol{X}^k$ is the object state vector at time $t = k$, $\mathbf{A}$ is the transition matrix and $\boldsymbol{W}^k$ is Gaussian noise for the motion model. For the rest of this section control aspects, $\mathbf{B}\boldsymbol{u}^{k-1}$, are ignored as there are no control inputs. A measurement or observation model is given by:

$$\boldsymbol{Y}^k = \mathbf{H}\boldsymbol{X}^k + \boldsymbol{V}^k \tag{3.2}$$

where $\boldsymbol{Y}^k$ is the observation vector, $\mathbf{H}$ is the observation matrix and $\boldsymbol{V}^k$ is Gaussian measurement noise. The noise $\boldsymbol{W}^k$ and $\boldsymbol{V}^k$ are zero mean Gaussians with covariances $\mathbf{Q}$ and $\mathbf{R}$. If the reader is unfamiliar with Gaussian distributions a discussion is provided in Chapter 5: Gaussian Distributions. There are two steps to Kalman filtering with the first being a prediction step.

Prediction step

The predicted location is described by Kalman as,

$$\tilde{\boldsymbol{X}}^k = \mathbf{A}\hat{\boldsymbol{X}}^{k-1}, \tag{3.3}$$

and,

$$\tilde{\boldsymbol{P}}^k = \mathbf{A}\hat{\boldsymbol{P}}^{k-1}\mathbf{A}^T + \mathbf{Q} \tag{3.4}$$

Here, the tilde $\tilde{\boldsymbol{X}}^k$ represents the priori estimated mean. It is calculated using a linear transform of the previous posterior mean $\hat{\boldsymbol{X}}^{k-1}$ and the transition matrix $\mathbf{A}$. The priori covariance $\tilde{\boldsymbol{P}}^k$ is similarly updated and its covariance changes due to the affect of $\mathbf{Q}$, system motion noise. After the prediction step, a measurement update step is performed.

Measurement update

Kalman describes the measurement update as,

$$\hat{\boldsymbol{X}}^k = \tilde{\boldsymbol{X}}^k + \mathbf{K}(\mathbf{y}^k - \mathbf{H}\tilde{\boldsymbol{X}}^k) \tag{3.5}$$

$$\hat{\boldsymbol{P}}^k = \tilde{\boldsymbol{P}}^k - \mathbf{K}\mathbf{H}\tilde{\boldsymbol{P}}^k \tag{3.6}$$

where the Kalman gain $\mathbf{K}$ is:

$$\mathbf{K}_g = \tilde{\mathbf{P}}_k \mathbf{H}^T (\mathbf{H}\tilde{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R})^{-1} \tag{3.7}$$

The posterior estimated mean $\hat{\boldsymbol{X}}^k$ is therefore a combination of both the measured state $\mathbf{y}^k$ and the predicted state $\tilde{\boldsymbol{X}}^k$. Similarily, the posterior estimated covariance $\hat{\boldsymbol{P}}^k$ is a combination of the predicted covariance $\tilde{\boldsymbol{P}}^k$ and measurement noise $\mathbf{R}$, which is captured in the Kalman gain $\mathbf{K}$. The posterior estimated mean and covariance $\hat{\boldsymbol{X}}^k$ and $\hat{\boldsymbol{P}}^k$ is then used in the prediction step for the next time sequence.

The result is an algorithm that takes into account both measurement and system noise and recursively updates the belief about the state.

3.2. Nearest neighbour (NN)

One of the simplest and most widely used methods for tracking multipled objects in clutter is the nearest neighbours (NN) filter [18]. When there is uncertainty about whether a measurement was caused by an object, the measurement that is closest to the predicted object state is called the nearest neighbour measurement $\mathbf{z^k}*$ and assumed to be the correct measurement update. For the validated measurements $\mathbf{Z}^k$ the NN measurement is defined as:

$$\mathbf{z^k}* = \arg\min_{z \in \mathbf{z}_k} D(z) \tag{3.8}$$

where the $D((Z))$ is the normalized distance squared (NDS) defined as:

$$D(\mathbf{Z}) = [\mathbf{z} - \tilde{\boldsymbol{X}}^k]' \tilde{\boldsymbol{P}}^k [\mathbf{z} - \tilde{\boldsymbol{X}}^k] \tag{3.9}$$

The predicted measurement means $\tilde{\boldsymbol{X}}^k$ and covariance matrix $\tilde{\boldsymbol{P}}^k$ are calculated via Kalman filtering, that is equations 3.3 and 3.4.

Gating can also be performed whereby measurements outside of a particular region are not validated. I.e. we do not calcualted a NDS for them as we assume they are not associate with a particular object. This is usually done via an ellipsoid at the predicted measurement location as shown in Figure 3.1 From all the measurements inside the validation gate the NN is found. Often additional constraints are placed such as in the global nearest neighbours (GNN) filter where a measurement can be assigned to at most one object [19].

In joint probabilistic data association (JPDA) a weighted average of the validated measurements are used [20]. In Figure 3.1, P1's nearest neighbour is O1 and P2's nearest neighbour is either O2 or O3. If O2 was closest to both P1 and P2 they would share measurements in NN whereas in GNN they could not. For JPDA a weighted average of O1, O2 and O3 for the first track and O2 and O3 for the second track would be used. In a cluttered environment this can lead to converging tracks.

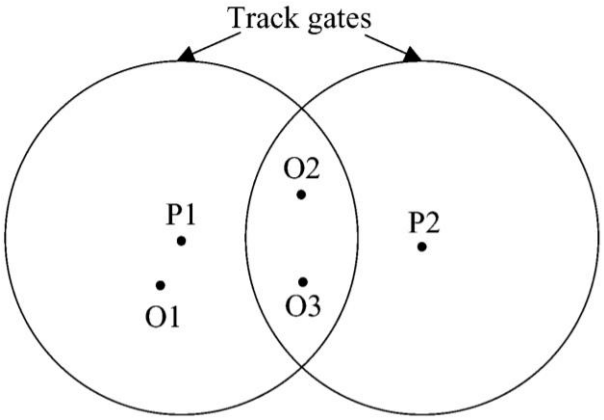


Figure 3.1: Track gates: O1, O2, O3 = Observation positions; P1, P2 = Predicted target positions. [19].

If there are no validated measurements, no measurement update takes place and only the prediction step is performed. This results in a loss of accuracy. If incorrect measurements are used to update an object's location, the belief of an objects location will be incorrect. Ultimately, due to the single scan nature of NN, GNN and JPDA, incorrect assignments will result in performance deteriorating over time.

3.3. Multiple Hypothesis Tracking (MHT)

Blackman states that "multiple hypothesis tracking (MHT) is generally accepted as the preferred method for solving the data association problem in modern multiple target tracking systems" [19]. The MHT algorithm was developed by Reid in 1979 [21] and provides a systematic approach that uses deferred logic to solve data association. Unlike GNN or JPDA, MHT forms multiple hypothesis at each step that allows future observations to aid in correct decision making.

Hypothesis generation

In the example from Figure 3.1 an objects location estimated location P1 and P2 will be based on tracks T1 and T2. A hypothesis could be that all the new observations O1, O2, and O3 are new object targets. Thus O1, O2 and O3 form new tracks denoted NT1, NT2 and NT3. This hypothesis, labelled H1 is presented below:

$$\text{H1: T1, T2, NT1, NT2, NT3} \tag{3.10}$$

Another hypothesis, H2, may be that T1 is associated with O1 and a new track T3(T1,O1) is formed. T2 is associated with 02 to form T4(T2,O2) and O3 starts a new track $NT3$.

$$\text{H2: T3, T4, NT3} \tag{3.11}$$

$$\text{H3: T3, T5, NT2} \tag{3.12}$$

$$\vdots$$

Hypotheses are generated for every possible association. These hypotheses are then expanded into more hypotheses when new measurements are made available. A log likelihood ratio is used as a track score and incorporates many aspects such as: the probability of false alarms, prior knowledge about object location and dynamic consistency. While this takes into account all the possible associations, the problem becomes computationally infeasible due to the large number of hypotheses.

Hypothesis reduction

To reduce the number of hypotheses, various techniques are used. When tracks have common observations, they can be clustered together to create a set of smaller problems. This clustering allows parallel processing where within each cluster, hypotheses are deleted if they have low track scores. N-scan pruning joins tracks together with a common root node. For example, in Figure 3.2, each node represents a target state. When a new possible

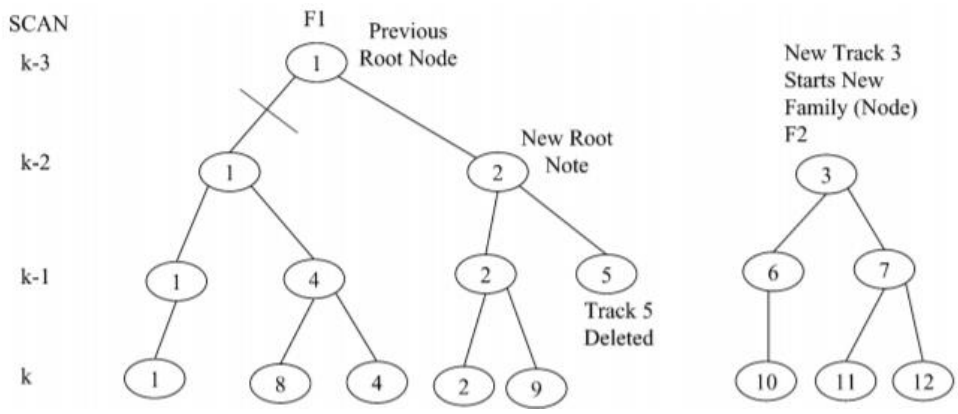


Figure 3.2: N-scan pruning and family structure. [19].

hypothesis arises, two tracks may form. A set of tracks with a common root node is said to be a family. Branches within a family stem from a root node and represent at most one target. Initially the family F1 had a root node 1 at scan k-3. At k-2 two branches were created which represents two conflicting hypothesis. At time k it was found that target 2 was the more likely hypothesis. At an agreed upon scan (normally n-5 or greater, but in this case n-2 for convenience) a new root node is chosen to represent the most likely hypothesis. Thus targets 1,4 and 8 are deleted. This is the process of n-scan pruning.

Chapter 4

Probabilistic Graphical Models

4.1. Introduction to PGMs

In a complex system there will be many interrelated aspects that need to be modelled so that a reasoning task can be performed. Uncertainty is one of these aspects that is well suited to being incorporated into PGMs. Additionally, Koller and Friendman [1, p. 1] emphasise that the declarative representation of PGMS, that is the ability to separate knowledge and reasoning, allows one to develop general algorithms which can be applied to many different models. There are therefore two different steps involved with PGMs, modelling and reasoning. This has lead to PGMs being used in a wide variaty of fields such as medicine, robotics, vision and education [22].

The framework discussed here is built on probability theory which helps to represent uncertainty and relationships. This is discussed first. Graphs are then used to condense complex distributions hence the term Probabilistic Graphical Models. Bayesian networks, which represent our knowledge, and cluster graphs, which are used for inference, are then discussed.

4.1.1. Probability theory refresher

Important concepts in probability theory are recapped here as they are fundamental to the use of PGMs. This section uses concepts explained by Barber in Chapter one of his book, Bayesian Reasoning and Machine Learning [23, p. 1-22].

Chain rule and Bayes' rule

The conditional distribution of a variable x given y is defined as:

$$P(x|y) = \frac{P(x,y)}{P(y)} \tag{4.1}$$

A simple rearrangement results in the chain rule,

$$P(x,y) = P(y)P(x|y) \tag{4.2}$$

which can be extended to:

$$P(x_1, ...x_n) = P(x_1)P(x_2|x_1)...P(x_n|x_1, ..., x_{n-1}) \tag{4.3}$$

This means that one can expand a joint probability into the probability of the first, the conditional probability of the second given the first, the conditional probability of the third given the first and second, and so on. Using the definition of conditional probability, the chain rule and the fact that $P(x, y) = P(y, x)$ one can arrive at Bayes' rule which allows one to transition between $P(x|y)$ and $P(y|x)$:

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)} \tag{4.4}$$

Random variables

Random variables are used to represent various properties and can be defined as a vector,

$$\boldsymbol{X} = (X_1, X_2, \cdots, X_n) \tag{4.5}$$

The notation used in this work consists of a subscript, usually used as an identifier, and a superscript, to denote time. Uppercase italics represent a random variable and bold uppercase italics represent a random vector. Random variables in the continuous space, for example *Height* (X_{H}), are used extensively in this work and can be modelled by Gaussian distributions. Chapter 5 provides a detailed description of these distributions. Discrete random variables, such as *Shoe size* (X_{s}), represent discrete possible outcomes and can be represented as a probability $P(X_{\mathrm{s}})$.

In this work, the task is to represent and then reason about one or more variables, given observations of some others. For example X_{Height} and X_{Weight} can be used to reason about $X_{Shoesize}$. This can be done by constructing conditional distributions over the space of random variables. One can reason about the system or condition on observed variables as such:

$$P(X_{\mathrm{s}}|X_{\mathrm{H}} = x_{\mathrm{H}}, X_{\mathrm{W}} = x_{\mathrm{W}}) \tag{4.6}$$

or more generally,

$$P(X_1|X_2 = x_2, \cdots, X_n = x_n) \tag{4.7}$$

Factors

An important concept for defining and manipulating distributions is a factor. A factor represents a function over random variables $\phi(X_1, ...X_k)$ that specify a piece of a model. Examples of factors include marginal distributions $P(X_1)$, joint distributions $P(X_1, X_2, X_3)$ and conditional distributions $P(X_1|X_2, X_3)$. We can also perform op-

erations on factors, such as multiplication $\phi(X1, X2) = \phi(X1)\phi(X2)$, marginalisation $\phi(X1) = \sum_{X2} \phi(X1, X2)$, and reduction where we can observe evidence and reduce the scope of a factor $\phi(X1|X2 = x_2)$. In Section 5 we discuss these operations on Gaussian factors.

4.1.2. Graphs

Graphs can be represented in a computer in a general format. Some graphs are used to help represent a model, such as a Bayesian network, whilst others are used to perform inference, such as a cluster graph.

An example from Koller [1, p. 35] is given in Figure 4.1. A set of nodes labelled $A, B, ..., I$ are represented as circles. Two nodes can be connected via an edge which can be either directed ($A \to C$) or undirected (C —D). For each pair of nodes there can be a maximum of one edge. When $A \to C$, C is said to be the child of A or A is the parent of C. For undirected graphs where C —D, C and D are said to be neighbours.

4.2. Bayesian networks

A Bayesian network is a directed acyclic graph (DAG) that exploits conditional independence properties to represent distributions. The nodes are random variables and the edges are factors which represent the relationship between random variables.

A well known example from Koller [1, p. 52] of a Bayesian network is the student example given in Figure 4.2. There are five random variables represented as nodes: Difficulty, Intelligence, Grade, SAT and Letter. The difficulty of a course ($P(D)$) and the intelligence of a student ($P(I)$) are marginal distributions while the grade of the student ($P(G|I, D)$) is a conditional distribution, conditioned on Intelligence and Difficulty. A letter of recommendation ($P(L|G)$) is dependant on the grade the student achieved, and the SAT score ($P(S|I)$) is dependant on their intelligence. An important characteristic of Bayesian networks is that dependency is encoded within the graph. For example, the grade

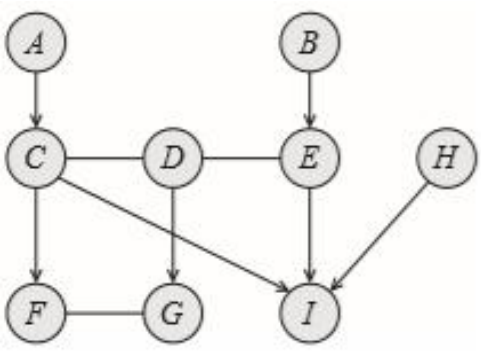


Figure 4.1: Example of partially directed graph, reproduced from Probabilistic grapical models by Koller and Friendman [1]

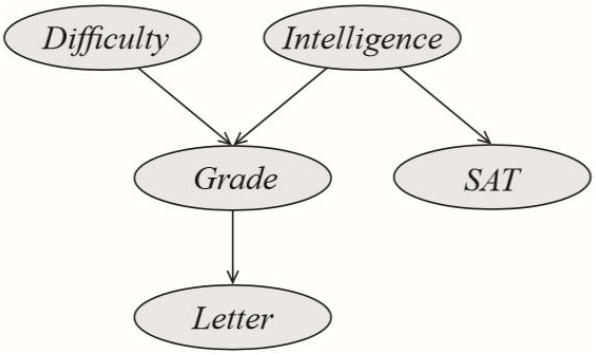


Figure 4.2: Bayesian network for student example from Koller and Friendman [1]

is dependent on both intelligence and difficulty of the course. The letter is dependant on the grade. Bayesian networks are therefore a natural way to model a system using the relationships between random variables.

Using the example of Figure 4.2 and the chain rule from equation 4.3 the joint distribution would be, $P(D, I, G, L, S) = P(D)P(I)P(G|D, I)P(L|G)P(S|I)$, which describes our knowledge of the system.

