Introduction to Graphical Models with an Application in Finding Coplanar Points

by

Jeanne-Marié Roux

Thesis presented in partial fulfilment of the requirements for the degree of

Master of Science in Applied Mathematics

at Stellenbosch University

Supervisor: Dr KM Hunter

Co-supervisor: Prof BM Herbst

March 2010

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF. The financial assistance of the NRF was used for living costs, and not to produce IP.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

This thesis provides an introduction to the statistical modeling technique known as graphical models. Since graph theory and probability theory are the two legs of graphical models, these two topics are presented, and then combined to produce two examples of graphical models: Bayesian Networks and Markov Random Fields. Furthermore, the max-sum, sum-product and junction tree algorithms are discussed. The graphical modeling technique is then applied to the specific problem of finding coplanar points in stereo images, taken with an uncalibrated camera. Although it is discovered that graphical models might not be the best method, in terms of speed, to use for this appliation, it does illustrate how to apply this technique in a real-life problem.

Uittreksel

Hierdie tesis stel die leser voor aan die statistiese modelerings-tegniek genoemd grafiese modelle. Aangesien grafiek teorie en waarskynlikheidsleer die twee bene van grafiese modelle is, word hierdie areas aangespreek en dan gekombineer om twee voorbeelde van grafiese modelle te vind: Bayesian Netwerke en Markov Lukrake Liggaam. Die maks-som, som-produk en aansluitboom algoritmes word ook bestudeer. Nadat die teorie van grafiese modelle en hierdie drie algoritmes afgehandel is, word grafiese modelle dan toegepas op 'n spesifieke probleem—om punte op 'n gemeenskaplike vlak in stereo beelde te vind, wat met 'n ongekalibreerde kamera geneem is. Alhoewel gevind is dat grafiese modelle nie die optimale metode is om punte op 'n gemeenskaplike vlak te vind, in terme van spoed, word die gebruik van grafiese modelle wel ten toongestel met hierdie praktiese voorbeeld.

Acknowledgements

I would firstly like to thank both my supervisors, Dr KM Hunter and Prof BM Herbst, for all their support, encouragement and guidance throughout the years that I was in the Applied Mathematics Department at the University of Stellenbosch.

MIH SWAT for the MIH Medialab, and all the opportunities that has come with it—opening our minds to new ideas and avenues.

NRF for funding—helping so many postgraduate students fulfil their studies (and dreams).

Dr Gert-Jan van Rooyen and Dr Herman Engelbrecht for not only being the supervisors of the MIH Medialab, but creating an environment full of different opportunities for us all, seeing to our needs, and being really cool people at that as well.

A big thanks to my fellow labrats—you guys are the best. Thank you not only for all the laughs, sarcasm, support, fixing my computer when I break it (and that has happened way too many times), but also for the really cool friends that you have become. A special thanks to Leendert Botha—I could not have asked for a better friend in a lab environment. Thank you for being there to take my mind off things when I needed a break, for protecting me from the big baddies and for being there when I just needed to bounce ideas off.

My friends—old and new—for good times, times that have formed me and moulded me. For being my family away from home. A special thanks goes to Ina Nel who has, through all the varsity years been there, supporting me and being a true friend. Also a special thanks to everyone from my cell in Somerset West—who have joined me in prayer so many times. A special thanks also to Rani van Wyk who has always been a friend that I can be completely honest with and who has always been such an example, and model of what a woman of GOD really is. And for being there, in the background, when I go on trips, covering me in prayer at all times, and giving it to me straight. Thanks too to Francesca Mountfort—though you've been far away for a year, I know I've always been able to press on your button, and you kept me from freaking out a couple of times.

I would also like to thank my entire family for their support, encouragement, love and also pushing me to greater things. The way you all have always believed in me, even though I've doubted myself so many times, have kept me going. Thank you especially to my mother, who I've always been able to phone at whatever time and share either laughter or tears.

Thank you for everything you have done, words will never explain it [I could write an entire thesis on it, and it would be longer than this one]. Thank you to all my dads—I love you more than I can describe, and the ways in which you taught me I will treasure. The lessons I have learnt from you all are deeply ingrained in me.

My dad always taught me when I was small to leave the best, and most important, till last. And so, I come to the most important. Thank you to my Father in heaven, Jesus Christ, the Holy Spirit. All honour and glory goes to You, for it is You alone that have guided me in all these things. You are my Love, my desire, the reason for my life. Thank You for leading me on the path that I have walked—a path on which I studied what I love, I enjoyed it, and I finished on time, no matter how long it took. Thank You for the strength to carry on, and the joy that surpasses all understanding. Thank You for being my life. Thank You for the cross.

Contents

1 Introduction **1**

2 Probability Theory **3**
- 2.1 Notation . . . 3
- 2.2 Basic Properties and Rules . . . 4
- 2.3 Probabilistic Inference . . . 5
- 2.4 Conditional Independence . . . 6
- 2.5 Memory and Computation Complexity . . . 8

3 Graphical Models **11**
- 3.1 Notation and Basic Graph Theory . . . 11
  - 3.1.1 Directed Graphs . . . 12
  - 3.1.2 Undirected Graphs . . . 13
- 3.2 Introduction to Graphical Models . . . 13
- 3.3 Bayesian Networks . . . 15
  - 3.3.1 From Probability Theory to Directed Graphs . . . 15
  - 3.3.2 Local Functions . . . 16
  - 3.3.3 From Directed Graphs to Probability Theory . . . 18
  - 3.3.4 Characterisation of Directed Graphical Models . . . 23
- 3.4 Markov Random Fields . . . 25
  - 3.4.1 From Undirected Graphs to Probability Theory . . . 26
  - 3.4.2 From Probability Theory to Undirected Graphs . . . 27
  - 3.4.3 Characterisation of Undirected Graphical Models . . . 30
- 3.5 Converting a Directed Graph to an Undirected Graph . . . 30
- 3.6 Local Messages from Distributive Law . . . 33

4 Algorithms **39**
- 4.1 Factor Graphs . . . 39
- 4.2 Sum-product Algorithm . . . 41
  - 4.2.1 Chain Example . . . 45
  - 4.2.2 Tree Example . . . 47
- 4.3 Max-sum Algorithm . . . 48
- 4.4 Junction Tree Algorithm . . . 52

4.4.1 Chordal Graphs 52
4.4.2 Clique Trees 53
4.4.3 Junction trees 55
4.4.4 Defining Potentials of a Junction Tree 55
4.4.5 Message Passing in Junction Tree 56
4.4.6 The Hugin Algorithm 61

5 Application 62
5.1 Transformations 62
5.1.1 Euclidean Transformation 63
5.1.2 Similarity Transformation 63
5.1.3 Affine Transformation 64
5.1.4 Projective Transformation 66
5.2 Graph Representing Variables and Interactions 66
5.3 Potentials Used in Model 68
5.4 Numerical Experiments Performed 68
5.4.1 Energy function: Euclidean distance 69
5.4.2 Potential function: Correlation 71
5.4.3 Potential function: Euclidean Distance and Correlation 71
5.5 Results of Experiments Discussed 72
5.5.1 Planes 1,2,3 72
5.5.2 Planes 3, 4, 5 73
5.5.3 General Discussion of Results 73

6 Conclusions 81

A Nomenclature 86
A.1 Symbols and Notation 86
A.2 Acronyms 87

List of Figures

3.1 Graphical model representing $X_3 \perp\!\!\!\perp X_1 \mid X_2$. 11
3.2 Basic graph theory. 14
3.3 Possible graphical model for $p(y_1, y_2, y_3)$. 16
3.4 The complete graph associated with the factorisation (2.8). 16
3.5 Graphical model examples. 17
3.6 Three canonical graphs used in discovering conditional independences. 20
3.7 Illustrating the need for both directed and undirected graphical models. . . . 26
3.8 MRF with three sets $X_A,\ X_B,\ X_C$ s.t. $X_A \perp\!\!\!\perp X_B \mid X_C$. 27
3.9 Markov Random Field (MRF) representing $p(x) = p(x_1)p(x_1|x_3)p(x_2|x_3)$. . . 29
3.10 Converting directed canonical graphs, in Figure 3.6, into undirected graphs. . 30
3.11 Moralised graphs of H_1 and H_2. 32
3.12 Graph of (3.16). 34
3.13 Illustration of how local messages are sent. 36
3.14 Graph illustrating local message passing graph-theoretically. 37

4.1 Factor graphs. 40
4.2 Fragment of a factor graph. 42
4.3 Chain example of sum-product algorithm. 45
4.4 Tree example of sum-product algorithm. 47
4.5 Example of max-sum algorithm. 51
4.6 Example of a graph with and without a chord. 53
4.7 Chordal graphs of examples H_1 and H_2. 54
4.8 Possible clique trees of H_1. 55
4.9 Junction tree of H_1 and H_2. 55
4.10 Graph used to explain message propagation in junction tree. 57

5.1 Euclidean transformation illustrated. 64
5.2 Similarity transformation illustrated. 65
5.3 Affine transformation illustrated. 65
5.4 Projective transformation illustrated. 66
5.5 General form of graphical model and junction trees for transformations. . . . 67
5.6 Graphical model used in application. 69
5.7 Euclidean transformation explained. 70

5.8 Example of correlation in images. 72
5.9 Images used in experiments. 75
5.10 Points used indicated on images. 76
5.11 Planes 1,2,3 calculated coplanar points using (5.5) and (5.6). 77
5.12 Planes 1,2,3 calculated coplanar points using (5.7) and (5.8) 78
5.13 Planes 3,4,5 calculated coplanar points using (5.5) and (5.6). 79
5.14 Planes 3,4,5 calculated coplanar points using (5.7) and (5.8). 80

List of Tables

2.1 Assumed conditional independence statements. 8

3.1 Full list of conditional independences for H_1. 23
3.2 Full list of conditional independences for H_2. 24
3.3 Conditional independences for moralised H_1 and H_2. 34

4.1 Joint distribution $p(x_1, x_2)$. 49

Chapter 1

Introduction

Graphical models is the framework that arises from the combination of graph theory and probability theory. Graph theory provides a way of modelling the set of variables and how they interact with each other, providing a visual representation, whilst probability theory is used to keep the system consistent. One of the advantages of using graphical models is that they are modular—a complex problem can be broken up into smaller parts using only local interactions while the probability theory keeps the model consistent. An example of this can be found in the music arena. When looking at chord progressions there are rules defining which chords should follow one another. A specific chord is dependent on the preceding chord and the accompanying melody note. These local interactions can be used to build a graphical model, with the relevant probability theory to keep the model consistent. We can then harmonise an entire piece of music using this graphical model. Another example is the weather. From today's weather one can make a prediction about tomorrow's weather, but we cannot use today's weather to predict next week Wednesday's weather. If you take all the different local interactions together in a graphical model, one can infer a specific season's weather.

Statistical physics [14], genetics [40] and the notion of interaction in a three-way contingency table [2] are three of the scientific areas that gave rise to graphical models, although at that time the term *graphical models* did not yet exist. The first book on the subject of graphical models and their application to multivariate data was [39]. There are various techniques that have been used for a number of years that are actually just specific cases of the general graphical model structure. Some of these techniques are Hidden Markov Models (HMM) [23, chapter 12], Kalman filters [23, chapter 15], Ising models ([26]) and factor analysis [23, chapter 14]. Graphical models can be, and is, used in many different areas. Some of these include image processing [42, 28, 34], computer vision [36, 10, 43, 9, 29], music [32, 30, 18], economics [33, 24, 12], social sciences [3, 38, 12] and even social networks [15, 21, 35]. For a more thorough history of the areas that led to the development of what is now known as graphical models, see [25] and [27].

In this thesis we aim to provide an introduction to the basics of the field of graphical models and their use in exact probabilistic inference problems. We start with some background probability theory in Chapter 2. In Chapter 3 we present some graph theory and show how to form graphical models. Having set up a graphical model, we can now use it to extract information about underlying data. For example, if we had a graphical model representing the weather as described previously, we could ask what the probability is that the 2010 Soccer World Cup final has good weather. Questions like these are answered using probabilistic inference on a model. One of the popular algorithms used to do exact probabilistic inference on graphical models is the junction tree algorithm, discussed in Chapter 4. Chapter 5 presents an application of graphical models in finding coplanar points in stereo images, with reference to all the preceding theory.

The problem of finding coplanar points remains relevant in computer vision. For example, in 3D reconstruction, knowledge of coplanar points allows us to identify planar regions in stereo images. This, in turn, improves the accuracy of the 3D reconstruction of featureless planes and reduces the computation time of planes with features, by reconstructing only a few points on the identified plane instead of using dense point reconstruction.

Finding planar regions in images has been well-documented, and there are various algorithms using different techniques. Most methods make use of information gathered from multiple images (more than 2). One could sweep through images using sets of planes at hypothesised depths, known as plane-sweeping algorithms [7]. An example of where this has been done is [41]. In [1] the authors hypothesise planes by using a single 3D line and surrounding texture, using six images. [11] finds planes by putting priors on shape and texture. Inter-image homographies are widely used in such articles as [13, 20]

In our experiments we specifically want to look at using only a single, uncalibrated camera and specifically using graphical models to solve the problem of finding coplanar points. In [6] a model is given for modelling projective transformations given two images. It is on this article that our experiments are based. This model also allows for jitter, which accomodates user input.

Our coplanar point experiments allow us to test the theory of graphical models. Our conclusions are that although graphical models is not the most efficient way to solve the problem, it yields satisfactory matches, using a single uncalibrated camera

Chapter 2

Probability Theory

Probability theory attaches values to, or beliefs in, occurrences of events. This enables us to model events much more realistically—assigning probabilities to certain events taking place, instead of it only being a yes/no model (a deterministic model). In this chapter we present the probability theory required to develop graphical models, including conditional independences—a concept that is vital to the efficient use of graphical models. For a more in-depth introduction to probability theory and statistics, see [37].

In Section 2.1 basic notation of probability theory is given, which is summarised in Appendix A, with basic properties and rules in probability theory following in Section 2.2. As mentioned in the introduction, we will be considering probabilistic inference problems, and thus will discuss general probabilistic inference in Section 2.3. After discussing which class of problems we would like to solve with graphical models, we look at one of the important aspects in graphical models, conditional independences, in Section 2.4. We will end this chapter with Section 2.5—a look at the computational and memory complexities that we are faced with.

2.1 Notation

Let X be a random variable such that $X = \{X_1, \ldots, X_N\}$ with $N > 1$ the total number of variables and let x_i represent a realisation of the random variable X_i. Every random variable may either be scalar-valued (univariate) or vector-valued (multivariate). In general, variables can either be continuous or discrete. For the sake of simplicity, and due to fact that we are looking at a discrete application, only the discrete case is considered here—but the results obtained can be generalised to continuous variables.

Discrete probability distributions can be expressed in terms of probability mass functions,

$$p(x_1, x_2, \ldots, x_N) := P(X_1 = x_1, X_2 = x_2, \ldots, X_N = x_n).$$

Since we have also previously stated that $X = \{X_1, \ldots, X_N\}$, we can naturally say that $x = \{x_1, \ldots, x_N\}$ and thereby shorten the notation of a joint distribution to

$$\begin{aligned} P(X_1 = x_1, X_2 = x_2, \ldots, X_N = x_n) &= P(X = x) \\ &= p(x). \end{aligned}$$

We sometimes use the notation X_A, where A indicates a set of indices, i.e. X_A indicates the random vector of variables indexed by $A \subseteq \{1, \ldots, N\}$. In this way, if $A = \{2, 3, 6\}$ then $X_A = \{X_2, X_3, X_6\}$.

If the total set of possible indices is $\{1, 2, \ldots, N\}$ and the set $A = \{1, 5, 6\}$ then the set of indices $\bar{A} = \{2, 3, 4, 7, \ldots, N\}$ is the set of indices not in A.

2.2 Basic Properties and Rules

There are two important, basic properties of probabilities, namely

$$0 \leq p(x) \leq 1, \forall x \in \mathcal{X}, \tag{2.1}$$

$$\sum_{x \in \mathcal{X}} p(x) = 1, \tag{2.2}$$

where $\mathcal{X}$ denotes the state space. The first property states that we represent probabilities by values between 0 and 1, with 0 being the value of an event definitely not occurring and 1 the value of an event that definitely occurs. For example, if one throws a completely unbiased coin, the probability of the coin landing with its head up is $\frac{1}{2}$. If we let y_1 be the case where the coin lands heads up, then $p(y_1) = 0.5$. The second property states that if one sums over all the possible events, then the sum has to be equal to one. Consider the above case again, but in addition, let y_2 be the case where the coin lands heads down. If the coin is thrown, either y_1 or y_2 will be realised, the probability of either y_1 or y_2 will be 1, thus $p(y_1) + p(y_2) = 1$.

The function $p(x_A, x_B)$ denotes the probability that events x_A and x_B both occur, whereas $p(x_A|x_B)$ is the probability of x_A, given that x_B is realised. We call $p(x_A|x_B)$ the conditional probability and say that the distribution $p(x_A, x_B)$ is *conditioned on* x_B and x_B is the *conditioning variable.* For example, if x_A is the event that it is raining and x_B that it is winter, then $p(x_A, x_B)$ would be the probability that it is both winter and raining, whereas $p(x_A|x_B)$ would be the probability that it is raining, given that it is winter. In a winter-rainfall area these probabilities would be higher than in a summer-rainfall area.

Four important rules of probability are

- the conditioning or product rule

$$p(x_A, x_B) = p(x_A|x_B)p(x_B) \tag{2.3}$$

for $p(x_B) > 0$,

- the marginalisation or sum rule

$$p(x_A) = \sum_{x_B \in \mathcal{X}_B} p(x_A, x_B), \tag{2.4}$$

- the chain rule of probability (application of product rule numerous times)

$$p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i | x_1, \ldots, x_{i-1}) \tag{2.5}$$

and

- Bayes' theorem,

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{2.6}$$

Applying the product and sum rule to Bayes' theorem, we also have that

$$p(x) = \sum_y p(x|y)p(y).$$

If two events X and Y are independent of each other, i.e. whether or not event X happens has no forbearance on the probability of event Y happening, then the joint probability $p(x, y)$ of the two events is given by

$$p(x, y) = p(x)p(y). \tag{2.7}$$

In probability distributions, factorisations are not necessarily unique. As an example of this, consider the distribution $p(x_1, x_2, x_3, x_4, x_5, x_6)$. This distribution can be factorised, using (2.5), in, for example, two ways,

$$p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_3, x_2, x_1)p(x_5|x_4, x_3, x_2, x_1)p(x_6|x_5, x_4, x_3, x_2, x_1) \tag{2.8}$$

and

$$p(x_1|x_6, x_5, x_4, x_3, x_2)p(x_2|x_6, x_5, x_4, x_3)p(x_3|x_6, x_5, x_4)p(x_4|x_6, x_5)p(x_5|x_6)p(x_6). \tag{2.9}$$

2.3 Probabilistic Inference

There are two situations in which probability distributions are used. The one situation is where $p(x)$ is known, and we want to do probabilistic inference on it:

1. determine marginal distributions of $p(x)$,
2. determine conditional distributions of $p(x)$,

3. determine the most probable data values from a set of given data (maximise the *a posterior* distribution by means of the maximum a posteriori (MAP) technique, written $x^* = \arg\max_x p(x)$) or
4. compute the probability of the observed data (the likelihood of the observed data, for the discrete case, is written as $p(x \in A) = \sum_{x \in A} p(x)$).

Depending on the application, we may want to do any or all of the items in the list, but they all fall under probabilistic inference. The other situation is called learning (also known as estimation or statistical inference), where the parameters of a distribution $p(x)$ is determined for given specific data.

For our purposes we consider probabilistic inference and focus on the first three items on the list. In our application of finding coplanar points we use item three in the list. Here the user supplies data points in two images. From these we need to determine which are the most probable coplanar points.

Calculating the marginals of a given probability density function is conceptually straightforward and calculating conditional distributions uses marginals in the following way. Let V, E, F and R indicate sets of indices, with V the set of all indices, E the set of indices of observed data (evidence), F the indices of the variables that we want to find the probability of, given E, and $R = V \backslash (E \cup F)$. Then X_E is the set of random variables that has been observed (and on which we will condition) and X_F is the random variables that we want to find, given the observed data. These two subsets are obviously disjoint. Thus we focus on the probabilistic inference problem of finding the probability $p(x_F|x_E)$. From the conditioning rule (2.3) we therefore want to find,

$$p(x_F|x_E) = \frac{p(x_E, x_F)}{p(x_E)}. \tag{2.10}$$

Marginalising $p(x_E, x_F, x_R)$ we find

$$p(x_E, x_F) = \sum_{x_R} p(x_E, x_F, x_R), \tag{2.11}$$

and, marginalising further,

$$p(x_E) = \sum_{x_F} p(x_E, x_F). \tag{2.12}$$

Now (2.11) and (2.12) can be used to calculate $p(x_F|x_E)$ in (2.10).

2.4 Conditional Independence

Conditional independences lead to a factorisation of the distribution (this is formally established later) that results in more effective algorithms than when only using the chain rule to provide a factorisation of the distribution. Before viewing this factorisation in more detail, let us first consider conditional independences.

Consider the case of having three variables $X_1,\ X_2,\ X_3$, with X_1 representing the probability of my father being in a university residence *Wilgenhof*, X_2 my older brother being in *Wilgenhof* and X_3 my younger brother being in *Wilgenhof* (assuming that they all attend the same university and are placed in the same residence as a family member). If we do not have any information of my older brother being in *Wilgenhof*, then my younger brother being in *Wilgenhof* depends on my father being there, thus X_1 has an influence on X_3's distribution. However, if we have information regarding whether my older brother was in *Wilgenhof*, then we do not need any information about my father being there or not, thus X_3 is independent of X_1 given X_2, written $X_3 \perp\!\!\!\perp X_1 \mid X_2$. Thus the conditional distribution of X_3 given X_2, does not depend on the value of X_1, which can be written as

$$p(x_3|x_1, x_2) = p(x_3|x_2).$$

Suppose the probability of x_1 and x_3 need to be found, given the value of x_2. Then using (2.3) we have

$$\begin{aligned} p(x_1, x_3|x_2) &= p(x_1|x_3, x_2)p(x_3|x_2) \\ &= p(x_1|x_2)p(x_3|x_2). \end{aligned}$$

Since this satisfies the condition of (2.7), we see that X_1 and X_3 are independent conditional to knowing X_2, ($X_1 \perp\!\!\!\perp X_3 \mid X_2$). Note that in the example above, there are three variables that have to be considered in the joint distribution $p(x_1, x_3|x_2)$ whilst in $p(x_1|x_2)$ and $p(x_3|x_2)$ there are only two variables in each. From this simple example we can see that conditional independence statements lead to a factorisation of a joint distribution. In general we can therefore say that if X_A is independent of X_B given X_C, with A, B and C sets of indices, then

$$p(x_A, x_B|x_C) = p(x_A|x_C)p(x_B|x_C) \tag{2.13}$$

or, alternatively,

$$p(x_A|x_B, x_C) = p(x_A|x_C). \tag{2.14}$$

Not only can we use independence and conditional independence statements to factorise joint probability distributions, but if a factorised joint probability distribution is given, we can extract independence statements from the factorised distribution. Though finding these independence statements algebraically can be computationally expensive.

We now return to the factorisation example at the end of Section 2.2, this time with the assumed conditional independence statements in the left column of Table 2.1. Then the factorisation in (2.8) can be simplified to

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_5, x_2). \tag{2.15}$$

As a further example, consider the distribution $p(y_1, y_2, y_3, y_4, y_5, y_6, y_7)$. If the conditional independence statements in the right column of Table 2.1 hold, then the factorised distribution is

$$p(y_1, y_2, y_3, y_4, y_5, y_6, y_7) = p(y_1)p(y_2|y_1)p(y_3|y_2)p(y_4)p(y_5|y_3, y_4)p(y_6|y_1, y_5)p(y_7|y_2, y_3, y_6).$$

Table 2.1: *Assumed conditional independence statements.*

Example 1	Example 2
$X_1 \perp\!\!\!\perp \emptyset \mid \emptyset$	$Y_1 \perp\!\!\!\perp \emptyset \mid \emptyset$
$X_2 \perp\!\!\!\perp \emptyset \mid X_1$	$Y_2 \perp\!\!\!\perp \emptyset \mid Y_1$
$X_3 \perp\!\!\!\perp X_2 \mid X_1$	$Y_3 \perp\!\!\!\perp Y_1 \mid X_2$
$X_4 \perp\!\!\!\perp \{X_1, X_3\} \mid X_2$	$Y_4 \perp\!\!\!\perp \{Y_1, Y_2, Y_3\} \mid \emptyset$
$X_5 \perp\!\!\!\perp \{X_1, X_2, X_4\} \mid X_3$	$Y_5 \perp\!\!\!\perp \{Y_1, Y_2\} \mid \{Y_3, Y_4\}$
$X_6 \perp\!\!\!\perp \{X_1, X_3, X_4\} \mid \{X_2, X_5\}$	$Y_6 \perp\!\!\!\perp \{Y_2, Y_3, Y_4\} \mid \{Y_1, Y_5\}$
	$Y_7 \perp\!\!\!\perp \{Y_1, Y_4, Y_5\} \mid \{Y_2, Y_3, Y_6\}$

2.5 Memory and Computation Complexity

Consider the case of representing the joint probability distribution $p(x_1, x_2, \ldots, x_N)$. Since we are focussing on the discrete case, and not the continuous case, one way of representing this distribution would be with the help of an N-dimensional look-up table. If every variable x_i has r realisations, then we must store and evaluate r^N numbers. Being exponential in N is problematic, since the numbers that need to be stored quickly grow to be large. Factorising the probability distribution by using conditional independences can reduce the size of the required look-up tables.

Each conditional probability is stored as a look-up table. The dimensions of the table representing a specific conditional probability then equates to the amount of variables in the probability. For example, $p(x_i)$ is represented in a one-dimensional table whereas $p(x_i|x_1, x_2, x_3)$, $i \neq 1, 2, 3$, is represented by a 4-dimensional table. The total number of different possibilities per conditional probability is therefore r^{m_i+1}, if every variable has r realisations and m_i is the number of variables being conditioned on. The associated table is of dimension $(m_i + 1)$, and size r^{m_i+1}. The number m_i is also known as the *fan-in* of variable X_i.

For now, consider the distribution $p(x) = p(x_1, x_2, x_3, x_4, x_5, x_6)$ and suppose we want to find $p(x_4)$. We therefore need to calculate

$$p(x_4) = \sum_{x_1,x_2,x_3,x_5,x_6} p(x_1, x_2, x_3, x_4, x_5, x_6) \tag{2.16}$$

where the associated look-up table has r^6 entries. However, if we assume the conditional independences as in the left column of Table 2.1, and thus use the simplified factorisation as in (2.15), we have

$$p(x_4) = \sum_{x_1,x_2,x_3,x_5,x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_2)p(x_6|x_2,x_5), \tag{2.17}$$

that requires storage space of $r + 4r^2 + r^3$, which is significantly smaller than r^6. Therefore exploiting conditional independences in factorisations reduces the required storage space.

Another advantage of using conditional independences is in reducing the computational complexity of a calculation. The reduction is gained by using a simplified factorisation together with the distributiv law. The distributive law states that if there are three variables a, b and c then

$$ab + ac = a(b + c). \tag{2.18}$$

On the left side of the equation, three calculations have to be performed (ab, ac and then the sum between the two products) whereas on the right side, there are only two calculations ($b+c$ and multiplying answer with a). This is only a minor improvement, but this example is small. Using the distributive law results in considerable space and time complexity improvements when looking at larger calculations.

