Variational Methods In Bayesian Image Restoration

DUC HUY DINH

MSc by Research in Pattern Analysis and Neural Networks

THE UNIVERSITY OF ASTON IN BIRMINGHAM

September 1997

THE UNIVERSITY OF ASTON IN BIRMINGHAM

Variational Methods In Bayesian Image Restoration

DUC HUY DINH

MSc by Research in Pattern Analysis and Neural Networks, 1997

Thesis Summary

We present a new approach for parameter estimation in the binary image restoration problem. Within the Bayesian context, we outline the basic conceptual principles of image restoration as well as evaluating some of the more common Monte Carlo techniques such as the Gibbs and Metropolis algorithms, and the much less well known Swendsen-Wang algorithm. We mainly concentrate on two key issues. First, we focus on the quality of restored images with respect to the choice of two restoration parameters that are generally not known. Second, we consider the difficulty in dealing with uncertainty in the restoration parameters. We compute the most likely parameters by maximising the distribution of the noisy image with respect to these parameters. This "evidence" procedure, developed by Gull and MacKay, is a computationally intractable problem. Rather than to resort to uncontrollable Monte Carlo methods to address this issue, we propose to rigorously approximate the evidence by using variational methods which have recently been developed in graphical models problems.

Keywords: Image restoration, Monte Carlo techniques, Recursive node-elimination, Decimation, Evidence, Thermodynamic integration, Mean field approximation

Acknowledgements

I would like to thank my supervisor David Barber for his constant help and considerable patience in answering my numerous questions. Without his insightful criticism and detailed comments, this thesis might not have been achieved.

I would also like to thank my school the Institut d'Informatique d'Entreprise IIE-CNAM and Aston University for allowing me to spend an exciting year in England.

Contents

1 Introduction **8**

1.1 Background 8

1.2 Bayesian formulation of the image reconstruction 10

1.2.1 Definition of the problem 10

1.2.2 Restoring binary images 11

1.3 The Bayesian framework for restoration parameters 13

2 Markov chain Monte Carlo techniques **15**

2.1 Why Monte Carlo methods 15

2.2 The Gibbs sampling algorithm 16

2.3 The Metropolis algorithm 18

2.4 Simulated Annealing 19

2.5 The Swendsen-Wang algorithm 20

2.6 Experimental results 23

2.6.1 Comparison of restored images 23

2.6.2 Influence of κ and β 27

2.6.3 Restoration of very noisy images 29

2.7 Discussion 29

3 Bounding techniques **31**

3.1 Boltzmann machines 31

3.1.1 Transformation of the partition function of the distribution for restored images 32

3.1.2 Transformation of the partition function of the Markov random field prior . . . 34

3.2 Recursive node-elimination 35

3.2.1 The lower bound 35

3.2.2 Alternative interpretation of the lower bound 37

3.2.3 The upper bound 40

4 Decimation **46**

4.1 Decimation of one node connected to two nodes 46

4.2 Decimation of one node connected to one node 48
4.3 Decimation of two nodes 48
4.4 Decimation of three nodes 49
4.5 Heuristic node-removal scheme 52

5 Parameter estimation **55**

5.1 Estimation using bounding techniques 55
5.2 Estimation using thermodynamic integration methods and mean field approximations 56
5.3 Prior determination 61
5.4 Comments 66

6 Conclusion **67**

List of Figures

1.1 True image T, noisy image D, restored image S 11
1.2 Illustration of the neighbourhood configuration 12

2.1 Sampling regions of high or low probability 20
2.2 Illustration of the Swendsen-Wang algorithm 22
2.3 True images and corrupted images generated with a noise level equal to 0.2 24
2.4 Images restored by simulated annealing (Gibbs, Metropolis) and the Swendsen-Wang algorithm for a same amount of CPU time $t = 50$ seconds 25
2.5 Images restored by simulated annealing (Gibbs, Metropolis) and the Swendsen-Wang algorithm for an equivalent amount of CPU time $t = 130$ seconds 26
2.6 Successive images from the Gibbs and Swendsen-Wang algorithms after an equivalent amount of CPU time t (5 seconds), for increasing t 27
2.7 Comparison of normalised Hamming distances for the images in Figure 2.6 from the Gibbs and Swendsen-Wang algorithms 28
2.8 Restored images for varying values of β 28
2.9 Restored images for varying values of q 29
2.10 Restoration of degraded images with a noise level equal to 0.3 30