4.3. Inference

Now that we have described our system we can begin to perform inference which involves asking certain questions. For example, what is the probability that a student gets a good letter? This can be answered by finding $P(L)$ via marginalising the joint distribution by summing out all other variables as in equations, 4.8 to 4.16.

$$P(L) = \sum_{D,I,G,S} P(D, I, G, L, S) \tag{4.8}$$

$$= \sum_{D,I,G,S} P(D)P(I)P(G|D, I)P(L|G)P(S|I) \tag{4.9}$$

$$= \sum_{D,I,G} P(D)P(G|D, I)P(L|G) \sum_{S} P(S|I)P(I) \tag{4.10}$$

$$= \sum_{D,I,G} P(D)P(G|D, I)P(L|G) \sum_{S} \psi_1(S, I) \tag{4.11}$$

$$= \sum_{D,I,G} P(L|G)P(G|D, I)P(D)\delta_{12}(I) \tag{4.12}$$

$$= \sum_{G} P(L|G) \sum_{D,I} P(G|D, I)P(D)\delta_{12}(I) \tag{4.13}$$

$$= \sum_{G} P(L|G) \sum_{D,I} \psi_2(G, D, I)\delta_{12}(I) \tag{4.14}$$

$$= \sum_{G} P(L|G)\delta_{23}(G) \tag{4.15}$$

$$= \sum_{G} \psi_3(L, G)\delta_{23}(G) \tag{4.16}$$

Factors with an overlapping score are multiplied to form a cluster, for example, $P(I)$ and $P(S|I)$ form a cluster $\psi_1(S, I)$. A cluster can then send and receive messages with an overlapping score to other clusters such as $\delta_{12}(I) = \sum_S \psi_1(S, I)$. The same variable elimination can be performed on each variable as shown in appendix C and this would result in the same clusters:

$$\psi_1(S, I) = P(I)P(S|I) \tag{4.17a}$$
$$\psi_2(G, D, I) = P(G|D, I)P(D) \tag{4.17b}$$
$$\psi_3(L, G) = P(L|G) \tag{4.17c}$$

where the messages between clusters are,

$$\delta_{12}(I) = \sum_S \psi_1(S, I) \tag{4.18a}$$
$$\delta_{23}(G) = \sum_{D,I} \psi_2(G, D, I)\delta_{12}(I) \tag{4.18b}$$
$$\delta_{32}(G) = \sum_L \psi_3(L, G) \tag{4.18c}$$
$$\delta_{21}(I) = \sum_{D,G} \psi_2(G, D, I)\delta_{32}(G). \tag{4.18d}$$

In this way a run of variable elimination can be represented as a graph with nodes being the clusters and edges being the messages, as shown in Figure 4.3.

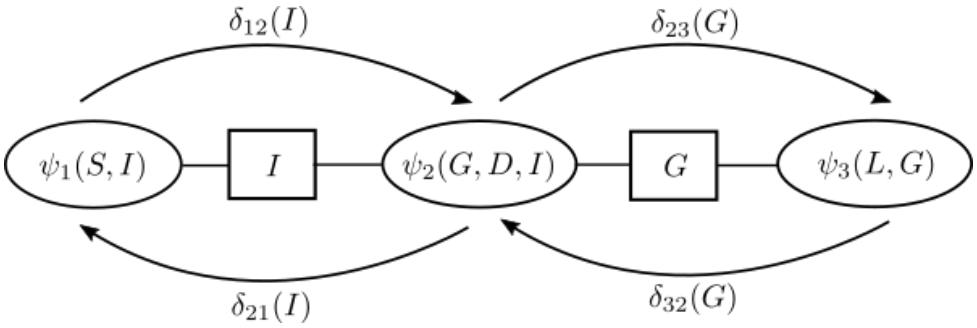
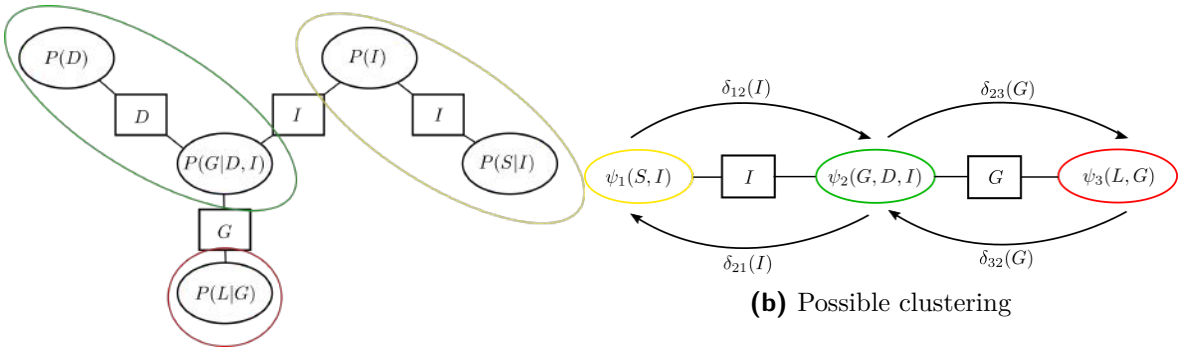


Figure 4.3: Cluster graph showing nodes as cluster, edges as sepsets and the messages sent between clusters

4.3.1. Cluster graphs

It is often faster to construct a cluster graph than it is to perform variable elimination. Given a Bayesian network, as in Figure 4.2, a graph showing overlapping factors is made in Figure 4.4a. Nodes whose scope is a subset of a neighbouring nodes' scope can be clustered together via multiplication. In Figure 4.4b the scope of $P(I)$ is a subset of $P(S|I)$ and thus they can be clustered together to form $\psi_1(S, I)$. Similarly, $P(D)$ and $P(G|D, I)$ are clustered into $\psi_2(G, D, I)$ and $P(L|G)$ is the third cluster $\psi_3(L, G)$. These clusters can

send messages between each other with the scope of their sepset. A sepset $S_{1,2}$ between cluster one and two, is the scope that two clusters share $S_{1,2} = I$. Cluster are connected via sepsets and the resulting graph is the same as in Figure 4.3. An important property



(a) Intermediate factor graph

(b) Possible clustering

Figure 4.4: (a) Bayesian network of student example with intermediate factors. (b) Possible clustering of nodes whose scope are a subset of a neighboring nodes scope.

of cluster graphs is that they satisfy the running intersection property which states that: Whenever there is a variable X that is an element of clusters $X \in C_i$ and $X \in C_j$ there is a single path between $C_i - C_j$. This ensures that the information about X will be sent to all clusters that directly require X.

4.3.2. Message passing: sum-product algorithm

A category of inference algorithms which use cluster graphs is message passing. This involves nodes sending and receiving messages to update distributions based on the belief of another cluster. Message passing via the sum-product algorithm begins with each cluster ψ_j generating a cluster potential $\psi_j(C_i)$ by multiplying all its factors. For the first cluster this would be $\psi_1(S, I) = \phi_I(I)\phi_S(S, I) = P(I)P(S|I)$. This cluster potential is then multiplied by any incoming messages $\delta_{i\to j}(S_{i,j})$ which will have the scope of their shared sepset $S_{i,j}$. An outgoing message $\delta_{j\to k}(S_{j,k})$ is then determined via summing out all variables not in the setpset $S_{j,k}$. For leaf nodes, nodes which are connected to at most one cluster, the outgoing message is simply equal to the cluster potential summing over the sepset of its neighbouring cluster. In summary an initial potential is calculated as

$$\psi_j(C_j) = \prod_n \phi_n(X_n, par(X_n)) \tag{4.19}$$

where n is an identifier for each factor $\phi_n(X_n, par(X_n))$ assigned to a cluster C_j. Any message from a leaf node is the cluster potential marginalised over the required sepset,

$$\delta_{j\to k}(S_{j,k}) = \sum_{C_j - S_{j,k}} \psi_j(C_j). \tag{4.20}$$

The outgoing messages from remaining clusters are equal to all incoming messages, multiplied by the cluster potential, and summed to have the scope of its sepset,

$$\delta_{j\to k} = \sum_{C_j - S_{j,k}} \psi_j(C_j) \prod_{i \neq k} \delta_{i\to j}(S_{i,j}) \tag{4.21}$$

where i identifies a list of all clusters which sent a message to cluster C_i apart from the cluster where the outgoing message is being sent to C_k. The process of summing and multiplying is why it is called the sum-product algorithm.

In tree-structured cluster graphs the sum-product algorithm result in a run of variable elimination. For example, if one had to query, "what is the probability of a student getting a good letter" we would calculated $P(L)$. This is done by multiplying the cluster potential containing L, $\psi_3(L, G)$, by its incoming messages $\delta_{2\to 3}(G)$, and then marginalising so $P(L) = \sum_G \psi_3(L, G)\delta_{23}(G)$. This allows a query to be answered using the same clusters and messages.

A more useful query is one based on evidence such as finding $P(L|G)$, the probability of a student getting a good letter given their grade. Evidence is incorporated by reduction which reduces the scope of a distribution when presenting evidence about a random variable $G = g$. $P(L|G)$ is therefore $P(L) = \psi_3(L, G = g)\delta_{23}(G = g)$, as we no longer have to marginalise over G. When a message is sent from a cluster where evidence has been observed, that evidence is shared with the graph.

4.4. Context

The work presented in this chapter allows one to model the relationship between random variables using a graph called a Bayesian network. This can then be converted into a cluster graph where message passing can be performed to answer queries and infer information about certain random variables. The operations which are required to perform message passing using the sum-product algorithm are: marginalisation through summation for discrete variables, multiplication for updating factors, creating joint distributions and performing reduction to observe evidence. In the next chapter, these operations are discussed in terms of continuous variables described by Gaussian distributions.

Chapter 5

Gaussian distributions

This chapter begins with a small discussion on continuous variables which are used throughout this report to represent, for example, a pixels location. Gaussian distributions, which can be used to describe a continuous variables distribution, are then introduced. The various operations on Gaussian distributions are explained, as well as how to parametrize a Gaussian in different forms. Linear Gaussian's are then discussed and a method for representing a mixture of Gaussian's concludes this chapter.

5.1. Continuous variables

The probability of any singular value occurring in continuous variables will be zero as there are an infinite number of possible values within its specified range. The probability distribution for continuous variables therefore needs a representation that is different to discrete variables.

Probability density function

A probability density function (PDF) represents the likelihood of a value in continuous variables. It integrates to a value of one, $\int_{Val(x)} p(x)dx = 1$ (meaning it is certain the value will occur over the full possible range of values) and is greater than or equal to zero, $p(x) \geq 0$ (so that there is never a negative probability of the value occurring). In a PDF the probability of an event occurring within a given range (a, b) is:

$$P(a \leq x \leq b) = \int_a^b p(x)dx \tag{5.1}$$

5.2. Univariate Gaussian distribution

The distribution for a continuous variable can be described by a univariate Gaussian distribution, also known as the normal distribution, which is described by the PDF:

$$\mathcal{N}(x; \mu, \sigma^2) = p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.2}$$

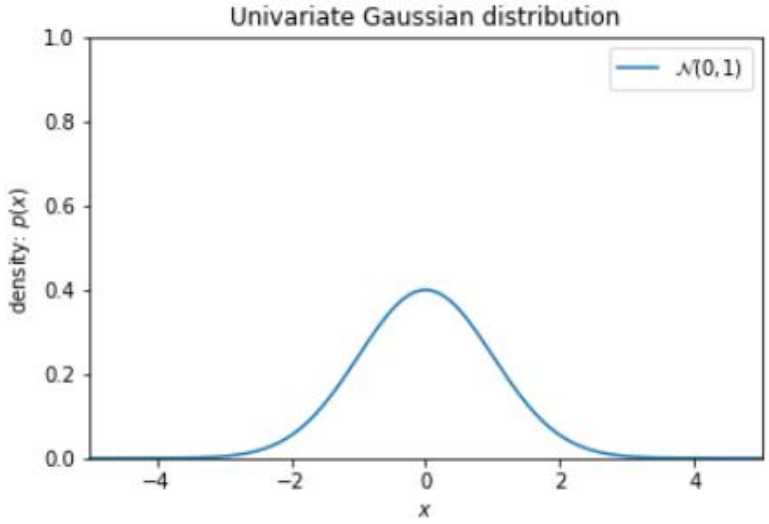


Figure 5.1: Univariate Gaussian Distribution

where μ represents the mean value of the distribution and σ represents its variance. Figure 5.1 plots this PDF with a mean ($\mu = 0$) and a variance ($\sigma^2 = 1$). This distribution is used to represent many naturally occurring instances where uncertainty is involved.

5.3. Multivariate Gaussian distribution

Univariate Gaussian distributions can be extended to contain multiple variables. In a simple case such as an objects location in an image plane there could be two coordinates. A multivariate distribution will represent the knowledge of its location by its mean values, which will be where it is most likely located, as well as by its covariance which represents our uncertainty.

5.3.1. Covariance form

The most common parametrization of a multivariate Gaussian is the covariance form. It represents random variables $\boldsymbol{X} = (X_1, X_2, \cdots, X_n)^T$, by an n-dimensional mean vector, $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_n)^T$, and an $n \times n$ covariance matrix,

$$\boldsymbol{\Sigma_{n,n}} = \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \cdots & \Sigma_{1,n} \\ \Sigma_{2,1} & \Sigma_{2,2} & \cdots & \Sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{n,1} & \Sigma_{n,2} & \cdots & \Sigma_{n,n} \end{pmatrix}. \tag{5.3}$$

The multivariate PDF in the covariance form is given as [1, p. 247],

$$\mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2\pi^{\frac{n}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})]. \tag{5.4}$$

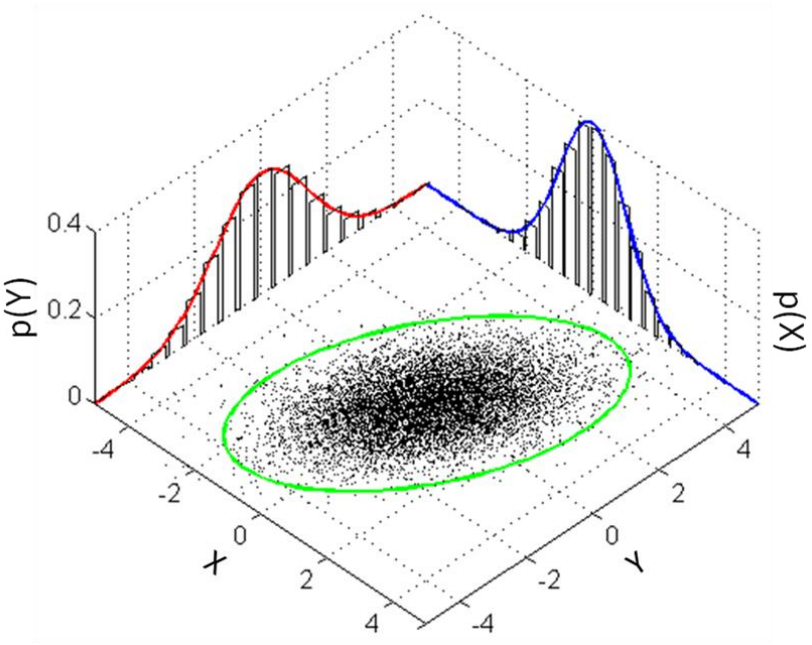


Figure 5.2: Multivariate Gaussian distribution reproduced from wikipedia, https://en.wikipedia.org/wiki/Multivariate_normal_distribution

For a two dimensional random vector represented by a multivariate Gaussian distribution with $\boldsymbol{\mu} = (0, 0,)^T$ and,

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.6 \\ 0.6 & 2 \end{pmatrix}, \tag{5.5}$$

The plot in Figure 5.2 is shown. The ellipse indicates the region where 95 percent of samples would fall in.