In (2.11) and (2.12) the marginals of joint distributions are calculated. Suppose that each variable in the distribution has r possible realisations. In (2.11) there are $r^{|R|}$ terms in the summation, where $|R|$ denotes the number of random variables in the set R. In (2.12) this becomes $r^{|F|}$ terms (given that $|F|$ is the size of F). This is not always a viable summation, due to the fact that the number of variables in F and R can be very large. In the discrete case a table of size $r^{|R|}$ is used and each element in R needs to be summed over, so the order complexity becomes $O(r^{|R|})$ operations to do a single marginalisation. If the distributive law is applied, and advantage taken of the factorisation of joint distributions into local factors these calculations usually decrease considerably. Exactly how much of an improvement there is, is of course dependent on the actual factorisation of a joint distribution.

Suppose we have a random variable with probability distribution

$$\begin{aligned} p(x) &= p(x_1, x_2)p(x_2, x_3)\dots p(x_{N-1}, x_N) \\ &= \prod_{i=1}^{N-1} p(x_i, x_{i+1}). \end{aligned} \tag{2.19}$$

Suppose, without loss of generality, that we want to find the marginal $p(x_1)$. Using the distributive law on the equation

$$p(x_1) = \sum_{i=2}^{N} \frac{1}{Z} \prod_{i=1}^{N-1} p(x_i, x_{i+1}) \tag{2.20}$$

we find

$$p(x_1) = \frac{1}{Z}\left[\sum_{x_2} p(x_1, x_2)\left[\sum_{x_3} p(x_2, x_3)\ldots\left[\sum_{x_N} p(x_{N-1}, x_N)\right]\right]\right]. \tag{2.21}$$

If every variable has r possible realisations, then the order complexity of finding (2.20) will be $O(r^N)$ and that of (2.21), $O(Nr^2)$. As N grows, the improvement that has been made in computational complexity will therefore become exponentially better. Thus it can be seen that using the local conditional probabilities, with the distributive law, improves the order-complexity of the problem, often significantly. When comparing the computational complexities of (2.16) and (2.17) we find that the complexity of (2.16) is $O(r^6)$ and of (2.17) is $O(r^3)$, and therefore again have a considerable improvement in this example.

Chapter 3

Graphical Models

Graphical models combine the probability theory of the previous chapter with graph theory. The benefits of this combination are that it provides a simple and effective way to visualise a problem, and allows insights into the properties of the underlying model.

As an example, consider the case of my father, younger and older brother in the residence *Wilgenhof*, as presented in Section 2.4. The graphical model in Figure 3.1 visually indicates the relationships between the variables X_1, X_2 and X_3. The exact meaning of the arrows and the nodes will be made clear in this chapter.

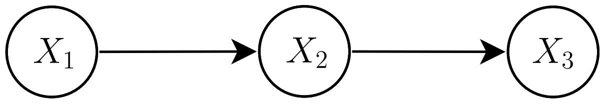


Figure 3.1: *Graphical model representing* $X_3 \perp\!\!\!\perp X_1 \mid X_2$.

In this chapter we give a brief introduction to graph theory, where we discuss directed and undirected graphs. We then present two types of graphical models: one with directed graphs and one with undirected graphs, in Sections 3.3 and 3.4. In these sections we will see how we use probability theory to find the undirected and directed graphs, but also how graph theory can be used to find the underlying probability theory in the graphs. In Section 3.4.3 we will see how we can convert a directed graph into an undirected graph, and also why we need both types of graphs. Finally, in Section 3.6, we will look at how local messages are sent around in a graph—an important element in the algorithms we will be discussing in Chapter 4.

3.1 Notation and Basic Graph Theory

A graph $G := G(V, E)$ consists of a finite, non-empty set of nodes (or vertices) V connected to each other by a set of edges (or links or arcs) E. The edges are unordered pairs of distinct vertices and an edge $e = \{u, v\}$ is said to join the vertices u and v. For example, the graph in Figure 3.2(a) has vertices $V = \{u, v, w\}$ and edges $E = \{uv, vw, wu\}$.

A *$u-v$ path* in a graph is a sequence of distinct vertices $v_0v_1v_2 \ldots v_n$, $u = v_0$ and $v = v_n$, with an edge existing between each consecutive pair of vertices on the graph. In Figure 3.2(a) and Figure 3.2(b) there are two $u - v$ paths, namely $u - v$ and $u - w - v$.

Often graphs have a certain structure, where one node is (or a group of nodes are) connected to the same structure many times. As an example, see Figure 3.2(d), where X_1 and X_2 are both connected to all other vertices. To write this compactly, we introduce the concept of a *plate*, a rectangular box shown in Figure 3.2(e). A plate is a structure where everything inside the rectangular box is repeated up to the specified number. Thus Figure 3.2(e) is a compact form of Figure 3.2(d), with $3 \leq j \leq N$.

Note that if X_F is the set of variables that we want to find the marginal of (finding $p(x_F|x_E)$), then X_F is referred to as the query node(s).

3.1.1 Directed Graphs

The edges in a graph can either be directed or undirected. If an edge is directed, it is pointing in a specific direction—indicated by an arrowhead. Figure 3.2(b) has $V = \{u, v, w\}$ and directed edges $E = \{uv, vw, wu\}$.

A $u - v$ path does not take the direction of edges into account, whereas a *directed path* is a path where the edges are also orientated in the same direction. In Figure 3.2(b) $u - w - v$ is a directed path. Although in general graph theory there can be cycles in directed graphs, in graphical models only directed graphs without cycles are used, known as directed acyclic graphs (DAGs). Directed graphs in graphical models are also known as Bayesian Networks (BN) (although BN can also mean Belief Networks).

In a directed graph there is the concept of *parents* and *children*. Suppose graph G has two nodes, v_1 and v_2, then v_1 is v_2's parent and v_2 is v_1's child if there is a directed edge going from v_1 to v_2. A node can have many (or no) children and/or parents. As an example, in Figure 3.2(b) u has both v and w as children while v's parents are the elements of the set $\{u, w\}$. Notice that u is parentless and v is childless. Let π_i denote the set of parents of node i, $\forall i \in V$. Accordingly, X_{π_i} denotes the parents of X_i. Furthermore, there is also the concept of *ancestors* and *descendants*. If there is a directed $u - v$ path in a graph, u is an ancestor of v and v is u's descendant. As an example, in Figure 3.2(f), g is a descendant of a, b, c and d (and thus a, b, c and d are g's ancestors) and f's ancestors are a, d and e (with f being a descendant of a, d and e).

In DAGs, there is the concept of a *topological ordering*, also known as a topological sorting. A topological ordering is an ordering where all parents come before their children, thus for all vertices i in V, π_i comes before i in the order. As an example, in Figure 3.2(f), a comes before nodes e, d and c because it is one of the parents of those nodes. One topological ordering of the graph in Figure 3.2(f) is a, b, c, d, e, f, g. However, topological orderings are not unique, since a, d, e, b, c, f, g is another topological ordering of the same graph.

A node can either be *head-to-head*, *tail-to-tail* or *head-to-tail* with regards to the path that it is on, and it depends on the incoming and outgoing arrows that it has. For example, in Figure 3.2(b), node v is head-to-head, since the arrows connected to this node both have their heads pointing at v. Similarly, node w is a head-to-tail node, and node u is a tail-to-tail node.

3.1.2 Undirected Graphs

Undirected graphs have undirected edges—edges where there is no direction, and thus no arrowheads. Figure 3.2(a) is a graph with vertex-set $V = \{u, v, w\}$ and undirected edge-set $E = \{uv, vw, wu\}$. An undirected graph in graphical models can also be referred to as an Markov Random Field (MRF).

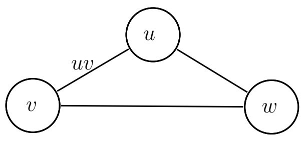
A *complete graph* is a graph where all the nodes are connected to each other, see Figure 3.2(c). A *tree* is an undirected graph that has no cycles, has a path between all nodes, and has $N-1$ edges, if there are a total of N nodes. For example, the graph in Figure 3.2(g) would be a tree if the edge between node a and e is removed. In this tree, nodes a, d and e are called the leaves, because they have only one edge.

An important concept in undirected graphs is that of *cliques* and *maximal cliques*. A clique is a complete subgraph of a graph. In Figure 3.2(g), the sets $\{a, b\}$, $\{b, c\}$, $\{c, d\}$ and $\{a, b, e\}$ are four of the cliques in the graph. A maximal clique is a clique that is not a proper subset of another clique, thus $\{a, b, e\}$ and $\{c, d\}$ are maximal cliques. Another way to define maximal cliques is that they are cliques such that if any other node is added to the set, it is not a clique any more.

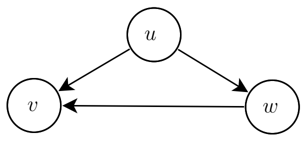
We now have all the background knowledge of graph theory and probability theory to delve into graphical models.

3.2 Introduction to Graphical Models

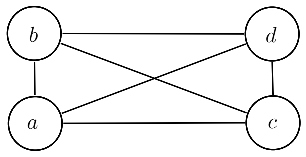
What are graphical models? The most concise explanation, quoted extensively, is Michael I. Jordan's explanation of graphical models: "…a marriage between probability theory and graph theory …", [22]. Combining graph theory's ability to model events and the relationship between events with probability theory's ability to attach values of (or beliefs in) these events, makes graphical models a powerful tool. Not only does it give us a way



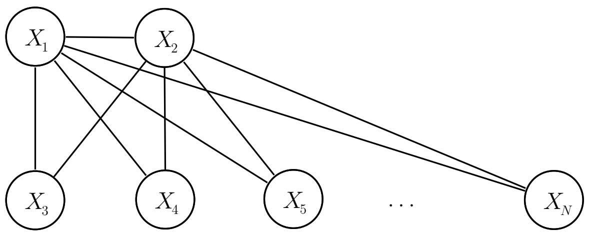
(a) Undirected Graph



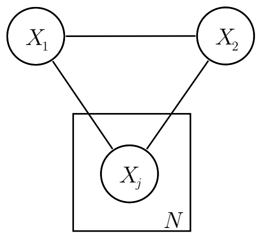
(b) Directed Graph



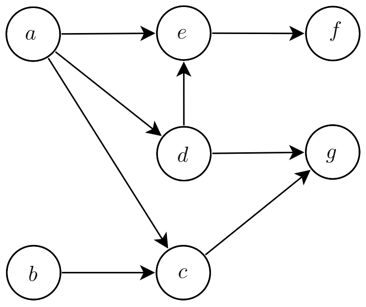
(c) Complete Graph



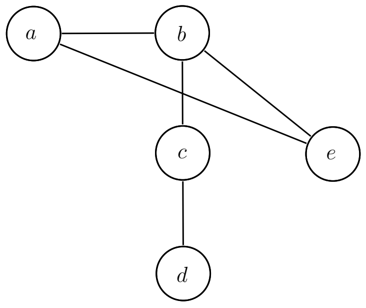
(d) Graph G



(e) Plate of G



(f) Topological sort



(g) Cliques

Figure 3.2: *Basic graph theory.*

to model the uncertainty of real life and encode algorithms to extract information, it is also a modular system—allowing complex problems to be broken down into smaller, more manageable pieces.

Addressing the probabilistic inference and learning problems described in Section 2.3 can be tricky in practice, especially when working with large data sets and/or complex probability distributions. Typically we have multivariate distributions (when multiple random variables are observed and analysed at the same time) that are structured according to conditional independence statements. The graphical model framework is one of the techniques that can be employed to solve these problems. Amongst other things, graphical models can therefore be used to find marginal distributions, conditional probabilities, MAP assignments, maximum likelihoods of parameters as well discovering conditional independences amongst various variables more effectively.

If there are conditional independence statements, i.e. the variables in the distribution interact with each other in some way, and the probability distribution can be factorised according to these conditional independences then the stage is set for graphical models to be used, as we illustrate in this chapter. If, however, the model does not simplify in terms of conditional independence statements, then graphical models may not be the correct framework to use.

In graphical models, graphs are used to visualise distributions, and although it would be ideal to only use one type of graph to illustrate any probability distribution, some distributions can only be represented by a BN whilst others can only be represented by an MRF (see Section 3.4), and thus they are both necessary in our study of graphical models.

In the next two sections, BNs and MRFs are presented for the discrete case (the concepts can easily be extended to the continuous case). Note that the theory and algorithms described here follows and summarises some of the work in [23, 4] and a workshop given by Tibério S. Caetano [5].

3.3 Bayesian Networks

For the sake of simplicity, the theory underlying the combination of graph theory and probability theory is discussed using directed graphs as backdrop—the same concepts however are also in MRFs and are dealt with in Section 3.4.

3.3.1 From Probability Theory to Directed Graphs

In graphical models, each node in a BN or MRF either represents a random variable, or a group of random variables and thus the nodes are named X_i, with i either a single index or a set of indices. If there is an edge linking nodes, it indicates that there is a relationship between the random variables in the distribution.

We assume that there is a one-to-one mapping between the nodes and random variables, and due to this, it is convenient to blur the distinction between the variable and node—referring to both as X_i, or x_i. At times it is necessary to indicate when a variable has been observed, i.e. when its value is specified. If variable x_i has been observed, we indicate this by $\bar{x}_i$.

Consider an arbitrary joint distribution $p(y_1, y_2, y_3)$. Applying the conditioning rule (2.3) twice, first conditioning on y_3 and y_2 and then again on y_3,

$$\begin{aligned} p(y_1, y_2, y_3) &= p(y_1|y_2, y_3)p(y_2, y_3) \\ &= p(y_1|y_2, y_3)p(y_2|y_3)p(y_3). \end{aligned}$$

To represent this in a directed graph, see Figure 3.3, place a directed edge from node y_i to y_j, $i \neq j$, $i, j \in V$ if the variables can be found together in a joint probability and y_i is the variable which is being conditioned on. For example, in the factorisation of $p(y_1, y_2, y_3)$, we have $p(y_1|y_2, y_3)$ as a factor. Thus there should be a directed edge from y_2 to y_1 as well as from y_3 to y_1, since both y_2 and y_3 were conditioned on. This is the method to construct directed graphical models from probability distributions.

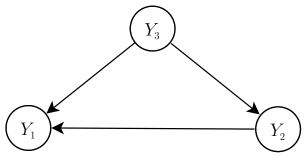


Figure 3.3: *Possible graphical model for* $p(y_1, y_2, y_3)$.

The probability density function $p(x_1, x_2, x_3, x_4, x_5, x_6)$ can be factorised as in (2.8) using the chain rule (2.5). The BN associated with this factorisation is given in Figure 3.4—it is a complete graph. However, if we use the conditional independence statements given in Table 2.1 in the left column, the factorisation simplifies to

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5), \tag{3.1}$$

and the associated graph simplifies to the graph H_1 given in Figure 3.5(a). This example illustrates the simplifications gained when using conditional independence statements.

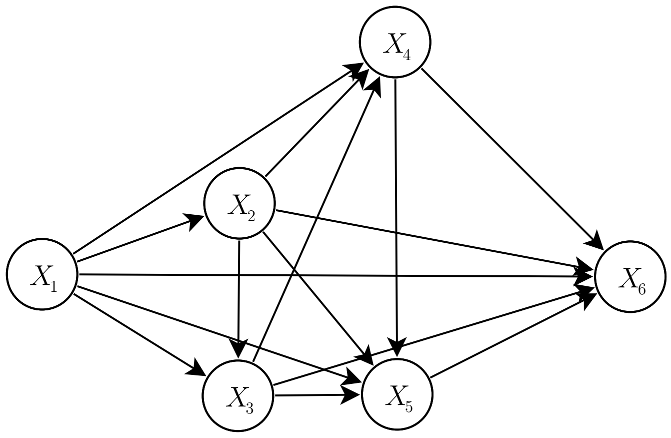


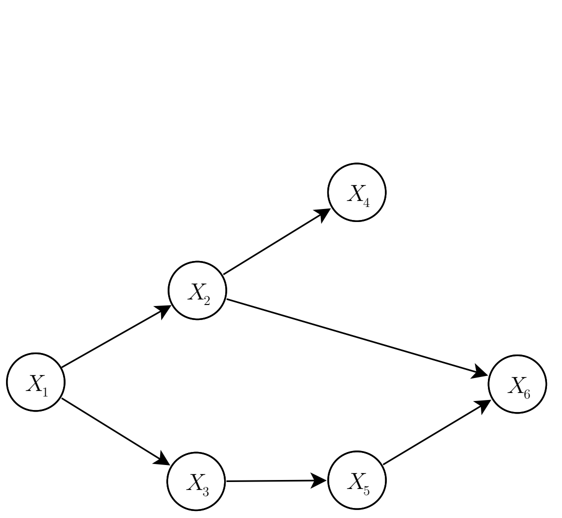
Figure 3.4: *The complete graph associated with the factorisation (2.8).*

Another example is the graph H_2, as given in Figure 3.5(b) associated with the factorised joint probability

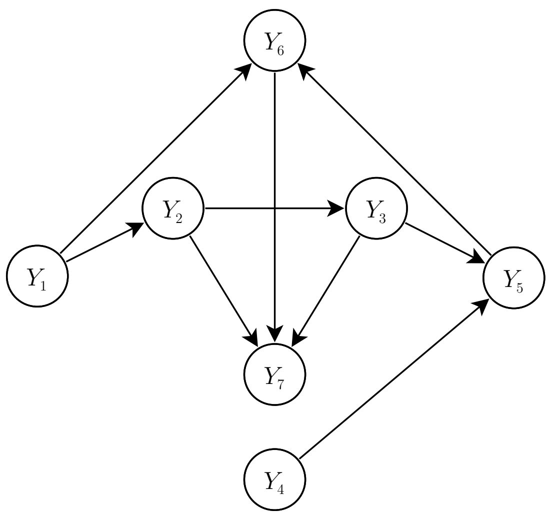
$$p(y_1, y_2, y_3, y_4, y_5, y_6, y_7) = p(y_1)p(y_2|y_1)p(y_3|y_2)p(y_4)p(y_5|y_3, y_4)p(y_6|y_1, y_5)p(y_7|y_2, y_3, y_6). \tag{3.2}$$

3.3.2 Local Functions

One of the advantages of using graphical models is that they are modular—the entire graph can be subdivided into smaller pieces for which certain relationships are defined, with the entire graph remaining consistent in a whole. To find these smaller pieces, a local area needs to be defined on a graph, and on this local area, a function is defined to represent



(a) Graph H_1



(b) Graph H_2

Figure 3.5: *Graphical model examples.*

the relationships. Since we have already looked at conditional independences and seen that they provide a good factorisation technique for the distribution, we would like to consider local areas in such a way that we can have the factorisation from conditional independences produce the functions over the areas.

It has already been stated that a distribution can have various factorisations. Suppose that we have a distribution, with its factorisation,

$$p(x) = f_a(x_1, x_2, x_3) f_b(x_1, x_3) f_c(x_4, x_6) f_d(x_5) f_e(x_4, x_6). \tag{3.3}$$

Consider the set of variables $\{x_1, x_2, \ldots, x_N\}$ and let $\mathcal{K}$ be a multiset of subsets of the indices $\{1, 2, \ldots, N\}$, i.e. the indices in the set $\mathcal{K}$ may be duplicated. For example, if we were to look at the distribution as in (3.3), the set of variables would be $\{x_1, x_2, \ldots, x_6\}$, the set of indices $\{1, 2, 3, 4, 5, 6\}$ and $\mathcal{K} = \{\{1, 2, 3\}, \{1, 3\}, \{4, 6\}, \{5\}, \{4, 6\}\}$. Furthermore, define $\mathcal{F}$ to be index set of the members of $\mathcal{K}$, such that $\mathcal{K} = \{K_s : s \in \mathcal{F}\}$. A factor $f_s(x_{K_s})$ is defined for every $s \in \mathcal{F}$. In (3.3), $\mathcal{F} = \{a, b, c, d, e\}$ and the factors are $f_a(x_1, x_2, x_3),\ f_b(x_1, x_3),\ f_c(x_4, x_6),\ f_d(x_5),\ f_e(x_4, x_6)$.

Choosing the functions to use depends on the application, however, there are two conditions that need to be met for the factorisation to lead to a valid probability distribution. The first is that the factors need to be nonnegative, and the second that the factorisation needs to be normalised (i.e. conditions (2.1) and (2.2) need to hold). Although (2.1) needs to be checked for every function chosen, (2.2) can be worked into the definition of choosing the functions. A normalisation factor, Z can be introduced such that, if the general form of representing the distribution is

$$p(x) = \frac{1}{Z} \prod_s f_s(x_{K_s}), \tag{3.4}$$

then

$$Z = \sum_x \prod_s f_s(x_{K_s}). \tag{3.5}$$

Using this definition, we now look at defining these functions for directed graphs. Firstly a local area needs to be defined. In a BN, the local area can be viewed to be a node and its parents. For each node $x_i \in V$ associate with it and all its parents, the function $p(x_i|x_{\pi_i})$. Due to the fact that conditional probabilities have the properties (2.1) and (2.2), this definition is valid. Furthermore, since they are already probabilistic distributions, they are already normalised, and therefore $Z = 1$, making the definition of the joint distribution associated with directed graphs

$$p(x_1, x_2, \ldots, x_N) := \prod_{i=1}^{N} p(x_i|x_{\pi_i}). \tag{3.6}$$

The functions $p(x_i|x_{\pi_i})$, together with the numerical values they can assume, create a family of joint distributions associated with the graph. This holds in general for any specific distribution and their associated graph. The fact that distributions can be characterised by such local functions, and also, as discussed later in Section 3.3.3, by the patterns of the edges in a graph, and the relationship between these characterisations are the underlying theory of probabilistic graphical models.

The conditional probabilities $p(x_i|x_{\pi_i})$ are called the local conditional probabilities associated with the graph. As can be seen, they are the building blocks which are used to make up the joint distribution associated with a graph.

3.3.3 From Directed Graphs to Probability Theory

Comparing (2.5) and (3.6) we see that there are possibly some variables that are left out of being conditioned on in (3.6), since x_{π_i} does not necessarily, and most often does not, include the entire set $\{x_1, \ldots, x_{i-1}\}$. From our knowledge of conditional independence thus far, we can conjecture that this implies that x_i is conditionally independent of those variables not in x_{π_i}, given x_{π_i}. We shall now investigate this conjecture.

Let I be a topological ordering of the nodes, then $\forall x_i \in V$, x_{π_i} comes before x_i in I. Without loss of generality, suppose that we want to find the marginalisation $p(x_j|x_{\pi_j})$, thus, we have, from (3.6),

$$p(x) = p(x_1)p(x_2|x_{\pi_2}) \ldots p(x_j|x_{\pi_j}) \ldots p(x_N|x_{\pi_N}). \tag{3.7}$$

Since $\sum_{x_i} p(x_i|x_{\pi_i}) = 1$, from discussion of local conditional factors in Section 3.3.1, we have, marginalising,

$$\begin{aligned} p(x_1,\dots,x_j) &= \sum_{x_{j+1}}\sum_{x_{j+2}}\dots\sum_{x_N} p(x_1,\dots,x_n) \\ &= p(x_1)p(x_2|x_{\pi_2})\dots p(x_j|x_{\pi_j})\sum_{x_{j+1}} p(x_{j+1}|x_{\pi_{j+1}})\dots\sum_{x_N} p(x_N|x_{\pi_N}) \\ &= p(x_1)p(x_2|x_{\pi_2})\dots p(x_j|x_{\pi_j}), \end{aligned}$$

and

$$\begin{aligned} p(x_1,\dots,x_{j-1}) &= \sum_{x_j} p(x_1)p(x_2|x_{\pi_2})\dots p(x_j|x_{\pi_j}) \\ &= p(x_1)p(x_2|x_{\pi_2})\dots p(x_{j-1}|x_{\pi_{j-1}}). \end{aligned}$$

Dividing $p(x_1,\dots,x_j)$ by $p(x_1,\dots,x_{j-1})$ and using the product rule (2.3), we get

$$p(x_j|x_1,\dots,x_{j-1}) = p(x_j|x_{\pi_j}), \tag{3.8}$$

and from our conditional independent discussion in Section 2.4 we gather that

$$X_j \perp\!\!\!\perp X_{\nu_i} \mid X_{\pi_i}, \tag{3.9}$$

with ν_i the set that contains all ancestors except the parents of x_i and also possibly some other non-descendant nodes (nodes that are neither ancestors nor descendants of x_i). Generally, for $x_i \in V$ the set of *basic conditional independence statements*, found from (3.9), associated with a graph G and specific topological ordering I are, for all $i \in V$,

$$\{X_i \perp\!\!\!\perp X_{\nu_i} \mid X_{\pi_i}\}. \tag{3.10}$$

Graph-theoretically interpreted, this means that the missing edges in a graph correspond to a basic set of conditional independences.

Before looking at *graph separation* and what it means in terms of conditional independences, consider the following example of the above concepts. Table 2.1 indicates the lists of basic conditional independences for the graphs in Figure 3.5 with their respective associated probability distributions (3.1) and (3.2) and the topological orderings $x_1, x_2, x_3, x_4, x_5, x_6$ and $y_1, y_2, y_3, y_4, y_5, y_6, y_7$. Note, for example, that $X_4 \perp\!\!\!\perp \{X_1, X_3\} \mid X_2$ is listed as a conditional independence statement in Table 2.1 and in Figure 3.5 there is no link between X_4 and either of X_1 and X_3 other than the link through X_2.

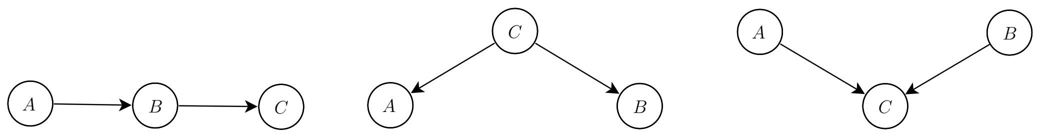
Graphical models can also be used to infer conditional independences that are not immediately obvious when looking at a factorisation of a joint probability distribution. If we want to know whether variables are conditionally independent, we can work it out algebraically, though this can be a rather arduous exercise. Using graphical models provides a simpler method to find out whether a certain set of variables is conditionally independent and can also be used to write down all the conditional independences implied by the basic set.

The general framework for using graphical models to read off conditional independences that are not explicitly stated is called *d-separation*, [31]. Consider a general DAG G with A, B and C subsets of V such that their intersection is empty, and their union may or may not be the entire set V. We want to find out whether $A \perp\!\!\!\perp B \mid C$ holds in G. To establish conditional independence, consider all paths from any node in A to any node in B. A path is said to be *blocked* if it includes a node such that one of the following conditions hold

- there is a node $i \in C$ that is on the $A{-}B$ path where the arrows meet either head-to-tail or tail-to-tail, or
- there is a node $i \notin C$ and none of i's descendants are in C that is on the $A - B$ path where the arrows meet head-to-head.

If all paths are blocked then the graph is said to be *d-separated*. D-separation means that the conditional independence statement $A \perp\!\!\!\perp B \mid C$ holds within the joint distribution characterised by the graph. Although this definition encompasses all we need, to make it more intuitive, let us look at how it works, and what it means.

Figure 3.6 summarises the three canonical graphs that were mentioned in the conditions above. Note that the independences that are arrived at below are the only ones that hold explicitly for all distributions with the structures considered. This does not mean that there are not other independences that might hold for some distributions, but these other independences are dependent on the particular case, and do not hold for all distributions with the structures as discussed.



(a) Head-to-tail (b) Tail-to-tail (c) Head-to-head

Figure 3.6: *Three canonical graphs used in discovering conditional independences.*

Head-to-Tail

In Figure 3.6(a) we have a model of this case and we want to establish that given B, A and C are conditionally independent. The factorisation of the distribution associated with the graph in Figure 3.6(a) can be written as

$$p(a, b, c) = p(a)p(b|a)p(c|b).$$

This factorisation implies that

$$\begin{aligned} p(c|a,b) &= \frac{p(a,b,c)}{p(a,b)} \\ &= \frac{p(a)p(b|a)p(c|b)}{p(a)p(b|a)} \\ &= p(c|b) \end{aligned}$$

From Section 2.4, if $p(c|a,b) = p(c|b)$ then $A \perp\!\!\!\perp C \mid B$. Thus we have shown the first case, where a head-to-tail connection blocks the path, creates the desired separation between sets A and B.