3.1 Geometry of the lower bound 36
3.2 Enforcing tractable networks 37
3.3 Illustration of the tangent bound 38
3.4 Comparison between the log-lower bound estimates using the first approach and the log-lower bound estimates using the second approach for 16 and 256 nodes 39
3.5 Comparison, for 16 nodes, between the log-exact value and the log-lower bound obtained by the second approach involving optimisation over a one-dimensional space of variational parameters 40
3.6 Geometry of the quadratic upper bound 41
3.7 Graphical modifications following the recursive quadratic method 42
3.8 Geometry of the linear upper bound 43
3.9 Configurations of the upper bound for different values of β 44

3.10 Comparison between the log-upper bound and the log-lower bound estimates for 16 and 256 nodes 45
3.11 Difference in the log-lower, log-upper bounds with the log-exact value for 16 nodes . . 45

4.1 Combining connections in series 47
4.2 Decimation of one node 48
4.3 Decimation of two nodes 48
4.4 Decimation of three nodes 50
4.5 Resulting configuration of the image after the recursive procedure 53

5.1 The bounded log-$P(D|\kappa,\beta)$ with respect to κ and β 56
5.2 The log-$Z_p(\beta)$ obtained by thermodynamic integration methods with respect to its lower and upper bounds 57
5.3 The log-$P(D|\kappa,\beta)$ as a function of κ and β, computed using thermodynamic integration methods and mean field approximations, for two different views 60
5.4 The log-$P(D|\kappa,\beta)$ obtained by thermodynamic integration methods and mean field approximations with respect to its lower and upper bounds 60
5.5 The expected energy $\langle V(S)\rangle$ with respect to β 61
5.6 The gamma distribution $\gamma(\beta,\alpha,\lambda)$ for β. The gamma distribution $\gamma(\kappa,\alpha,\lambda)$ for κ . . . 62
5.7 The bounded log-posterior $P(\kappa,\beta|D)$ as a function of κ and β 63
5.8 The log-posterior $P(\kappa,\beta|D)$ as a function of κ and β, computed using thermodynamic integration methods and mean field approximations, for two different views 64
5.9 The log-posterior $P(\kappa,\beta|D)$ as a function of κ. The log-posterior $P(\kappa,\beta|D)$ as a function of β 65
5.10 The original image, the restored image when $P(\kappa,\beta)$ is disregarded, the restored image when $P(\kappa,\beta)$ is taken into account, the restored image obtained in Chapter 2 65

Chapter 1

Introduction

1.1 Background

The field of image restoration is continually evolving and highlights some key problems in modern science. Since Geman and Geman [2] introduced the use of stochastic models and adopted the Bayesian approach, there has been considerable interest in this area due to its evident practical and theoretical importance. The potential application of automatic image restoration algorithms is wide, ranging from medical image enhancement to cleaning up of noisy video images.

The objective of image restoration consists of finding an estimate of an original image from an imperfect version of that image. The image is regarded as an array of sites or pixels, each having a particular value representing the intensity or the colour. In this thesis, the images will be binary, i.e. black and white. The principles of image restoration require two different types of models. The first one involves the process by which an assumed true, "clean" image is degraded into the corrupted, "noisy" observed image. This noise model is defined by some assumed known statistical conditional distribution. The second model expresses prior knowledge about the characteristics of the underlying true image. It supposes that a pixel value depends only on its neighbours such that pixels close together tend to have similar colour. The formalism of this model stems from the analogy with the Ising model in statistical mechanics [2]. The Ising model is a simple model of a magnetic system which consists of a lattice of locally interacting binary spin variables. Many of the techniques we use were pioneered in the attempt to understand the macroscopic properties of such interacting spin systems.

There are several methods of restoring corrupted images and nearly all of them can be traced back to the seminal papers by Geman and Geman [2] and Besag [3]. Most of them resort to the Bayesian framework and Monte Carlo methods (mainly Gibbs and Metropolis) to explore stochastic models. The first method, proposed by Geman and Geman [2], outlined the main principles of image restoration and served as a basis for much work related to this field. This method aims at computing the maximum *a posteriori* (MAP) estimate of the underlying image given the degraded observations. In other words, it consists of finding an image which maximises the posterior distribution for reconstructed

images using a stochastic relaxation algorithm referred to as simulated annealing. Besag [3] proposed another method called Iterated Conditional Modes (ICM) to find the MAP estimate. This method has proved to be much faster and less computation-intensive than simulated annealing. Dubes and Jain [6] compared these two methods and stated that they both yield the same quality of reconstructed images. Another attempt [5] to evaluate the MAP estimate is to resort to applying the Ford-Fulkerson algorithm which uses graph theory to maximise the logarithm of the posterior distribution for restored images. However, the surge of activity in Markov Chain Monte Carlo techniques allows realisations from the posterior distribution to be generated and the restored image to be obtained over the *average* of these realisations. In a Bayesian sense, the posterior average over restored images is the ideal estimate of the true image. The development and the use of powerful sampling algorithms such as the Swendsen-Wang algorithm, which, contrary to the single site algorithms like Gibbs or Metropolis, update large like coloured patches in the image at the same time, have greatly facilitated a Bayesian treatment of image restoration [12].