5.3.2. Canonical form

Often it is better to perform certain operations on Gaussians using different parametrizations. For example, when marginalizing it is easier to perform using the covariance form but for reduction it is easier to perform using the canonical form. The parametrisation is as follows [1, p. 609],

$$\mathcal{C}(\boldsymbol{X}; \mathbf{K}, \mathbf{h}, g) = e(-\frac{1}{2}\boldsymbol{X}^T\mathbf{K}\boldsymbol{X} + \mathbf{h}^T\boldsymbol{X} + g) \tag{5.6}$$

with an information matrix defined as the inverse of the covariance matrix, $\mathbf{K} = \boldsymbol{\Sigma}^{-1}$, the information vector being, $\mathbf{h} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$, and a normalisation constant equal to, $g = -\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \log((2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}})$

5.4. Gaussian operations

The benefit of representing Gaussian's in the canonical form is that it becomes easier to perform many operations. A Gaussian factor is a function that involves Gaussian distributions and describes a piece of a model in terms of Gaussian's distributions. During this section a matrix may be defined by a subscript to show which factor it belongs to. For example $\mathbf{K}_1$ is from $\boldsymbol{\phi}_1(X, Y)$. In the case where one is referring to an entry within $\mathbf{K}_1$ it will be done as such:

$$\mathbf{K}_1 = \begin{bmatrix} K_{1,XX} & K_{1,XY} \\ K_{1,YX} & K_{1,YY} \end{bmatrix} \tag{5.7}$$

5.4.1. Matching the scope

The scope of a canonical Gaussian can be extended by padding the relevant entries with zeros. This is useful for performing operations on two factors with different scopes. The factors $\phi_1(X, Y)$ and $\phi_2(Y, Z)$ contain joint distributions between their relevant random variables described in the canonical form.

$$\boldsymbol{\phi_1}(X, Y) = \mathcal{C}\left(\begin{bmatrix} X \\ Y \end{bmatrix}; \begin{bmatrix} K_{1,XX} & K_{1,XY} \\ K_{1,YX} & K_{1,YY} \end{bmatrix}, \begin{bmatrix} h_{1,X} \\ h_{1,Y} \end{bmatrix}, g_1\right) \tag{5.8}$$

$$\boldsymbol{\phi_2}(Y, Z) = \mathcal{C}\left(\begin{bmatrix} Y \\ Z \end{bmatrix}; \begin{bmatrix} K_{2,YY} & K_{2,YZ} \\ K_{2,ZY} & K_{2,ZZ} \end{bmatrix}, \begin{bmatrix} h_{2,Y} \\ h_{2,Z} \end{bmatrix}, g_2\right) \tag{5.9}$$

The scope of $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$ can be extended,

$$\boldsymbol{\phi_1}(X, Y, Z) = \mathcal{C}\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}; \begin{bmatrix} K_{1,XX} & K_{1,XY} & 0 \\ K_{1,YX} & K_{1,YY} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} h_{1,X} \\ h_{1,Y} \\ 0 \end{bmatrix}, g_1\right) \tag{5.10}$$

$$\boldsymbol{\phi_2}(Y, Z, X) = \mathcal{C}\left(\begin{bmatrix} Y \\ Z \\ X \end{bmatrix}; \begin{bmatrix} K_{2,YY} & K_{2,YZ} & 0 \\ K_{2,ZY} & K_{2,ZZ} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} h_{2,Y} \\ h_{2,Z} \\ 0 \end{bmatrix}, g_2\right) \tag{5.11}$$

and $\boldsymbol{\phi_2}$ can be then also be re-arranged to match the variable ordering of $\boldsymbol{\phi_1}$,

$$\boldsymbol{\phi_2}(X, Y, Z) = \mathcal{C}\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}; \begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{2,YY} & K_{2,YZ} \\ 0 & K_{2,ZY} & K_{2,ZZ} \end{bmatrix}, \begin{bmatrix} 0 \\ h_{2,Y} \\ h_{2,Z} \end{bmatrix}, g_2\right) \tag{5.12}$$

5.4.2. Canonical form product

Two canonical Gaussians with the same scope can then be easily multiplied [1, p. 610]:

$$\mathcal{C}(\boldsymbol{X};\mathbf{K}_1,\mathbf{h}_1,g_1)\mathcal{C}(\boldsymbol{X};\mathbf{K}_2,\mathbf{h}_2,g_2)=\mathcal{C}(\boldsymbol{X};\mathbf{K}_1+\mathbf{K}_2,\mathbf{h}_1+\mathbf{h}_2,g_1+g_2) \tag{5.13}$$

The multiplication of the factors $\boldsymbol{\phi_1}$ in equation (5.10) and $\boldsymbol{\phi_2}$ in equation (5.12) is,

$$\boldsymbol{\phi_1}\times\boldsymbol{\phi_2}=\mathcal{C}\left(\begin{bmatrix}X\\Y\\Z\end{bmatrix};\begin{bmatrix}K_{1,XX} & K_{1,XY} & 0\\ K_{1,YX} & K_{1,YY}+K_{2,YY} & K_{2,YZ}\\ 0 & K_{2,ZY} & K_{2,ZZ}\end{bmatrix},\begin{bmatrix}h_{1,X}\\ h_{1,Y}+h_{2,Y}\\ h_{2,Z}\end{bmatrix},g_1+g_2\right) \tag{5.14}$$

5.4.3. Canonical form division

Similarly, two canonical Gaussians can be divided [1, p. 610]:

$$\frac{\mathcal{C}(\boldsymbol{X};\mathbf{K}_1,\mathbf{h}_1,g_1)}{\mathcal{C}(\boldsymbol{X};\mathbf{K}_2,\mathbf{h}_2,g_2)}=\mathcal{C}(\boldsymbol{X};\mathbf{K}_1-\mathbf{K}_2,\mathbf{h}_1-\mathbf{h}_2,g_1-g_2) \tag{5.15}$$

For $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$ the division is then:

$$\frac{\boldsymbol{\phi_1}}{\boldsymbol{\phi_2}}=\mathcal{C}\left(\begin{bmatrix}X\\Y\\Z\end{bmatrix};\begin{bmatrix}K_{1,XX} & K_{1,XY} & 0\\ K_{1,YX} & K_{1,YY}-K_{2,YY} & K_{2,YZ}\\ 0 & K_{2,ZY} & K_{2,ZZ}\end{bmatrix},\begin{bmatrix}h_{1,X}\\ h_{1,Y}-h_{2,Y}\\ h_{2,Z}\end{bmatrix},g_1-g_2\right) \tag{5.16}$$

5.4.4. Marginalization

If one considers a canonical Gaussian with the scope $[\boldsymbol{X},\boldsymbol{Y}]$, $\mathcal{C}([\boldsymbol{X},\boldsymbol{Y}];\mathbf{K},\mathbf{h},g)$, the marginalization onto $\boldsymbol{X}$ is the integral over $\boldsymbol{Y}$,

$$\int\mathcal{C}(\boldsymbol{X},\boldsymbol{Y};\mathbf{K},\mathbf{h},g)d\boldsymbol{Y}=\mathcal{C}(\boldsymbol{X};\mathbf{K}',\mathbf{h}',g'), \tag{5.17}$$

where, $\mathbf{K}'=K_{\boldsymbol{XX}}-K_{\boldsymbol{XY}}K_{\boldsymbol{YY}}^{-1}K_{\boldsymbol{YX}}$, $\mathbf{h}'=h_{\boldsymbol{X}}-K_{\boldsymbol{XY}}K_{\boldsymbol{YY}}^{-1}h_{\boldsymbol{Y}}$ and, $g'=g+\frac{1}{2}(log|2\pi K_{\boldsymbol{YY}}^{-1}|+h_{\boldsymbol{Y}}^{T}K_{\boldsymbol{YY}}^{-1}h_{\boldsymbol{Y}})$. Note that for marginalization it is far simpler to perform using the covariance form [1, p. 250]. Given a joint distribution,

$$P(\boldsymbol{X},\boldsymbol{Y})=\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu_X}\\ \boldsymbol{\mu_Y}\end{bmatrix};\begin{bmatrix}\boldsymbol{\Sigma_{XX}} & \boldsymbol{\Sigma_{XY}}\\ \boldsymbol{\Sigma_{YX}} & \boldsymbol{\Sigma_{XX}}\end{bmatrix}\right), \tag{5.18}$$

the marginalized distribution onto $\boldsymbol{Y}$ is,

$$P(\boldsymbol{Y})=\mathcal{N}(\boldsymbol{Y};\boldsymbol{\mu_Y},\boldsymbol{\Sigma_{YY}}) \tag{5.19}$$

and the marginalized distribution onto $\boldsymbol{X}$ is,

$$P(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{XX}}). \tag{5.20}$$

5.4.5. Canonical reduction

To observe evidence, a canonical Gaussian $\mathcal{C}(\boldsymbol{X}, \boldsymbol{Y}; \mathbf{K}, \mathbf{h}, g)$ can be reduced by setting $\boldsymbol{Y} = \mathbf{y}$. The resulting factor is,

$$\mathcal{C}(\boldsymbol{X}; \mathbf{K}', \mathbf{h}', g') \tag{5.21}$$

where, $\mathbf{K}' = K_{\boldsymbol{XX}}$, $\mathbf{h}' = h_{\boldsymbol{X}} - K_{\boldsymbol{XY}}\mathbf{y}$, and $g' = g + h_{\boldsymbol{Y}}^T\boldsymbol{y} - \frac{1}{2}\mathbf{y}^T K_{\boldsymbol{YY}}\mathbf{y}$. The operations required for inference with Gaussian distributions have now been explained.

5.5. Linear Gaussian distributions

In this work Gaussian random variables such as state $\boldsymbol{X}^k$ at time $t = k$ will be linearly dependant on its previous state $P(\boldsymbol{X}^k|\boldsymbol{X}^{k-1})$. This can be seen in the linear motion and measurement update equations:

$$\boldsymbol{X}_k = \mathbf{A}\boldsymbol{X}_{k-1} + \mathbf{B}\boldsymbol{u}_{k-1} + \boldsymbol{W}_k \tag{3.1}$$

and

$$\boldsymbol{Y}_k = \mathbf{H}\boldsymbol{X}_{k-1} + \boldsymbol{V}_k \tag{3.2}$$

Linear Gaussian of parents

Koller explains in Theorem 7.3 [1, p. 251] that a linear Guassian network is defined by a joint multivariate Gaussian distribution. Let $\boldsymbol{X} = \mathcal{N}(\boldsymbol{\mu}_X; \boldsymbol{\Sigma}_X)$. When $\boldsymbol{Y}$ is a linear Gaussian of its parents $\boldsymbol{X} = X_1, X_2, \ldots X_n$ it is denoted:

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{Y}; \beta_0 + \boldsymbol{\beta}^T\boldsymbol{X}, \boldsymbol{\sigma}^2) \tag{5.22}$$

The distribution of $\boldsymbol{Y}$ is therefore a multivariate Gaussian $P(\boldsymbol{Y}) = \mathcal{N}(\boldsymbol{Y}; \boldsymbol{\mu}_{\boldsymbol{Y}}, \boldsymbol{\Sigma}_{\boldsymbol{Y}})$ with,

$$\boldsymbol{\mu}_{\boldsymbol{Y}} = \boldsymbol{\beta}_{\mathbf{0}} + \boldsymbol{\beta}^T\boldsymbol{\mu}_{\boldsymbol{X}} \tag{5.23}$$

and,

$$\boldsymbol{\Sigma}_{\boldsymbol{Y}} = \boldsymbol{\sigma}^2 + \boldsymbol{\beta}^T\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{\beta} \tag{5.24}$$

Linear Gaussian joint distributions

In the case where one wanted the joint distribution $P(\boldsymbol{X}, \boldsymbol{Y})$ it can be represented as,

$$P(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu_X} \\ \boldsymbol{\mu_Y} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma_{XX}} & \boldsymbol{\Sigma_{XY}} \\ \boldsymbol{\Sigma_{YX}} & \boldsymbol{\Sigma_{YY}} \end{bmatrix}\right) \tag{5.25}$$

where the variances are,

$$\boldsymbol{\Sigma_{XX}} = \boldsymbol{\Sigma_{XX}} \tag{5.26}$$

$$\boldsymbol{\Sigma_{YY}} = \boldsymbol{\Sigma_{YY}} \tag{5.27}$$

and the covariances are,

$$\boldsymbol{\Sigma_{X_i,Y}} = \sum_{j=1}^{n} \boldsymbol{\beta}_j \boldsymbol{\Sigma}_{i,j} \tag{5.28}$$

for the number of dimensions, n. Then with $\boldsymbol{\Sigma_{YX}} = \boldsymbol{\Sigma_{XY}^T}$:

$$P(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu_X} \\ \boldsymbol{\beta}_0 + \boldsymbol{\beta}^T \boldsymbol{\mu_X} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma_X} & \boldsymbol{\Sigma_X \beta}^T \\ \boldsymbol{\beta \Sigma_X} & \boldsymbol{\sigma}^2 + \boldsymbol{\beta}^T \boldsymbol{\Sigma_X \beta} \end{bmatrix}\right) \tag{5.29}$$

This section has shown how to construct linear Gaussian distribution based on a Gaussian parent, as well as constructing a joint distribution with its parents.

5.6. Sum of Gaussian estimation

If a density function $P(\boldsymbol{X})$ is represented by a mixture of k Gaussians, $P(\boldsymbol{X}) \sim \sum_{i=1}^{k} w_i \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ it can be approximated as $Q(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where,

$$\boldsymbol{\mu} = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i, \tag{5.30}$$

and,

$$\boldsymbol{\Sigma} = \sum_{i=1}^{k} w_i \boldsymbol{\Sigma}_i + \sum_{i=1}^{k} w_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})((\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{5.31}$$

This approximation has the same means and covariances as the orignal distribution but is only represented by a single Gaussian. This approximation is used so that the number of Gaussians in a Gaussian mixture does not grow indefinitely and become intractable.

Chapter 6

Kalman filter PGM for single object tracking

We now have the tools required to perform single object tracking. This section considers the case where there is one object to be tracked and one measurement at each time step. Later, the model is expanded to include cases where the measurement is missing and where there are false measurements. Due to the temporal nature of tracking the notation of using a superscript to indicate the time step is used. $\boldsymbol{X}_1^0$ refers to object 1 at time 0 whereas $\boldsymbol{X}_1^t$ is the same object at time t.

The chapter begins by modelling the Kalman filter as a PGM. Inference via the sum product algorithm is then performed to arrive at the Kalman filter update and prediction equations. The model is then expanded to include missed detections and false measurements so that a single object tracker can be implemented.

6.1. Model representation

When interpreting the Kalman filter as a PGM we first describe the state of an object. An object's state $\boldsymbol{X}$ is a random vector of pixel locations in the x and y direction S_x and S_y, as well as velocities V_x and V_y. The velocity is measured in $\frac{pixels}{timestep}$ where the *timestep* is the amount of time between each frame. For a video with a standard frame rate of 24fps, the *timestep* is $\frac{1}{24}$s or 0.04167s. The velocity can be thought of as the number of pixels an object moves between frames.

The object's state is never observed and is thus drawn from a multivariate Gaussian distribution $\boldsymbol{X}^t \sim \mathcal{N}(\boldsymbol{\mu}^t, \mathbf{P}^t)$. The mean vector $\boldsymbol{\mu}^t$ contains the mean value of each random variable, whereas the covariance matrix $\mathbf{P}^t$ contains the variance of each random variabe in its diagonaml elements, and the covariance between random variables in the other elements

$$\boldsymbol{\mu}^t = \begin{bmatrix} \mu_{sx} \\ \mu_{sy} \\ \mu_{vx} \\ \mu_{vy} \end{bmatrix}, \quad \mathbf{P}^t = \begin{bmatrix} \sigma_{sx}^2 & 0 & \sigma_{sx,vx} & 0 \\ 0 & \sigma_{sy}^2 & 0 & \sigma_{sy,vy} \\ \sigma_{vx,sx} & 0 & \sigma_{vx}^2 & 0 \\ 0 & \sigma_{vy,sy} & 0 & \sigma_{vy}^2 \end{bmatrix}. \tag{6.1}$$

Note that each element in $\boldsymbol{\mu}^t$ and $\mathbf{P}^t$ should contain a superscript so that μ_{sx} should be μ_{sx}^t. However, these are omitted for the sake of brevity. Inspecting the covariance matrix $\mathbf{P}^t$, it can be seen that the objects position is related to its velocity so $\sigma_{sx,vx}$ and $\sigma_{sy,vy}$ are $\neq 0$, and vice-versa, $\sigma_{vx,sx}$ and $\sigma_{vy,sy} \neq 0$. The positions themselves are independent so $\sigma_{sx,sy}$ and $\sigma_{sy,sx} = 0$ and same with the velocities $\sigma_{vx,vy}$ and $\sigma_{vy,vx} = 0$. This is explained by the motion model in Section 3.1 Kalman filter,

$$\boldsymbol{X}^t = \mathbf{A}\boldsymbol{X}^{t-1} + \boldsymbol{W}^t \tag{6.2}$$

where $\boldsymbol{W}^t$ is zero mean Gaussian noise $\boldsymbol{W}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^t)$ and $\mathbf{A}$ is the transition matrix. The transition matrix $\mathbf{A}$ is defined in such a way so that the position at the next frame will be equal to the previous position plus the velocity and some noise,

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

There is therefore a relationship between positions and their respective velocities but the x and y dimensions are independent of each other. A constant velocity is assumed and any change in velocity is modelled as noise. An alternative way of viewing the motion model, more aligned with PGMs, is that the object state $\boldsymbol{X}^t$ is a linear Gaussian of its parents $\boldsymbol{X}^{t-1}$ with added noise $\boldsymbol{W}^t$,

$$P(\boldsymbol{X}^t|\boldsymbol{X}^{t-1}) = \mathcal{N}(\boldsymbol{X}^t; \mathbf{A}\boldsymbol{X}^{t-1}, \mathbf{Q}^t) \tag{6.4}$$

as discussed in Section 5.5 Linear Gaussian distributions.

The measurements consist of only the centroid location of an object mask $\mathbf{y} = [c_x, c_y]^T$. A random vector for the measurement $\boldsymbol{Y}$ is a linear transform of the object state shown in the measurement model from Section 3.1 Kalman filter,

$$\boldsymbol{Y}^t = \mathbf{H}\boldsymbol{X}^t + \boldsymbol{V}^t. \tag{6.5}$$

Therefore the measurement matrix $\mathbf{H}$ selects the position variables from the object state and is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.6}$$

In a similar fashion to equation 6.4, the measurement $\boldsymbol{Y}^t$ can also be described as a linear

Gaussian of its parents $\boldsymbol{X}^t$ and some noise $\boldsymbol{V}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^t)$,

$$P(\boldsymbol{Y}^t|\boldsymbol{X}^t) = \mathcal{N}(\boldsymbol{Y}^t; \mathbf{H}\boldsymbol{X}^t, \mathbf{R}^t) \tag{6.7}$$

The relationships between random the variables can be represented as a Bayesian network where each measurement is caused by the state of the object and the updated object state is derived from the previous object state. This is shown in Figure 6.1.

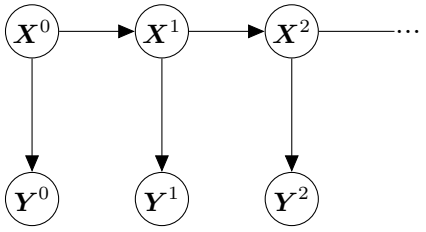


Figure 6.1: Bayesian network of a Kalman filter.

6.2. Inference

The Bayesian network in Figure 6.1 is used to create a cluster graph shown in Figure 6.2. It is now possible to perform inference on this graph via message passing using the sum-product algorithm.