The example mentioned in the introduction of Section 2.4, where *Wilgenhof* is a residence at a university and we want to find out whether my younger brother will be in *Wilgenhof*, is an example of this case. A represents my father being in *Wilgenhof*, B my older brother being in *Wilgenhof* and C my younger brother being in *Wilgenhof*. We can see that the path between my father and younger brother is blocked, given my older brother, and thus there is a conditional independence, as per the d-separation algorithm.

Tail-to-Tail

Figure 3.6(b) represents the case of when we have a tail-to-tail connection and we see that the factorisation of the joint distribution can be given by

$$p(a,b,c) = p(c)p(a|c)p(b|c).$$

This factorisation implies

$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(c)p(a|c)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c), \end{aligned}$$

and again, from earlier discussions, if $p(a,b|c) = p(a|c)p(b|c)$ then $A \perp\!\!\!\perp B \mid C$.

For example, suppose that A represents the type of food that an animal receives and B the type of treat that an animal receives and C the type of animal. We can see that there is a connection between these two variables—if the dry food that a pet receives is large pellets, we would assume that it is more likely that it receives a large ostrich-bone as a treat than catnip, since we would think that catnip would be more of a treat for a pet that receives smaller pellets as dry food (preferably smelling a bit more like tuna). Alternatively, a pet that receives seeds as food would more likely be treated by giving it a seed ball. However, if we know that the animal is a dog, a cat or a bird, there is no additional information between the type of food that the animal gets and what they get as a treat. If we know it is a dog, then we know that it gets an ostrich bone, regardless of the type of food that it receives. Similarly for the other cases.

Head-to-Head

The last case, with the connections as in Figure 3.6(c), is slightly different. The joint distribution that is found via the graph is

$$p(a, b, c) = p(a)p(b)p(c|a, b).$$

From this factorisation, and using the marginalisation rule (2.4), we have that

$$\begin{aligned} p(a, b) &= \sum_c p(a, b, c) \\ &= p(a)p(b) \sum_c p(c|a, b) \\ &= p(a)p(b). \end{aligned}$$

Thus, if C is not observed, then A and B are independent of one another, $A \perp\!\!\!\perp B \mid \emptyset$. This is known as marginal independence. However, if C is observed, then A and B are not necessarily independent any more. This can be seen if we condition the joint distribution

$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a, b)}{p(c)}, \end{aligned}$$

and this does not, in general, factorise to $p(a)p(b)$.

An example will most likely explain this type of relationship between the three variables better. Suppose A is the event that there is a special on test-driving black cars today, B is the event that my dad owns a black car and C the event that I am driving a black car today. A could be the reason for C, but so could B. Thus we have the structure of having a head-to-head connection as in Figure 3.6(c). A and B are marginally independent of each other: the fact that there is a special on test-driving black cars near my house and my dad owning a black car are not really dependent on each other. However, if I am seen driving a black car, event C, it could either be because there is a special on test-driving black cars, or it could be because my dad owns a black car and I am borrowing it. Notice $P(A = \text{‘yes’})$ is smaller than $P(A = \text{‘yes’}|C = \text{‘yes’})$. However, $P(A = \text{‘yes’}|B = \text{‘yes’}, C = \text{‘yes’})$ is smaller than $P(A = \text{‘yes’}|C = \text{‘yes’})$. Thus, though it is more likely that there is a special on test-driving black cars if I am seen driving a black car today, if my dad also owns a black car, it is less likely that there is a test-driving special on, and therefore the probability of there being a sale is dependent on whether my dad owns a black car or not, given that I am seen driving one.

In Table 3.1 and Table 3.2 the full list of conditional independences for the graphs H_1 and H_2 as in Figure 3.5 are given. The lists were compiled in such a way that the ‘main’ independence is given in the first column, with other nodes that could be combined with the main independence to form new sets in the second column. In the second column, any or all of the nodes, or sets of nodes, can be included. As an example, in Table 3.1 the conditional independence $X_1 \perp\!\!\!\perp X_5 \mid X_3$ is stated in the first column, with $X_2, X_4, \{X_2, X_6\}$ in the second column. The independence $X_1 \perp\!\!\!\perp X_5 \mid X_3$ is valid on its own, but if either X_2, X_4, or $\{X_2, X_6\}$ is added, the conditional independence still holds. Note that if X_6 is added, then X_2 has to be added as well, that is why they are grouped together in a set.

As can be seen by comparing the basic conditional independences (in Table 2.1) and these tables, there are some conditional independences that were not captured in the basic list, such as, for H_1, that $X_1 \perp\!\!\!\perp X_6 \mid \{X_2, X_3\}$. This would be due to the fact that X_3 is not a parent of X_6, but still 'blocks' the path to X_6 from X_1 if observed.

Table 3.1: *Full list of conditional independences for H_1.*

Conditional Independences	Other nodes that can be in observed set
$X_1 \perp\!\!\!\perp X_4 \mid X_2$	X_3, X_5, X_6
$X_1 \perp\!\!\!\perp X_5 \mid X_3$	$X_2, X_4, \{X_2, X_6\}$
$X_1 \perp\!\!\!\perp X_6 \mid \{X_2, X_5\}$	X_3, X_4
$X_1 \perp\!\!\!\perp X_6 \mid \{X_2, X_3\}$	X_4, X_5
$X_2 \perp\!\!\!\perp X_3 \mid X_1$	$X_4, X_5, \{X_5, X_6\}$
$X_2 \perp\!\!\!\perp X_5 \mid X_3$	X_1, X_4
$X_2 \perp\!\!\!\perp X_5 \mid X_1$	X_3, X_4
$X_3 \perp\!\!\!\perp X_4 \mid X_1$	$X_2, X_5, \{X_2, X_6\}, \{X_5, X_6\}$
$X_3 \perp\!\!\!\perp X_4 \mid X_2$	X_1, X_5, X_6
$X_3 \perp\!\!\!\perp X_6 \mid \{X_2, X_5\}$	X_1, X_4
$X_3 \perp\!\!\!\perp X_6 \mid \{X_1, X_5\}$	X_2, X_4
$X_4 \perp\!\!\!\perp X_5 \mid X_3$	X_1, X_2
$X_4 \perp\!\!\!\perp X_5 \mid X_1$	X_2, X_3
$X_4 \perp\!\!\!\perp X_5 \mid X_2$	X_1, X_3, X_6
$X_4 \perp\!\!\!\perp X_6 \mid X_2$	X_1, X_3, X_5

Bayes' ball theorem

Another way to examine graph separation intuitively is Bayes' ball theorem, though formally it is the same as d-separation. For a discussion on this algorithm, see [23, chapter 2].

3.3.4 Characterisation of Directed Graphical Models

As previously stated, graphical models are associated with a family of joint distributions. Let us look at two ways of defining this family.

Table 3.2: *Full list of conditional independences for H_2.*

Conditional Independences	Other nodes that can be in observed set
$Y_1 \perp\!\!\!\perp Y_3 \mid Y_2$	$Y_4, \{Y_5, Y_6\}, \{Y_5, Y_6, Y_7\}$
$Y_1 \perp\!\!\!\perp Y_4 \mid \emptyset$	Y_2, Y_3, Y_7
$Y_1 \perp\!\!\!\perp Y_5 \mid Y_2$	$Y_3, Y_4, \{Y_3, Y_7\}$
$Y_1 \perp\!\!\!\perp Y_5 \mid Y_3$	Y_2, Y_4, Y_7
$Y_1 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_6\}$	Y_3, Y_4, Y_5
$Y_2 \perp\!\!\!\perp Y_4 \mid \emptyset$	Y_1, Y_3, Y_7
$Y_2 \perp\!\!\!\perp Y_5 \mid Y_3$	Y_1, Y_4, Y_7
$Y_2 \perp\!\!\!\perp Y_6 \mid \{Y_1, Y_3\}$	Y_4, Y_5
$Y_2 \perp\!\!\!\perp Y_6 \mid \{Y_1, Y_5\}$	Y_3, Y_4
$Y_3 \perp\!\!\!\perp Y_5 \mid \emptyset$	Y_1, Y_2
$Y_3 \perp\!\!\!\perp Y_6 \mid \{Y_1, Y_5\}$	Y_2, Y_4
$Y_3 \perp\!\!\!\perp Y_6 \mid \{Y_2, Y_5\}$	Y_1, Y_4
$Y_4 \perp\!\!\!\perp Y_6 \mid \{Y_1, Y_5\}$	$Y_2, Y_3, \{Y_3, Y_7\}$
$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_1, Y_6\}$	Y_2, Y_3
$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_6\}$	Y_1, Y_3
$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_3, Y_5\}$	Y_1, Y_2, Y_6
$Y_5 \perp\!\!\!\perp Y_7 \mid \{Y_1, Y_3, Y_6\}$	Y_2, Y_4
$Y_5 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_3, Y_6\}$	Y_1, Y_4

Let a family, say $\mathcal{D}_1$, be a family of distributions that is found from ranging over all the possible values of $\{p(x_i|x_{\pi_i})\}$ in the joint distribution definition

$$p(x) = \prod_{i=1}^{N} p(x_i|x_{\pi_i}).$$

Let the second family be found via the list of the conditional independent statements found from the graph associated with the same joint probability distribution. This list is always finite, since there are a finite amount of nodes and edges in the graph. Once this list is made, consider all the joint probability distributions $p(x_1, \ldots, x_N)$, making no restrictions at first. For every distribution, consider the conditional independence statements that are in the list, and if the joint probability distribution under consideration fulfils all the conditional independence statements, then it is in the second family $\mathcal{D}_2$. Note that a distribution may have more conditional independence statements than those in the list, but this does not matter, since if a distribution is more restricted than necessary, it still means that it falls in a larger family with less restrictions. One can check whether these conditional independent statements hold by viewing the factorisations of the joint distribution.

These two definitions of directed graphical models are, in fact, equivalent. For a proof of this, see [23, chapter 16]. Unfortunately, even though these characterisations of directed graphical models are at the centre of the graphical model formalism and provides the strong link between graph theory and probability theory, there lies too much theory behind it for it to be proven here. The link it provides is being able to view probability distributions via numerical parametrisations ($\mathcal{D}_1$) and conditional independence statements ($\mathcal{D}_2$) as being completely equivalent, and being able to use both to analyse and infer from the probability distribution.

3.4 Markov Random Fields

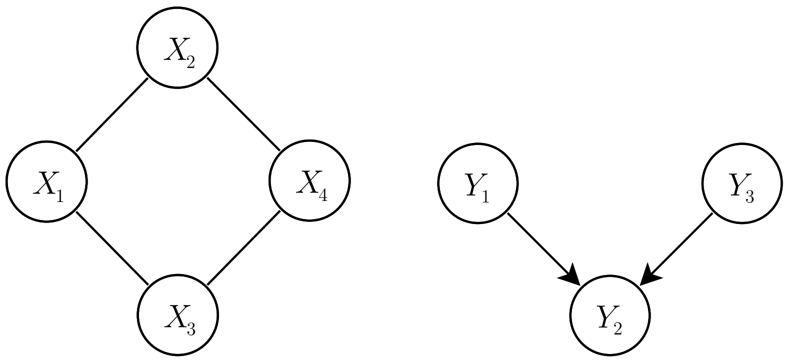
The only difference between a directed graph and an undirected graph, graph-theoretically, is that the undirected graph has undirected edges, i.e. they have no arrows. However, when looking at them from a graphical model point of view, there are other, important, differences. For example, the way conditional independences are dealt with in an MRF, and the factorisation of the joint distributions which they represent, differ from BNs. MRFs are useful when the distribution that we are looking at has global constraints which can easily be separated into sets of local constraints.

We have seen how probability distributions can be characterised by directed graphical models but there are some distributions that cannot be represented by a directed graphical model, and for these one can use undirected graphical models. We see in Section 3.4.1 how to characterise an MRF by means of conditional independent statements, but for now assume that the graph in Figure 3.7(a) has the conditional independent statements

$$\begin{aligned} X_1 &\perp\!\!\!\perp X_4 \quad | \quad \{X_2, X_3\}, \\ X_2 &\perp\!\!\!\perp X_3 \quad | \quad \{X_1, X_4\}. \end{aligned}$$

Using the same nodes and trying to model these conditional independences via a directed graphical model results in the following problem. Since we use DAGs, there is at least one node with a head-to-head connection, since cycles are not allowed. Assume, without loss of generality, that this node is X_1. From our discussion in Section 2.4, X_2 and X_3 therefore may not necessarily be conditionally independent given X_1. $X_2 \perp\!\!\!\perp X_3 \mid \{X_1, X_4\}$ may therefore not hold, because, given X_1, it is possible that X_2 and X_3 become dependent, no matter whether X_4 is a head-to-tail or tail-to-tail node. It is clear, therefore, that there are joint distributions that can be represented by an MRF but not by a DAG.

But what about representing all graphs as MRFs? Figure 3.7(b) is an example of why this also cannot happen—but to understand the example, the conditional independence statement characterisation of MRFs first needs to be established.



(a) MRF that cannot be modelled as a BN

(b) BN that cannot be modelled as an MRF

Figure 3.7: *Illustrating the need for both directed and undirected graphical models.*

3.4.1 From Undirected Graphs to Probability Theory

When we discussed directed graphs, we first looked at factorised parametrisation before looking at conditional independence. In undirected graphs, however, first looking at the graph and how conditional independences can be represented in it is more intuitive, and thus we first discuss conditional independent axioms, and then derive the factorisation.

As (3.9) held for directed graphs, we want to find a way of representing the conditional independences in an undirected graphical model. We thus want to say whether $X_A \perp\!\!\!\perp X_B \mid X_C$ is true for a specific graph or not. In undirected graphs, as in directed graphs, we look at whether X_C separates the graph, causing us to be unable to move along a path from X_A to X_B. Here the idea of graph separation is much simpler—if a group of nodes X_C separates one group of nodes X_A from another group of nodes X_B, then $X_A \perp\!\!\!\perp X_B \mid X_C$ holds, there are no special cases like head-to-head connections in directed graphs. Looking at it again in terms of paths, as in the directed case, if every path from any node in X_A to any node in X_B includes a node that lies in X_C, then $X_A \perp\!\!\!\perp X_B \mid X_C$ holds, else $X_A \perp\!\!\!\perp X_B \nmid X_C$. As an illustration, in Figure 3.8 $X_A \perp\!\!\!\perp X_B \mid X_C$ holds, where the set of indices A, B and C are $A = \{1, 2, 3, 4, 5\}$, $B = \{9, 10, 11, 12\}$ and $C = \{6, 7, 8\}$. If the nodes X_1 and X_{11} were to be connected, however, then $X_A \perp\!\!\!\perp X_B \nmid X_C$, since there would then be a path from the set X_A to the set X_B that does not include a node from the set X_C, namely the path $X_1 - X_{11}$. Seeing whether a conditional independence holds in a certain distribution using an undirected graph is therefore merely a reachability problem in graph-theoretical terms, with only needing to take out X_C from the graph and see whether, starting from a node in X_A, one can move along a path to a node in X_B.

Returning to the example in the introduction of this section, we know from our discussion that the graph in Figure 3.7(b) is characterised by $Y_1 \perp\!\!\!\perp Y_3$, but that $Y_1 \perp\!\!\!\perp Y_3 \mid Y_2$ does not necessarily hold. To represent the independence $Y_1 \perp\!\!\!\perp Y_3$ on an undirected graph, there

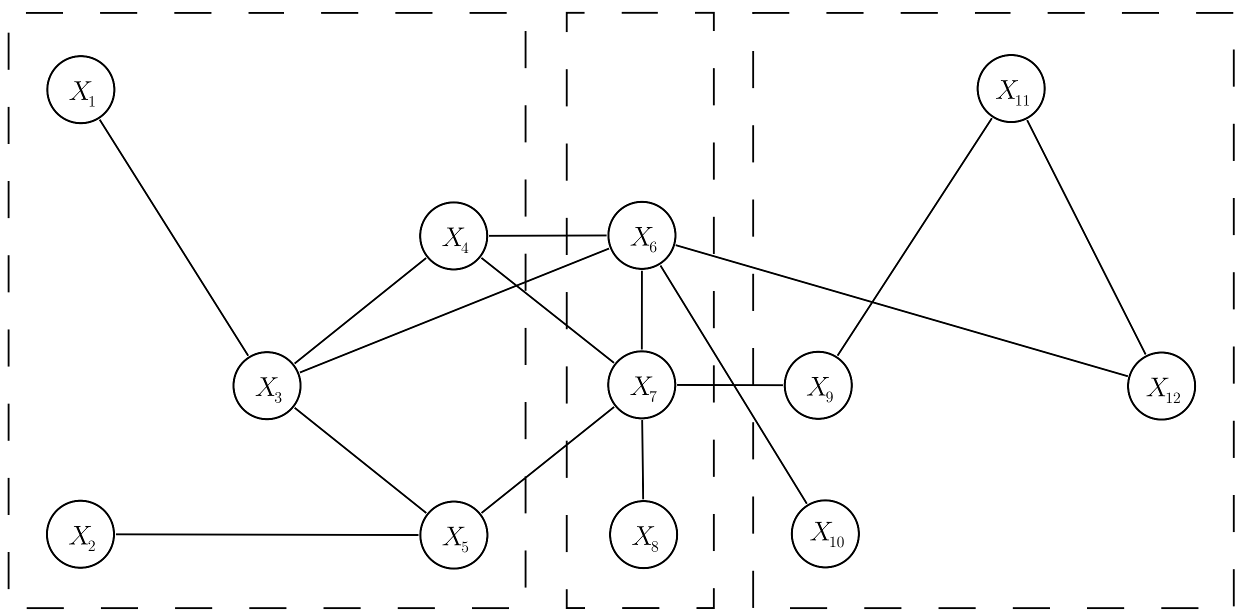


Figure 3.8: *MRF with three sets* $X_A,\ X_B,\ X_C$ *s.t.* $X_A \perp\!\!\!\perp X_B \mid X_C$.

should be no edge between the nodes Y_1 and Y_3. However, if Y_2 is given, then it is possible that Y_1 and Y_3 are no longer independent, and to represent this possible dependence, there should be an edge between Y_1 and Y_3. We are therefore in the position that the one condition states that there should be an edge and the other that there should not be an edge. It is not possible to represent this using an undirected graph and therefore it is not possible to illustrate this case using an MRF. We have therefore seen that both undirected and directed graphs are necessary, where some conditional independences can only be represented by one of the types, and not the other one.

3.4.2 From Probability Theory to Undirected Graphs

In the directed graphical model, we found a parametrisation which saw the local functions chosen as the joint probabilities. In the undirected case, although the conditional independences are easier, the parametrisation is not as simple as in the directed case, and a few concessions that, ideally we would not want to make, must be made. The first difference in the way that we view parametrisations in the different graphs, is the definition of the local area. In the undirected graph there is no notion of ancestors, parents and children as in the directed case, and therefore we need to find another way of defining the local area. Intuitively looking at the node and its neighbours as a local area would be a logical avenue to explore. However, this definition has major consistency problems. These problems lead to our choice of functions not being independent or arbitrary, two properties we deem important.

So what are the properties that we want in a local area? When looking at the conditional independences, we see that if a node X_i is not directly connected to X_j, then they are conditionally independent given all the other nodes. So from the 'independence' point of view, we want the nodes that are not connected to be in different functions. On the other hand, if nodes are dependent on each other, with dependence being the lack of independence, one would want them to be in the same function. Therefore having local areas as the areas in a graph that are all connected to each other fulfils our needs. Recall from Section 3.1.2 that cliques are defined to include all the nodes that are connected to each other and exclude all those that are not directly connected. Using cliques as defining the locality of the functions becomes a valid option. Due to every clique being the subset of at least one maximal clique, we only need to look at the maximal cliques when defining the local functions. Although in some instances one would want to relax this condition, in theory one does not need to consider cliques that are not maximal, and also, in our application of graphical models, we only consider maximal cliques. In summary, we defined the local functions over maximal cliques.

Let C indicate the set of indices of a maximal clique in graph G, and let $\mathcal{C}$ be the entire set of all C. Furthermore, let $\psi_{X_C}(x_C)$ be the local potential function on the possible realisations x_C of the maximal clique X_C, with the restriction that it has to be nonnegative and real-valued. As mentioned earlier, there are some concessions that have to be made with defining this arbitrary function as is. One of these is finding the normalisation factor when working out the joint distribution, due to the fact that there is no reason for arbitrary functions, unlike conditional probabilities, to be normalised. The joint probability is then defined as

$$p(x) := \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_{X_C}(x_C), \tag{3.11}$$

with normalisation factor

$$Z := \sum_{x} \prod_{C \in \mathcal{C}} \psi_{X_C}(x_C). \tag{3.12}$$

From (3.12) we see that finding Z can become quite expensive, if the number of configurations becomes large, since it sums over all configurations of x. Using the distributive law on (3.12) does speed up finding Z, but, in general, it may still take too long for practical purposes to find Z. However, it is not always necessary to find Z. When we find the conditional probability of a variable, it is the ratio of two marginalised probabilities—both containing the factor Z. Due to the fact that they both have Z, Z in effect 'cancels out'. One could possibly look at finding the conditional probabilities by summing across the unnormalised probabilities and thereafter taking the ratio of the sums, in so doing not having to deal with the normalisation factor at all.

Unfortunately, the potential functions defined on MRF have no local probabilistic interpretations, unless the MRF was found from a BN. As will be explained in Section 3.5 one can convert a directed graph into an undirected graph but the price to pay is that some conditional independences may be lost. When looking at an MRF that was converted from a BN, the potential functions $\psi_{X_C}(x_C)$ are given the values of the conditional probabilities

$p(x_C)$ in such a way that, as long as the variables in the potential function $p(x_C)$ are all part of clique C, for every potential function, they can be arbitrarily assigned to the cliques.

Arbitrarily chosen functions are not necessarily nonnegative, but because potential functions are required to be, the exponential function is often employed. Doing this, the potential function becomes

$$\psi_{X_C}(x_C) = e^{-E(x_C)},$$

with $E(x_C) = \sum_{C \in \mathcal{C}} E_C(x_C)$ called an energy function. This exponential representation of $\psi_{X_C}(x_C)$ is also known as the *Boltzmann distribution*. The joint probability distribution now becomes

$$\begin{aligned} p(x) &= \frac{1}{Z} \prod_{C \in \mathcal{C}} e^{-E_C(x_C)} && (3.13) \\ &= \frac{1}{Z} e^{-\sum_{C \in \mathcal{C}} E_C(x_C)} && (3.14) \\ &= \frac{1}{Z} e^{-E(x_C)}. && (3.15) \end{aligned}$$

The Boltzmann distribution originates in Physics where it relates energy levels to probabilities. This is also what we are doing here—taking arbitrary functions and relating them to probabilities.

Suppose that we have the distribution $p(x) = p(x_1)p(x_1|x_3)p(x_2|x_3)$. We see that there are two maximal cliques that need to be defined, $X_1 - X_3$ and $X_2 - X_3$, if we choose the potential functions as

$$\begin{aligned} \psi_{1,3}(x_1, x_3) &= p(x_1)p(x_1|x_3), \\ \psi_{2,3}(x_2, x_3) &= p(x_2|x_3). \end{aligned}$$

To draw the graphical model representing this choice of potential function, we draw an edge between all the variables in a joint or conditional probability. Therefore there is an edge between X_1 and X_3, as well as X_2 and X_3. This graph is illustrated in Figure 3.9. Note that we could have also just defined one maximal clique $X_1 - X_2 - X_3$ on which all the conditional probabilities would then have been defined.

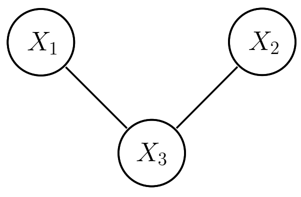


Figure 3.9: *MRF representing* $p(x) = p(x_1)p(x_1|x_3)p(x_2|x_3)$.

3.4.3 Characterisation of Undirected Graphical Models

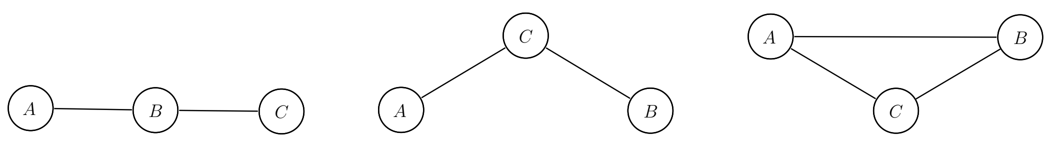
In Section 3.3.4 we defined the two families of directed graphical models—one found from the parametrisation, and one from the conditional independence statements. We now do the same for undirected graphical models.

Let U_1 be the family of probability distributions found by ranging over all possible choices of positive functions on the maximal cliques of a undirected graph G. Let U_2 be the family of probability distributions found via the conditional independence statements associated with graph G. Again, list all the conditional independent statements $X_A \perp\!\!\!\perp X_B \mid X_C$ associated with the graph. Thereafter, iterate through the possible probability distributions, and discard a probability distribution as soon as there is a conditional independent statement that does not hold in it that is in the list, if all the statements hold, then the probability distribution is in the family U_2. Again it is found that the above families are equivalent. The Hammersley-Clifford theorem states this equivalence, and a discussion on this theorem can be found in [23, chapter 16].

3.5 Converting a Directed Graph to an Undirected Graph

Although neither MRF nor BN have the ability to represent all conditional independences, and thus both are needed, one can turn a directed graph into an undirected graph quite easily. The price that is paid for doing this is the loss of some of the conditional independent properties that are held in the directed graph.

Consider again the canonical directed graphs in Figure 3.6. We now try to find a way to convert BNs into MRFs in a general way, keeping as many conditional independences as possible.



(a) Head-to-tail (b) Tail-to-tail (c) Head-to-head

Figure 3.10: *Converting directed canonical graphs, in Figure 3.6, into undirected graphs.*

Head-to-Tail

Figure 3.6(a) has the joint distribution $p(a, b, c) = p(a)p(c|a)p(b|c)$. In Figure 3.10(a) the joint distribution is given, over maximal cliques $\{A, C\}$ and $\{C, B\}$, by $p(a, b, c) =$

$\psi_{A,C}(a,c)\psi_{C,B}(c,b)$. If one chooses

$$\begin{aligned}\psi_{A,C}(a,c) &= p(a)p(c|a)\\ \psi_{C,B}(c,b) &= p(b|c)\end{aligned}$$

then it is clear that the undirected graph in Figure 3.10(a) keeps the same conditional independences as the directed graph in Figure 3.6(a). For all head-to-tail connections this conversion holds.

Tail-to-Tail

Figure 3.6(b) has the joint distribution $p(a,b,c) = p(a)p(a|c)p(b|c)$. In Figure 3.10(b) the joint distribution is given, over maximal cliques $\{A,C\}$ and $\{C,B\}$, by $p(a,b,c) = \psi_{A,C}(a,c)\psi_{C,B}(c,b)$. If one chooses

$$\begin{aligned}\psi_{A,C}(a,c) &= p(c)p(c|a)\\ \psi_{C,B}(c,b) &= p(b|c)\end{aligned}$$

then it is clear that the undirected graph in Figure 3.10(b) keeps the same conditional independences as the directed graph in Figure 3.6(b). Note that $p(c)$ was chosen at random to be included in $\psi_{A,C}$ and could just as well been chosen to be incorporated into $\psi_{C,B}$, the result would have been the same—an undirected graph where the conditional independences hold. This conversion naturally holds for all tail-to-tail connections.