The message sent from one state transition cluster to the next, $\delta_{X,X}(\boldsymbol{X}^t) = P(\boldsymbol{X}^t|\mathbb{Y}^{t-1})$, where $\mathbb{Y}^{t-1} = (\boldsymbol{Y_0} = \mathbf{y}^0, \boldsymbol{Y}^1 = \mathbf{y}^1, \ldots, \boldsymbol{Y}^{t-1} = \mathbf{y}^{t-1})$ is used to represent past observed evidence. This contains the predicted belief of the state before evidence at that time step has been included. The message sent from the measurement cluster to the state transition cluster $\delta_{Y,X}(\boldsymbol{X}^t)$ is the belief of the state $\boldsymbol{X}^t$ after observing this evidence $P(\boldsymbol{X}^t|\boldsymbol{Y}^t = \mathbf{y}^t)$. Subsections 6.2.1 and 6.2.2 show how when certain queries are made, it is possible to arrive at the Kalman filter update equations in Section 3.1.

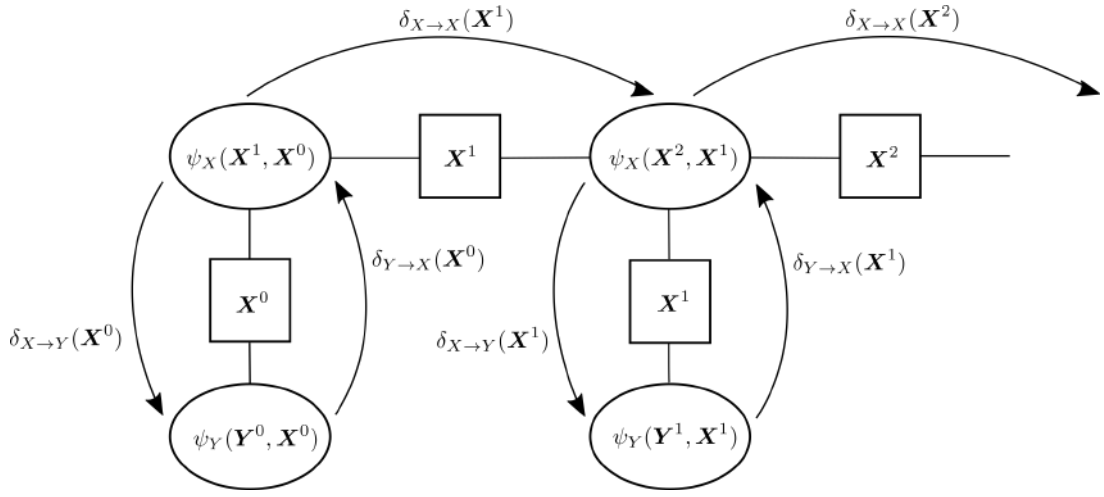


Figure 6.2: Cluster graph of a Kalman filter

6.2.1. Measurement Update

To get to the measurement update equations,

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}(\mathbf{y}_k - \mathbf{H}\tilde{\mathbf{x}}_k), \quad \text{and} \quad \hat{\mathbf{P}}_k = \tilde{\mathbf{P}}_k - \mathbf{K}\mathbf{H}\tilde{\mathbf{P}}_k, \tag{6.8}$$

sum-product message passing can be performed by taking the product of the incoming message to the measurement cluster $\delta_{X,Y}(\boldsymbol{X}^t) = \delta_{X,X}(\boldsymbol{X}^t)$, and the measurement cluster potential $\psi_Y(\boldsymbol{Y}^t, \boldsymbol{X}^t)$. Evidence can be observed $\boldsymbol{Y}^t = \mathbf{y}^t$ so the updated belief about the state $P(\boldsymbol{X}^t|\mathbb{Y}^t)$ is determined. This is also equivalent to simply multiplying the predicted state belief $\delta_{X,X}(\boldsymbol{X}^t) = P(\boldsymbol{X}^t|\mathbb{Y}^{t-1})$ by the measurement cluster message $\delta_{Y,X}(\boldsymbol{X}^t) = P(\boldsymbol{Y}^t = \mathbf{y}^t|\boldsymbol{X}^t)$ having observed the evidence $\boldsymbol{Y}^t = \mathbf{y}^t$. This approach is now performed using the following steps.

Joint distribution

The joint distribution $P(\boldsymbol{X}^t, \boldsymbol{Y}^t)$ is the result of applying the chain rule to the multiplication of $\delta_{X,X}(\boldsymbol{X}^t) = P(\boldsymbol{X}^t|\mathbb{Y}^{t-1})$ and $\delta_{Y,X}(\boldsymbol{Y}^t, \boldsymbol{X}^t) = P(\boldsymbol{Y}^t|\boldsymbol{X}^t)$. Using the joint linear Gaussian distributions in Section 5.5,

$$P(\boldsymbol{X}^t, \boldsymbol{Y}^t|\mathbb{Y}^{t-1}) = \mathcal{N}([\boldsymbol{X}^t, \boldsymbol{Y}^t]^T; \boldsymbol{\mu}_j^t, \boldsymbol{\Sigma}_j^t), \tag{6.9}$$

equation 5.29 defines the joint mean vector $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$,

$$\boldsymbol{\mu}_j^t = \begin{bmatrix} \boldsymbol{\mu}^t \\ \mathbf{H}\boldsymbol{\mu}^t \end{bmatrix}, \boldsymbol{\Sigma}_j^t = \begin{bmatrix} \mathbf{P}^t & \mathbf{P}^t\mathbf{H}^T \\ \mathbf{H}\mathbf{P}^t & \mathbf{R}^t + \mathbf{H}\mathbf{P}^t\mathbf{H}^T \end{bmatrix}. \tag{6.10}$$

Joint distribution in canonical form

Before observing evidence it is helpful to convert to a canonical Gaussian as in equation 5.6,

$$\mathcal{C}([\boldsymbol{X}^t, \boldsymbol{Y}^t]^T; \mathbf{K}_j^t, \mathbf{h}_j^t, g_j^t) = \mathcal{N}([\boldsymbol{X}^t, \boldsymbol{Y}^t]^T; \boldsymbol{\mu}_j^t, \boldsymbol{\Sigma}_j^t), \tag{6.11}$$

with the information matrix and vector,

$$\begin{bmatrix} \mathbf{K}_{j,\boldsymbol{XX}}^t & \mathbf{K}_{j,\boldsymbol{XY}}^t \\ \mathbf{K}_{j,\boldsymbol{YX}}^t & \mathbf{K}_{j,\boldsymbol{YY}}^t \end{bmatrix} = \mathbf{K}_j^t = (\boldsymbol{\Sigma}_j^t)^{-1} \quad , \quad \begin{bmatrix} \mathbf{h}_{j,\boldsymbol{X}}^t \\ \mathbf{h}_{j,\boldsymbol{Y}}^t \end{bmatrix} = \mathbf{h}_j^t = (\boldsymbol{\Sigma}_j^t)^{-1}\boldsymbol{\mu}_j^t \tag{6.12}$$

and the normalization constant,

$$g_j^t = -\frac{1}{2}(\boldsymbol{\mu}_j^t)^T(\boldsymbol{\Sigma}_j^t)^{-1}\boldsymbol{\mu}_j^t - \log((2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}_j^t|^{\frac{1}{2}}). \tag{6.13}$$

The superscripts denoting time are dropped as we are discussing all elements at time t. Matrix inversion lemma (see D.4) are used to find each element in $\boldsymbol{K}_j$,

$$\mathbf{K}^t_{j,\boldsymbol{XX}} = (\boldsymbol{\Sigma}_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}})^{-1} \tag{6.14}$$

$$\mathbf{K}^t_{j,\boldsymbol{XY}} = -(\boldsymbol{\Sigma}_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}})^{-1}\boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}} \tag{6.15}$$

$$\mathbf{K}^t_{j,\boldsymbol{YX}} = -\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}}(\boldsymbol{\Sigma}_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}})^{-1} \tag{6.16}$$

$$\mathbf{K}^t_{j,\boldsymbol{YY}} = \boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}} + \boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}}(\boldsymbol{\Sigma}_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}})^{-1}\boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}. \tag{6.17}$$

The information vectors can be found via simple multiplication:

$$\mathbf{h}^t_{j,\boldsymbol{X}} = \mathbf{K}_{j,\boldsymbol{XX}}\boldsymbol{\mu}_{j,\boldsymbol{X}} + \mathbf{K}_{j,\boldsymbol{XY}}\boldsymbol{\mu}_{j,\boldsymbol{Y}} \tag{6.18}$$

$$\mathbf{h}^t_{j,\boldsymbol{Y}} = \mathbf{K}_{j,\boldsymbol{YX}}\boldsymbol{\mu}_{j,\boldsymbol{X}} + \mathbf{K}_{j,\boldsymbol{YY}}\boldsymbol{\mu}_{j,\boldsymbol{Y}} \tag{6.19}$$

Observing evidence via reduction in canonical form

The belief of $P(\boldsymbol{X}^t, \boldsymbol{Y}^t = \mathbf{y}^t|\mathbb{Y}^{t-1})$ can be determined via reduction as in Section 5.4.5 to observe the evidence $\boldsymbol{Y}^t = \mathbf{y}^t$. This results in an updated information matrix $\mathbf{K}'_j$, information vector $\mathbf{h}'_j$ and normalisation constant g' with the superscripts t left out as all elements are in terms of t:

$$\mathbf{K}'_j = \mathbf{K}_{j,\boldsymbol{XX}} \quad , \quad \mathbf{h}'_j = \mathbf{h}_{j,\boldsymbol{X}} - \mathbf{K}_{j,\boldsymbol{XY}}\mathbf{y} \quad \text{and} \quad g' = g + \mathbf{h}^T_{j,\boldsymbol{Y}}\mathbf{y} - \frac{1}{2}\mathbf{y}^T\mathbf{K}_{j,\boldsymbol{YY}}\mathbf{y}. \tag{6.20}$$

Therefore,

$$\mathbf{K}'_j = (\boldsymbol{\Sigma}_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}})^{-1} \tag{6.21}$$

and,

$$\mathbf{h}'_j = \mathbf{K}_{j,\boldsymbol{XX}}\boldsymbol{\mu}_{j,\boldsymbol{X}} + \mathbf{K}_{j,\boldsymbol{XY}}\boldsymbol{\mu}_{j,\boldsymbol{Y}} - \mathbf{K}_{j,\boldsymbol{XY}}\mathbf{y} \tag{6.22}$$

$$= \mathbf{K}_{j,\boldsymbol{XX}}\boldsymbol{\mu}_{j,\boldsymbol{X}} - \mathbf{K}_{j,\boldsymbol{XY}}(\mathbf{y} - \boldsymbol{\mu}_{j,\boldsymbol{Y}}) \tag{6.23}$$

Observed evidence in covariance form

The results can be converted back into the covariance form. The conditional distribution is the normalisation of the joint distribution. The Gaussian parameters will not change with normalisation and, with the time superscripts omitted, are,

$$P(\boldsymbol{X}^t|\mathbb{Y}^t) \approx P(\boldsymbol{X}^t, \mathbb{Y}^t) = \mathcal{N}(\boldsymbol{X}^t; \boldsymbol{\mu}', \boldsymbol{\Sigma}'), \tag{6.24}$$

where the covariance matrix $\boldsymbol{\Sigma}'$ is the inverse of the information matrix,

$$\begin{aligned}\boldsymbol{\Sigma}' &= (\mathbf{K}'_j)^{-1} && (6.25)\\ &= \boldsymbol{\Sigma}^t_{j,\boldsymbol{XX}} - \boldsymbol{\Sigma}^t_{j,\boldsymbol{XY}}(\boldsymbol{\Sigma}^t_{j,\boldsymbol{YY}})^{-1}\boldsymbol{\Sigma}_{j,\boldsymbol{YX}} && (6.26)\\ &= \mathbf{P}^t - \mathbf{P}^t\mathbf{H}^T(\mathbf{R} + \mathbf{H}^T\mathbf{P}^t\mathbf{H})^{-1}\mathbf{H}\mathbf{P}^t && (6.27)\end{aligned}$$

and the mean vector $\boldsymbol{\mu}'$ is,

$$\begin{aligned}\boldsymbol{\mu}' &= (\mathbf{K}'_j)^{-1}\mathbf{h}'_j && (6.28)\\ &= \mathbf{K}^{-1}_{j,\boldsymbol{XX}}(\mathbf{K}_{j,\boldsymbol{XX}}\boldsymbol{\mu}_{j,\boldsymbol{x}} - \mathbf{K}_{j,\boldsymbol{XY}}(\mathbf{y} - \boldsymbol{\mu}_{j,\boldsymbol{y}})) && (6.29)\\ &= \boldsymbol{\mu}_{j,\boldsymbol{X}} + \boldsymbol{\Sigma}_{j,\boldsymbol{XY}}\boldsymbol{\Sigma}^{-1}_{j,\boldsymbol{YY}}(\mathbf{y} - \boldsymbol{\mu}_{j,\boldsymbol{Y}}) && (6.30)\end{aligned}$$

reintroducing the time superscript for the last step,

$$= \boldsymbol{\mu}^t + \mathbf{P}^t\mathbf{H}^T(\mathbf{R} + \mathbf{H}^T\mathbf{P}^t\mathbf{H})^{-1}(\mathbf{y}^t - \mathbf{H}\boldsymbol{\mu}^t). \quad (6.31)$$

Noting that the Kalman gain is:

$$\mathbf{K}_g = \mathbf{P}^t\mathbf{H}^T(\mathbf{R} + \mathbf{H}^T\mathbf{P}^t\mathbf{H})^{-1}, \quad (6.32)$$

the above equations are simplified,

$$\boldsymbol{\mu}' = \boldsymbol{\mu}^t + \mathbf{K}_g(\mathbf{y}^t - \mathbf{H}\boldsymbol{\mu}^t), \quad (6.33)$$

and,

$$\boldsymbol{\Sigma}' = \mathbf{P}^t - \mathbf{K}_g\mathbf{H}\mathbf{P}^t. \quad (6.34)$$

We have just used the sum-product algorithm to determine the belief of state $\boldsymbol{X}$ given all its measurements up until time t, $P(\boldsymbol{X}^t|\mathbb{Y}^t)$. This belief will be compared with the Kalman filter update equations where it will be found to be the same.

6.2.2. Prediction update

To predict the state of the object at the next time step the $\delta_{X,X}(\boldsymbol{X}^{t+1})$ message sent between state clusters can be determined. This is done by taking all the messages received by the state cluster, and marginalising out the necessary variables. The product of $\delta_{Y,X}(\boldsymbol{X}^t)$ and $\delta_{X,X}(\boldsymbol{X}^t)$ has just been calculated and is now multiplied by $P(\boldsymbol{X}^{t+1}|\boldsymbol{X}^t)$ to construct a joint distribution.

Joint distribution using motion model

Similarly to the joint distribution constructed in the measurement update, the equation 5.29 is used to determine the joint mean vector for the prediction step $\boldsymbol{\mu}_{jp}$ and joint covariance matrix for the prediction step $\boldsymbol{\Sigma}_{jp}$.

$$P(\boldsymbol{X}^t, \boldsymbol{X}^{t+1}|\mathbb{Y}^t) = \mathcal{N}\left(\boldsymbol{\mu}_{jp} = \begin{bmatrix} \boldsymbol{\mu}' \\ \mathbf{A}\boldsymbol{\mu}' \end{bmatrix}, \boldsymbol{\Sigma}_{jp} = \begin{bmatrix} \boldsymbol{\Sigma}' & \boldsymbol{\Sigma}'\mathbf{A} \\ \mathbf{A}^T\boldsymbol{\Sigma}' & \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}'\mathbf{A}^T \end{bmatrix}\right) \tag{6.35}$$

Marginalization

Now that we have the product of all the necessary factors represented in the joint distribution, marginalization can take place to determine the output message. This is performed as in equation 5.19 resulting in $\delta_{X,X}(\boldsymbol{X}^{t+1}) = P(\boldsymbol{X}^{t+1}|\mathbb{Y}^t)$, where,

$$P(\boldsymbol{X}^{t+1}|\mathbb{Y}^t) = \mathcal{N}(\boldsymbol{X}^{t+1}; \boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1}) \tag{6.36}$$

and the mean vector is,

$$\boldsymbol{\mu}^{t+1} = \boldsymbol{\mu}_{jp,Y} = \mathbf{A}\boldsymbol{\mu}' \tag{6.37}$$

while the covariance matrix is,

$$\boldsymbol{\Sigma}^{t+1} = \boldsymbol{\Sigma}_{jp,YY} = \mathbf{Q} + \mathbf{A}^T\boldsymbol{\Sigma}'\mathbf{A}. \tag{6.38}$$

Using the sum-product algorith, it was shown that we can predict the belief of the object state at in the next time step, $P(\boldsymbol{X}^{t+1}|\mathbb{Y}^t)$. This is belief corresponds the the update equation from the Kalman filter.

6.2.3. Summary

The belief about the object state after observing evidence, $P(\boldsymbol{X}^t|\mathbb{Y}^t)$ has been determined via message passing using a cluster graph. This belief is modelled as a Gaussian $P(\boldsymbol{X}^t|\mathbb{Y}^t) \sim \mathcal{N}(\boldsymbol{X}^t; \boldsymbol{\mu}', \boldsymbol{\Sigma}')$ with

$$\boldsymbol{\mu}' = \boldsymbol{\mu}^t + \mathbf{K}_g(\mathbf{y} - \mathbf{H}\boldsymbol{\mu}^t), \tag{6.33}$$

and,

$$\boldsymbol{\Sigma}' = \mathbf{P}^t - \mathbf{K}_g\mathbf{H}\mathbf{P}^t. \tag{6.34}$$

Similarly, the predicted belief of an object $P(\boldsymbol{X}^{t+1}|\mathbb{Y}^t) \sim \mathcal{N}(\boldsymbol{X}^{t+1}; \boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1})$ has a mean vector,

$$\boldsymbol{\mu}^{t+1} = \mathbf{A}\boldsymbol{\mu}' \tag{6.37}$$

and a covariance matrix,

$$\boldsymbol{\Sigma}^{t+1} = \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}'\mathbf{A}^T \tag{6.38}$$

Table 6.1: Comparison between the Kalman Filter equations and the PGM derivations

Kalman filter equations	PGM derivations
Measurement update:	
$\hat{\boldsymbol{X}}^k = \tilde{\boldsymbol{X}}^k + \mathbf{K}(\mathbf{y}^k - \mathbf{H}\tilde{\boldsymbol{X}}^k)$	$\boldsymbol{\mu}' = \boldsymbol{\mu}^t + \mathbf{K}_g(\mathbf{y} - \mathbf{H}\boldsymbol{\mu}^t)$
$\hat{\boldsymbol{P}}^k = \tilde{\boldsymbol{P}}^k - \mathbf{K}\mathbf{H}\tilde{\boldsymbol{P}}^k$	$\boldsymbol{\Sigma}' = \mathbf{P}^t - \mathbf{K}_g\mathbf{H}\mathbf{P}^t$
Prediction update:	
$\tilde{\boldsymbol{X}}^k = \mathbf{A}\hat{\boldsymbol{X}}^{k-1}$	$\boldsymbol{\mu}^{t+1} = \mathbf{A}\boldsymbol{\mu}'$
$\tilde{\boldsymbol{P}}^k = \mathbf{A}\hat{\boldsymbol{P}}^{k-1}\mathbf{A}^T + \mathbf{Q}$	$\boldsymbol{\Sigma}^{t+1} = \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}'\mathbf{A}^T$

These beliefs are the same as the equations for the Kalman Filter from Section 3.1. This is seen in table 6.1. This model forms the basis of this project and will be expanded so that additional uncertainties in object tracking are accounted for.

6.3. Missed Detections

One type of inaccuracy in the measurement system is a missed detection. This occurs when an object is present but not detected in a time step. Looking at the Bayesian network for the Kalman filter in Figure 6.3 this can be incorporated by not performing the measurement update. If a detection is not generated for the time step $t = 1$,it will result in the cluster graph shown in 6.4.

The measurement update is skipped and so the prediction takes place using the prior state estimate $P(\boldsymbol{X}^1|\mathbb{Y}^0)$ instead of the updated observed state estimate $P(\boldsymbol{X}^1|\mathbb{Y}^1)$. In other words, when $\boldsymbol{Y}^t$ is not present the message $\delta_{X,X}(\boldsymbol{X}^{t+1})$ is calculated as such:

$$P(\boldsymbol{X}^{t+1}|\mathbb{Y}^0) = \mathcal{N}(\boldsymbol{X}^{t+1}; \boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1}) \tag{6.36}$$

where the prediction uses $\boldsymbol{\mu}' = \boldsymbol{\mu}^t$ and $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}^t$

$$\boldsymbol{\mu}^{t+1} = \mathbf{A}^T\boldsymbol{\mu}^t \tag{6.39}$$

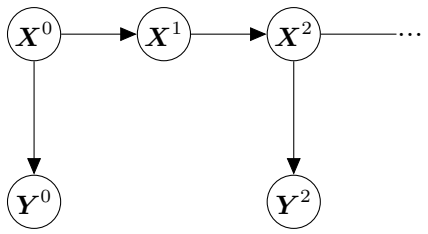


Figure 6.3: Kalman Filter with a missed detection $\boldsymbol{Y}^1$

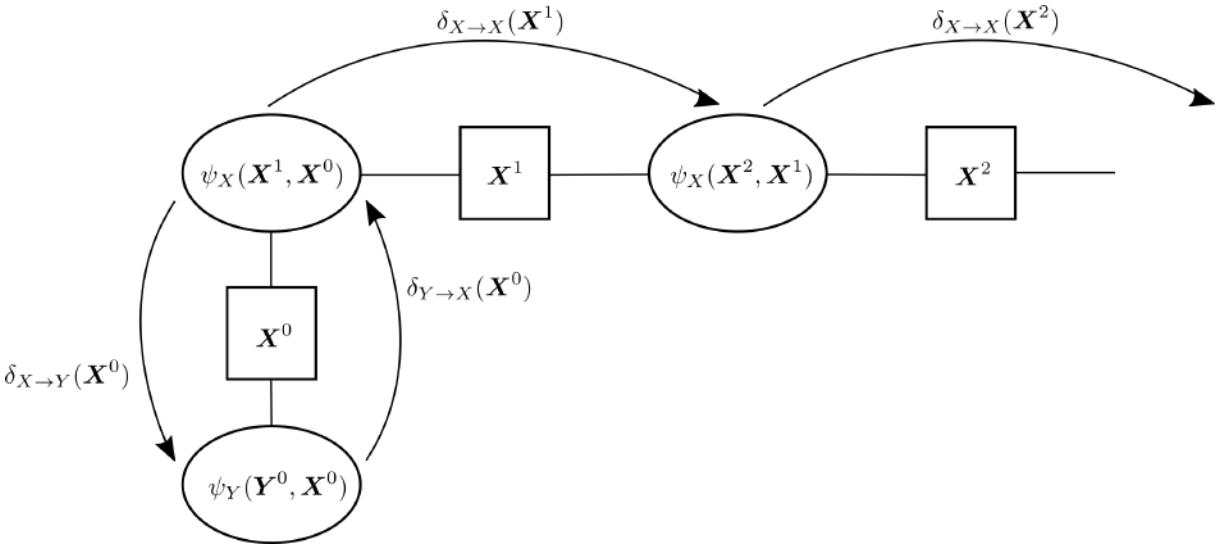


Figure 6.4: Cluster graph for tracking with missed detections

$$\boldsymbol{\Sigma}^{t+1} = \mathbf{Q} + \mathbf{A}^T \boldsymbol{\Sigma}^t \mathbf{A} \tag{6.40}$$

Missed detections result in increased uncertainty. The prediction noise $\mathbf{Q}$ is added at each time step and there is no reduction in noise due to evidence observed.

6.4. False detections

When a detection occurs that is not caused by an object it is called a false detection.To include false detections, an additional discrete random variable A is introduced which represents the association of a measurement with an object. The random variable's subscript denotes which measurement is being referred to, whereas the possible values indicate which object it is associated with. If $P(A_1 = 1) = 1$, measurement one is caused by object one. If $P(A_1 = 0) = 1$, measurement one is not caused by any already existing object and no existing object's state will be updated with this measurement. We take the approach that each false detection is caused by an additional object which is not yet being tracked. A new object $\boldsymbol{X}_{new}$ is therefore added in this model. This is a conservative approach for applications such as autonomous navigation. In these applications it is important that, if a detection is generated, the possibility that it came from a new object is included. This object tracker therefore does not distinguish between clutter being tracked and real objects being tracked. This topic is futher discussed in 7.2.3. The priors are estimated as $P(A_1 = 1) = 0.95$ and $P(A_1 = 0) = 0.05$, corresponding to a 5 percent chance of a false detection.

A new measurement factor ϕ_Y is defined where this distribution of A is incorporated, $\phi_Y = P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{X}_{new}, A)$. Note this term now contains both continuous and discrete variables and is described in Table 6.2.

Table 6.2: Measurement factor distribution

$\phi_Y(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_{new}^t, A^t)$	
$A = 1$	$P(\boldsymbol{Y}^t \vert \boldsymbol{X}_1^t)$
$A = 0$	$P(\boldsymbol{Y}^t \vert \boldsymbol{X}_{new}^t)$

A Bayesian Network incorporating the new discrete random variable A and possible new object $\boldsymbol{X}_{new}$, is created and shown in Figure 6.5.

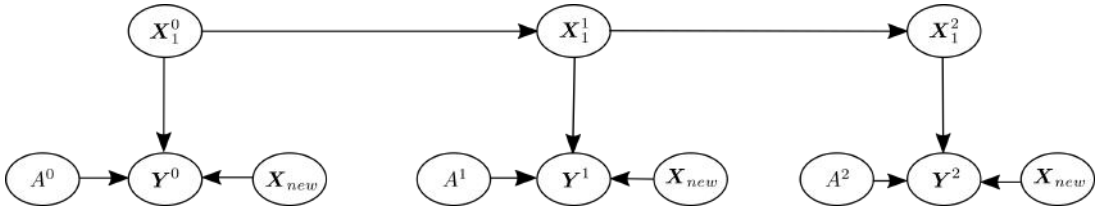


Figure 6.5: Bayesian Network incorporating false detections

6.4.1. Inference

When converting into a cluster graph the association variable is absorbed into the measurement cluster $\psi_Y(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_{new}^t, A^t) = P(\boldsymbol{Y}^t \vert \boldsymbol{X}^t, \boldsymbol{X}_{new}^t, A^t)P(A)$, resulting in Figure 6.6. As per the sum-product algorithm, the message sent from the measurement cluster to the state transition cluster is the product of all incoming messages, except for the message received from the state transition cluster, multiplied by its cluster potential and marginalised over the shared septset $S_{Y,X} = \boldsymbol{X}$. With evidence taken into account this results in $\delta_{Y\to X}(\boldsymbol{X}^t) = \sum_A \int P(\boldsymbol{Y}^t = \mathbf{y} \vert \boldsymbol{X}_1^t \boldsymbol{X}_{new}, A)P(A)P(\boldsymbol{X}_{new})d\boldsymbol{X}_{new}$, which is described in Table 6.3 for each case. When the first term is sent to the state cluster and multiplied by the prior belief of the state we arrive at the same Kalman filter update equation, weighted by $P(A = 1)$. The second term, which is a constant when marginalised, will be multiplied by the prior belief. The result is that the updated belief is a weighted

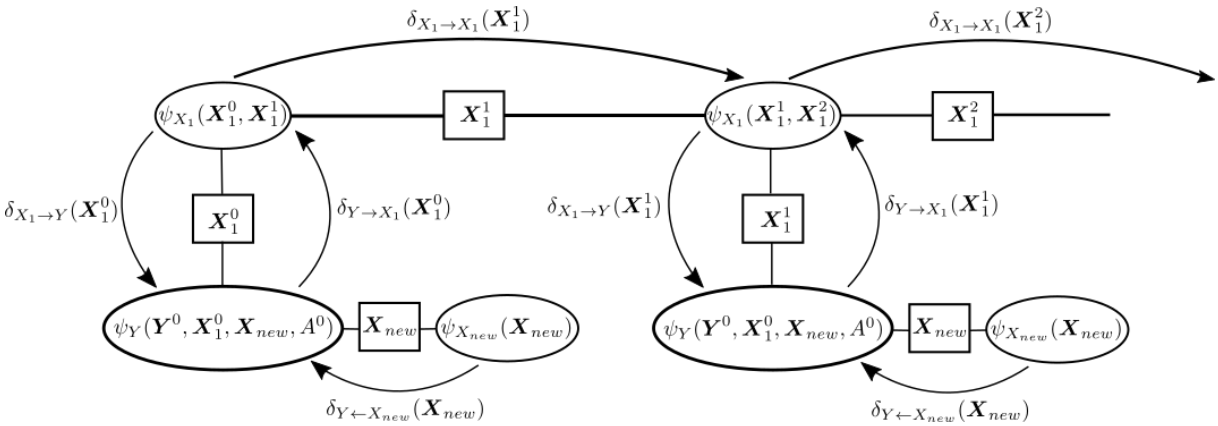


Figure 6.6: Cluster graph incorporating false detections

Table 6.3: Message sent from measurement cluster to state cluster

	$\delta_{Y\to X}(\boldsymbol{X}_1^t)$
$A=1$	$\int P(\boldsymbol{Y}^t=\mathbf{y}^t\|\boldsymbol{X}_1^t)P(A=1)P(\boldsymbol{X}_{new})d\boldsymbol{X}_{new} \approx P(\boldsymbol{Y}^t=\mathbf{y}^t\|\boldsymbol{X}_1^t)w_{update}$
$A=0$	$+\int P(\boldsymbol{Y}^t\|\boldsymbol{X}_{new}^t)P(A=0)P(\boldsymbol{X}_{new})d\boldsymbol{X}_{new} \approx w_{prior}$

Gaussian of the Kalman filter update and the prior distribution. When it is certain that an object caused a measurement this results in the same belief as the Kalman filter case, and when it is certain the measurement is not caused by the object, the same belief for a missed detections happens.

$$\begin{aligned} P(\boldsymbol{X}_1^t|\boldsymbol{Y}^t=\mathbf{y}^t,\mathbb{Y}^{t-1}) &\approx \delta_{X_1\to X_1}(\boldsymbol{X}_1^t)\delta_{Y\to X_1}(\boldsymbol{X}_1^t) \\ &\approx P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1})(P(\boldsymbol{Y}^t=\mathbf{y}^t|\boldsymbol{X}_1^t)w_{update}+w_{prior}) \\ &\approx w_{update}P(\boldsymbol{X}_1^t,\boldsymbol{Y}^t=\mathbf{y}^t)+w_{prior}P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1}) \end{aligned} \tag{6.41}$$

Estimating weights for measurement inclusion

We now estimate these two parameters using the following intuition. When a measurement is observed $\boldsymbol{Y}=\mathbf{y}$, we can compare how *close* it is to the predicted distribution of an object $\boldsymbol{X}\sim\mathcal{N}(\boldsymbol{\mu},\mathbf{P})$. However, defining the *closeness* of a vector to a distribution should be done cautiously. The scope of the measurement $\mathbf{y}=[c_x,c_y]^T$, where c_x and c_y are the x and y pixel coordinates of the detected masks' centroid, should match the scope of the object state distribution $\boldsymbol{X}$. We therefore marginalise the object state vector $\boldsymbol{X}\sim\mathcal{N}(\boldsymbol{\mu},\mathbf{P})$ to create an adjusted object distribution with the same scope as $\mathbf{y}$. The adjusted object state distribution $\boldsymbol{X}_{adj}=\mathcal{N}(\boldsymbol{\mu}_{adj},\mathbf{P}_{adj})$ contains

$$\boldsymbol{\mu}_{adj}=\begin{bmatrix}\mu_{sx}\\ \mu_{sy}\end{bmatrix} \text{ and } \quad \mathbf{P}_{adj}=\begin{bmatrix}(\sigma_{sx})^2 & 0\\ 0 & (\sigma_{sy})^2\end{bmatrix}. \tag{6.42}$$

A mahalanobis distance (MD) is a distance metric between a vector and a distribution that De Maesschalck et al. [24] applies to outlier detection. This is metric now used as the basis for determining a score between an object and a measurement. The use of the MD allows for more features to be included in the measurements in future such as in Section 7.4 where the numbers of pixels in an object mask is also measured. In general the MD can compare a vector to a distribution, and not merely two vectors, by taking into account the covariance matrix. This also allows for distributions where there are dependencies between random variables. The MD in this case is

$$MD(\mathbf{y})=\sqrt{(\mathbf{y}-\boldsymbol{\mu}_{adj})^T\mathbf{P}_{adj}^{-1}(\mathbf{y}-\boldsymbol{\mu}_{adj})}. \tag{6.43}$$

Brereton shows that MDs can be characterised by a chi squared distribution [25]. This means that an MD interval where 99 percent of measurements will be expected to fall within can be found. This is done by consulting the standard chi squared distribution table from Brereton [25] which is attached in appendix E. Using a p value of 0.01 (which corresponds to a cumulative probability of 99 percent), and 2 degrees of freedom (because there are two dimensions in this case), the chi squared critical value is found to be 9.21. Brereton explains that the MD critical value is then calculated as the square root of the chi squared value $MD = \sqrt{9.21} \approxeq 3.035$. This means that 99 percent of measurements are expected to fall within an MD of less than 3.035 from an object. A cost for a measurement not being assigned to an object is then approximated as this critical value $MD = 3.035$. The normalised cost factor for associating a measurement with an object,

$$\phi_{cost}(w_{update}) = \frac{MD(\mathbf{y})}{3.035 + MD(\mathbf{y})}, \tag{6.44}$$

and for assuming the measurement is not assigned to any existing object,

$$\phi_{cost}(w_{prior}) = \frac{3.035}{3.035 + MD(\mathbf{y})} \tag{6.45}$$

is now fully described. This cost can also be converted into a score factor $\phi_{score}(w)$ where

$$\phi_{score}(w_{update}) = 1 - \frac{MD(\mathbf{y})}{3.035 + MD(\mathbf{y})}, \tag{6.46}$$

and,

$$\phi_{score}(w_{prior}) = 1 - \frac{3.035}{3.035 + MD(\mathbf{y})} \tag{6.47}$$

Measurement update

The belief about object $\boldsymbol{X}$ is described by two Gaussians in equation 6.41, $P(\boldsymbol{X}_1^t|\boldsymbol{Y}^t = \mathbf{y}^t, \mathbb{Y}^{t-1}) \approx w_{update}P(\boldsymbol{X}_1^t, \boldsymbol{Y}^t = \mathbf{y}^t) + w_{prior}P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1})$. The first update term is drawn from a Gaussian with the mean and covariances of $\boldsymbol{\mu}_1 = \boldsymbol{\mu}' = \boldsymbol{\mu}^t + \mathbf{K}_g(\mathbf{y} - \mathbf{H}\boldsymbol{\mu}^t)$ and, $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}' = \mathbf{P}^t - \mathbf{K}_g\mathbf{H}\mathbf{P}^t$, and the weighting $w_1 = w_{update} = \phi_{score}(w_{update})$. The prior represented as a Gaussian has the parameters, $\boldsymbol{\mu}_2 = \boldsymbol{\mu}^t$ and $\boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}^t$, and $w_2 = w_{prior} = \phi_{score}(w_{prior})$. If we send the message along, the mixture of Gaussians will grow indefinitely at each measurement update and become computationally intractable. We therefore estimate this Gaussian mixture as a single Gaussian using the process described in Section 5.6. We calculate our estimated belief as,

$$P(\boldsymbol{X}_1^t|\boldsymbol{Y}^t = \mathbf{y}^t, \mathbb{Y}^{t-1}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{6.48}$$