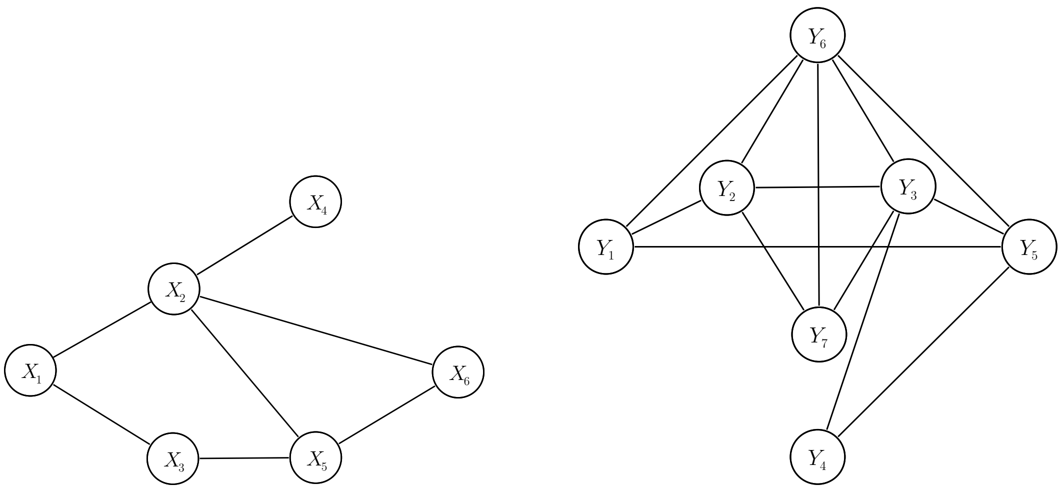
Head-to-Head

Figure 3.6(c) has the joint distribution $p(a,b,c) = p(a)p(b)p(c|a,b)$. In Figure 3.10(c) the joint distribution is given, over maximal cliques $\{A,C\}$ and $\{C,B\}$, by $p(a,b,c) = \psi_{A,B,C}(a,b,c)$.

As in discussing conditional independences in the directed graphical model, this is the case where a bit more work needs to be done. From the example in Figure 3.7 we know that simply converting the directed edges of a head-to-head node into undirected edges as in the previous cases is not sufficient. We know that when there is a probability defined over various variables, then those variables need to be together in at least one clique. Thus, because the term $p(c|a,b)$ has all three variables a, b and c in it, there needs to be a maximal clique such that A, B and C are connected. We connect the parents of C—which ensures that they are all three in a maximal clique together. This process, of *marrying the parents*, is called *moralising* the graph, and the resulting undirected graph, after converting the other directed edges into undirected edges, is known as a *moral* graph. Once we have moralised the graph, it appears as in Figure 3.10(c). Due to the fact that it is now a complete graph, we see that there are no special conditional independences that hold. It is due to this fact, of moralising a graph, that converting a directed graph into an undirected graph causes some of the conditional independences to fall away. By moralising the graph one adds the fewest possible edges whilst retaining the maximal number of independence properties that is held in the directed graph. The clique potential for Figure 3.10(c) is therefore

$$\psi_{A,B,C}(a,b,c) = p(a)p(b)p(c|a,b).$$

After looking at the three canonical graphs, a procedure for converting a BN to a MRF is now forthcoming. First draw undirected edges connecting the parents of nodes and then convert all directed edges into undirected edges such that all edges are now undirected (moralizing the graph). To find the potential functions of the undirected graph, initialise all ψ_i's to 1. Thereafter, for every conditional distribution factor in the joint distribution function of the directed graph, multiply it into one of the clique potentials, since there is at least one maximal clique that contains all the nodes in a specific conditional distribution now that we have a moral graph. Since there might be some factors that can be placed in more than one clique, arbitrarily chose any one to put conditional probability in. However, do not put these probabilities in more than one ψ_i.



(a) Graph H_1 moralised (b) Graph H_2 moralised

Figure 3.11: *Moralised graphs of H_1 and H_2.*

Figure 3.11(a) and Figure 3.11(b) are the moralised graphs of Figure 3.5(a) and Figure 3.5(b), respectively, and therefore also the undirected graphs converted from the directed graphs. Notice that in H_1, X_2 and X_5 have been married, and in H_2, the parent nodes of Y_6, Y_1 and Y_5, are married. The clique potentials, as we chose them, associated with H_1 in Figure 3.11(a) are

$$\begin{aligned}
\psi_{1,2}(x_1,x_2) &= p(x_1)p(x_2|x_1)\\
\psi_{1,3}(x_1,x_3) &= p(x_3|x_1)\\
\psi_{2,4}(x_2,x_4) &= p(x_4|x_2)\\
\psi_{3,5}(x_3,x_5) &= p(x_5|x_3)\\
\psi_{2,5,6}(x_2,x_5,x_6) &= p(x_6|x_2,x_5),
\end{aligned}$$

and clique potentials, as we chose them, associated with H_2 in Figure 3.11(b) are

$$\begin{aligned}
\psi_{1,2,6}(y_1,y_2,y_6) &= p(y_2|y_1)\\
\psi_{1,5,6}(y_1,y_5,y_6) &= p(y_6|y_1,y_5)p(y_1)\\
\psi_{2,3,6,7}(y_2,y_3,y_6,y_7) &= p(y_3|y_2)p(y_7|y_2,y_3,y_6)\\
\psi_{3,4,5}(y_3,y_4,y_5) &= p(y_5|y_3,y_4)p(y_4).
\end{aligned}$$

Note, that in the clique potential of H_2, $p(y_1)$ could also have been included in clique potential $\psi_{1,2,6}$. Furthermore, notice that there is a clique in this graph over Y_3, Y_5 and Y_6 that is not included in the list of clique potentials above. The reason for this would be that there are no factors that involve those three variables and thus it is not necessary to.

The list of conditional independences that hold in the moralised graphs of H_1 and H_2 are given in Table 3.3. When these lists are compared to those of the directed graphical models, in Table 2.1, we can see that there are some conditional independences that do not hold at all any more, like $Y_3 \perp\!\!\!\perp Y_4 \mid \emptyset$ that held in the directed graph H_2, but in the moralised graph, Y_3 and Y_4 are linked, and are therefore dependent on each other. In other conditional independences, the variables might still be conditionally independent, but on a different set. As an example of this, in H_1, $X_1 \perp\!\!\!\perp X_5 \mid X_3$ holds but in the moralised H_1 this conditional independence does not hold, but if the set is enlarged to $\{X_2, X_3\}$, then X_1 and X_5 are conditionally independent.

This conversion of directed graphs to undirected graphs plays an important role in exact inference techniques—where one would prefer to work with undirected graphs as far as possible. Converting an undirected graph into a directed graph, although possible, is much less common, since there is again the issue of the normalisation factor that needs to be incorporated into the joint probability distribution.

3.6 Local Messages from Distributive Law

In Section 2.5 we illustrated how using the distributive law helps to find the marginals of a joint distribution more efficiently. However, using the distributive law is also a key element in

Table 3.3: *Conditional independences for moralised H_1 and H_2.*

moralised H_1	moralised H_2
$X_1 \perp\!\!\!\perp X_4 \mid X_2$	$Y_1 \perp\!\!\!\perp Y_3 \mid \{Y_2, Y_5, Y_6\}$
$X_1 \perp\!\!\!\perp X_5 \mid \{X_2, X_3\}$	$Y_1 \perp\!\!\!\perp Y_4 \mid \{Y_2, Y_5, Y_6\}$
$X_1 \perp\!\!\!\perp X_6 \mid \{X_2, X_3\}$	$Y_1 \perp\!\!\!\perp Y_4 \mid \{Y_3, Y_5\}$
$X_1 \perp\!\!\!\perp X_6 \mid \{X_2, X_5\}$	$Y_1 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_3, Y_6\}$
$X_2 \perp\!\!\!\perp X_3 \mid \{X_1, X_5\}$	$Y_1 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_5, Y_6\}$
$X_3 \perp\!\!\!\perp X_4 \mid \{X_1, X_5\}$	$Y_2 \perp\!\!\!\perp Y_4 \mid \{Y_1, Y_3, Y_6\}$
$X_3 \perp\!\!\!\perp X_4 \mid X_2$	$Y_2 \perp\!\!\!\perp Y_4 \mid \{Y_3, Y_5\}$
$X_3 \perp\!\!\!\perp X_6 \mid \{X_1, X_5\}$	$Y_2 \perp\!\!\!\perp Y_5 \mid \{Y_1, Y_3, Y_6\}$
$X_3 \perp\!\!\!\perp X_6 \mid \{X_2, X_5\}$	$Y_4 \perp\!\!\!\perp Y_6 \mid \{Y_3, Y_5\}$
$X_4 \perp\!\!\!\perp X_5 \mid X_2$	$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_1, Y_3, Y_6\}$
$X_4 \perp\!\!\!\perp X_6 \mid X_2$	$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_3, Y_6\}$
	$Y_4 \perp\!\!\!\perp Y_7 \mid \{Y_3, Y_5\}$
	$Y_5 \perp\!\!\!\perp Y_7 \mid \{Y_1, Y_3, Y_6\}$
	$Y_5 \perp\!\!\!\perp Y_7 \mid \{Y_2, Y_3, Y_6\}$

the process of sending local messages around in a graph. To illustrate this concept, consider the distribution,

$$p(x_1, x_2, \ldots, x_N) = \frac{1}{Z} \prod_{i}^{N-1} \psi(x_i, x_{i+1}). \tag{3.16}$$

This distribution is given by the graph in Figure 3.12. Note that henceforth the subscripts are dropped when looking at MRF potentials, thus $\psi_{k,m}(x_k, x_m) = \psi(x_k, x_m)$.

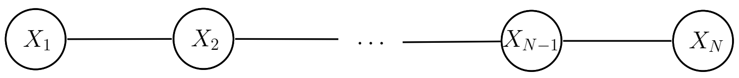


Figure 3.12: *Graph of (3.16).*

Firstly, suppose that there is no observed data, and we are interested in finding the marginal

$p(x_j)$, where $j \in V$. Then using the distributive law

$$\begin{aligned}
p(x_j) &= \frac{1}{Z}\sum_{x_1}\psi(x_1,x_2)\cdot\sum_{x_2}\psi(x_2,x_3)\cdot\ldots\cdot\sum_{x_{j-1}}\psi(x_{j-1},x_j)\cdot\sum_{x_{j+1}}\psi(x_j,x_{j+1})\cdot\ldots \\
&\quad \cdot\sum_{x_{N-1}}\psi(x_{N-2},x_{N-1})\cdot\sum_{x_N}\psi(x_{N-1},x_N) \qquad (3.17)\\
&= \frac{1}{Z}\left[\sum_{x_{j-1}}\psi(x_{j-1},x_j)\cdot\ldots\cdot\left[\sum_{x_2}\psi(x_2,x_3)\cdot\left[\sum_{x_1}\psi(x_1,x_2)\right]\right]\ldots\right]\cdot \\
&\quad \left[\sum_{x_{j+1}}\psi(x_j,x_{j+1})\cdot\ldots\cdot\left[\sum_{x_{N-1}}\psi(x_{N-2},x_{N-1})\cdot\left[\sum_{x_N}\psi(x_{N-1},x_N)\right]\right]\ldots\right] \qquad (3.18)\\
&= \frac{1}{Z}\mu_\alpha(x_j)\mu_\beta(x_j), \qquad (3.19)
\end{aligned}$$

with

$$\begin{aligned}
\mu_\alpha(x_j) &= \left[\sum_{x_{j-1}}\psi(x_{j-1},x_j)\cdot\ldots\cdot\left[\sum_{x_2}\psi(x_2,x_3)\cdot\left[\sum_{x_1}\psi(x_1,x_2)\right]\right]\ldots\right] \\
&= \sum_{x_{j-1}}\psi(x_{j-1},x_j)\left[\sum_{x_{j-2}}\psi(x_{j-2},x_{j-1})\ldots\right] \\
&= \sum_{x_{j-1}}\psi(x_{j-1},x_j)\mu_\alpha(x_{j-1}), \\
\mu_\beta(x_j) &= \left[\sum_{x_{j+1}}\psi(x_j,x_{j+1})\cdot\ldots\cdot\left[\sum_{x_{N-1}}\psi(x_{N-2},x_{N-1})\cdot\left[\sum_{x_N}\psi(x_{N-1},x_N)\right]\right]\ldots\right] \\
&= \sum_{x_{j+1}}\psi(x_j,x_{j+1})\left[\sum_{x_{j+2}}\psi(x_{j+1},x_{j+2})\right] \\
&= \sum_{x_{j+1}}\psi(x_j,x_{j+1})\mu_\beta(x_{j+2}).
\end{aligned}$$

The calculation of $\mu_\alpha(x_j)$ and $\mu_\beta(x_j)$ follow the same logic, and thus we only describe the method of finding $\mu_\alpha(x_j)$. The evaluation of $\mu_\alpha(x_j)$ starts by summing over the variable x_1, thus finding $\mu_\alpha(x_2)$ first. The reason for this is that x_2 is in both the last and second last terms, and thus one would want to sum over both of these terms when summing over x_2. Once x_1 has been summed over, we sum over x_2, such that $\mu_\alpha(x_3) = \sum_{x_2}\psi(x_2,x_3)\mu_\alpha(x_2)$. In general, $\mu_\alpha(x_j) = \sum_{x_{j-1}}\psi(x_{j-1},x_j)\mu_\alpha(x_{j-1})$ is recursively calculated until the desired node is reached (unless $j = 1$ in which case $\mu_\alpha(x_j) = 0$ and only $\mu_\beta(x_j)$ is calculated).

Looking at this from a graph-theoretical point of view, a message, $\mu_\alpha(x_j)$ is sent from one node to the next, starting at a leaf-node. In each summand, the incoming message $\mu_\alpha(x_{j-1})$ is multiplied by the potential on the node before the summing over the node variable. This passing of messages is illustrated in Figure 3.13. Note that each node, once it has been

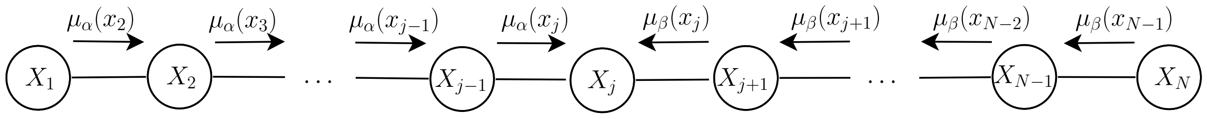


Figure 3.13: *Illustration of how local messages are sent.*

summed over, can be viewed as being removed from the graph, until only the desired node, X_j is left.

The above example, explaining the sending of messages on a chain graph, gives us an intuitive idea of what happens when the marginal of a specific node needs to be found, and forms the basis of the elimination algorithm. This method, however, is not yet optimal, since the same procedure needs to be repeated for every marginal of the distribution. This is remedied in the junction tree algorithm, discussed in Section 4.4.

Although the elimination algorithm is not defined formally here, we present two more examples to illustrate its detail.

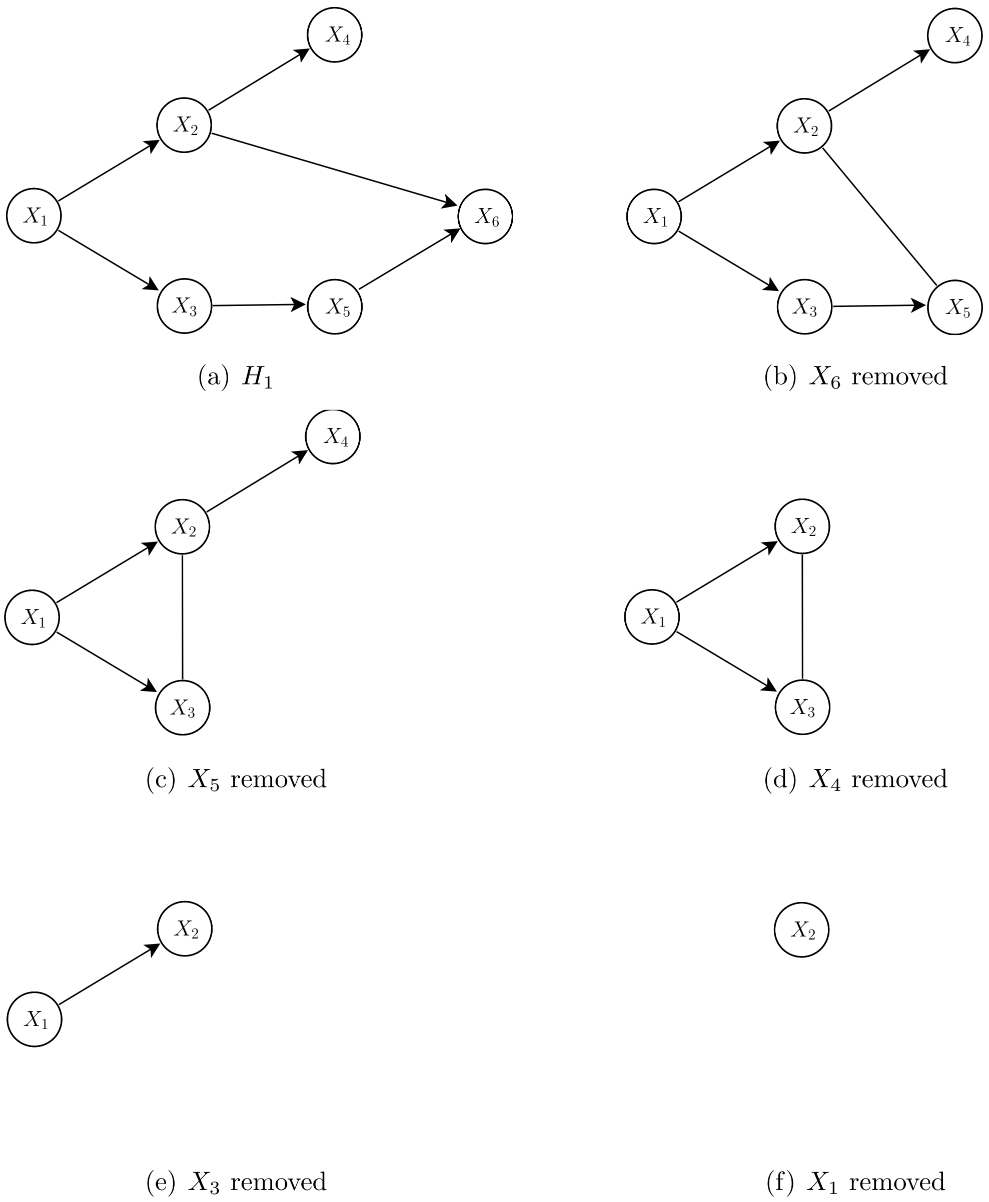
Suppose we want to find $p(x_2)$ given the distribution

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2,x_5).$$

Note that this is our, now familiar, example associated with graph H_1 (see Figure 3.5(a)). Let the notation for messages, for now, be $m_i(x_j)$, where i is the variable that is summed over, and x_j is the node(s) that is in the summand excluding the variable that is summed over. For example, $m_1(x_2) = \sum_{x_1} \psi(x_1,x_2)$. For illustrative purposes, let us 'forget' that $\sum_{x_1} p(x_1) = 1$. Then, from the distributive law,

$$\begin{aligned}
p(x_2) &= \sum_{x_1} p(x_1)p(x_2|x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_4} p(x_4|x_2)\sum_{x_5} p(x_5|x_3)\sum_{x_6} p(x_6|x_2,x_5)\\
&= \sum_{x_1} p(x_1)p(x_2|x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_4} p(x_4|x_2)\sum_{x_5} p(x_5|x_3)m_6(x_2,x_5)\\
&= \sum_{x_1} p(x_1)p(x_2|x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_4} p(x_4|x_2)m_5(x_2,x_3)\\
&= \sum_{x_1} p(x_1)p(x_2|x_1)\sum_{x_3} p(x_3|x_1)m_4(x_2)m_5(x_2,x_3)\\
&= \sum_{x_1} p(x_1)m_3(x_1,x_2)m_4(x_2)\\
&= m_1(x_2)m_4(x_2).
\end{aligned}$$

In Figure 3.14 the message passing is illustrated in more detail. When summing over a particular variable, an intermediate factor $m_i(x_k)$, $i \in V$ and x_k, which is possibly a set of variables, is created. This message then creates a relationship between the variables that are in the summand, excluding the variable being summed over. For example, when summing over x_6 the message $m_6(x_2, x_5)$ is formed, which depends on both x_2 and x_5. This is represented by connecting X_2 and X_5 when removing X_6 from the graph, as illustrated in Figure 3.14(b). Since there is no information about the induced dependency created by the summation between the variables, the edge we add is undirected. Note that in the elimination algorithm, directed graphs are always moralised before running the algorithm. These undirected edges that are added are 'legal' in the sense that all edges in the graph are undirected. Likewise, when summing over x_5, an induced dependency is created between x_2 and x_3.



(a) H_1 (b) X_6 removed

(c) X_5 removed (d) X_4 removed

(e) X_3 removed (f) X_1 removed

Figure 3.14: *Graph illustrating local message passing graph-theoretically.*

When considering the marginal distribution $p(x_2) = m_1(x_2)m_4(x_2)$, it can be seen that there is a message sent from the direction of X_4 and also from the direction of X_1. This is intuitive when looking at the graph H_1, since there are two parts to the graph. The node X_2

separates the graph into two parts—one containing X_1, X_2, X_3, X_5 and X_6 and the other part containing X_2 and X_4. These are then also the two different directions that the messages come from: from X_1, and from X_4. Had we started the marginalisation at node X_1, the two messages would have come from X_6 and from X_4. The *elimination order*, the order in which summations are performed, in our example is $(6, 5, 4, 3, 1, 2)$. Other elimination orders could have been used, though there are good and bad elimination orders, [23]. The only restriction on choosing an elimination order is that the query node must be last .

Note that if there are observed nodes in the graph, the corresponding variables' values are set, and therefore there is no summation over that variable.

Chapter 4

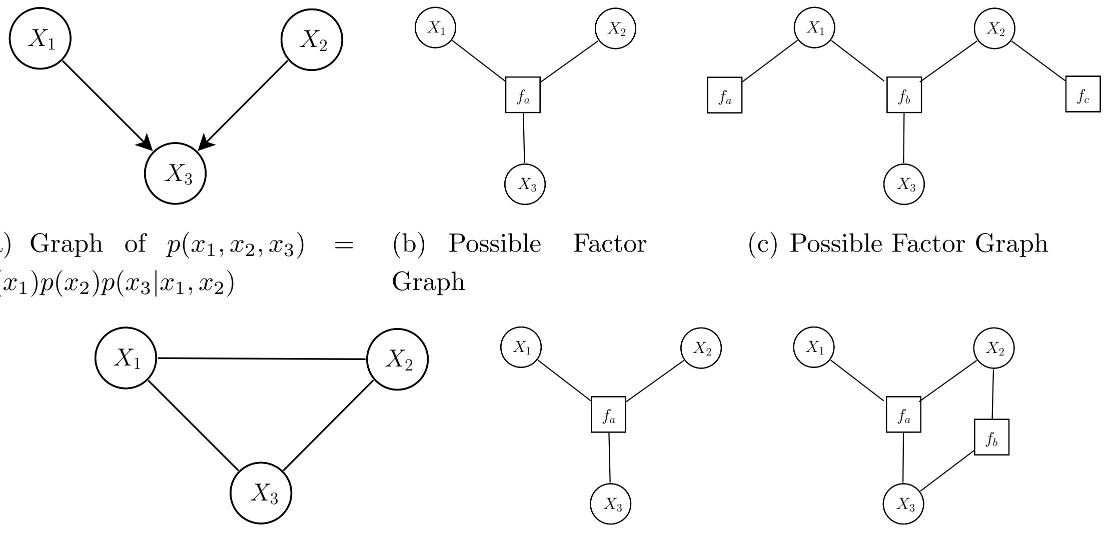
Algorithms

Thus far we have discussed the graphical model structures, how the models are created and how the interaction between probability theory and graph theory is represented. Once we have the model that represents the problem that we are looking at, however, we want to do inference on it (or in the case of statistical inference, find the parameters of the model). To do this, we use algorithms that have been developed either for graph theory in general, or specifically for graphical models. In Section 4.4, we will focus on one algorithm in particular—the junction tree algorithm. This algorithm is very general, and to implement it, the sum-product and max-sum algorithms are discussed first, in Sections 4.2 and 4.3. First however, we need to specify a type of graph, a factor graph, that is used in the algorithms. Note that the algorithms described here follows and summarises some of the work in [23, 4] and a workshop given by Tibério S. Caetano [5].

4.1 Factor Graphs

Up until now we have worked only with directed and undirected graphs—which were constructed in such a way that they represented the probability distributions in terms of conditional independences. Factor graphs, which can also be directed or undirected, are also used to represent probability distributions. Whereas in BNs and MRFs conditional independences were used to indicate relationships, in factor graphs any factorisation can be used. Factor graphs are used to represent distributions in the form of (3.4), with normalisation constant Z defined as in (3.5). Our definitions of the distribution according to conditional independences—(3.6) for BN and (3.11) for MRF—are special cases of (3.4).

Graphically, a factor graph has two types of nodes—a circular node representing variables and a square node representing the factors. In Figure 4.1(b) a basic factor graph for distribution $p(x_1, x_2, x_3) = f_a(x_1, x_2, x_3)$ is given. The edges of a factor graph are always between the variable nodes and the factor nodes which they are related to, with no edges between variable nodes, since the relationships in a factor graph are centred around the factors. Due to the fact that the connections in a factor graph is between factors and variables and no conditional independences are represented by using directed edges, even though one could write a factor graph as directed, its structure will be exactly the same as that of an undirected graph, with the exception that there will be directed edges. However, if the direction of the edges are disregarded, no information is lost, and for our purposes, we will therefore only use undirected factor graphs. Since both BNs and MRFs can be represented by a factor graph, which will have the same structure in both cases, algorithms developed to work for BNs can be applied directly to MRFs as well, without any changes.



(a) Graph of $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$

(b) Possible Factor Graph

(c) Possible Factor Graph

(d) Graph of $\psi(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$

(e) Possible Factor Graph

(f) Possible Factor Graph

Figure 4.1: *Factor graphs.*

We will use factor graphs to represent the distributions in the factored form we find from using the conditional independences. Suppose we have the distribution $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$, illustrated by a DAG in Figure 4.1(a), then Figure 4.1(b) represents the factor graph with

$$f_a(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2),$$

and 4.1(c) represents the factor graph with

$$\begin{aligned} f_a(x_1) &= p(x_1), \\ f_b(x_2) &= p(x_2), \\ f_c(x_1, x_2, x_3) &= p(x_3|x_1, x_2). \end{aligned}$$

It is clear that a factor graph is therefore not unique, it depends on how the factors are chosen for a specific distribution. The procedure to convert a DAG into a factor graph is to assign a node to every variable, insert the factor nodes such that they correspond to conditional distributions and then insert the links between the factor nodes and the variable nodes.

A very similar approach is used to convert a MRF to a factor graph. In Figure 4.1(d) the distribution $p(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$ is given. Two possible factor graphs are given in Figure 4.1(e), with factorisation

$$f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3),$$

and Figure 4.1(f), with factorisation

$$\psi(x_1, x_2, x_3) = f_a(x_1, x_2, x_3) f_b(x_2, x_3).$$

The procedure to convert a MRF to a factor graph is therefore to assign a node to every variable, insert factor nodes such that they correspond to cliques and then add appropriate links between the factor nodes and the variable nodes.

4.2 Sum-product Algorithm

In Section 3.6 we described message-passing, using the distributive law. We showed how summing over a variable can be seen as passing a message to its neighbours. The neighbouring nodes then multiply this message with the function defined over them to generate a new message to send, in turn, to their other neighbours. The same idea proves to hold in factor graphs.