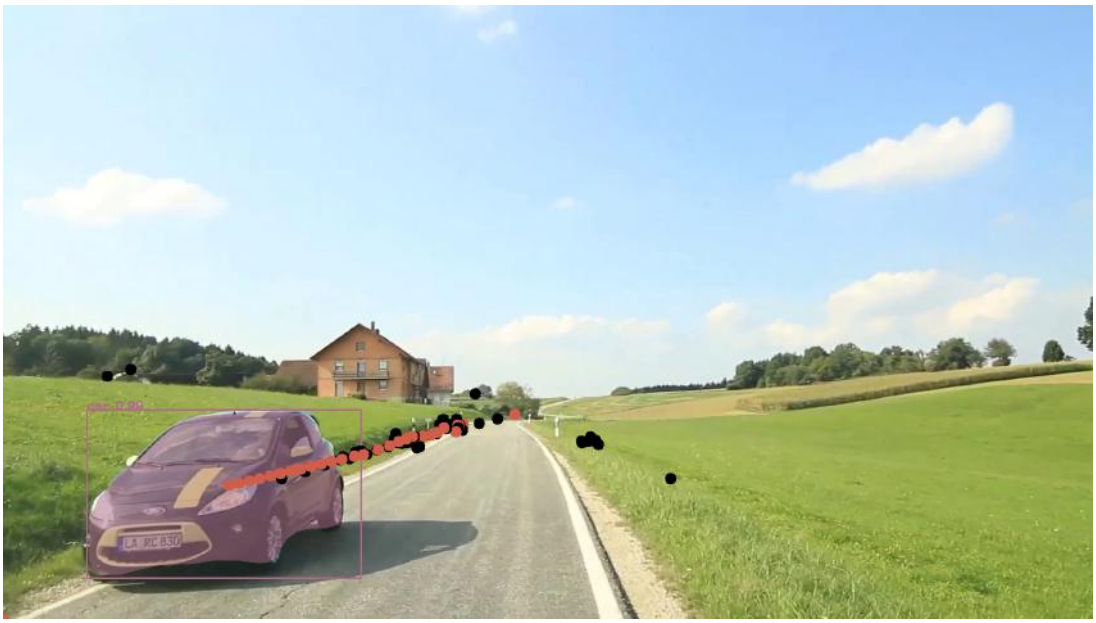


Figure 6.7: Tracking a single object: The output from the detector is the mask shown on the car. The centroids of all previous masks are plotted in black, while the predicted object location at each time step is given in orange.

1.2

with,

$$\boldsymbol{\mu} = \sum_{i=1}^{k} w_i \boldsymbol{\mu}_i \tag{6.49}$$

and,

$$\boldsymbol{\Sigma} = \sum_{i=1}^{k} w_i \boldsymbol{\Sigma}_i + \sum_{i=1}^{k} w_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})((\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \tag{6.50}$$

Prediction update

The prediction update remains exactly the same as before.

We have now defined a process for tracking a single object that contains measurement noise, dynamic uncertainty, missed detections and false detections.

6.5. Results

The algorithm developed is implemented in python. A Github repository will be created with the project file including this implementation[1]. The estimation of the measurement and state transition covariances are discussed in Section 7.3. We now show, in Figure 1.2, a single frame with the mask generated from Mask R-CNN locating the object. This has been done on each frame in the past and their centroids are shown by a black dot. The belief about the objects location at each time step is then shown by the orange dot.

[1]Github repository found at https://github.com/Daniel-De-Freitas/18345662_Skripsie.git

At each point in time the belief about the object is represented as a Gaussian. We can therefore show a confidence ellipse which is where, if we had to sample it's true location, 99 percent of the time it would fall within this green ellipse. In figure 6.8 frames of a video are shown where there are missed detectionss. When this occurs the uncertainty grows and so does the size of this ellipse. When a measurement is observed, the uncertainty is reduced.

Figure 6.8: Images showing the confidence ellipse of the object in green. When we don't have a measurement the uncertainty increases. When a measurement is received we become more certain of its location.

In Figure 6.9 an example of a false measurement is shown. The belief about the object is given by the green ellipse and the detection is given by the black dot. In this case the weighting for the measurement update would be low, and the prior would be high. False

Figure 6.9: Image showing the confidence ellipse of the object, and a false measurement

measurements and missed detections have been taken into account to successfully track the object.

Chapter 7

Multiple Object Tracking

So far we have solved the first problem with object tracking - modelling measurement and dynamic uncertainty to try and infer the true object state. This has been applied to a single object and measurement. Additionally, we have incorporated an association variable and the possibility that a detection is caused by a new object, as well as the case where an object's detection is missed. Now the association variable is adjusted, and additional objects and measurements included, to solve the second problem with multiple object tracking - the data association problem. The model previously developed is extended in Section 7.1 to track two objects with two measurements. This is then generalised and the process of initiating and deleting tracks is discussed. An update to the measurement is then added, and the results are given.

7.1. Tracking two objects with two measurements

Firstly, the model in Chapter 6 is further developed to include two objects. Afterwards an additional measurement is added and a solution is derived.

7.1.1. Modelling two object tracking with a single measurement

A score factor $\phi_{score}(A)$ describing the association between a measurement and an object is determined in a similar fashion to Section 6.4 False measurements. The difference here is that there are two objects and thus A can be either $0, 1$ or 2. A new measurement model $P(\boldsymbol{Y}|\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_{new}, \boldsymbol{A})$ is a linear Gaussian of its object parent, conditioned on the discrete association variable which can take on three different values:

$$P(\boldsymbol{Y}|\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_{new}, A = 1) = P(\boldsymbol{Y}|\boldsymbol{X}_1) = \mathcal{N}(\boldsymbol{Y}; \mathbf{H}\boldsymbol{X}_1^t, \mathbf{R}^t), \tag{7.1}$$

$$P(\boldsymbol{Y}|\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_{new}, A = 2) = P(\boldsymbol{Y}|\boldsymbol{X}_2) = \mathcal{N}(\boldsymbol{Y}; \mathbf{H}\boldsymbol{X}_2^t, \mathbf{R}^t) \tag{7.2}$$

and

$$P(\boldsymbol{Y}|\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_{new}, A = 0) = P(\boldsymbol{Y}|\boldsymbol{X}_{new}) = \mathcal{N}(\boldsymbol{Y}; \mathbf{H}\boldsymbol{X}_{new}^t, \mathbf{R}^{\mathbf{t}}). \tag{7.3}$$

A Bayesian network containing the new object is shown in Figure 7.1.

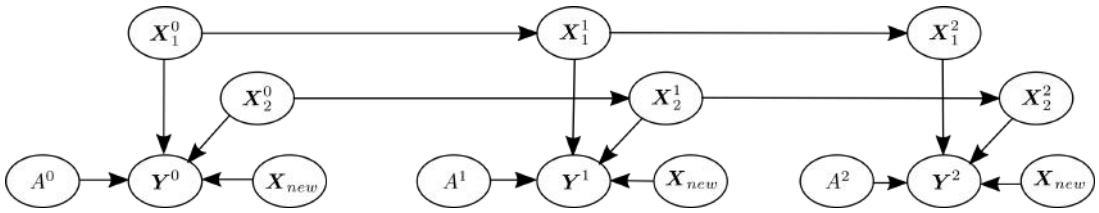


Figure 7.1: Bayesian Network for tracking two objects

The concept of plates are introduced, where a repeated structure can be put inside a plate. In this case the information for each temporal time step is the same. This will be useful when a structure is complex and repeating it can be cumbersome.

7.1.2. Inference for two objects with a single measurement

We begin by constructing the cluster graph in Figure 7.2 from the Bayesian network in Figure 7.2. The measurement cluster potential is initialised, $\psi_Y(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t) = P(\boldsymbol{Y}^t|\boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)P(A^t)$, and its distribution is represented in Table 7.1.

Table 7.1: Measurement cluster potential distribution

$\psi_Y(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)$	
$A = 1$	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_1^t, \mathbb{Y}^{t-1})P(A^t = 1)$
$A = 2$	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_2^t, \mathbb{Y}^{t-1})P(A^t = 2)$
$A = 0$	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_{new}^t, \mathbb{Y}^{t-1})P(A^t = 0)$

This cluster then receives all the messages from surrounding clusters and multiplies them together with its cluster potential to form a factor $\phi_{\boldsymbol{Y}+\text{messages}}(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t) = \psi_{\boldsymbol{Y}}(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)\delta_{\boldsymbol{X}_1\to\boldsymbol{Y}}(\boldsymbol{X}_1^t, \mathbb{Y}^{t-1})\delta_{\boldsymbol{X}_2\to\boldsymbol{Y}}(\boldsymbol{X}_2^t, \mathbb{Y}^{t-1})\delta_{\boldsymbol{X}_{new}\to\boldsymbol{Y}}(\boldsymbol{X}_{new})$. We look at each case for the discrete conditions in Table 7.2.

Table 7.2: Measurement cluster distribution with messages received

$\phi_{\boldsymbol{Y}+\text{messages}}(\boldsymbol{Y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)$	
$A = 1$:	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_1^t, \mathbb{Y}^{t-1})P(A^t = 1)P(\boldsymbol{X}_1^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_2^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_{new}^t)$
$A = 2$:	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_2^t, \mathbb{Y}^{t-1})P(A^t = 2)P(\boldsymbol{X}_1^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_2^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_{new}^t)$
$A = 0$:	$P(\boldsymbol{Y}^t\|\boldsymbol{X}_{new}^t, \mathbb{Y}^{t-1})P(A^t = 0)P(\boldsymbol{X}_1^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_2^t\|\mathbb{Y}^{t-1})P(\boldsymbol{X}_{new}^t)$

The measurement evidence can then be incorporated via reduction, $\phi_{Y+\text{messages}}(\boldsymbol{Y}^t = \mathbf{y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)$. For outgoing messages we multiply the measurement cluster potential with all its incoming messages, except for the message that comes from the cluster we are sending a message to. A factor for each message $\phi_{\boldsymbol{Y}\to\boldsymbol{X}_i}$ can be determined by dividing the factor containing the measurement cluster potential and all the messages and by the

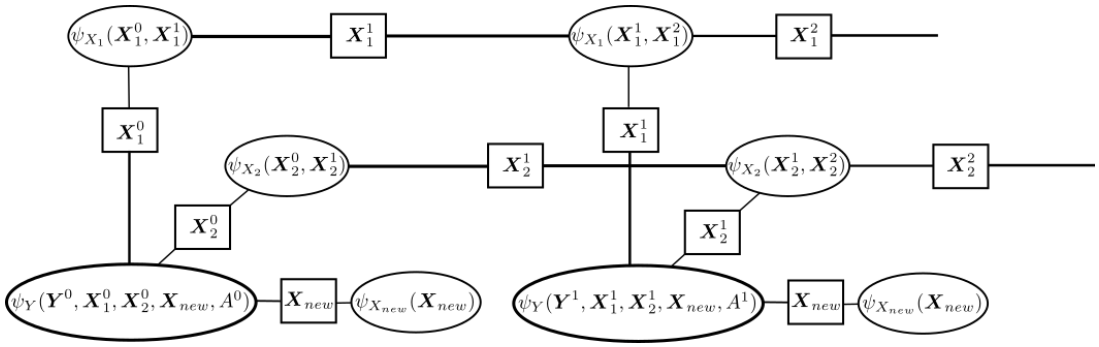


Figure 7.2: Cluster Graph for tracking two objects

message from the cluster we are sending a message to, $\phi_{\boldsymbol{Y}\to\boldsymbol{X}_i} = \phi_{Y+\text{messages}}/\delta_{\boldsymbol{X}_i\to\boldsymbol{Y}}$. The outgoing message is therefore this factor after marginalising over the scope of each sepset.

$$\delta_{\boldsymbol{Y}\to\boldsymbol{X}_1} = \sum_A \int [\phi_{(\boldsymbol{Y}\to\boldsymbol{X}_i)}(\boldsymbol{Y}^t = \mathbf{y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)] d\boldsymbol{X}_2 d\boldsymbol{X}_{new} \tag{7.4}$$

$$\delta_{\boldsymbol{Y}\to\boldsymbol{X}_2} = \sum_A \int [\phi_{\boldsymbol{Y}\to\boldsymbol{X}_2}(\boldsymbol{Y}^t = \mathbf{y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)] d\boldsymbol{X}_1 d\boldsymbol{X}_{new} \tag{7.5}$$

$$\delta_{\boldsymbol{Y}\to\boldsymbol{X}_{new}} = \sum_A \int [\phi_{\boldsymbol{Y}\to\boldsymbol{X}_{new}}(\boldsymbol{Y}^t = \mathbf{y}^t, \boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \boldsymbol{X}_{new}^t, A^t)] d\boldsymbol{X}_1 d\boldsymbol{X}_2 \tag{7.6}$$

When we sum over A we will get three terms for each message. These terms are marginalised via integration. The result is that two of these terms become constants, representing the possibility of other objects causing the measurement. The other term will become the measurement update for the standard Kalman filter weighted by $P(A)$. The process for the first message is shown:

$$\begin{aligned}
\delta_{\boldsymbol{Y}\to\boldsymbol{X}_1}(\boldsymbol{X}_1^t) = & \\
& \int [P(\boldsymbol{Y}^t = \mathbf{y}^t | \boldsymbol{X}_1^t, \mathbb{Y}^{t-1}) P(A^t = 1) P(\boldsymbol{X}_2^t | \mathbb{Y}^{t-1}) P(\boldsymbol{X}_{new}^t)] d\boldsymbol{X}_2 d\boldsymbol{X}_{new} \\
& + \int [P(\boldsymbol{Y}^t = \mathbf{y}^t | \boldsymbol{X}_2^t, \mathbb{Y}^{t-1}) P(A^t = 2) P(\boldsymbol{X}_2^t | \mathbb{Y}^{t-1}) P(\boldsymbol{X}_{new}^t)] d\boldsymbol{X}_2 d\boldsymbol{X}_{new} \\
& + \int [P(\boldsymbol{Y}^t = \mathbf{y}^t | \boldsymbol{X}_{new}^t, \mathbb{Y}^{t-1}) P(A^t = 0) P(\boldsymbol{X}_2^t | \mathbb{Y}^{t-1}) P(\boldsymbol{X}_{new}^t)] d\boldsymbol{X}_2 d\boldsymbol{X}_{new}
\end{aligned} \tag{7.7}$$

Similarly as before, we approximate this message as,

$$\delta_{\boldsymbol{Y}\to\boldsymbol{X}_1}(\boldsymbol{X}_1^t) \approx w_{update,1} P(\boldsymbol{Y}^t = \mathbf{y}^t | \boldsymbol{X}_1^t, \mathbb{Y}^{t-1}) + w_{prior,1} \tag{7.8}$$

To determine the updated belief $P(\boldsymbol{X}_1^t | \boldsymbol{Y}^t = \mathbf{y}^t, \mathbb{Y}^{t-1})$, we multiply this message by the prior message as before in the derivation of the updated belief given false detections in

Section 6.4,

$$\begin{aligned} P(\boldsymbol{X}_1^t|\boldsymbol{Y}^t = \mathbf{y}^t, \mathbb{Y}^{t-1}) &\approx \delta_{X_1 \to X_1}(\boldsymbol{X}_1)\delta_{Y \to X_1}(\boldsymbol{X}_1^t) \\ &\approx P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1})(w_{1,update}P(\boldsymbol{Y}^t = \mathbf{y}^t|\boldsymbol{X}_1^t) + w_{prior}) \\ &\approx w_{1,update}P(\boldsymbol{X}_1^t, \boldsymbol{Y}^t = \mathbf{y}^t) + w_{1,prior}P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1}) \end{aligned} \tag{7.9}$$

The measurement update can therefore be performed as before, the only difference is that there are two objects and so the other object must also be updated,

$$\begin{aligned} P(\boldsymbol{X}_2^t|\boldsymbol{Y}^t = \mathbf{y}^t, \mathbb{Y}^{t-1}) &\approx \delta_{X_2 \to X_2}(\boldsymbol{X}_2)\delta_{Y \to X_2}(\boldsymbol{X}_1^t) \\ &\approx P(\boldsymbol{X}_2^t|\mathbb{Y}^{t-1})(w_{2,update}P(\boldsymbol{Y}^t = \mathbf{y}^t|\boldsymbol{X}_2^t) + w_{2,prior}) \\ &\approx w_{2,update}P(\boldsymbol{X}_2^t, \boldsymbol{Y}^t = \mathbf{y}^t) + w_{2,prior}P(\boldsymbol{X}_2^t|\mathbb{Y}^{t-1}). \end{aligned} \tag{7.10}$$

The method for approximating weights for multiple object tracking is discussed in Section 7.2.2. The prediction update remains the same and will be performed on each object

7.2. Tracking multiple objects with multiple measurements

We include an additional measurement as shown in the Bayesian network in Figure 7.3. A simplified cluster graph is then created but without showing the $\boldsymbol{X}_{new}$ clusters in Figure 7.4.

7.2.1. Determining the updated beliefs after measurements

If we had to run inference we would arrive at an expression similar to before, only now there will be multiple measurement update terms:

$$\begin{aligned} P(\boldsymbol{X}_i^t|\boldsymbol{Y}_1^t = \mathbf{y}_1^t, \boldsymbol{Y}_2^t = \mathbf{y}_2^t, \mathbb{Y}^{t-1}) &\approx \delta_{X_i \to X_i}(\boldsymbol{X}_i)\delta_{Y_1 \to X_i}(\boldsymbol{X}_1^t)\delta_{Y_2 \to X_i}(\boldsymbol{X}_1^t) \\ &\approx w_{i,1}P(\boldsymbol{X}_1^t, \boldsymbol{Y}_1^t = \mathbf{y}_1^t) + w_{i,2}P(\boldsymbol{X}_1^t, \boldsymbol{Y}_2^t = \mathbf{y}_2^t) + w_{1,0}P(\boldsymbol{X}_1^t|\mathbb{Y}^{t-1}) \end{aligned} \tag{7.11}$$

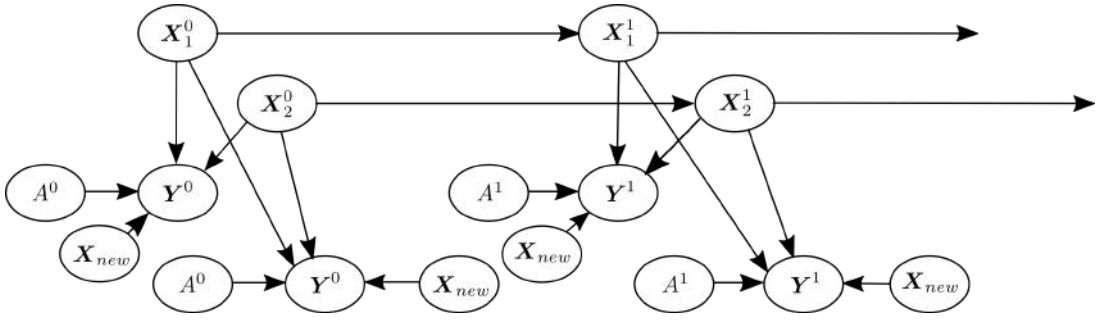


Figure 7.3: Bayesian Network for tracking two objects with two measurements

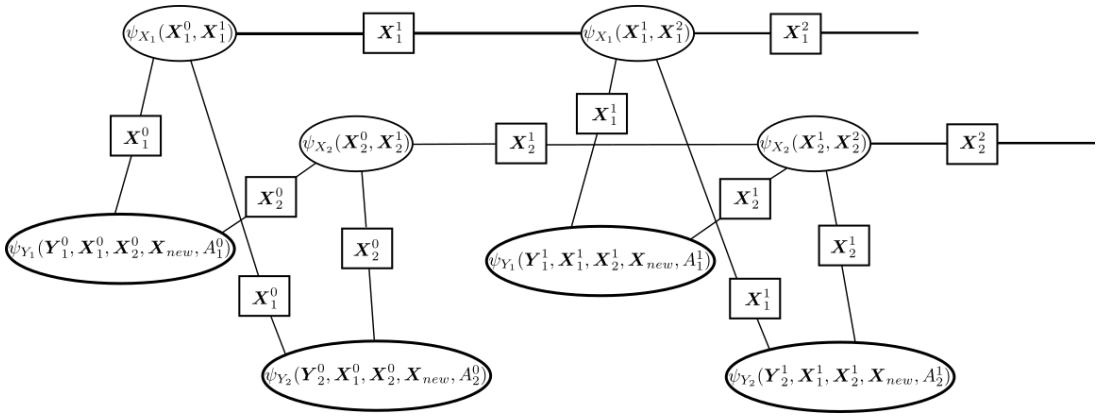


Figure 7.4: Simplified cluster graph for tracking two objects with two measurements. The $\boldsymbol{X}_{new}$ clusters are not shown

The update term for each object i, with a measurement j, is $P(\boldsymbol{X}_i^t, \boldsymbol{Y}_j^t = \mathbf{y}_j^t) \sim \mathcal{N}(\boldsymbol{\mu}_{i,j}, \boldsymbol{\Sigma}_{i,j})$ where,

$$\boldsymbol{\mu}_{i,j} = \boldsymbol{\mu}_i^t + \mathbf{K}_{g,i-j}(\mathbf{y}_j - \mathbf{H}\boldsymbol{\mu}_i^t) \tag{7.12}$$

and,

$$\boldsymbol{\Sigma}_{i,j} = \mathbf{P}_i^t - \mathbf{K}_{g,i-j}\mathbf{H}\mathbf{P}_i^t \tag{7.13}$$

The prior for each object i is, $P(\boldsymbol{X}_i^t|\mathbb{Y}^{t-1}) \sim \mathcal{N}(\boldsymbol{\mu}_{i,0}, \boldsymbol{\Sigma}_{i,0})$ where,

$$\boldsymbol{\mu}_{i,0} = \boldsymbol{\mu}_i = \boldsymbol{\mu}_i^t \tag{7.14}$$

and,

$$\boldsymbol{\Sigma}_{i,0} = \boldsymbol{\Sigma}_i = \mathbf{P}_i^t \tag{7.15}$$

The weight for an object i, updated by a measurement j, is $w_{i,j}$. The weight for the prior of each object is $w_{i,0}$. The updated belief estimated as a single Gaussian $P(\boldsymbol{X}_i^t|\mathbb{Y}^t)$, with the number of measurements equal to k is,

$$P(\boldsymbol{X}_i^t|\mathbb{Y}^t) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{7.16}$$

with,

$$\boldsymbol{\mu}_i = \sum_{n=0}^{k} w_{i,n}\boldsymbol{\mu}_{i,n} \tag{7.17}$$

and,

$$\boldsymbol{\Sigma}_i = \sum_{n=0}^{k} w_{i,n}\boldsymbol{\Sigma}_{i,n} + \sum_{n=0}^{k} w_{i,n}(\boldsymbol{\mu}_{i,n} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_{i,n} - \boldsymbol{\mu}_i)^T \tag{7.18}$$

The prediction uses this updated belief and is still unchanged. We have now fully describe our interpretation of tracking multiple object state beliefs given multiple measurements. The only aspect we have not touched on is how we estimate the weighting parameters for the measurements.

7.2.2. Weight estimation

We compare two methods for estimating the weights. The first is to take the cost factor of each measurement with each object, creating a cost matrix. We then apply the Hungarian algorithm to determine the most likely assignments. In this way each object is updated by at most one measurement, similarly to the GNN approach. Alternatively, the cost factor is converted into a score factor, and normalised for each object. These scores are used as weights. If we did not approximate the mixture of Gaussians as a single Gaussian this would be similar to MHT, where a weighted Gaussian of each possible association is created. By using a single Gaussian estimation we instead perform an approach more similar to JPDA and get an average of the weighted Gaussian.

Hungarian algorithm

The Hungarian algorithm solves the assignment problem where we must minimise the cost, in a cost matrix, of assigning each column to a row. Objects will be represented in the rows and measurements in the columns. We therefore include the possibility that each measurement is caused by $\boldsymbol{X}_{new}$ so that if a measurement is assigned to a new object, it does not prevent another measurement also being assigned to a new object. The MD for an object i compared to a measurement j is,

$$MD_{i,j}(\mathbf{y}) = \sqrt{(\mathbf{y}_j - \boldsymbol{\mu}_i)^T \mathbf{P}_i^{-1} (\mathbf{y}_j - \boldsymbol{\mu}_i)}, \tag{7.19}$$

and the MD for a new object is set to 3.035 as discussed before. The cost factor $\phi_{cost}(\mathbf{w})$, is then determined as,

$$\phi_{cost}(w_{i,j}) = MD_{i,j}(\mathbf{y_j}) \tag{7.20}$$

$$\phi_{cost}(w_{z,j}) = 3.035 \tag{7.21}$$

where z ranges from the number of objects, to the number of objects and measurements. The Hungarian algorithm takes this matrix as the input and outputs an array with $w_{i,j} = 1$ if an object caused a measurement, and $w_{i,j} = 0$ if it did not. A new object will need to be created for each measurement if $w_{n,j} = 1$.

Score matrix

The MD for an object i compared to a measurement j is, A normalised cost factor $\phi_{cost}(\mathbf{w})$, in terms of columns, for o number of objects, is then determined as,

$$\phi_{cost}(w_{i,j}) = \frac{MD_{i,j}(\mathbf{y}_j)}{3.035 + \sum_{n=1}^{k} MD_{n,j}(\mathbf{y}_n)} + 3.035 \tag{7.22}$$

$$\phi_{cost}(w_{o+1,j}) = 3.035 \tag{7.23}$$

This is then transformed into a score factor $\phi_{score}(\mathbf{w})$ equal to $\mathbf{1} - \phi_{cost}(\mathbf{w})$. These are the scores used to update the belief of an objects state. If the maximum score in a measurement column occurs at $k+1$ the measurement is likely to be caused by a new object and so a new object will be created. If the maximum score in a row is less than the maximum score in that column, i.e. the maximum score of an object being associated with a measurement, is less than that measurement being caused by a new object, it is probable that that object did not generate a detection. If the converse is true, the object did cause a detection.

7.2.3. Track maintenance

Whenever a measurement cannot be associated to an existing object it is modelled to have been caused by a new, not yet tracked object $\boldsymbol{X}_{new}$. This conservative approach ensures that any detection is taken into account by an object. This is essential in an application such as autonomous navigation where a detection not being tracked could be disastrous. When a new object is determined to have caused a measurement, as discussed in the previous section, it is instantiated with the measurement covariance and means.

$$\boldsymbol{X}_{o+1} = \boldsymbol{X}_{new,j} \sim \mathcal{N}(\boldsymbol{X}_{o+1}; \boldsymbol{mu}_j, \boldsymbol{\Sigma}_j) \tag{7.24}$$

Two additional variables $n_{detections}$, n_{misses} are included which increment each time an object generates a detection or has no detection. If an object has at least 10 detections, and experiences 10 missed detections in a row, its track is saved and no longer updated. If it has less than 10 detections it is filtered out.

7.3. Results and implementation

The algorithm developed in this work is implemented using python and uploaded to Github[1]. The correct parameter estimation procedure, for the covariance matrices representing dynamic noise and measurement noise, would be to annotate a sample of images, from various videos. The detected centroids would then be compared to the annotated true

[1]Github repository found at https://github.com/Daniel-De-Freitas/18345662_Skripsie.git

object locations to determine the measurement noise covariance. For the state transition noise, the true object state would be used as the measurement, and the predicted location in the next frame would be compared to its true location. This requires a large amount of time for annotation and in this work, constructing a proof of concept, this was not considered necessary. Instead a qualitative analysis was used. Initial parameters are chosen and then, after testing various videos, manipulated. When objects were moving too erratically the velocity variance values in the dynamic model were reduced. When a detection was not assigned to the correct object because it was too far away, the variance for the measurement location was increased.

7.3.1. Hungarian assignment tracking

The weights from the Hungarian assignment allow for one object to be assigned to one measurement. When performing object tracking on a video, as in Figure 7.5, it was noted that when an incorrect assignment occurs, it is propagated throughout the video. This is shown in the figure where the colour representing each object's track is swapped when their paths cross. This represents a negative feature of making the hard assignment decision.

7.3.2. Score factor weightings in Gaussian mixture approximation tracking

When using the score to determine weightings for a mixture of Gaussians approach, it is shown in Figure 7.6 that object tracks can converge on each other. This is due the estimation of the Gaussian mixture as a single Gaussian. If we sent the Gaussian mixture to the next time step, the mixture component corresponding to the correct assignment would gain a larger weighting as more evidence is observed. This would however, lead to intractability and so at some stage approximations or assumptions would need to be made. An example of this was shown in the hypothesis reduction in Section 3.3.

7.3.3. Occlusion noise

When an object is occluded its mask is distorted. This results in its centroid being shifted. In Figure 7.8 we see a stationary dog that is predicted to have moved to the left due to occlusion. This problem is exacerbated when an occluded object, that was moving to the right, has its mask distorted and its centroid begins shifting to the left. The velocity that we infer would be in the opposite direction to its true velocity. This occlusion inaccuracy is not handled particularly well as we have a linear assumption that says our object location is based off the previous location and a velocity.

(a) Two objects just before crossing paths **(b)** Two objects just after crossing paths

Figure 7.5: Hungarian assignment tracking: Two objects cross where the person on the right was initially being tracked by the blue dots. In the process of crossing, the objects have swapped tracks which is an assignment error.

(a) Two objects just before crossing paths **(b)** Two objects just after crossing paths

Figure 7.6: Score weighted Gaussian approximation tracking: After the paths have crossed both tracks merge.

(a) Two objects just before crossing paths **(b)** Two objects just after crossing paths

Figure 7.7: Hungarian assignment tracking with area measurement included: Each object is assigned a consist label which is represented by the same colour following their tracks.

Figure 7.8: Two separate frames showing how the dog is predicted to have moved to the left even though it is stationary. The dogs location is given by the green ellipse. A purple ellipse in the second frame is an additional object track which was the result of a false detection.

7.4. Improvements

When two objects are close to each other, and their states are similar, it can be difficult to assign the correct measurements to the obejcts. This is seen in the previous section. We now introduce an additional measurement consisting of the square root of the number of pixels in a mask. The resulting object state is:

$$\boldsymbol{\mu}^t = \begin{bmatrix} \mu_{sx} \\ \mu_{sy} \\ \mu_{vx} \\ \mu_{vy} \\ \mu_{px} \end{bmatrix}, \quad \mathbf{P}^t = \begin{bmatrix} \sigma_{sx}^2 & 0 & \sigma_{sx,vx} & 0 & 0 \\ 0 & \sigma_{sy}^2 & 0 & \sigma_{sy,vy} & 0 \\ \sigma_{vx,sx} & 0 & \sigma_{vx}^2 & 0 & 0 \\ 0 & \sigma_{vy,sy} & 0 & \sigma_{vy}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{px}^2 \end{bmatrix}. \tag{7.25}$$

There is now an additional dimension, and so when calculating the MD for a false detection, a p value of 0.01 and three degrees of freedom is now used. This changes the critical MD value to 3.368. Other than this change, we can now track as before. Applying this extra measurement to the the Gaussian mixture approximation tracking in Section 7.3.2 did not yield any improved results and the object tracks still converged. However, when applying this to the Hungarian assignment tracking, the correct assignments were now made as shown in Figure 7.7. This improvement is due to the number of pixels in the occluded object being much smaller than the object in front. This additional pixel measurement now allows the correct assignment to be made.

7.5. Further development

Incorporating occlusion

To better incorporate the occlusion distortion in 7.3.3, it is suggested that the measurement and state transition model be updated to include an additional random variable that represents whether or not there is occlusion. When it is likely that there is occlusion, the measurement and prediction noise should increase. Additionally, one could encode the fact that when occlusion takes place, the inferred velocity is inaccurate and a prediction should be taken using the objects inferred velocity from before occlusion.

Taking a non linear approach

A non linear motion model could also be defined so that the object tracker can perform in non-linear environments. This can be done via implementing an extended Kalman filter.

Method for propagating Gaussian mixtures

A method for sending Gaussian mixtures without the degree of simplification used in this project would greatly improve its performance. When the motion update takes place, the predicted location should be represented as a mixture of Gaussians and not simply a single Gaussian. A method for routine Gaussian mixture reduction would need to be implemented to ensure tractability.

Changing the detection method for greatly applicability

Lastly, the detection method used could be swapped out with a new one. For example, if one trained Mask R-CNN to detect certain objects in certain environments, such as astrological objects briefly mentioned in Chapter 2.3, this tracking application could determine the past trajectory of planets or stars. Alternatively, YOLACT from Section 2.2 could be implemented to detect in real time. It would then need to be determined if this tracking algorithm could be performed in real time.

Chapter 8

Summary and Conclusion

In Chapter 2 we showed how instance segmentation can be used to generate a detection for an object and selected an appropriate neural network, Mask R-CNN. We then implemented this in python to generate measurements, thus completing our first objective.

We then modelled the uncertainty in our measurements due to various sources of noise in a measurement model. A motion model was created which also takes into account our uncertainty of the object motion. These uncertainties are incorporated into a Bayesian network in Chapter 6 to represent our knowledge of the system. A cluster graph is constructed and inference performed using the sum-product algorithm. The result is a probabilistic, principled approach that allowed us to successfully track a single object, in the presence of measurement noise and dynamic uncertainty which was our second objective.

In Chapter 7 we extended this single object tracking to include multiple measurements and objects. Two different methods were compared for solving the data association problem. The first method used the Hungarian algorithm to solve an assignment problem, and the second uses a Gaussian mixture with weightings determined by a factor containing Mahalanobis distances. This Gaussian mixture is then estimated as a single Gaussian. We then improved our tracking by incorporating new evidence in our measurements, the amount of pixels in a mask. This allowed us to successfully track two objects which cross paths. Ultimately we showed that our third objective, to solve the data association problem was performed successfully.

We then briefly examine possible improvements such as; incorporating occlusions more effectively, methods to mitigate the loss of information with Gaussian mixture approximations, the use of a non linear motion model, the effect of changing the method of detection for improved applicability, and the possibility of real time tracking.

This project has therefore shown a proof of concept, that the output of instance segmentation combined with PGMs, can perform robust multiple object tracking.

Bibliography

[1] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, 2006.

[3] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *CoRR*, vol. abs/1704.06857, pp. 1–23, 2017. [Online]. Available: http://arxiv.org/abs/1704.06857

[4] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[5] K. Granström, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview, and applications," *Journal of Advances in Information Fusion*, vol. 12, no. 2, pp. 139–174, 2017.

[6] A. S. Jalal and V. Singh, "The state-of-the-Art in visual object tracking," *Informatica (Slovenia)*, vol. 36, no. 3, pp. 227–248, 2012.

[7] A. M. Hafiz and G. M. Bhat, "A survey on instance segmentation: state of the art," *International Journal of Multimedia Information Retrieval*, 2020.