Suppose we have a factorisation of a joint distribution, as given in (3.4),

$$p(x_1, \dots, x_N) = \prod_s f_s(x_{K_s}),$$

with the normalisation factor defined as one of the factors f_s defined over an empty set of variables. Furthermore, suppose that we want to find the marginal $p(x_i)$, then

$$\begin{aligned} p(x_i) &= \sum_{\{x_1,x_2,\dots,x_N\}\backslash x_i} p(x_1, x_2, \dots, x_N) \\ &= \sum_{\{x_1,x_2,\dots,x_N\}\backslash x_i} \prod_s f_s(x_{K_s}) \\ &= \prod_s \sum_{\{x_1,x_2,\dots,x_N\}\backslash x_i} f_s(x_{K_s}). \end{aligned} \tag{4.1}$$

Without loss of generality, assume that the graph in Figure 4.2 is a fragment of the graph representing a probability distribution. Let the query node be X_i and its neighbouring factor nodes f_s and the nodes to the right of X_i. The factor node f_s has neighbouring nodes X_i and the set X_j, $j = 1, 2, \dots, J$. $F_s(x_i, X_s)$ represents the product of all the factors in the group associated with factor f_s, with X_s the set $\{X_1, X_2, \dots, X_J\}$ of all variables in the subgraph connected to the variable node X_i via factor node f_s. On the other hand, $G_j(x_j, X_{sj})$ represents the products of all the factors in the graph associated with the neighbours of f_s,

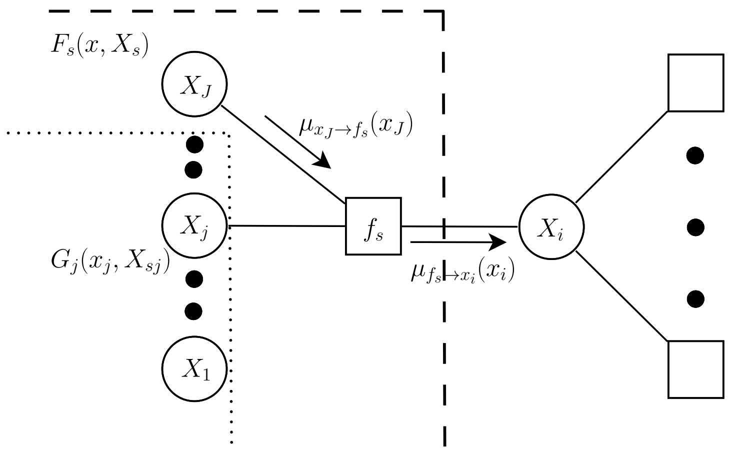


Figure 4.2: *Fragment of a factor graph.*

excluding X_J and X_i. We consider only the part played by $F_s(x_i, X_s)$, but the same logic holds for all the factor nodes to the right on the graph of node X_i.

The marginal $p(x_i)$ is given by

$$\begin{aligned} p(x_i) &= \prod_{s \in \mathrm{ne}(x_i)} \sum_{X_s} F_s(x_i, X_s) \\ &= \prod_{s \in \mathrm{ne}(x_i)} \mu_{f_s \to x_i}(x_i), \end{aligned}$$

with

$$\mu_{f_s \to x_i}(x_i) = \sum_{X_s} F_s(x_i, X_s) \tag{4.2}$$

and $\mathrm{ne}(x_i)$ the set of factor nodes neighbouring x_i. If $\mu_{f_s \to x_i}(x_i)$ is viewed as a message from a factor node f_s to variable node x_i then (4.2) indicates that the marginal $p(x_i)$ is found by multiplying all the incoming messages with the v6ariable X_i. This is similar to the elimination algorithm, as discussed in Section 3.6. Since $F_s(x_i, X_s)$ is the product of all the factors in the group associated with the factor f_s,

$$F_s(x_i, X_s) = f_s(x_i, x_1, \ldots, x_J) G_1(x_1, X_{s1}) \ldots G_J(x_J, X_{sJ}). \tag{4.3}$$

Thus,

$$\begin{aligned}
\mu_{f_s \to x_i}(x_i) &= \sum_{X_s} F_s(x, X_s) \\
&= \sum_{X_s} \left[f_s(x_1, \ldots, x_i, \ldots, x_J) G_1(x_1, X_{s1}) \ldots G_J(x_J, X_{sJ}) \right] \\
&= \sum_{X_s} \left[f_s(x_1, \ldots, x_i, \ldots, x_J) \prod_{j \in \mathrm{ne}(f_s) \backslash x_i} G_j(x_j, X_{sj}) \right] \\
&= \left[\sum_{X_s} f_s(x_1, \ldots, x_i, \ldots, x_J) \right] \left[\prod_{j \in \mathrm{ne}(f_s) \backslash x_i} \sum_{X_{sj}} G_j(x_j, X_{sj}) \right] \\
&= \sum_{X_s} f_s(x_1, \ldots, x_i, \ldots, x_J) \prod_{j \in \mathrm{ne}(f_s) \backslash x_i} \mu_{x_j \to f_s}(x_j),
\end{aligned}$$

with

$$\mu_{x_j \to f_s}(x_j) = \sum_{X_{sj}} G_j(x_j, X_{sj}), \tag{4.4}$$

$\mathrm{ne}(f_s)$ the set of variable nodes that are neighbours of factor node f_s, and $\mathrm{ne}(f_s) \backslash x_i$ the same set excluding the variable x_i. Let $\mu_{x_j \to f_s}(x_j)$ indicate the message from variable node x_j to a factor node f_s.

The function $G_j(x_j, X_{sj})$ is given by the product of all the incoming messages to the factor nodes that are connected to the variable node x_j, excluding factor node f_s. Thus, it will be the product of terms $F_m(x_j, X_{jm})$ which are each associated with one of the factor nodes f_m that is connected to the variable node x_j, excluding factor node f_s. Therefore

$$G_j(x_j, X_{sj}) = \prod_{m \in \mathrm{ne}(x_j) \backslash f_s} F_m(x_j, X_{jm}),$$

where every $F_m(x_j, X_{jm})$ is again a subgraph of the same type as we had at the beginning, and so we have a recursive situation.

We therefore have two messages, $\mu_{f_s \to x_i}(x_i)$ which is a message sent from a factor node to a variable node and $\mu_{x_j \to f_s}(x_j)$ which is a message sent from a variable node to a factor node. However, if we look further, we see these messages are closely interlinked, since

$$\begin{aligned}
\mu_{x_j \to f_s}(x_j) &= \sum_{X_{sj}} G_j(x_j, X_{sj}) \\
&= \sum_{X_{jm}} \prod_{m \in \mathrm{ne}(x_j) \backslash f_s} F_m(x_j, X_{jm}) \\
&= \prod_{m \in \mathrm{ne}(x_j) \backslash f_s} \sum_{X_{jm}} F_m(x_j, X_{jm}) \\
&= \prod_{m \in \mathrm{ne}(x_j) \backslash f_s} \mu_{f_m \to x_j}(x_j).
\end{aligned}$$

The messages $\mu_{x_j \to f_s}(x_j)$ and $\mu_{f_m \to x_j}(x_j)$ therefore depend on each other in the sense that for a node (whether variable or factor) to send a message, it must first receive all its messages, and the nodes that it receives its messages from have the same condition. This is known as the message-passing protocol. The marginal is calculated recursively, with x_i viewed as a root node, and the recursion starting at leaf nodes. This is similar to the elimination algorithm described in Section 3.6.

To start the recursion, the leaf nodes need to be initialised. If a leaf node is a variable node, then the message that passes from it is initialised to the value 1, thus, supposing x_1 is a leaf node with f_s a factor node which it is connected to, $\mu_{x_1 \to f_s}(x_1) = 1$. If the leaf node is a factor node, say f_s with x_i connected to it, then it is initialised to the function it represents over the variable(s) it is connected to, such that $\mu_{f_s \to x_i}(x_i) = f(x_i)$.

Although the message-passing protocol of the sum-product algorithm has been discussed, the general sum-product algorithm is not yet complete. One of the shortcomings of the elimination algorithm, as illustrated in Section 3.6, is that the elimination algorithm has to be rerun for every marginal that needs to be found. One of the advantages of the general sum-product algorithm is that it can be used to avoid these repetitions. This is done by not only passing messages to a specified variable, but to then also send messages out from this variable again.

Choose any arbitrary node to be the root node. Once it has received all its messages from its neighbours, the root node will now create a message that will again recursively pass through the graph. In this manner, each node has received a message from all its neighbours. When the messages are sent from the leaves to the root, for any node, albeit a variable or factor node, it receives messages from all its neighbours except one—the one neighbour that it will be sending the message along to. However, when the message now comes from the root, that same variable will now receive a message from the neighbour it previously sent a message to, and thus has now received a message from all its neighbours. If the previous messages are stored before sending them along, and together with this new message that has come from the root variable, every marginal can now be calculated. Comparing complexities, working out the marginal for every node separately will grow quadratically with the size of the graph, whereas using the sum-product, finding the marginal for every node in the graph is only twice as expensive as finding a single marginal.

Previously it was stated that the normalisation factor is considered to be built into the definition of $p(x)$ to be a factor defined over an empty set of variables. To find Z, as previously discussed, any of the marginals must be normalised, since the same Z holds for the entire distribution.

4.2.1 Chain Example

To illustrate the sum-product algorithm, first consider the chain graph as in Figure 4.3, which is a specific example of the graph in Figure 3.12.

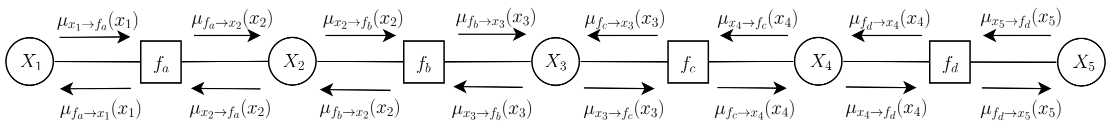


Figure 4.3: *Chain example of sum-product algorithm.*

The joint distribution is given by

$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_3, x_4) f_d(x_4, x_5). \tag{4.5}$$

To perform the sum-product algorithm, the choice of root node is arbitrary, thus designate X_3 as the root node. From the graphical representation it is clear that the leaf nodes are variable nodes X_1 and X_5, and thus the message passing starts at these nodes. There are two strands of message-passing, which one is started first is immaterial, and they can be seen to run in parallel, since they have no influence on each other. The first strand starts at X_1 and the second at X_5. Viewing the message passings from X_1 first, the messages are sent in the following order

$$\begin{aligned}
\mu_{x_1\to f_a}(x_1) &= 1,\\
\mu_{f_a\to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2),\\
\mu_{x_2\to f_b}(x_2) &= \mu_{f_a\to x_2}(x_2),\\
\mu_{f_b\to x_3}(x_3) &= \sum_{x_2} \left[f_b(x_2, x_3)\mu_{f_a\to x_2}(x_2)\right],
\end{aligned}$$

and the messages passing from X_5 to X_3 will be

$$\begin{aligned}
\mu_{x_5\to f_d}(x_5) &= 1,\\
\mu_{f_d\to x_4}(x_4) &= \sum_{x_5} f_d(x_4, x_5),\\
\mu_{x_4\to f_c}(x_4) &= \mu_{f_d\to x_4}(x_4),\\
\mu_{f_c\to x_3}(x_3) &= \sum_{x_4} \left[f_c(x_3, x_4)\mu_{f_d\to x_4}(x_4)\right].
\end{aligned}$$

After the messages above have been passed, the root node X_3 has received all its messages. Now the root node can send messages out to the leaves again, to complete the sum-product algorithm, and the marginals of any node can easily be found. The root node's 'messages' will also follow two paths—one to leaf-node X_1 and the other to X_5. The initialization-step will be slightly different in sending the messages from the root to the leaves, however. The message that X_3 has received from the left-hand side, $\mu_{f_b\to x_3}(x_3)$, contains information that needs to be sent throughout the graph. Similarly, $\mu_{f_c \to x_3}(x_3)$ contains information that needs to be sent to the left-hand side of the graph. The initialisation of the message sent from X_3 to X_5 will therefore incorporate the message $\mu_{f_b \to x_3}(x_3)$, and from X_3 to X_1 will incorporate $\mu_{f_c \to x_3}(x_3)$. The messages going to leaf-node X_1 will be

$$\begin{aligned}
\mu_{x_3 \to f_b}(x_3) &= \mu_{f_c \to x_3}(x_3) \\
\mu_{f_b \to x_2}(x_2) &= \sum_{x_3} f_b(x_2, x_3)\mu_{x_3 \to f_b}(x_3) \\
\mu_{x_2 \to f_a}(x_2) &= \mu_{f_b \to x_2}(x_2) \\
\mu_{f_a \to x_1}(x_1) &= \sum_{x_2} \left[f_a(x_1, x_2)\mu_{f_b \to x_2}(x_2) \right],
\end{aligned}$$

and to leaf-node X_5 will be

$$\begin{aligned}
\mu_{x_3 \to f_c}(x_3) &= \mu_{f_b \to x_3}(x_3) \\
\mu_{f_c \to x_4}(x_4) &= \sum_{x_3} f_c(x_3, x_4)\mu_{x_3 \to f_c}(x_3) \\
\mu_{x_4 \to f_d}(x_4) &= \mu_{f_c \to x_4}(x_4) \\
\mu_{f_d \to x_5}(x_5) &= \sum_{x_4} \left[f_d(x_4, x_5)\mu_{f_c \to x_4}(x_4) \right].
\end{aligned}$$

Furthermore, the normalisation factor can be found by normalising any of the marginals. If using x_4, then

$$Z = \sum_{x_4} \mu_{f_c \to x_4}(x_4). \tag{4.6}$$

To check whether this method gives us the correct answer, consider finding the marginal of X_4, which is not the chosen root node. Then we have that, with normalisation factor Z as in (4.6),

$$\begin{aligned}
p(x_4) &= \frac{1}{Z}\mu_{f_d \to x_4}(x_4)\mu_{f_c \to x_4}(x_4) \\
&= \frac{1}{Z}\left[\sum_{x_5} f_d(x_4, x_5)\right]\left[\sum_{x_3} f_c(x_3, x_4)\mu_{x_3 \to f_c}(x_3)\right] \\
&= \frac{1}{Z}\sum_{x_5} f_d(x_4, x_5)\left[\sum_{x_3} f_c(x_3, x_4)\mu_{f_b \to x_3}(x_3)\right] \\
&= \frac{1}{Z}\sum_{x_5} f_d(x_4, x_5)\left[\sum_{x_3} f_c(x_3, x_4)\sum_{x_2} f_b(x_2, x_3)\mu_{f_a \to x_2}(x_2)\right] \\
&= \frac{1}{Z}\sum_{x_5} f_d(x_4, x_5)\left[\sum_{x_3} f_c(x_3, x_4)\sum_{x_2} f_b(x_2, x_3)\sum_{x_1} f_a(x_1, x_2)\right] \\
&= \frac{1}{Z}\sum_{x_5}\sum_{x_3}\sum_{x_2}\sum_{x_1} f_d(x_4, x_5)f_c(x_3, x_4)f_b(x_2, x_3)f_a(x_1, x_2) \\
&= \frac{1}{Z}\sum_{x_5}\sum_{x_3}\sum_{x_2}\sum_{x_1} p(x).
\end{aligned}$$

We see therefore that the sum-product algorithm gives us the correct marginalisation for x_4.

4.2.2 Tree Example

Consider another example, this time a tree, as in Figure 4.4.

The probability distribution that the graph in Figure 4.4 represents can be given by

$$p(x) = \frac{1}{Z} f_a(x_1, x_2, x_3) f_b(x_3, x_4) f_c(x_3, x_5). \tag{4.7}$$

Choose X_5 as the root of the graph. As illustrated in Figure 4.4, the messages will be passed

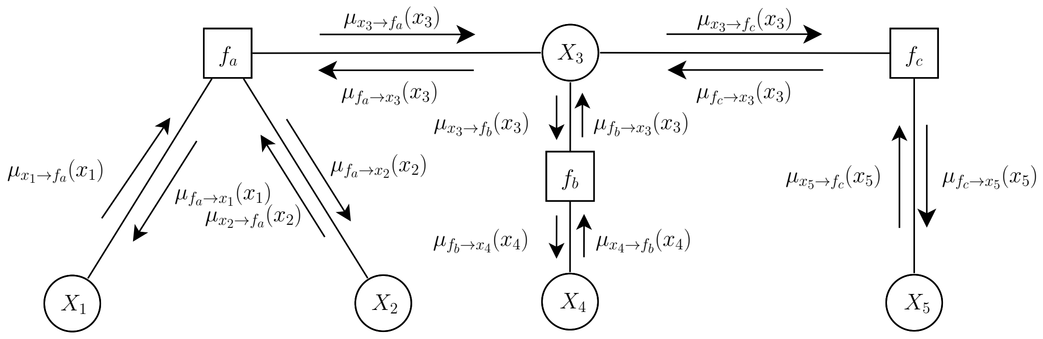


Figure 4.4: *Tree example of sum-product algorithm.*

along from the three leaf-nodes X_1, X_2 and X_4, through the respective factors f_a and f_b, via X_3 and f_c to the chosen root node X_5. The messages that will be passed from X_1 and X_2 to X_3 are,

$$\begin{aligned} \mu_{x_1\to f_a}(x_1) &= 1, \\ \mu_{x_2\to f_a}(x_2) &= 1, \\ \mu_{f_a\to x_3}(x_3) &= \sum_{x_1}\sum_{x_2} f_a(x_1, x_2, x_3), \end{aligned}$$

the messages passed from X_4 to X_3 are

$$\begin{aligned} \mu_{x_4\to f_b}(x_4) &= 1, \\ \mu_{f_b\to x_3}(x_3) &= \sum_{x_4} f_b(x_3, x_4) \end{aligned}$$

and the message passed from X_3 to X_5 are

$$\begin{aligned} \mu_{x_3\to f_c}(x_3) &= \mu_{f_a\to x_3}(x_3)\mu_{f_b\to x_3}(x_3), \\ \mu_{f_c\to x_5}(x_5) &= \sum_{x_3} \left[f_c(x_3, x_5)\mu_{f_a\to x_3}(x_3)\mu_{f_b\to x_3}(x_3)\right]. \end{aligned}$$

Once $\mu_{f_c \to x_5}(x_5)$ has arrived at X_5, we now have to send messages from X_5 back to the leaf nodes. Note that this time the root node only received a message from one branch, and thus there are no other messages from other branches that need to be incorporated, thus the message sent out from X_5 will be initialised to 1. Therefore, the message sent from X_5 to X_3 will be

$$\begin{aligned}\mu_{x_5 \to f_c}(x_5) &= 1, \\ \mu_{f_c \to x_3}(x_3) &= \sum_{x_5} f_c(x_3, x_5),\end{aligned}$$

the message sent from X_3 to X_4

$$\begin{aligned}\mu_{x_3 \to f_b}(x_3) &= \mu_{f_c \to x_3}(x_3), \\ \mu_{f_b \to x_4}(x_4) &= \sum_{x_3} f_b(x_3, x_4)\mu_{x_3 \to f_b}(x_3)\end{aligned}$$

and from X_3 to X_1 and X_2 will be

$$\begin{aligned}\mu_{x_3 \to f_a}(x_3) &= \mu_{f_c \to x_3}(x_3), \\ \mu_{f_a \to x_2}(x_2) &= \sum_{x_2} f_a(x_1, x_2, x_3)\mu_{x_3 \to f_a}(x_3), \\ \mu_{f_a \to x_1}(x_1) &= \sum_{x_1} f_a(x_1, x_2, x_3)\mu_{x_3 \to f_a}(x_3).\end{aligned}$$

Again the normalisation factor Z will be found by marginalising further over any of the variables. Thus, if we want to find the marginal of say, x_3, we will have

$$\begin{aligned}p(x_3) &= \frac{1}{Z}\mu_{f_a \to x_3}(x_3)\mu_{f_b \to x_3}(x_3)\mu_{f_c \to x_3}(x_3) \\ &= \frac{1}{Z}\left[\sum_{x_1}\sum_{x_2} f_a(x_1, x_2, x_3)\right]\left[\sum_{x_4} f_b(x_3, x_4)\right]\left[\sum_{x_5} f_c(x_3, x_5)\right] \\ &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} f_a(x_1, x_2, x_3)f_b(x_3, x_4)f_c(x_3, x_5) \\ &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_4}\sum_{x_5} p(x).\end{aligned}$$

And thus the sum-product algorithm as described also gives us the correct answer when using a tree-structured factor graph.

4.3 Max-sum Algorithm

Suppose we have the distribution $p(x_1, x_2)$ with the numeric values given in Table 4.1.

We can easily see that $p(x_1 = 1, x_2 = 0) = 0.4$ is the maximum value that the distribution can assume. However, if we were to try and find the maximum value by using the sum-product

Table 4.1: *Joint distribution $p(x_1, x_2)$.*

	$x_1 = 0$	$x_1 = 1$
$x_2 = 0$	0.0	0.4
$x_2 = 1$	0.3	0.3

algorithm, finding first the marginal of all the variables and then choosing the values that maximise the marginal, we would have found the wrong answer. For example, if looking at the marginals and finding the values of x_1 and x_2 that would maximise the marginals separately, $x_1 = 0$ and $x_2 = 1$, but $p(x_1 = 0, x_2 = 1) = 0.3 < 0.4 = p(x_1 = 1, x_2 = 0)$. We will therefore need another way to find the values that maximise joint distributions.

It has been stated previously that in our application, we are looking for the MAP values. The max-sum algorithm, described in this section, is used to help us in this endeavour. Not only does the max-sum algorithm help us find the set of values for x that will give us the largest joint probability, but also what this probability is. Furthermore, the max-sum algorithm is an application of dynamic programming when looking at the field of graphical models, [8]. Let

$$x^{\max} = \arg\max_x p(x) \tag{4.8}$$

be the values of x that maximises the joint distribution $p(x)$, i.e.

$$p(x^{\max}) = \max_x p(x) \tag{4.9}$$

is the maximum value that the joint distribution $p(x)$ can assume.

There are three general properties regarding finding the maximum values of a function that we are using. The first is analogous to the distributive law, as in (2.18), if $a \geq 0$,

$$\max(ab, ac) = a \max(b, c) \tag{4.10}$$

and the second is

$$\max(a + b, a + c) = a + \max(b, c). \tag{4.11}$$

During the max-sum algorithm we come across the multiplication of small probabilities. Although theoretically this does not cause a problem, in practice this can lead to numerical underflow problems. To counter numerical instabilities, the logarithm of the joint distribution is used. Since ln is an increasing function,

$$\ln\left(\max_x p(x)\right) = \max_x \ \ln p(x) \tag{4.12}$$

holds.

Let us first consider the problem of finding the maximum of the joint probability. If we substitute (3.4) into (4.9) and use (4.10) we have the same structure as in the sum-product algorithm. We can therefore use the same definitions and results with regards to message-sending, substituting maxima where we had summations to give us the max-product algorithm.

If we want to apply this process on the distribution of the chain graph as described by (3.16) and in Figure 3.12, we find

$$\begin{aligned}\max_x p(x) &= \frac{1}{Z}\max_{x_1} \dots \max_{x_N} \psi(x_1, x_2)\psi(x_2, x_3)\dots\psi(x_{N-1}, x_N) \\ &= \frac{1}{Z}\max_{x_1}\left[\psi(x_1, x_2)\left[\dots \max_{x_N} \psi(x_{N-1}, x_N)\right]\dots\right].\end{aligned}$$

Note however, that we will only be propogating the messages in the forward direction, and not in both directions as in the sum-product algorithm, the reason for this will be given later on in the section. What is important to realise is that the reason we sent messages back from the root to the leaves in the sum-product was so that we could find all the variables' marginalisations. However, in the max-sum algorithm we want to find the MAP values and the value that the probability distribution assumes when x is set to the MAP values. No matter which node we designate as the root, we always get the same answer, and therefore we do not have to do it for every variable. We therefore do not need to do backward propogation of messages.

As previously stated, the product of small probabilities could cause numerical problems, and therefore we will use the logarithms. From (4.12) and (4.11) we therefore have the products in the max-product algorithm change into summations, and thus we have the max-sum algorithm.

Factor graphs are again used to implement the max-sum algorithm. The messages sent from factor nodes to variable nodes and from variable nodes to factor nodes have the same definition as in the sum-product algorithm, taking into account that the summations become maximisations and the products become summations. In the max-sum algorithm the general definitions of the messages are, given that we are looking at the messages from/to variable node X_i and the other variable nodes being the set $\{X_1, \dots, X_J\}$

$$\begin{aligned}\mu_{f_s \to x_i}(x_i) &= \max_{x_1,\dots,x_J}\left[\ln f(x_i, x_1, \dots, x_J) + \sum_{j \in \mathrm{ne}(f_s)\backslash x_i} \mu_{x_j \to f_s}(x_j)\right], && (4.13) \\ \mu_{x_i \to f_s}(x_i) &= \sum_{m \in \mathrm{ne}(x)\backslash f_s} \mu_{f_m \to x_i}(x_i). && (4.14)\end{aligned}$$

Analagous to the sum-product algorithm, the messages from the leaf-nodes are initialised. For example, if x_1 is the leaf node we are sending a message from to factor node f_s, the message is initialised to $\mu_{x_1 \to f_s}(x_1) = 0$. Similarly, if f_s is a leaf factor node we are sending a message from, to variable node x_i, then $\mu_{f_s \to x_i}(x_i) = \ln f(x_i)$.

Once all the messages have been sent to the root node, the maximum value of the joint probability distribution can be calculated. Since we choose the root variable arbitrarily, and whichever one we choose yields the same result, it is the same as finding any marginal in the sum-product algorithm. Thus, similar to (4.1), we have that, if x_i is designated as the root note,

$$p(x^{\max}) = \max_{x_i} \left[\sum_{s \in \mathrm{ne}(x_i)} \mu_{f_s \to x_i}(x_i) \right]. \tag{4.15}$$

We now turn to the problem of finding the configuration of x that results in the maximum value of the joint distribution $p(x)$. If x_i is the root node, finding $x_i^{\max}$ is trivial, with

$$x^{\max} = \arg\max_{x_i} \left[\sum_{s \in \mathrm{ne}(x_i)} \mu_{f_s \to x_i}(x_i) \right]. \tag{4.16}$$

If we were to follow the method of the sum-product algorithm, we would now do backward propagation of messages so as to use (4.16) at every node. However, as stated before we will not do this. The reason backward propagation of messages is not done is because it can lead to a mix between optimal configurations, that would lead to a final non-optimal configuration. To overcome this problem we use a method known as *back-tracking.*

Simply seen, back-tracking results from keeping track of which values of the variables give rise to the maximum state of each variable. A function $\phi(x_i)$ is defined such that it stores the arguments that leads to the maximum state of each variable. In other words, it stores the quantities given by

$$\phi(x_i) = \arg\max_{x_{i-1}} [\ln f(x_i, x_1, \ldots, x_J) + \sum_{m \in \{x_1, \ldots, x_J\}} \mu_{x_m \to f_s}(x_m)],$$

with $\{x_1, \ldots, x_J\}$ the set of variable nodes connected to the factor node f_s other than x_i.