[8] H. Kumar. Quick intro to instance segmentation: Mask r-cnn. [Online]. Available: https://kharshit.github.io/blog/2019/08/23/quick-intro-to-instance-segmentation

[9] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[10] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO:

common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[12] C. J. Burke, P. D. Aleo, Y. C. Chen, X. Liu, J. R. Peterson, G. H. Sembroski, and J. Y. Y. Lin, "Deblending and classifying astronomical sources with Mask R-CNN deep learning," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 3, pp. 3952–3965, 2019.

[13] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific Reports*, vol. 10, no. 1, pp. 1–7, 2020. [Online]. Available: http://dx.doi.org/10.1038/s41598-020-61808-3

[14] Z. M. Chen, "Efficient Multi-Target Tracking Using Graphical Models," *Review Literature And Arts Of The Americas*, 2008.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960.

[16] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, p. 52–57, Mar. 2002. [Online]. Available: https://doi.org/10.1145/504729.504754

[17] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," *Proceedings - 8th International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2015*, pp. 74–77, 2016.

[18] X. Rong Li, "Tracking in clutter with nearest neighbor filters: analysis and performance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 3, pp. 995–1010, 1996.

[19] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1 II, pp. 5–18, 2004.

[20] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, 1980, pp. 807–812.

[21] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[22] L. E. Sucar, "Probabilistic graphical models," *Advances in Computer Vision and Pattern Recognition. London: Springer London. doi*, vol. 10, pp. 978–1, 2015.

[23] D. Barber, "Bayesian Reasoning and Machine Learning (DRAFT: Nov 9, 2017)," p. 690, 2017. [Online]. Available: http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.HomePage

[24] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.

[25] R. G. Brereton, "The chi squared and multinormal distributions," *Journal of Chemometrics*, vol. 29, no. 1, pp. 9–12, 2015.

[26] R. W. Cottle, "Manifestations of the schur complement," *Linear Algebra and its Applications*, vol. 8, no. 3, pp. 189 – 211, 1974. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0024379574900664

Appendix A

Techno-economic analysis

A.1. Project planning and execution

A Gantt chart shown in Figure A.1 shows how the project was planned, and how it ended up being executed. We see that the review of probabilistic techniques and representing our model as a PGM ran over time. However, the implementation of the pre-trained neural network came was faster than the baseline. Overall the object was completed on in the predicted time. This is an appendix.

A.2. Budget

From Figures A.2 and A.3 we see that the actual cost was exactly the same as the estimated costs. This is because we spent the exact same amount of time as we had predicted for the overall project and this project did not require any additional costs other than time.

A.3. Technical impact

The algorithm developed could be applied to many industries where an objects location is required. This form of tracking would allow mechatronic systems to use this measurement in determining if an action should be taken. This can be done without needing to spend large amounts of time on training neural networks for tracking.

A.4. Return on investment

By saving time on training, and being able to apply various detection methods, this tracker can be applied to multiple different applications. The only cost for investment was time.

A.5. Commercialisation

The work here is not yet ready for commercialisation but it is shown how it could be further improved so that it could be commercialised.

Figure A.1: Project planning represented a Gantt chart

Activity	**Hours**	**Junior engineer cost [R]** (@ R400/Hr)
Review neural network literature	25	R10 000,00
Review probabilistic techniques literature	50	R20 000,00
Determine applicability of probabilistic techniques and instance segmentation	35	R14 000,00
Implement pretrained neural network-based instance segmentation	145	R58 000,00
Model the object tracking problem	150	R60 000,00
Represent the model as a PGM	95	R38 000,00
Perform inference	265	R106 000,00
Review algorithm design	35	R14 000,00
Update algorithm deisgn	160	R64 000,00
Finalize algorithm	55	R22 000,00
Compare results	20	R8 000,00
Finalize report	135	R54 000,00
Total	**1170**	**R468 000,00**

Figure A.2: Estimated budget

Activity	**Hours**	**Junior engineer cost [R]** (@ R400/Hr)
Review neural network literature	40	R16 000,00
Review probabilistic techniques literature	100	R40 000,00
Determine applicability of probabilistic techniques and instance segmentation	35	R14 000,00
Implement pretrained neural network-based instance segmentation	75	R30 000,00
Model the object tracking problem	140	R56 000,00
Represent the model as a PGM	75	R30 000,00
Perform inference	300	R120 000,00
Review algorithm design	25	R10 000,00
Update algorithm deisgn	125	R50 000,00
Finalize algorithm	60	R24 000,00
Compare results	60	R24 000,00
Finalize report	135	R54 000,00
Total	**1170**	**R468 000,00**

Figure A.3: Estimated budget

Appendix B

Safety report

B.1. Overview of Testing

Robust object tracking using neural-network-based instance segmentation testing requires the analysis of real life video footage with single and multiple objects in motion. Videos are required to suitably test the software's ability to identify and track objects. The ability of the developed code to accurately identify and track the path of objects in the videos is tested.

B.2. Equipment Required

Equipment	**Quantity**
Personal Computer (PC)	1
Camera	1
SD Card	1

B.3. General Lab Safety

The following general lab safety instructions are applicable:

- No after hours testing may be performed without the necessary permissions.
- An induction is required before testing may be undertaken.
- Closed shoes must be worn at all times.
- Loose clothing may not be worn.
- Remove jewellery, metal watches, or other metal accessories while using lab equipment, as these can be dangerous in the vicinity of electrical connections.
- Good housekeeping practices should be kept during testing.
- No food or drink is permitted in the lab.
- Safety report must be visible and accessible during testing.
- If uncertain, ask for help – it will be willingly provided.

B.4. Activity Based Risk Assessment

Activity	**Risk**	**Risk Type**	**Mitigating Steps**
Entering the office	Hand injury from door	P	Be careful not to get your hand stuck in the door.
Turning PC on	Electrical shock	P	Inspect power cords to ensure there are no exposed wiring.
Movement in office	Tripping	P/E	Be aware of surrounding equipment. Do not trip over power cables.
Working with equipment on a desk	Equipment falling and being damaged	P/E	Ensure all lab equipment is placed near the center of the counter to prevent them from falling.
Recording video footage	Dropping the camera	E	Place a camera strap around your neck during filming.
Backing up data	Data loss	P	Store data in a Google Drive Cloud and ensure that sufficient independent backups are kept.
Storing PC and camera	Damage or theft	E	Store equipment in a locked room/cupboard.

Appendix C

Variable Elimination

$$
\begin{aligned}
P(L) &= \sum_{D,I,G,S} P(D,I,G,L,S) && \text{(C.1a)}\\
&= \sum_{D,I,G,S} P(D)P(I)P(G|D,I)P(L|G)P(S|I) && \text{(C.1b)}\\
&= \sum_{D,I,G} P(D)P(G|D,I)P(L|G)\sum_{S} P(S|I)P(I) && \text{(C.1c)}\\
&= \sum_{D,I,G} P(D)P(G|D,I)P(L|G)\sum_{S} \psi_1(S,I) && \text{(C.1d)}\\
&= \sum_{D,I,G} P(L|G)P(G|D,I)P(D)\delta_{12}(I) && \text{(C.1e)}\\
&= \sum_{G} P(L|G)\sum_{D,I} P(G|D,I)P(D)\delta_{12}(I) && \text{(C.1f)}\\
&= \sum_{G} P(L|G)\sum_{D,I} \psi_2(G,D,I)\delta_{12}(I) && \text{(C.1g)}\\
&= \sum_{G} P(L|G)\delta_{23}(G) && \text{(C.1h)}\\
&= \sum_{G} \psi_3(L,G)\delta_{23}(G) && \text{(C.1i)}
\end{aligned}
$$

$$P(G) = \sum_{D,I,L,S} P(D,I,G,L,S) \tag{C.2a}$$
$$= \sum_{D,I,L,S} P(D)P(I)P(G|D,I)P(L|G)P(S|I) \tag{C.2b}$$
$$= \sum_{D,I,L} P(D)P(G|D,I)P(L|G)\sum_{S} P(S|I)P(I) \tag{C.2c}$$
$$= \sum_{D,I,L} P(D)P(G|D,I)P(L|G)\sum_{S} \psi_1(S,I) \tag{C.2d}$$
$$= \sum_{L} P(L|G)\sum_{D,I} P(G|D,I)P(D)\delta_{12}(I) \tag{C.2e}$$
$$= \sum_{L} P(L|G)\sum_{D,I} \psi_2(G,D,I)\delta_{12}(I) \tag{C.2f}$$
$$= \sum_{L} P(L|G)\delta_{23}(G) \tag{C.2g}$$
$$= \sum_{L} \psi_3(L,G)\delta_{23}(G) \tag{C.2h}$$

$$P(G) = \sum_{D,I,L,S} P(D,I,G,L,S) \tag{C.3a}$$
$$= \sum_{D,I,L,S} P(D)P(I)P(G|D,I)P(L|G)P(S|I) \tag{C.3b}$$
$$= \sum_{D,I,L} P(D)P(G|D,I)P(L|G)\sum_{S} P(S|I)P(I) \tag{C.3c}$$
$$= \sum_{D,I,L} P(D)P(G|D,I)P(L|G)\sum_{S} \psi_1(S,I) \tag{C.3d}$$
$$= \sum_{D,I} P(G|D,I)P(D)\sum_{L} P(L|G)\delta_{12}(I) \tag{C.3e}$$
$$= \sum_{D,I} P(G|D,I)P(D)\sum_{L} \psi_3(L,G)\delta_{12}(I) \tag{C.3f}$$
$$= \sum_{D,I} P(G|D,I)P(D)\delta_{32}(G)\delta_{12}(I) \tag{C.3g}$$
$$= \sum_{D,I} \psi_2(G,D,I)\delta_{32}(G)\delta_{12}(I) \tag{C.3h}$$

$$P(D) = \sum_{I,G,L,S} P(D,I,G,L,S) \tag{C.4a}$$
$$= \sum_{I,G,L,S} P(D)P(I)P(G|D,I)P(L|G)P(S|I) \tag{C.4b}$$
$$= \sum_{I,G,L} P(D)P(G|D,I)P(L|G)\sum_{S} P(I)P(S|I) \tag{C.4c}$$
$$= \sum_{I,G,L} P(D)P(G|D,I)P(L|G)\sum_{S} \psi_1(S,I) \tag{C.4d}$$
$$= \sum_{I,G,L} P(D)P(G|D,I)P(L|G)\delta_{12}(I) \tag{C.4e}$$
$$= \sum_{I,G} P(D)P(G|D,I)\sum_{L} P(L|G)\delta_{12}(I) \tag{C.4f}$$
$$= \sum_{I,G} P(D)P(G|D,I)\sum_{L} \psi_3(L,G)\delta_{12}(I) \tag{C.4g}$$
$$= \sum_{I,G} P(D)P(G|D,I)\delta_{32}(G)\delta_{12}(I) \tag{C.4h}$$
$$= \sum_{I,G} \psi_2(G,D,I)\delta_{32}(G)\delta_{12}(I) \tag{C.4i}$$

$$P(I) = \sum_{D,G,L,S} P(D,I,G,L,S) \tag{C.5a}$$
$$= \sum_{D,G,L,S} P(D)P(I)P(G|D,I)P(L|G)P(S|I) \tag{C.5b}$$
$$= \sum_{D,G,L} P(D)P(G|D,I)P(L|G)\sum_{S} P(I)P(S|I) \tag{C.5c}$$
$$= \sum_{D,G,L} P(D)P(G|D,I)P(L|G)\sum_{S} \psi_1(S,I) \tag{C.5d}$$
$$= \sum_{D,G,L} P(D)P(G|D,I)P(L|G)\delta_{12}(I) \tag{C.5e}$$
$$= \sum_{D,G} P(D)P(G|D,I)\sum_{L} P(L|G)\delta_{12}(I) \tag{C.5f}$$
$$= \sum_{D,G} P(D)P(G|D,I)\sum_{L} \psi_3(L|G)\delta_{12}(I) \tag{C.5g}$$
$$= \sum_{D,G} P(D)P(G|D,I)\delta_{32}(G)\delta_{12}(I) \tag{C.5h}$$
$$= \sum_{D,G} \psi_2(G,D,I)\delta_{32}(G)\delta_{12}(I) \tag{C.5i}$$

$$P(I) = \sum_{D,G,L,S} P(D, I, G, L, S) \tag{C.6a}$$
$$= \sum_{D,G,L,S} P(D)P(I)P(G|D, I)P(L|G)P(S|I) \tag{C.6b}$$
$$= \sum_{D,G,S} P(D)P(I)P(G|D, I)P(S|I) \sum_{L} P(L|G) \tag{C.6c}$$
$$= \sum_{D,G,S} P(D)P(I)P(G|D, I)P(S|I) \sum_{L} \psi_3(L, G) \tag{C.6d}$$
$$= \sum_{D,G,S} P(D)P(I)P(G|D, I)P(S|I)\delta_{32}(G) \tag{C.6e}$$
$$= \sum_{S} P(I)P(S|I) \sum_{D,G} P(D)P(G|D, I)\delta_{32}(G) \tag{C.6f}$$
$$= \sum_{S} P(I)P(S|I) \sum_{D,G} \psi_2(G, D, I)\delta_{32}(G) \tag{C.6g}$$
$$= \sum_{S} P(I)P(S|I)\delta_{21}(I) \tag{C.6h}$$
$$= \sum_{S} \psi_1(S, I)\delta_{21}(I) \tag{C.6i}$$

$$P(S) = \sum_{D,I,G,L} P(D, I, G, L, S) \tag{C.7a}$$
$$= \sum_{D,I,G,L} P(D)P(I)P(G|D, I)P(L|G)P(S|I) \tag{C.7b}$$
$$= \sum_{D,I,G,L} P(D)P(I)P(G|D, I)P(S|I) \sum_{L} P(L|G) \tag{C.7c}$$
$$= \sum_{D,I,G} P(D)P(I)P(G|D, I)P(S|I) \sum_{L} \psi_3(L, G) \tag{C.7d}$$
$$= \sum_{D,I,G} P(D)P(I)P(G|D, I)P(S|I)\delta_{32}(G) \tag{C.7e}$$
$$= \sum_{I} P(S|I) \sum_{D,G} P(D)P(I)P(G|D, I)\delta_{32}(G) \tag{C.7f}$$
$$= \sum_{I} P(S|I) \sum_{D,G} \psi_2(G, D, I)\delta_{32}(G) \tag{C.7g}$$
$$= \sum_{I} P(S|I)\delta_{21}(I) \tag{C.7h}$$
$$= \sum_{I} \psi_1(S, I)\delta_{21}(I) \tag{C.7i}$$

Appendix D

Matrix Lemma Inversion

Given a matrix $\boldsymbol{M}$:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} \tag{D.1}$$

When it is square and non singular, with $\boldsymbol{A}$ and $\boldsymbol{D}$ nonsingular, then using Schurs complement the inverse can be written as [26]:

$$\boldsymbol{M}^{-1} = \begin{pmatrix} (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1} & -(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \end{pmatrix} \tag{D.2}$$

In the context of covariant to canonical conversions:

$$\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_{a,a} & \boldsymbol{\Sigma}_{a,b} \\ \boldsymbol{\Sigma}_{b,a} & \boldsymbol{\Sigma}_{b,b} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{K}_{a,a} & \boldsymbol{K}_{a,b} \\ \boldsymbol{K}_{b,a} & \boldsymbol{K}_{b,b} \end{pmatrix} = \boldsymbol{K} \tag{D.3}$$

so,

$$\boldsymbol{K}_{a,a} = (\boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a})^{-1} \tag{D.4}$$

$$\boldsymbol{K}_{a,b} = -(\boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a})^{-1}\boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1} \tag{D.5}$$

$$\boldsymbol{K}_{b,a} = -\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a}(\boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a})^{-1} \tag{D.6}$$

$$\boldsymbol{K}_{b,b} = \boldsymbol{\Sigma}_{b,b}^{-1} + \boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a}(\boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a})^{-1}\boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1} \tag{D.7}$$

Appendix E

Chi squared distribution table

The standard Chi squared critical values table from [25] is given below:

Table I. Critical points of the Chi squared distribution

	0.99	0.975	0.95	0.9	0.1	0.05	0.025	0.01
1	0.00016	0.00098	0.00393	0.01579	2.706	3.841	5.024	6.635
2	0.0201	0.0506	0.1026	0.2107	4.605	5.991	7.378	9.21
3	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345
4	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277
5	0.554	0.831	1.145	1.61	9.236	11.07	12.832	15.086
6	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812
7	1.239	1.69	2.167	2.833	12.017	14.067	16.013	18.475
8	1.647	2.18	2.733	3.49	13.362	15.507	17.535	20.09
9	2.088	2.7	3.325	4.168	14.684	16.919	19.023	21.666
10	2.558	3.247	3.94	4.865	15.987	18.307	20.483	23.209
11	3.053	3.816	4.575	5.578	17.275	19.675	21.92	24.725
12	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217
13	4.107	5.009	5.892	7.041	19.812	22.362	24.736	27.688
14	4.66	5.629	6.571	7.79	21.064	23.685	26.119	29.141
15	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578
20	8.26	9.591	10.851	12.443	28.412	31.41	34.17	37.566
25	11.524	13.12	14.611	16.473	34.382	37.652	40.646	44.314
30	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892
40	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691
50	29.707	32.357	34.764	37.689	63.167	67.505	71.42	76.154
100	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807

The rows represent the degrees of freedom and the columns the probability levels, the cells of the values of chi squared or the squared Mahalanobis distance. The probabilities are conventionally represented by *p*-values.

Figure E.1: Critical points of the Chi squared distribution