As an example of this, consider the distribution

$$p(x) = f_a(x_1, x_2) f_b(x_2, x_3)$$

with the factor graph representing the distribution in Figure 4.5

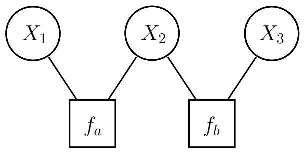


Figure 4.5: *Example of max-sum algorithm.*

The message that will be sent from X_1 to X_3, if X_3 is chosen as the root node, will be

$$\begin{aligned}
\mu_{x_1 \to f_a}(x_1) &= 0, \\
\mu_{f_a \to x_2}(x_2) &= \max_{x_1} \ln f_a(x_1, x_2), \\
\mu_{x_2 \to f_b}(x_2) &= \mu_{f_a \to x_2}(x_2), \\
\mu_{f_b \to x_3}(x_3) &= \max_{x_2} \ln f_b(x_2, x_3) + \mu_{x_2 \to f_b}(x_2).
\end{aligned}$$

The back-tracking information stored is

$$\begin{aligned}\phi(x_2) &= \arg\max_{x_1} [\ln f_a(x_1,x_2)],\\ \phi(x_3) &= \arg\max_{x_2} [\ln f_b(x_2,x_3) + \mu_{f_a\to x_2}(x_2)],\\ &= \arg\max_{x_2} [\ln f_b(x_2,x_3) + \max_{x_1} \ln f_a(x_1,x_2)].\end{aligned}$$

To find the maximum configuration of the joint distribution we therefore find

$$\begin{aligned}p(x^{\max}) &= \max_{x_3}\left[\sum_{s\in\mathrm{ne}(x_3)} \mu_{f_s\to x_3}(x_3)\right]\\ &= \max_{x_3}[\max_{x_2} \ln f_a(x_1,x_2) + \mu_{x_2\to f_b}(x_2)]\\ &= \max_{x_3}[\max_{x_2} \ln f_a(x_1,x_2) + \max_{x_1} \ln f_a(x_1,x_2)].\end{aligned}$$

To find the configuration $x^{\max}$ we now first find

$$x_3^{\max} = \arg\max_{x_3}[\max_{x_2} \ln f_a(x_1,x_2) + \max_{x_1} \ln f_a(x_1,x_2)].$$

Then using this information, we track back the values that were set as the values that caused the maximum value in x_2 and then x_1.

4.4 Junction Tree Algorithm

Both the sum-product algorithm and the max-sum algorithm are defined on factor graphs, and by implication, trees. Although these algorithms are useful, the restriction of only being able to use trees is a burden we do not want to bear. Although there are graphical models that are trees, like HMMs, in general we do not have this property when looking at BNs or MRFs. The junction tree algorithm provides a structure and general framework in which we can apply the max-sum algorithm, sum-product algorithm and other algorithms to do probabilistic inference. As the junction tree algorithm is a general framework we will also look at a specific implementation of the junction tree—the Hugin algorithm, though there are others, such as the Shafer-Shenoy algorithm. Both of these are dealt with in more depth in [23, chapter 17]. The key idea with the junction tree is to define locality, and then find a way of representing a general problem in terms of local computations. For the junction tree algorithm to be used, the graphical models that we use will first need to be converted into chordal graphs, and form there, into clique trees with the junction tree property.

4.4.1 Chordal Graphs

A *chord* is defined as an edge connecting nodes in a cycle, without being on the path of the cycle. As an example, consider the graph in Figure 4.6. The cycle $a-b-d-c-a$

has a chord, namely the edge bc, whereas the cycle $c-d-f-e-c$ has no chord, and is therefore chordless. A chordal graph is a graph where every cycle greater than three has a chord. Looking at our example in Figure 4.6, we see that the graph is not chordal, since the cycle $c-d-f-e-c$ is longer than three nodes and has no chord. If, however, either the edge cf and/or de is added, then the graph becomes a chordal graph.

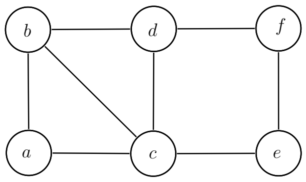


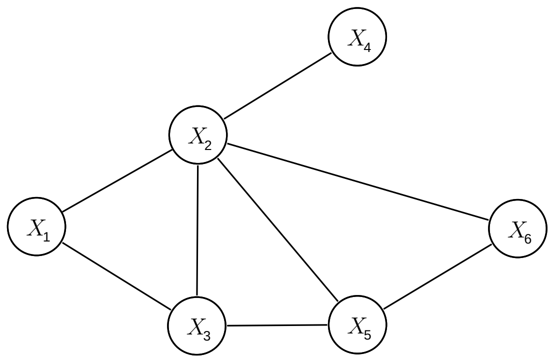
Figure 4.6: *Example of a graph with and without a chord.*

From Figure 3.11 it is clear that our example graphs H_1 and H_2 are not chordal. However, we can convert non-chordal graphs into chordal graphs, by a process known as *triangulation.* Note that to use the junction tree algorithm we first moralise the graphs, so as to have undirected graphs. As was previously stated, one of the reasons for using undirected graphs is that a directed graph can, with relative ease, be converted into an undirected graph, and thus we can stipulate one algorithm to be used in both cases. Furthermore, chordal graphs are necessary for the junction tree algorithm, and moralising a graph will not add any edges that would not be added during triangulation. Triangulating a graph involves adding edges to an undirected graph until it is chordal. For the same reason that moralising a graph does not lead to a misrepresentation of the distribution, though there is some loss of conditional independence properties, as discussed in Section 3.5, so too will adding edges to create a chordal graph lead to the loss of some independences, but it will not misrepresent our distribution. In Figure 4.7 the chordal graphs of H_1 and H_2 are given. In H_1 only the edge between node X_2 and X_3 is added, whilst in H_2 the edge between nodes Y_2 and Y_5 or the edge between the nodes Y_1 and Y_3 had to be added, since there was a cycle of length five, $Y_1-Y_2-Y_3-Y_5-Y_1$, that had no chords. We arbitrarily chose to add the chord between Y_1 and Y_3.

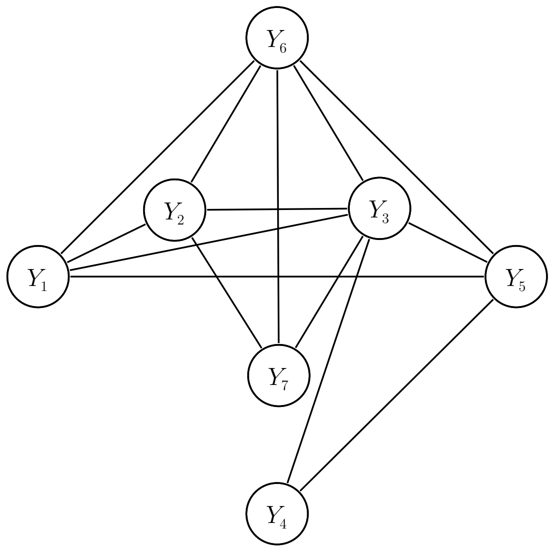
Although we will not go into the details of it, the elimination algorithm as discussed in Section 3.6 leads to a chordal graph being formed. For a detailed explanation of why this is so, see [23, chapter 17]. For our purposes it suffices to point out that the process of eliminating variables when summing over them creates extra edges, as discussed in Section 3.6. These edges that are added are the same edges that will be added when creating a chordal graph.

4.4.2 Clique Trees

From a chordal graph we can now create a clique tree. Note that because a clique tree is built on an underlying graph, the chordal graph, it is a hypergraph.



(a) Chordal Graph of H_1



(b) Chordal Graph of H_2

Figure 4.7: *Chordal graphs of examples H_1 and H_2.*

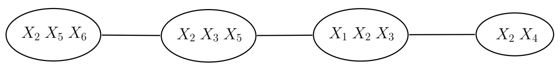
The nodes in a clique tree are the cliques in a graph. For example, the cliques in the chordal graph of H_1 are the sets

$$\{X_2, X_5, X_6\}, \{X_2, X_3, X_5\}, \{X_1, X_2, X_3\}, \{X_2, X_4\}. \tag{4.17}$$

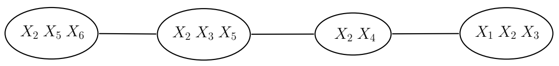
Graphically the clique tree is as illustrated in Figure 4.8(a), with the nodes represented by ellipses. The edges between the cliques is an indication of how information will flow through the clique tree.

In Section 3.6 we discussed how the local messages flowed in the graph of H_1 during marginalising to find $p(x_2)$. When looking how these messages flowed, we see that the message $m_6(x_2, x_5)$ was created when summing over x_6, thus 'in' clique $\{X_2, X_5, X_6\}$, and was used again when summing over x_5, which is represented by the clique $\{X_2, X_3, X_5\}$. The message $m_5(x_2, x_3)$ was sent from clique $\{X_2, X_3, X_5\}$ to $\{X_1, X_2, X_3\}$. The message $m_4(x_2)$ was also sent to clique $\{X_1, X_2, X_3\}$. Note that although we could make separate cliques for the sets $\{X_1, X_2\}$ and $\{X_2\}$ to represent the flow of messages between X_1, X_2 and X_3, the reason we do not do this is because the messages that were sent around within this clique was due to our elimination ordering, as well as which variable we chose to marginalise over. If, for instance, we wanted to find $p(x_6)$, then the information from the different cliques can still be sent around the cliques as illustrated in Figure 4.8(a). Thus the cliques $\{X_1, X_2\}$ and $\{X_2\}$ are not stated explicitly, even though they could be.

Clique trees, however, are not unique. For example, the graph in Figure 4.8(b) can also represent the flow of messages for graph H_1. However, there is a subtle, but important for our purposes, difference. In Figure 4.8(a) all the nodes containing a certain variable are connected, whereas in Figure 4.8(b) this is not the case, as can be seen when looking at the



(a) Clique tree with junction tree property.



(b) Clique tree without junction tree property.

Figure 4.8: *Possible clique trees of H_1.*

variable x_3. The sets containing X_3 are

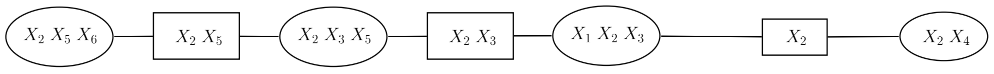
$$\{X_2, X_3, X_5\}, \{X_1, X_2, X_3\},$$

and in Figure 4.8(a) these cliques are connected by an edge, whereas in Figure 4.8(b) clique $X_2 - X_4$ separates the two cliques containing X_3. The property of all nodes containing a specific variable, for all variables, being connected is called the *junction tree property.* The clique tree in Figure 4.8(a) therefore has the junction tree property.

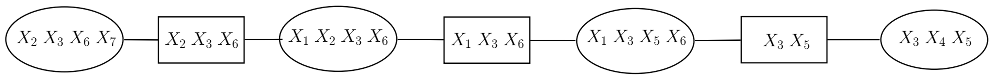
4.4.3 Junction trees

To represent a junction tree, we add another type of node to the clique trees we have defined in Section 4.4.2. These nodes are seen as separator nodes, since they will separate the clique nodes. Representing the variables that are common between the two clique nodes they separate, they also indicate what the domain is of the message that is sent between the cliques. An example of a clique tree with separator sets, called a junction tree, see Figure 4.9(a), which represents the junction tree of H_1. Note that the message that is sent from clique $\{X_2,\ X_5,\ X_6\}$ to clique $\{X_2,\ X_3,\ X_5\}$ is $m_6(x_2, x_5)$, which has as its domain x_2 and x_5. This is represented by the separator node that now separates these two cliques.

The junction tree for H_2 is given in Figure 4.9(b).



(a) Junction tree of H_1.



(b) Junction tree of H_2.

Figure 4.9: *Junction tree of H_1 and H_2.*

4.4.4 Defining Potentials of a Junction Tree

A junction tree is built up from an MRF. As described previously, the potential functions on an MRF are defined over cliques—though we only looked at defining them over maximal cliques. Intuitively it seems logical to define the potential functions of a junction tree using the potential functions of its underlying MRF. However, since the clique tree might have some extra cliques that are not in the underlying moralised graph, we have to make provision for the situation where the clique tree contains different cliques to the underlying graph. We do know, however, that the cliques in the underlying graph will be proper subsets of the maximal cliques. As an example, from Figure 3.11(a), Figure 4.7(a) and Figure 4.9(a) we see that the maximal cliques $\{X_1,\ X_2\}$ and $\{X_1,\ X_3\}$ in the moralised graph of H_1 are proper subsets of the clique $\{X_1,\ X_2,\ X_3\}$ in the chordal graph of H_1. Therefore, the potential on the clique $\{X_1,\ X_2,\ X_3\}$ in the junction tree will be the product of the potentials over the cliques $\{X_1,\ X_2\}$ and $\{X_1,\ X_3\}$ in the moralised graph. For now we define the potential as $\psi_C(x_C)$, but will drop the subscripts of ψ_C after this section. Note that every potential in the moralised graph is assigned to only one potential in the clique tree. If there are more than one possible cliques they can be assigned to, arbitrarily choose one to assign the potential to.

The potential functions for the clique nodes for the example H_1 are

$$\begin{aligned}
\psi_{1,2,3} &= p(x_1)p(x_2|x_1)p(x_3|x_1),\\
\psi_{2,3,5} &= p(x_5|x_3),\\
\psi_{2,5,6} &= p(x_6|x_2,x_5)\\
\psi_{2,4} &= p(x_4|x_2).
\end{aligned}$$

The potential functions for the clique nodes in junction tree of example H_2 are

$$\begin{aligned}
\psi_{1,2,5,6}(y_1,y_2,y_5,y_6) &= p(y_1)p(y_2|y_1)p(y_6|y_1,y_5),\\
\psi_{2,3,6,7}(y_2,y_3,y_6,y_7) &= p(y_3|y_2)p(y_7|y_2,y_3,y_6),\\
\psi_{3,4,5}(y_3,y_4,y_5) &= p(y_4)p(y_5|y_3,y_4).
\end{aligned}$$

Note that adding the chord $Y_2 - Y_5$ has had no influence on the clique potentials—they are exactly the same as for the moralised graph.

The potential functions defined on the separator sets $\phi_S(x_S)$ will be initialised to unity.

4.4.5 Message Passing in Junction Tree

After constructing the junction tree, and initialising the potential functions over the clique and separator nodes, messages need to be propagated throughout the tree. For illustrative purposes, consider the graph in Figure 4.10. Although only two clique nodes and one separator node is considered, the example will illustrate how messages are sent throughout the tree. Let V and W be sets of indices and let $S = V \cup W$. Then $\psi_V(x_V)$ will be the potential function over the clique X_V, $\psi_W(x_W)$ the potential function over the clique X_W and $\phi_S(X_S)$ the separator potential, initialised to unity. The joint distribution over this simple graph is

$$p(x) = \frac{1}{Z}\frac{\psi_V(X_V)\psi_W(X_W)}{\phi_S(X_S)}.$$

To see why the distribution is given by this joint probability, see [23, chapter 17]. The general idea is to transform potentials into marginals, and thus to maintain consistency, potential functions that are defined on common variables need to have the same marginal on those variables. Thus there is a forward and a backward pass in the junction tree algorithm, to maintain consistency in the same manner as discussed for the sum-product algorithm.

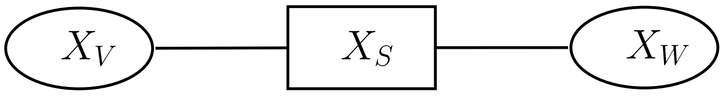


Figure 4.10: *Graph used to explain message propagation in junction tree.*

For now, drop the arguments of the potential functions, since it is clear from the subscripts what the domain is of the functions, i.e. let $\psi_V = \psi_V(X_V)$. Let an asterisk indicate an updated version of a potential. Then ψ_V^* will indicate an updated version of ψ_V and ϕ_S^{**} will indicate an updated version of ϕ_S^*. If we start at X_V and send a message to X_W, the forward pass will start by passing a message from X_V to X_W. In the forward pass ψ_V remains the same, $\psi_V^* = \psi_V$, since the message is sent out from ψ_V. The message creates the following updates

$$\phi_S^* = \sum_{V\backslash S} \psi_V^*, \tag{4.18}$$

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W. \tag{4.19}$$

Since

$$\begin{aligned} p(x) &= \frac{1}{Z}\frac{\psi_V^*\psi_W^*}{\phi_S^*} \\ &= \frac{1}{Z}\frac{\psi_V\frac{\psi_S^*}{\phi_S}\psi_W}{\phi_S^*} \\ &= \frac{1}{Z}\frac{\psi_V\psi_W}{\psi_S}, \end{aligned}$$

the joint probability has not been changed by doing the forward pass.

During the backward pass $\psi_W^{**} = \psi_W^*$ and the messages that are sent from X_W to X_V will do the following updates

$$\phi_S^{**} = \sum_{W\backslash S} \psi_W^*, \tag{4.20}$$

$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*}\psi_V^*. \tag{4.21}$$

To again check that our joint probability have not changed, we check

$$\begin{aligned} p(x) &= \frac{1}{Z}\frac{\psi_V^{**}\psi_W^{**}}{\phi_S^{**}} \\ &= \frac{1}{Z}\frac{\frac{\phi_S^{**}}{\phi_S^{*}}\psi_V^{*}\psi_W}{\phi_S^{**}} \\ &= \frac{1}{Z}\frac{\psi_V^{*}\psi_W^{*}}{\phi_S^{*}}. \end{aligned}$$

To ensure that the system is indeed consistent, the marginal $p(x_S) = \frac{1}{Z}\sum_{V\backslash S}\psi_V^{**}$ must be equal to $p(x_S) = \frac{1}{Z}\sum_{W\backslash S}\psi_W^{**}$. We have, using (4.18) and (4.20)

$$\begin{aligned} \sum_{V\backslash S}\psi_V^{**} &= \sum_{V\backslash S}\frac{\phi_S^{**}}{\phi_S^{*}}\psi_V^{*} \\ &= \frac{\phi_S^{**}}{\phi_S^{*}}\sum_{V\backslash S}\psi_V^{*} \\ &= \phi_S^{**} \qquad (4.22) \\ &= \sum_{W\backslash S}\psi_W^{*} \\ &= \sum_{W\backslash S}\psi_W^{**}. \end{aligned}$$

Note that from (4.22) we see that the marginal $p(x_S) = \frac{1}{Z}\phi_S^{**}$.

The normalisation factor Z can be found in three ways, after the junction tree algorithm has been run. From the separator potential,

$$Z = \sum_S \phi_S^{**},$$

whilst if one finds the marginal $p(x_W)$ and then further sums over W one could also find Z, since

$$\begin{aligned} p(x_W) &= \frac{1}{Z}\sum_{V\backslash S}\psi_V\psi_W \\ &= \frac{1}{Z}\phi_S^{*}\psi_W \\ &= \frac{1}{Z}\psi_W^{*} \\ &= \frac{1}{Z}\psi_W^{**}. \end{aligned}$$

Similarly, from $p(x_V)$ it is possible to find Z by further summing over x_V.

The footwork to discover message propagation in the junction tree algorithm has now been done. If we consider a chain clique tree, it is clear from the above explanation that if we start at one end with message passing and do forward and backward propagation of messages, the system will be consistent and we will be able to find the information we need. However, what happens if the clique tree is not a chain?

The solution is much like the situation that plays out in the sum-product algorithm. We again want to update one node (here a clique node) based on the information stored, or passed on, from other nodes. As with the sum-product algorithm, we state that a node can only send a message to its neighbours once it has received all the messages from its other neighbours. Again this is called the message passing protocol. Also, this is again a recursive function, and we can arbitrarily start at a node, and work out to find the leaves of the graph. The messages will then be passed from the leaf-nodes inward (forward pass) and then again from the root node outwards (backward pass).

One can also introduce evidence into the junction tree algorithm. In essence, when we introduce evidence we direct our attention from the whole clique tree to only a subset thereof—those variables for which there is not observed data. The potentials will now be redefined on only these subsets, which are just slices of the original potentials. In general, however, there is no fundamental difference between how conditional and joint distributions are dealt with.

Looking at our examples H_1 and H_2 we will now illustrate the propagation of the messages in the junction tree algorithm. The junction trees of H_1 and H_2 are given in Figure 4.9. For H_1, the messages that are sent are, from $X_2 - X_5 - X_6$ to $X_2 - X_4$ in the forward propagation stage,

$$\begin{aligned}
\psi^*_{2,5,6} &= \psi_{2,5,6} \\
\phi^*_{2,5} &= \sum_{x_6} \psi^*_{2,5,6} \\
\psi^*_{2,3,5} &= \frac{\phi^*_{2,5}}{\phi_{2,5}}\psi_{2,3,5} = \sum_{x_6} \psi^*_{2,5,6}\psi_{2,3,5} \\
\phi^*_{2,3} &= \sum_{x_5} \psi^*_{2,3,5} = \sum_{x_5}\sum_{x_6} \psi^*_{2,5,6}\psi_{2,3,5} \\
\psi^*_{1,2,3} &= \frac{\phi^*_{2,3}}{\phi_{2,3}}\psi_{1,2,3} = \sum_{x_5}\sum_{x_6} \psi^*_{2,5,6}\psi_{2,3,5}\psi_{1,2,3} \\
\phi^*_{2} &= \sum_{x_1,x_3} \psi^*_{1,2,3} = \sum_{x_1,x_3}\sum_{x_5}\sum_{x_6} \psi^*_{2,5,6}\psi_{2,3,5}\psi_{1,2,3} \\
\psi^*_{2,4} &= \frac{\phi^*_{2}}{\phi_{2}}\psi_{2,4} = \psi_{2,4}\sum_{x_1,x_3}\sum_{x_5}\sum_{x_6} \psi^*_{2,5,6}\psi_{2,3,5}\psi_{1,2,3}.
\end{aligned}$$

Notice that, from Section 4.4.4 where we defined the potentials,

$$\phi^*_{2,5} = \sum_{x_6} \psi^*_{2,5,6} = \sum_{x_6} p(x_6|x_2, x_5),$$

and similarly for the other messages.

In the backward propagation phase, the messages that are sent are

$$\begin{aligned}
\psi_{2,4}^{**} &= \psi_{2,4}^{*} = \psi_{2,4} \sum_{x_1,x_3} \sum_{x_5} \sum_{x_6} \psi_{2,5,6}^{*} \psi_{2,3,5} \psi_{1,2,3} \\
\phi_{2}^{**} &= \sum_{x_4} \psi_{2,4}^{**} = \sum_{x_4} \psi_{2,4} \sum_{x_1,x_3} \sum_{x_5} \sum_{x_6} \psi_{2,5,6}^{*} \psi_{2,3,5} \psi_{1,2,3} \\
\psi_{1,2,3}^{**} &= \frac{\phi_2^{**}}{\phi_2^{*}} \psi_{1,2,3}^{*} \\
\phi_{2,3}^{**} &= \sum_{x_1} \psi_{1,2,3}^{**} \\
\psi_{2,3,5}^{**} &= \frac{\phi_{2,3}^{**}}{\phi_{2,3}^{*}} \psi_{2,3,5}^{*} \\
\phi_{2,5}^{**} &= \sum_{x_3} \psi_{2,3,5}^{**} \\
\psi_{2,5,6}^{**} &= \frac{\phi_{2,5}^{**}}{\phi_{2,5}^{*}} \psi_{2,5,6}^{*}.
\end{aligned}$$

Similarly, for H_2, if the messages are sent along from $X_2 - X_3 - X_6 - X_7$ to $X_3 - X_4 - X_5$, then

$$\begin{aligned}
\psi_{2,3,6,7}^{*} &= \psi_{2,3,6,7} \\
\phi_{2,3,6}^{*} &= \sum_{x_7} \psi_{2,3,6,7}^{*} \\
\psi_{1,2,3,6}^{*} &= \frac{\phi_{2,3,6}^{*}}{\phi_{2,3,6}} \psi_{1,2,3,6} = \psi_{1,2,3,6} \sum_{x_7} \psi_{2,3,6,7} \\
\phi_{1,3,6}^{*} &= \sum_{x_2} \psi_{1,2,3,6}^{*} \\
\psi_{1,3,5,6}^{*} &= \frac{\phi_{1,3,6}^{*}}{\phi_{1,3,6}} \psi_{1,3,5,6} = \sum_{x_2} \psi_{1,2,3,6} \sum_{x_7} \psi_{2,3,6,7} \\
\phi_{3,5}^{*} &= \sum_{x_1,x_6} \psi_{1,3,5,6}^{*} = \sum_{x_1,x_6} \sum_{x_2} \psi_{1,2,3,6} \sum_{x_7} \psi_{2,3,6,7} \\
\psi_{3,4,5}^{*} &= \frac{\phi_{3,5}^{*}}{\phi_{3,5}} \psi_{3,4,5} = \psi_{3,4,5} \sum_{x_1,x_6} \sum_{x_2} \psi_{1,2,3,6} \sum_{x_7} \psi_{2,3,6,7},
\end{aligned}$$

and the backward propagation will result in the messages

$$\begin{aligned}
\psi^{**}_{3,4,5} &= \psi^{*}_{3,4,5} = \psi_{3,4,5} \sum_{x_1,x_6} \sum_{x_2} \psi_{1,2,3,6} \sum_{x_7} \psi_{2,3,6,7} \\
\phi^{**}_{3,5} &= \sum_{x_4} \psi^{**}_{3,4,5} \\
\psi^{**}_{1,3,5,6} &= \frac{\phi^{**}_{3,5}}{\phi^{*}_{3,5}} \psi^{*}_{1,3,5,6} \\
\phi^{**}_{1,3,6} &= \sum_{x_5} \psi^{**}_{1,3,5,6} \\
\psi^{**}_{1,2,3,6} &= \frac{\phi^{**}_{1,3,6}}{\phi^{*}_{1,3,6}} \psi^{*}_{1,2,3,6} \\
\phi^{**}_{2,3,6} &= \sum_{1} \psi^{**}_{1,2,3,6} \\
\psi^{**}_{2,3,6,7} &= \frac{\phi^{**}_{2,3,6}}{\phi^{*}_{2,3,6}} \psi^{**}_{2,3,6,7}.
\end{aligned}$$

4.4.6 The Hugin Algorithm

The junction tree algorithm, as mentioned previously, is a general framework, and there are several implementations of it. The implementation that we used is the Hugin algorithm [17]. The steps described in the preceding sections are, in fact, the steps of the Hugin algorithm. In summary, there are four or five steps in the algorithm, depending on whether we start with a undirected or directed graph. The steps are as follow

1. moralise (only necessary if the graph is directed),
2. introduce evidence,
3. triangulate the graph if necessary to produce a chordal graph, and define the potentials from the underlying graphs,
4. construct a junction tree and
5. propagate probabilities throughout the junction.

Another implementation of general junction tree framework is the Shafer-Shenoy algorithm, in which no separator potentials are used and thus the propagation of messages differs from the Hugin algorithm. Although there are a number of implementations of the junction tree, and although the way they propagate messages tend to differentiate them from each other, the core remains the same—chordal graphs and junction tree property.

Chapter 5

Application

We now have all the background knowledge of graphical models that we need to apply graphical models to find coplanar points. In our application we consider two images, with the user choosing points on both images. On the first image, the points are chosen on planar regions: there can be multiple planes per image. On the second image, a large set of points is chosen. This set must contain the points corresponding to those chosen in the first image and any other randomly selected points. Although the system currently needs user input from both images, it is possible to use corner detection on the second image, or both images, to find candidate corresponding points. The problem with using a corner detection algorithm to find points on the second image is that the points chosen on the first image might not be picked up by the corner detector. Also, since the focus of this thesis is not on image processing, but rather the application of graphical models to find coplanar points, corner detection algorithms were abandoned after some initial experiments. An advantage of the graphical model approach is that the points do not have to match perfectly, i.e. allowance for jitter is made. This is important since no human input data is perfect.

At least six points per plane were specified. The algorithm developed then matches these points, per plane, to the randomly selected points on the second image.

Before looking at the graphical model we used in our experiments, we first need to consider some computer vision theory—looking at the different transformations between images, in Section 5.1. We then define our model using graph theory in Section 5.2, and using probability theory in Section 5.3. The experiments are dealt with in more detail in Section 5.4 and the results are discussed in Section 5.5.

5.1 Transformations

When two different images are taken of the same scene from different viewpoints, there is a transformation of the position of the points in the images. There are four different types of transformations—Euclidean, similarity, affine and projective—and each of these require a different number of point correspondences. In general, a homography matrix can be defined to represent these transformations. If we have the point $\mathbf{x}$ on the first image, in homogeneous coordinates, transformed to the point $\mathbf{x}'$ on the second image, we represent this transformation by matrix H, such that $\mathbf{x}' = H\mathbf{x}$, i.e.

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{5.1}$$

We now present the four different transformations and describe how they are represented. Note that although we are considering transformations between two images in our application, for illustrative purposes, we plot the transformations on one image when we illustrate their properties. For more details, see for example [19].

5.1.1 Euclidean Transformation

The first of our four transformations is the Euclidean transformation. It is the simplest of the four, accommodating only a rotation and translation. In Figure 5.1 an Euclidean transformation is illustrated. The original house is indicated by a solid line and the transformed house by a dashed line. It is clear that in this transformation, the length, angles and area are preserved. In addition, parallel lines stay parallel. An Euclidean transformation is written as

$$\begin{aligned} \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} &= \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \end{aligned}$$

with R a 2×2 rotation matrix and $\mathbf{t}$ a translation vector.

There are therefore three degrees of freedom in this transformation—one for rotation and two for translation. Two point correspondences are therefore needed to determine the parameters of this transformation.

5.1.2 Similarity Transformation

A similarity transformation is a Euclidean transformation with a scaling factor. In Figure 5.2 a similarity transformation is illustrated. The original house is again indicated by a solid

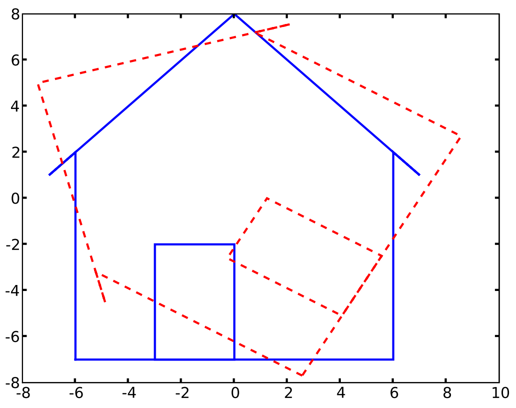


Figure 5.1: *Euclidean transformation illustrated.*

line and the transformed house by a dashed line. In a similarity transformation angles, the ratio of the lengths, and ratios of the area are preserved. Parallel lines again stay parallel. A similarity transformation is written as

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

with R a 2×2 rotation matrix, s a scalar and $\mathbf{t}$ a translation vector.

There are therefore four degrees of freedom in this transformation—one for rotation, one for the scaling variable and two for translation. We therefore need two point correspondences to determine the parameters of this transformation.

5.1.3 Affine Transformation

In an affine transformation, the ratio of the areas and the ratio of lengths of parallel lines are invariant, whilst parallel lines still remain parallel. It is a nonsingular linear transformation which is then followed by a translation. When looking at an affine transformation in matrix form, it is given by

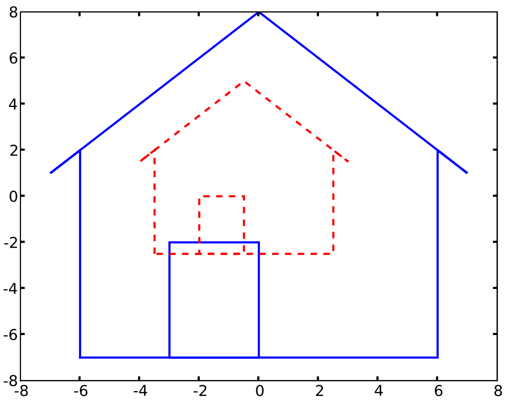


Figure 5.2: *Similarity transformation illustrated.*

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$
$$= \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

with A a nonsingular 2×2 matrix and $\mathbf{t}$ a translation vector.

An affinity has six degrees of freedom, and we therefore need three point correspondences to compute the transformation. In Figure 5.3 is an example of an affine transformation. Note that parallel lines (the walls of the house) do indeed remain parallel under this transformation.

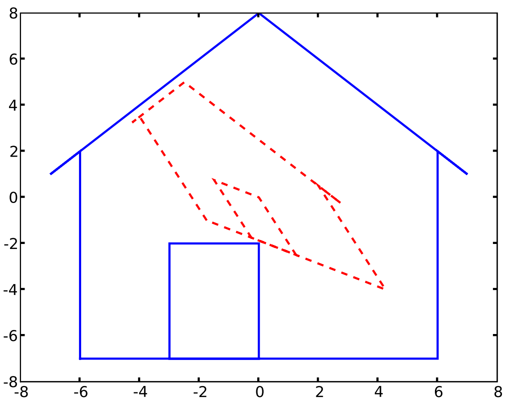


Figure 5.3: *Affine transformation illustrated.*

5.1.4 Projective Transformation

The projective transformation is the most general transformation presented here. In a projective transformation there are no invariances except for the cross ratio of four points. In matrix form, a projective transformation is given by

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

with A and $\mathbf{t}$ defined as before and $\mathbf{v}^T$ and v not necessarily nonzero.

Even though there are nine elements, we only have eight degrees of freedom, due to the fact that we are using homogeneous coordinates. We thus need four point correspondences to find the transformation. An example of a projective transformation is given in Figure 5.4. Note that the parallel lines of the house's walls are now transformed to non-parallel lines.

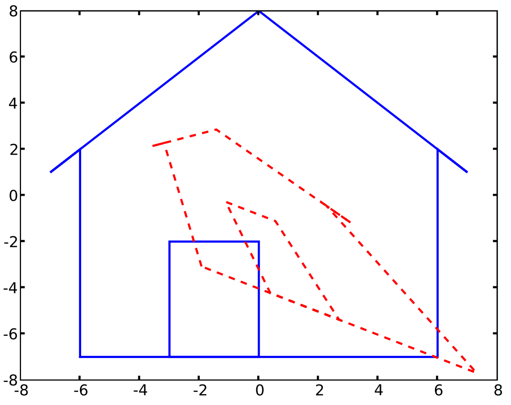


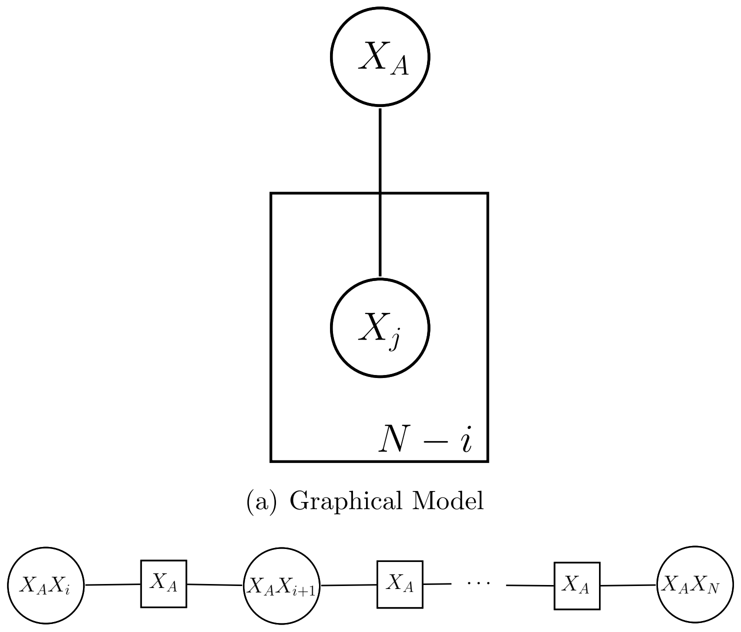
Figure 5.4: *Projective transformation illustrated.*

5.2 Graph Representing Variables and Interactions

The model that we use is based on the model derived in [6]. We consider the interactions between the five points (the four points used in finding the homography together with the fifth point which is then transformed and plotted on the second image) to be local interactions, and taking all these interactions together, we get the global interactions between the points. As discussed earlier, graphical models provide us with a tool to model modular systems, which this is. We use a MRF to represent the distribution $p(x)$, and thus, in general the distribution is given by (3.11), as discussed in Section 3.4.2.

Let the set of variables $\{x_1, x_2, \ldots, x_N\}$ indicate the points chosen on the first image, and the variables $\{y_1, y_2, \ldots, y_M\}$ the points chosen on the second image, with $N \leq M$. The variables $\{x_1, x_2, \ldots, x_N\}$ are seen as the random variables in the model, and the variables $\{y_1, y_2, \ldots, y_M\}$ as the possible realisations of the random variables.

The general form of the graphical model we use to represent transformations is given in Figure 5.5(a). The set of nodes X_A, where A is a set of indices, is representative of a complete set of nodes of either size two, three or four, depending on which of the transformations is used. Let the size of the complete set of nodes be i, then the number of nodes represented by the plate is $N - i$, since the size of the set of nodes in the first image is N, and if i of them are used in the complete set of nodes, then there are $N - i$ nodes left that need to be represented.



(a) Graphical Model

(b) Junction Tree

Figure 5.5: *General form of graphical model and junction trees for transformations.*

The size of the complete set of nodes is determined by the type of transformation we use, due to the nodes in this set representing the point correspondences. We saw in Section 5.1 that we either need two (Euclidean and similarity transformations), three (affine transformation) or four (projective transformation) point correspondences, and thus $i = 2, 3, 4$.

To illustrate how this method works, consider the situation where we are modelling an Euclidean transformation. The points represented in the complete set is used to calculate the transformation, i.e. the values of θ and $\mathbf{t}$. This information is then used to calculate the energy function for the other variables. Consider the situation where $x_1 = y_1$ and $x_2 = y_2$, where '=' denotes 'correspond to'. We then find the corresponding θ and $\mathbf{t}$ by using the equations in Section 5.1.1. We now use this information to transform every point in the set $\{x_3, x_4, \ldots, x_N\}$ onto the second image, and then use an error-function to define how well the transformation performs, for every variable. For every joint realisation, we find a probability, and then use MAP to decide which is the configuration that is most likely.

Note that after we use the junction tree algorithm to find the MAP values. The junction tree algorithm, for the general model, is depicted in Figure 5.5(b).

5.3 Potentials Used in Model

Now that we have seen what the graphical model looks like, we need to define the potentials. From the graph in Figure 5.5(a) we see that the cliques are defined as $X_A - X_j$ with $j = \{i+1, \ldots, N\}$. The general definition of a joint probability over an MRF, (3.11), can then be restructured for this case to be, with $i = 1, 2, 3, 4$,

$$p(x_1, \ldots, x_i, \ldots, x_N) = \frac{1}{Z} \prod_{j=i+1}^{N} \psi_j(x_1, \ldots, x_i, x_j). \tag{5.2}$$

We now define an energy function $U(x)$ such that $U_k(x) = -\log \psi_k(x)$. The maximiser of 5.2 will therefore be the minimiser of

$$U(x_1, \ldots, x_i, \ldots, x_N) = \frac{1}{Z} \prod_{j=i+1}^{N} U_j(x_A, x_j). \tag{5.3}$$

Thus, we can see the joint probability distribution as

$$p(x) = \frac{1}{Z} \exp[-U(x)]. \tag{5.4}$$

As discussed in Section 4.3, the logarithms are used because multiplying small probabilities with each other can lead to numerical instability. Our choices for the energy functions $U(x)$ are discussed in the next section, where the implementation of our work is looked at.

5.4 Numerical Experiments Performed

Due to the nature of the images that we work with, we have to work with projective transformations when we look at point correspondences between the images. We therefore have a clique of size five, and the graphical model is as in Figure 5.6. Furthermore, the joint probability is given by (5.4) with

$$U(x) = \frac{1}{Z} \prod_{j=5}^{N} U_j(x_1, x_2, x_3, x_4, x_j).$$

The article that this work is based on, [6], defined as its potential function the Euclidean distance between points. We looked at this potential function, as well as three others.

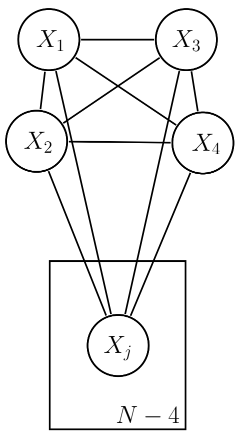


Figure 5.6: *Graphical model used in application.*

5.4.1 Energy function: Euclidean distance

To use the Euclidean distance as an energy function, the homography between the points first had to be established.

It was mentioned earlier that the points that are entered by the user are given in planar regions. We know therefore that the points in the first image are on a plane, and we want to find the plane in the second image corresponding to the plane in the first image. Note that thereafter the second image can be used to determine the corresponding planes in sequential images. Firstly, four points are selected from the second image, and the homography between the four points on the first image and the four points on the second image is calculated. These four points are represented by the set X_A in the general graphical model. The homography that is then found is used to transform all the other points on the same plane in the first image to points on the second image. The potential is then found by finding the Euclidean distances between the points defined by the user on the first image and the transformed points of $\mathbf{x}$ on the second image. For example‘, consider Figure 5.7. Assume Figure 5.7(a) be part of the first image and the dots in Figure 5.7(b) the part of the second image that we look at. Let the points $(0,0)$, $(0,1)$, $(1,1)$ $(1,0)$ be the points on the second image that are used in finding the homography. Suppose that we find a homography, H_{p_1}, that transforms the points $(4,6)$, $(5,3)$ to the points represented by the ‘x’ on the graph. Furthermore, let there be a second homography, H_{p_2} that transforms the points $(4,6)$, $(5,3)$ to that represented by the ‘+’. We see that the homography H_{p_1} transforms the points to points that lie much closer to the points on the second image than the homography H_{p_2}. If we now define the energy function to be the Euclidean distance between the points, the points indicated by a ‘x’ have a smaller distance than those represented by the ‘+’. When we find the minima for the Euclidean distances, the homography H_{p_1} is seen to perform the best, and therefore, the points that were used from the second image to find this homography is viewed as the realisations for the points on the first image.

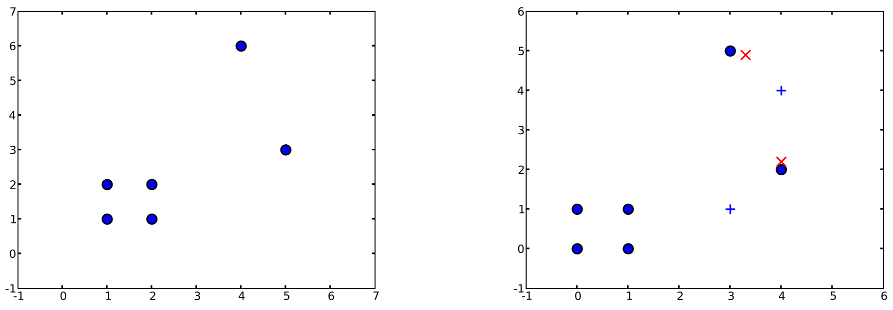
To illustrate how the energy function U_k is defined, consider the case where $x_1 = y_1$, $x_2 = y_4$, $x_3 = y_9$ and $x_4 = y_2$, where the x_i's are the points on the first image and y_j's points on the second image. We find the homography H_p from these specified realisations of x_1, x_2, x_3 and x_4. Furthermore, suppose we now want to find the energy function $U_k(x_1, x_2, x_3, x_4, x_5)$. We first find

$$\mathbf{x}' = H_p \mathbf{x}_5.$$

We then find the Euclidean distance between x' and each of the realisations y_j's that are not already considered in the realisations of $x_1, \ldots, x_4$. Therefore, if we consider the case where y_5 is considered the realisation of x_5,

$$U_k(x_1, x_2, x_3, x_4, x_5) = \sqrt{(x_1' - y_{5_1})^2 + (x_2' - y_{5_2})^2}. \tag{5.5}$$

Note that we do not consider the case where a point on the second image can be used for more than one realisation in a particular energy function, when setting up the potential matrix. The reason for this is that we specifically are choosing points such that they correspond to differing points in the second image. Due to this being a discrete probability, we set up a matrix to represent the data, and then choose the minimising arguments for all the points in the first image so as to find the best fit to the data.



(a) Image 1 (b) Image 2

Figure 5.7: *Euclidean transformation explained.*

Finding a homography is relatively expensive, and since the homography between all the combinations of four points of the second image needs to be found for every plane, using the Euclidean distance between points can become quite expensive.

Using the Euclidean distance only to define the potential sometimes leads to results that are less favourable, due to factors as geometry in an image. For instance, if there are two planes with similar dimensions on the images, often the algorithm cannot correctly specify which one of the two planes a point belongs to. To improve this situation, the grey-scale values of the image was considered. One way of incorporating the grey-scale values of an image, is by the correlation of segments of the image.

5.4.2 Potential function: Correlation

Correlation can be seen as to how well two images match each other. The grey-scale values are looked at, and for every pixel, these values are then compared. If the grey-scale values are close together, there will be a high correlation, but the further apart they are, the lower the correlation becomes. Although in general rotations of the images are also taken into consideration when finding the correlation, because we have chosen our images to follow each other relatively closely, we do not consider rotations and translations. For a more precise definition and details, see [16, chapter 4].

We used a neighbourhood of twenty pixels on each side of a specified point and compared it to the neighbourhood of the point it is being compared to. An average for the correlation is then found and used as the correlation coefficient of the two images. Note that when a point was within the boundaries of twenty pixels away from an edge, the neighbourhood was shifted such that the same amount of grey-scale values were still used, even though the chosen point would then not be at the centre of the images being correlated. We define correlations to lie between zero and one, with zero being that there is no correlation—for instance the grey-scale value in the first image is 0 whilst in the second image is 250—and one being that they are perfectly correlated—pixels in both images have the same value.

As an example, consider the images in Figure 5.8. By inspection it is clear that Figure 5.8(a) and Figure 5.8(c) are very similar and in fact their correlation is 0.96497841. Figure 5.8(a) and Figure 5.8(b) are not highly correlated, and have a correlation coefficient of 0.50647062.

Due to the fact that correlations are already scaled to be between 0 and 1, we do not find the logarithms when finding the potentials ψ. The potential function is, if $\mathrm{corr}(a, b)$ represents the correlation coefficient between the points a and b,

$$\psi(x_1, x_2, x_3, x_4, x_i) = 0.2(\mathrm{corr}(x_1, y_j) + \mathrm{corr}(x_2, y_k) + \mathrm{corr}(x_3, y_l) + \mathrm{corr}(x_4, y_m) + \mathrm{corr}(x_i, y_n)), \tag{5.6}$$

with $i \neq k \neq l \neq m \neq n$.

Finding the correlations between points is not as expensive as finding the homography between points, and an added bonus is that the correlations between all the pairs of points can simply be calculated and stored once. Once that is done, finding the potentials become faster since one then just needs to look up the correlation between two points in a look-up table.

5.4.3 Potential function: Euclidean Distance and Correlation

Neither of the two methods, either using the Euclidean distance or correlations, results in the correct results at all times. As already mentioned, geometry of a scene plays a major role in the Euclidean distance potential function, whereas with correlations, sometimes there are

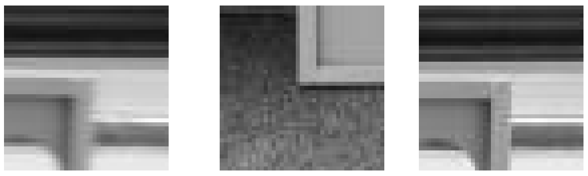

(a) Image One (b) Image Two (c) Image Three

Figure 5.8: *Example of correlation in images.*

areas that are very similar but they are not the same point. Using a combination between the two methods is therefore something that could improve the results.

Firstly, consider the situation where the Euclidean distance method is used, as described in Section 5.4.1. The correlation between the point that was found via the transformation and the different realisations are then used in finding the potential function. Thus, if the energy function is (5.5), then the potential ψ will be

$$\psi_k = 0.5(\exp(U_k(x_1, x_2, x_3, x_4, x_k)) + \text{corr}(x_k, y_m)), \tag{5.7}$$

where y_m is the possible realisation of x_k.

If we want to weigh the correlation probability more than the Euclidean distance, we add a weight of 0.75 to the correlation and 0.25 to the Euclidean distance, causing the potential to become

$$\psi_k = 0.25 \exp(U_k(x_1, x_2, x_3, x_4, x_k)) + 0.75\text{corr}(x_k, y_m). \tag{5.8}$$

5.5 Results of Experiments Discussed

In Figure 5.9 the images that were used for the experiments are given, and in Figure 5.10 the points chosen on the images are illustrated. Note that although the points are indicated per plane in Figure 5.10(b), they were not indicated as lying on the planes when they were given as input to the algorithm. The planes in Figure 5.10(b) indicate what the correct coplanar points are.

Due to the fact that there are memory errors in the software that we used, only three planes could be considered at a time, and thus we considered two sets of planes. In the first iteration we considered planes one, two and three, and in the second iteration planes three, four and five.

5.5.1 Planes 1,2,3

In Figure 5.11 the coplanar points found by using the potentials as defined in Section 5.4.1 and Section 5.4.2, are shown, respectively. Note that although we specify in the potential matrix that we use for the junction tree algorithm we specify that there cannot be two variables with the same realisation, unfortunately when running the junction tree algorithm, this is not specified, and in some instances the algorithm then chooses a configuration where two variables have the same realisation. An example of this can clearly be seen in Figure 5.12(b), where it appears that one of the points of plane 3 has no realisation. In truth, when looking at the numerical results, it is because the fifth and sixth points on the plane have the same realisation, and we therefore cannot see it. When only discussing coplanar points, this is not too much of a problem, seeing as it still means that the points are viewed to be on the same plane, however. In the same trend, the reason we are not looking at the specific realisations of the points is due to the fact that we are only interested in planes matching planes, and not necessarily in point matching.

For the examples in Figure 5.11 we can clearly see that using only the correlation outperforms using the Euclidean Distance to find the coplanar points. One of the reasons for this could be that the structures are all very rectangular, and in actual fact, the corner points were chosen to represent the variables X_1, X_2, X_3, X_4. The points are all also very close together, and this could lead to the a sensitivity when using the Euclidean distance method.

When viewing the combination of the Euclidean distance and correlation methods, as discussed in Section 5.4.3, we have the results as in Figure 5.12.

It is interesting to note that although the results for the correlation and Euclidean distance methods separately did well, combining the two methods have produced less favourable results, although if the correlations are weighed heavier, then the results are not too bad.

5.5.2 Planes 3, 4, 5

In Figure 5.13 the coplanar points found by using the potentials as defined in Section 5.4.1 and Section 5.4.2, respectively. For the planes in Figure 5.13, both the Euclidean distance method and the correlation method results in the correct coplanar regions.

Inspecting the combination of the Euclidean distance and correlation methods, as discussed in Section 5.4.3, we obtain the results as in Figure 5.14.

Again, weighing the correlation method more than the Euclidean distance method produces better results, but these results are still not as good as those when using either one of the two methods alone. In general, this was the result.

5.5.3 General Discussion of Results

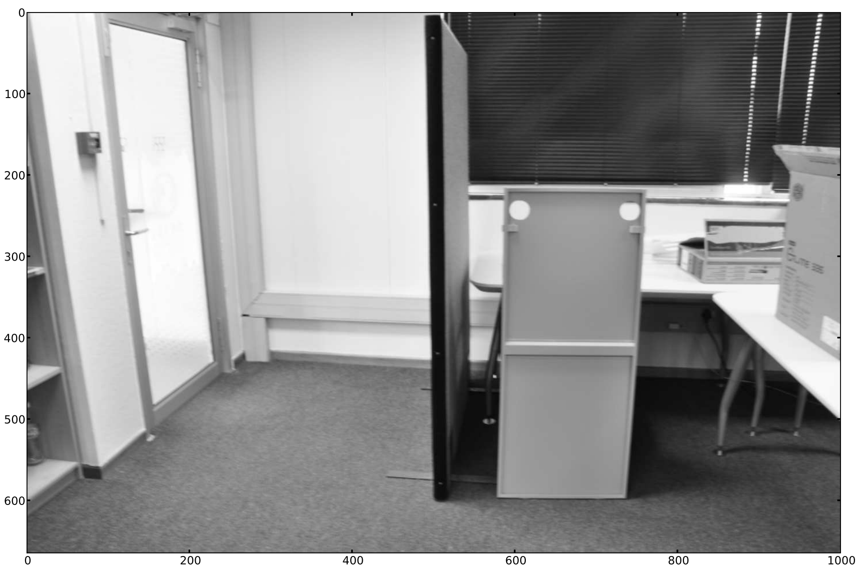
It was previously mentioned that there can be any amount of points on the second image, as long as the planar points of the first image is on the second image too. Although there were no extra points chosen on the second image, it can be seen that the points corresponding to the other planar regions are extra-points when looking at a single plane.

After some experimentation, it became clear that six points were the ‘optimal’ when looking at finding planar regions. The algorithm was quite unstable when only chosing five points, and the more points where chosen on a specific plane, the longer the algorithm ran, and memory leaks occurred within the software that we used. In addition to this, we stated earlier that one of the reasons we would like to find coplanar regions is to improve the 3D reconstruction of featureless areas. We would therefore prefer to choose as few points as possible.

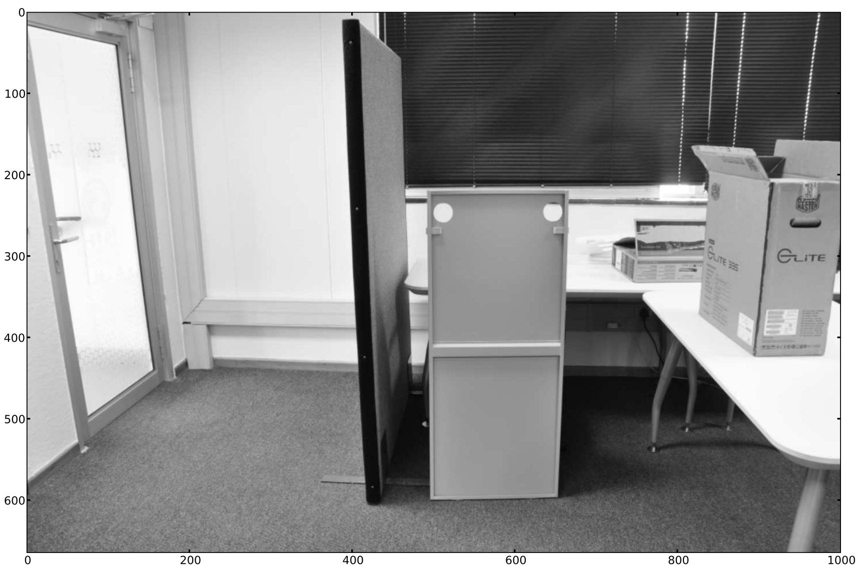
Ideally we would want the points chosen to be automatically generated. However, initial experiments with image processing techniques such as corner, edge and blob detection did not meet our needs—they did not provide the required information to fulfil our criteria on points chosen. Since the focus of this thesis is on graphical models and not on automating graphical models software, we opted for manual user input.

During experimentation, it was noticed, as previously mentioned, that the closer the points are chosen on a plane, the greater the possibility of the results being less accurate. In practise, if this method would be used, one would therefore want to choose the points in such a way that they are relatively far apart, and also that geometric structures of the planes are not too much alike, since, as discussed earlier, this will also lead to less accurate results.

The average time taken for the algorithm to run, per plane, was 150 seconds. This using the Python implementation of [17]. Although the junction tree algorithm itself did not take long to run—on average about 1 second—setting up the potential matrix took quite a lot of time, since we had to iterate through all the possible realisations for every variable. This method is therefore too slow to be implemented in practise, if real-time rendering is needed/wanted.

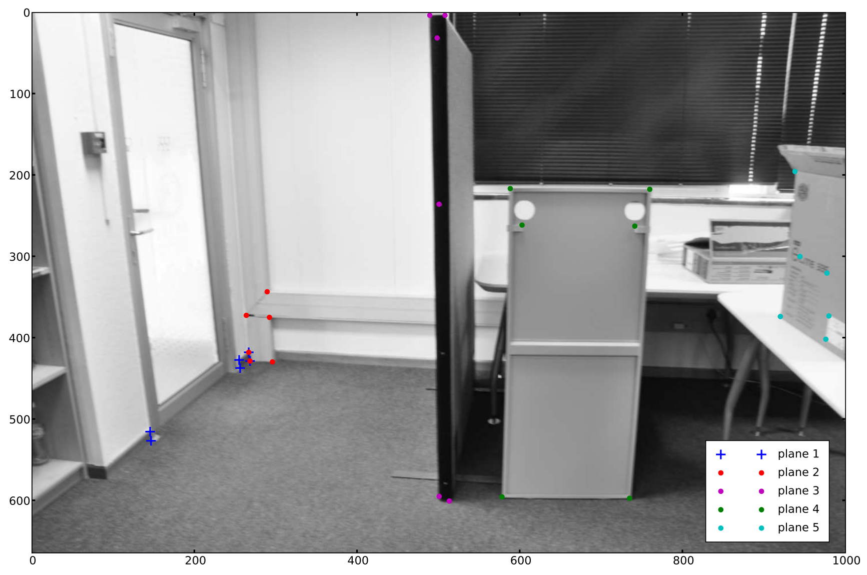


(a) First Image

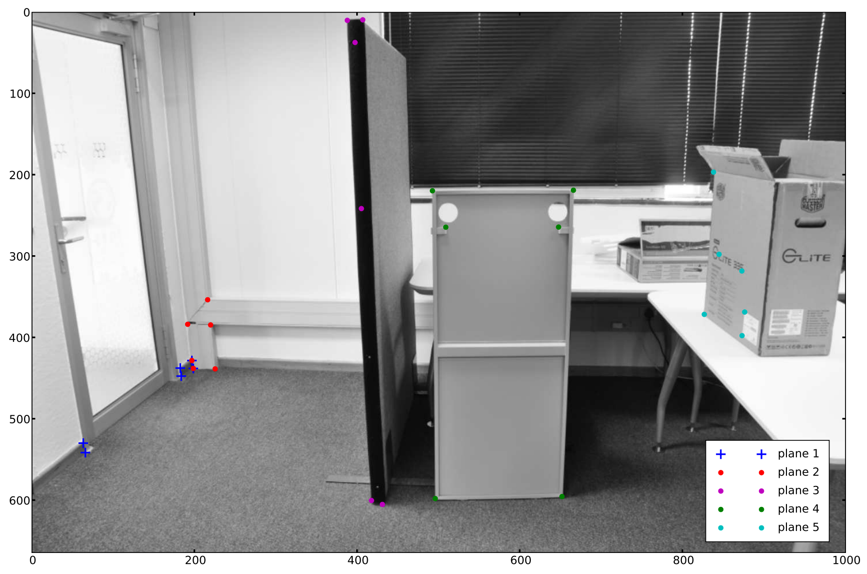


(b) Second Image

Figure 5.9: *Images used in experiments.*

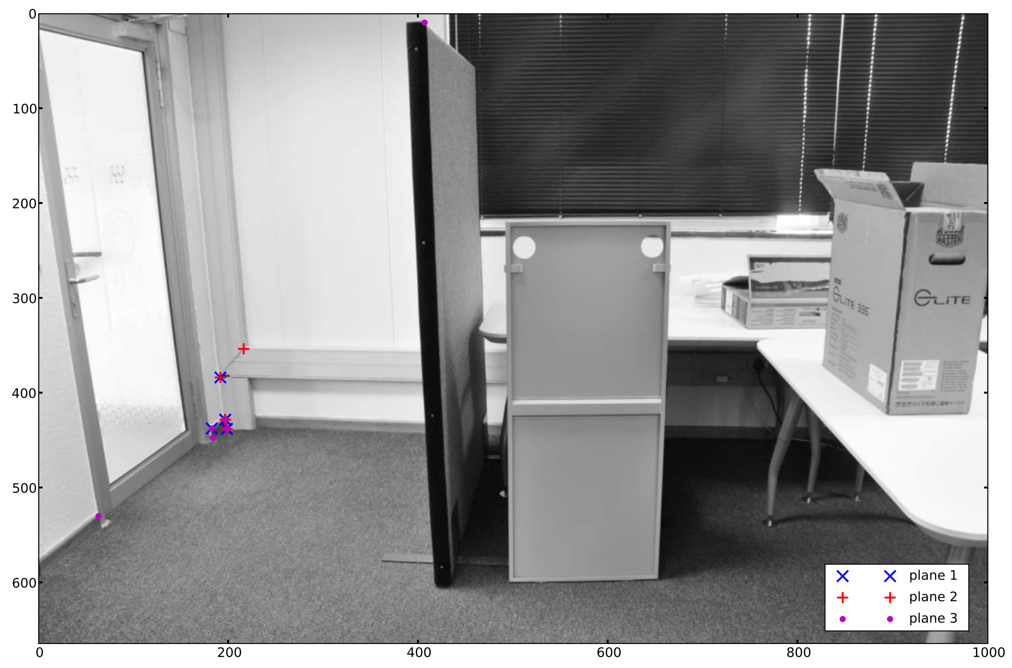


(a) First image with chosen points.

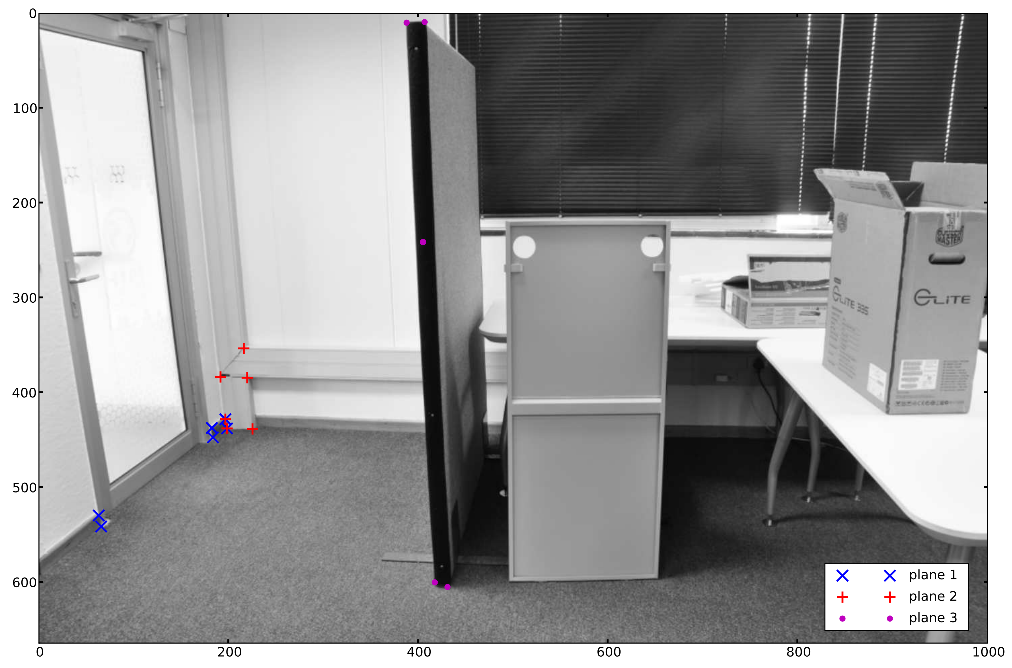


(b) Second image with chosen points.

Figure 5.10: *Points used indicated on images.*

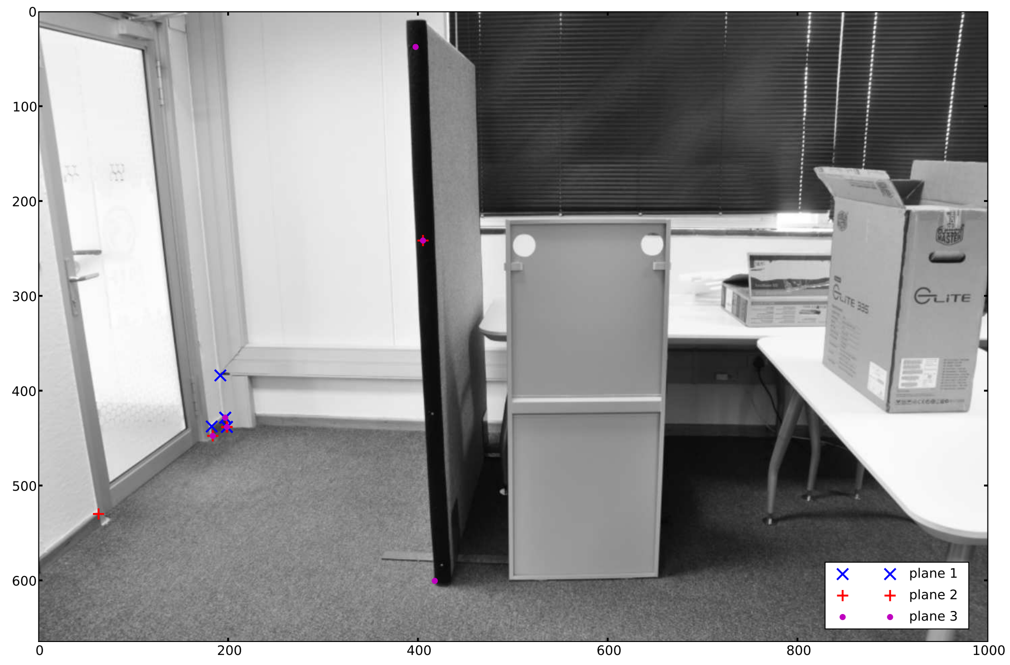


(a) Euclidean Distance

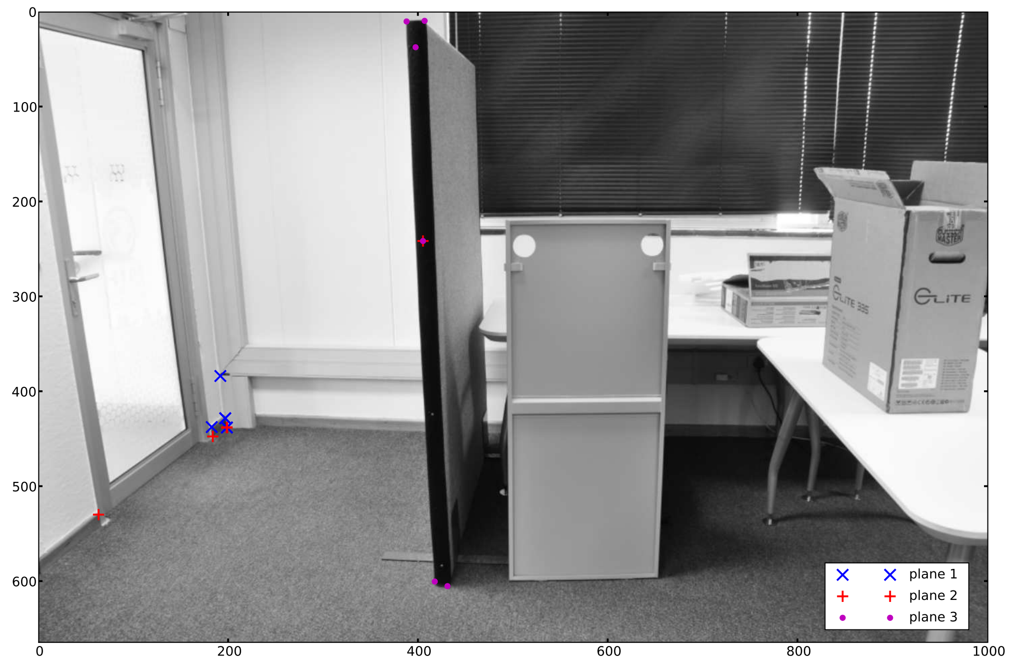


(b) Correlation

Figure 5.11: *Planes 1,2,3 calculated coplanar points using (5.5) and (5.6).*

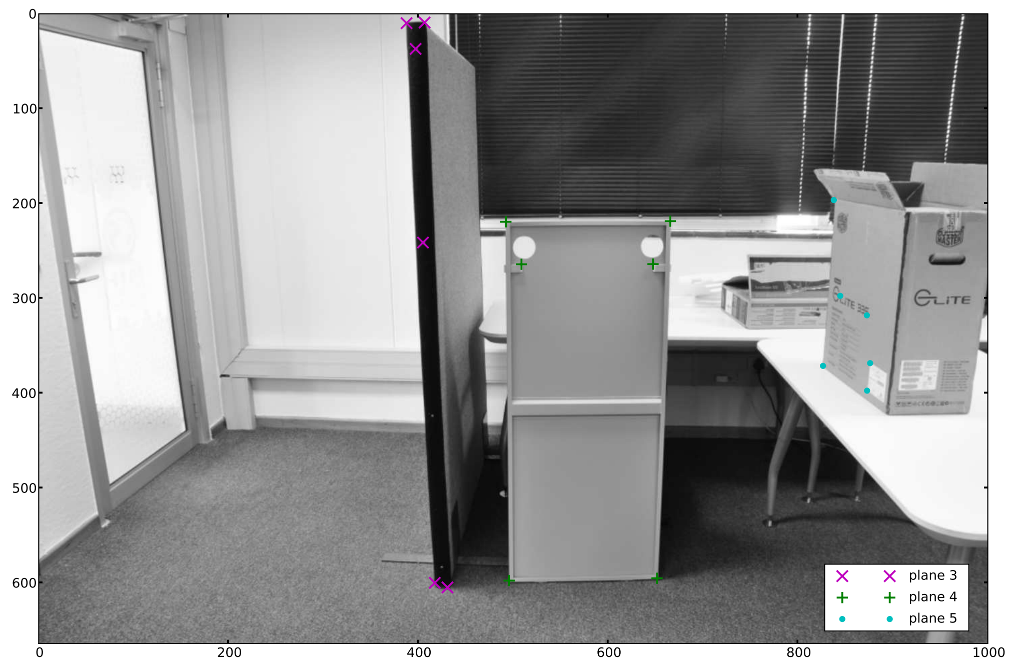


(a) Potential as in (5.7)

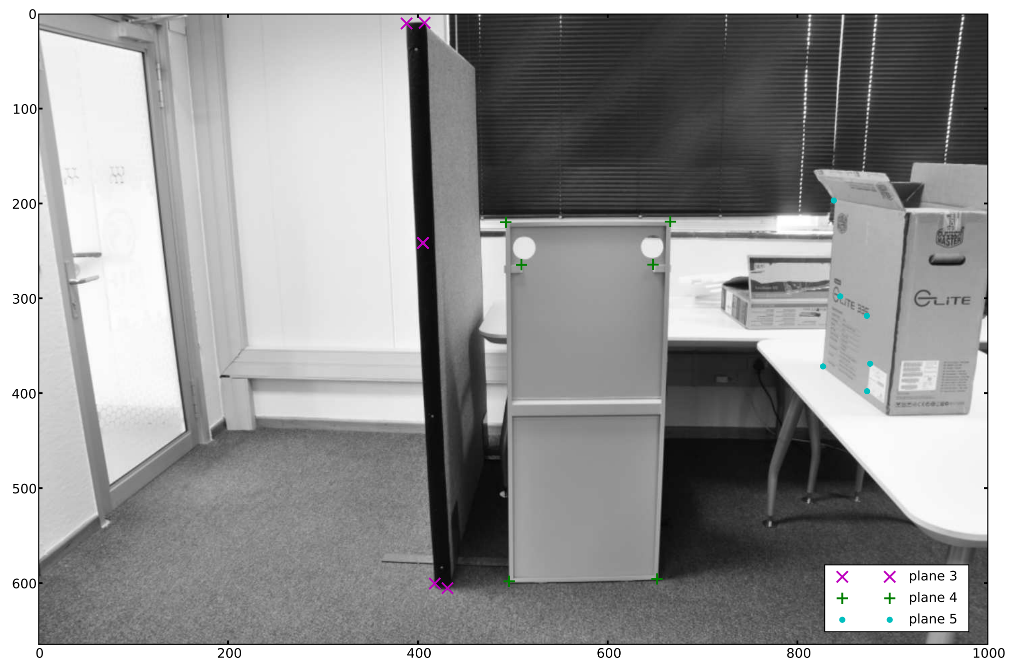


(b) Potential as in (5.8)

Figure 5.12: *Planes 1,2,3 calculated coplanar points using (5.7) and (5.8)*

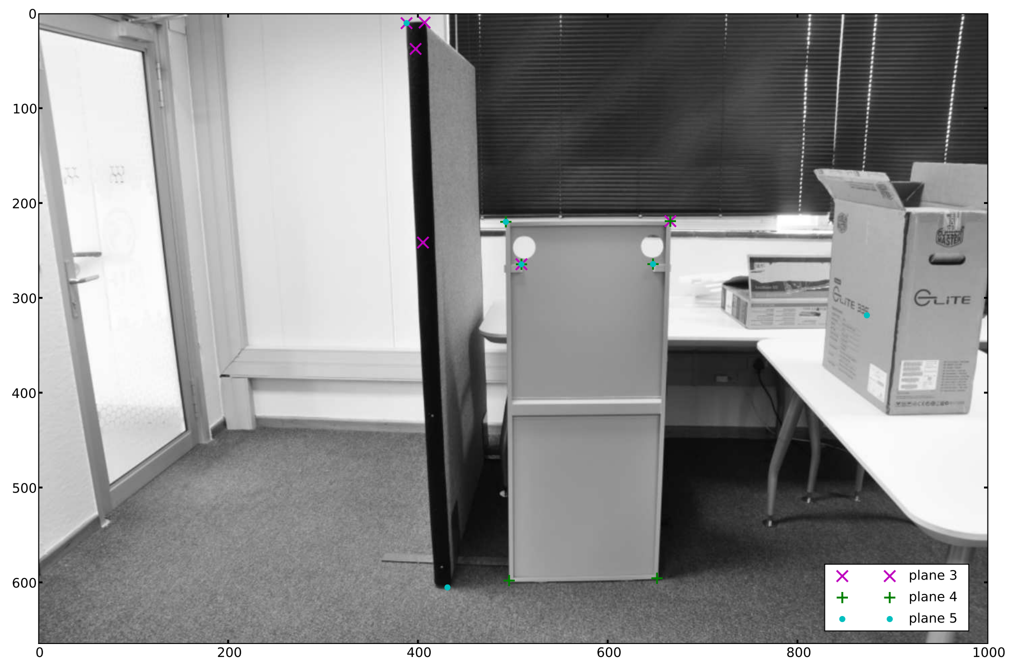


(a) Euclidean Distance

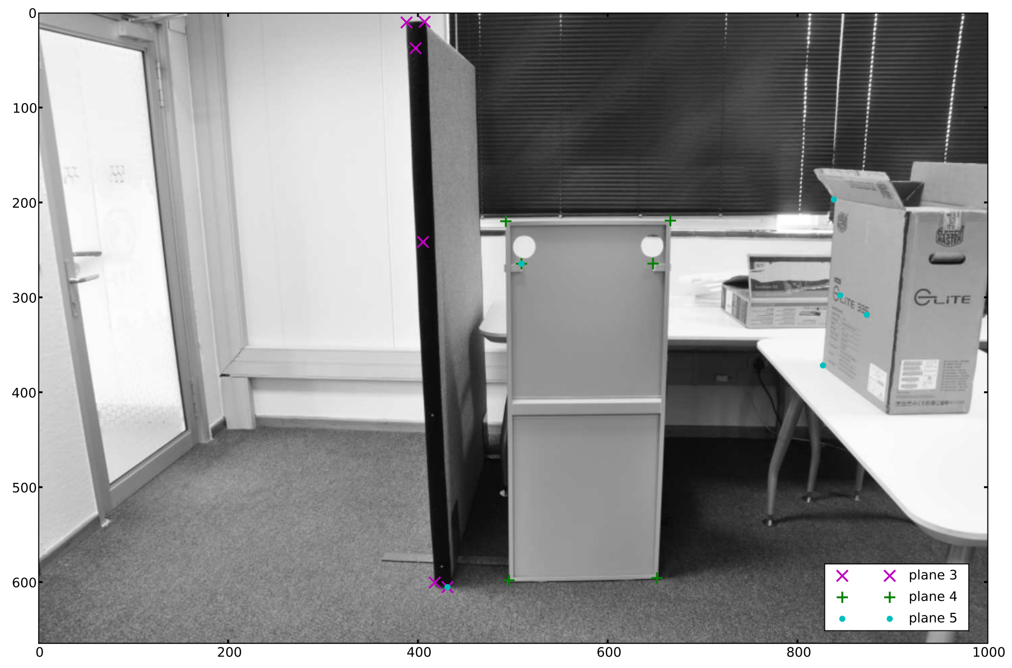


(b) Correlation

Figure 5.13: *Planes 3,4,5 calculated coplanar points using (5.5) and (5.6).*



(a) Potential as in (5.7)



(b) Potential as in (5.8)

Figure 5.14: *Planes 3,4,5 calculated coplanar points using (5.7) and (5.8).*

Chapter 6

Conclusions

Although the focus of the thesis is not finding coplanar points, but rather graphical models, we have shown in our experiments that graphical models can be used to find coplanar points using images from a single uncalibrated camera. The main drawbacks of this technique is the runtime and the sensitivity to small planar regions. We used a Python implementation, which implies nonoptimality. A C++ implementation would only give limited improvements: the model is inherently complex.

The order of the runtime of the junction tree algorithm depends on the size of the largest maximal clique in the relevant graphical model, in our case size five. Therefore we could speed up the runtime by reducing the size of the maximal clique. For example, if we used calibrated camers we could use affine transformations instead of projective transformations, and thus reduce the size of the maximal clique to four and therefore reduce the order of the runtime.

Future experiments could include learning a graphical model from a large set of data containing coplanar points. Thereafter using the learned model to draw certain conclusions.

In conclusion, graphical models are flexible and can be used in a variety of applications, though it is not always the optimal modelling technique.

Bibliography

[1] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2559, 1999.

[2] M.S. Bartlett. Contingency table interactions. *Journal of the Royal Statistical Society Supplement*, 2:248–252, 1935.

[3] Berrington, Ann, Hu, Yongjian, Smith, W. F. Peter, Sturgis, and Patrick. A graphical chain model for reciprocal relationships between women's gender role attitudes and labour force participation. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 171(1):89–108, January 2008.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, October 2007.

[5] Tibério S. Caetano. `http://dip.sun.ac.za/~vision/?q=GraphicalModels`, 2008.

[6] Tibério S. Caetano and Terry Caelli. A unified formulation of invariant point pattern matching. In *ICPR (3)*, pages 121–124, 2006.

[7] Robert Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Vision and Pattern Recognition*, pages 358–363, June 1996.

[8] T.H. Cormen, C.E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2 edition, 2001.

[9] Dragos Datcu and Leon Rothkrantz. Automatic recognition of facial expressions using Bayesian Belief Networks. In *Proceedings of IEEE SMC 2004*, pages 2209–2214, October 2004.

[10] Peerapong Dhangwatnotai and Ting Zhao. Image segmentation using graphical models.

[11] A. R. Dick, P. H. S. Torr, S. J. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. *Computer Vision, IEEE International Conference on*, 1:268, 2001.

[12] Ronja Foraita, Stephan Klasen, and Iris Pigeot. Using graphical chain models to analyze differences in structural correlates of undernutrition in Benin and Bangladesh. *Economics and Human Biology*, 6(3):398–419, December 2008.

[13] F. Fraundorfer, K. Schindler, and H. Bischof. Piecewise planar scene reconstruction from sparse correspondences. *IVC*, 24(4):395–406, April 2006.

[14] W. Gibbs. *Elementary Principles of Statistical Mechanics.* Yale University Press, NewHaven, Connecticut, 1902.

[15] A. Goldenberg. *Scalable Graphical Models for Social Networks.* PhD, School of Computer Science, Carnegie Mellon University, 2007.

[16] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[17] A. Gouws. *A Python Implementation of Graphical Models.* MSc(Eng), Department of Electrical Engineering, University of Stellenbosch, South Africa, 2010.

[18] Graham Grindlay and David Helmbold. Modeling, analyzing, and synthesizing expressive piano performance with graphical models. *Mach. Learn.*, 65(2-3):361–387, 2006.

[19] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[20] Qiang He and Chee-Hung Henry Chu. Planar surface detection in image pairs using homographic constraints. In *ISVC (1)*, pages 19–27, 2006.

[21] P.D. Hoff, A.E Raftery, and M.S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

[22] Michael I. Jordan. *Learning in graphical models.* MIT Press, 1998.

[23] Michael I. Jordan. An introduction to probabilistic graphical models. 2003.

[24] Sham Kakade, Michael J. Kearns, and Luis E. Ortiz. Graphical economics. In *COLT*, pages 17–32, 2004.

[25] S.L. Lauritzen. *Graphical Models.* USA Oxford University Press, 1996.

[26] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003. available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

[27] D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review / Revue Internationale de Statistique*, 63(2):215–232, 1995.

[28] Julian J. McAuley, Tibério S. Caetano, Alex J. Smola, and Matthias O. Franz. Learning high-order MRF priors of color images. In *ICML '06: Proceedings of the 23rd*

international conference on Machine learning, pages 617–624, New York, NY, USA, 2006. ACM.

[29] Julian J. Mcauley, Tib&#x00e9;rio S. Caetano, and Marconi S. Barbosa. Graph rigidity, cyclic belief propagation, and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2047–2054, 2008.

[30] Jean-François Paiement, Douglas Eck, Samy Bengio, and David Barber. A graphical model for chord progressions embedded in a psychoacoustic space. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 641–648, New York, NY, USA, 2005. ACM.

[31] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[32] Christopher Raphael. Aligning music audio with symbolic scores using a hybrid graphical model. *Mach. Learn.*, 65(2-3):389–409, 2006.

[33] P. Spirtes. Graphical models, causal inference, and econometric models. *Journal of Economic Methodology*, 12(1):3–34, March 2005.

[34] Marshall F. Tappen, Bryan C. Russell, and William T. Freeman. Efficient graphical models for processing images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:673–680, 2004.

[35] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002.

[36] O. Tremois and J.P. Le Cadre. Maneuvering target motion analysis using Hidden Markov Model. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 3:317–320, 1994.

[37] Dennis D. Wackerly, William Mendenhall, and Richard L. Scheaffer. *Mathematical Statistics with Applications.* Duxbury, 6 edition.

[38] N. Wermuth. Analysing social science data with graphical Markov models. In *Highly Structured Stochastic Systems*, pages 33–39. Oxford University Press, 2003.

[39] J. Whittaker. Graphical models in applied multivariate statistics. 1990.

[40] S. Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.

[41] Q. Yang, C. Engels, and A. Akbarzadeh. Near real-time stereo for weakly-textured scenes. In *British Machine Vision Conference*, 2008.

[42] L. Zhang and Q. Ji. Image segmentation with a unified graphical model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 5555.

[43] Lei Zhang and Qiang Ji. Image segmentation with a unified graphical model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 5555.

Appendix A

Nomenclature

A.1 Symbols and Notation

X	Random variable with $X = (X_1, \ldots, X_N)$, $N > 1$, other letters may also be used to specify random variables.
X_A	Random vector of variables indexed by set $A \subseteq \{1, \ldots, N\}$.
x	A realisation of random variable X with $x = (x_1, \ldots, x_N)$, note that it is not a specific realisation.
x_A	A realisation of variables indexed by $A \subseteq \{1, \ldots, N\}$.
$\bar{x}$	A particular realisation of X with $x = (x_1, \ldots, x_N)$.
$\mathcal{X}$	Set of all realisations (sample space).
$p(x)$	The joint distribution of x.
$p(x_A \vert x_B)$	The probability of x_A, given x_B. x_B is the conditioning variable.
N	Total number of random variables
$G(V, E)$	A graph G with vertex set V and edge set E.
V	Vertex set of a graph.
E	Edge set of a graph.
π_i	Set of parents of node/variable i. Thus X_{π_i} is the set of parents of X_i
ν_i	Set containing all ancestors except the parents of i, and (possibly) some other nondescendant nodes as well.
$X_1 \perp\!\!\!\perp X_2 \mid X_3$	X_1 is conditionally independent of X_2 given X_3.
C	Set of indices of a maximal clique.
$\mathcal{C}$	Set of all C in a graph G.
Z	Normalisation factor for MRF joint probability.

A.2 Acronyms

BN	Bayesian Networks
DAG	directed acyclic graph
HMM	Hidden Markov Models
MRF	Markov Random Field
MAP	maximum a posteriori