Irrespective of the sampling strategy chosen to tackle the image restoration, the quality of any reconstructed image depends heavily on prior knowledge, both in regard to form and parametrisation. The problem of prior parameter estimation is unquestionably the core and challenging problem in image treatment and has been studied extensively over the last few years. The difficulty in dealing with uncertainty in the restoration parameters is due to the intractable partition function of the posterior distribution for restored images. The parameters have then often been chosen on merely an ad hoc basis. Some early work concentrated on using maximum likelihood estimation in the context of Besag's "coding scheme" which turns out to be unreliable due to the complexity of the distribution of corrupted data [2]. Most of the recent image restoration work addresses this issue by assuming that a set of uncorrupted images is available which can be analysed in an attempt to find the parameters correctly, but little is known of the convergence properties of this method [3]. The iterative expectation-maximisation (EM) algorithm is now being applied [19] and involves simultaneous analysis of both the underlying image and the prior parameters. However, there is no guarantee that such a re-estimation method converges to a global maximum of the parameters and the reconstruction simultaneously. Moreover, it is unlikely to find the global maximum and, in general, the results depend on the initial choice of parameters [8]. Pryce and Bruce [8] explored another approach expressed in the "evidence" framework developed by Gull in statistical physics problems and later by MacKay in the context of neural network model selection. The computation of the most likely restoration parameters, by maximising the evidence, is an intractable problem. Typically the evidence calculation involves approximations using Monte Carlo methods, although this is not satisfactory as such approximations are essentially uncontrolled. However, the effectiveness of this method is limited by the quality of the forms of the priors [8], but this approach holds out interesting possibilities in calculating restoration parameters (if the prior beliefs about the noise and image generation process are not well matched to reality, estimates of the restoration parameters can be suboptimal).

In this thesis, following the work carried out by Pryce and Bruce [8], we propose an alternative

approach for evaluating the evidence by borrowing variational methods that have recently been developed in the graphical models literature. Our aim is to calculate rigorous approximations of the evidence based on consistent lower and upper bounds which are computed using the recursive-node elimination technique devised by Jaakkola and Jordan [16] and decimation techniques as advocated by Saul and Jordan [10], Rüger, Weinberger and Wittchen [17], Nijman and Kappen [18]. The effectiveness of this approach relies however on the way the bounds are calculated, the processing time necessary to obtain them and, most importantly, the ability of combining these two methods.

This project aims at restoring degraded binary images and is organised in five chapters. In Chapter 1, we outline the principles of image restoration and develop an evidence framework that enables us to estimate the restoration parameters. In Chapter 2, the main purpose is to provide a general insight of the quality of restored images with respect to the choice of parameters and to develop some aspects that can be used as a basis for further study and research in Chapter 5. To achieve this objective, we use Monte Carlo methods, especially the Gibbs sampling, Metropolis and Swendsen-Wang algorithms. We will show the differences between them, their strengths and weaknesses concerning their computational power: we argue that the Swendsen-Wang algorithm is much faster than single pixel update schemes but can be disadvantageous in restoring the fine details of the images. In Chapter 3, we focus on the recursive node-elimination scheme for rigorously bounding the evidence from both below and above. Chapter 4 outlines the method of removing the pixels in the image so that the computation of the evidence is in time polynomial in the number of remaining pixels and no longer in exponential time. In Chapter 5, we consider the question of how best to choose the parameters in order to obtain an optimal restored image. We will show that the evidence approach does not provide reliable restoration parameters when the prior assumptions about the noise and image generation process are not well matched to reality.

We have developed a new bounding approach for approximating normalising constants which are otherwise computationally intractable in large densely networks. A naive method to compute them is to sum over all the possible configuration states, but then the time complexity is exponential in a number of states. Work in the past has resorted to uncontrollable Monte Carlo techniques to approximate them. In this thesis, we have applied variational methods to estimate them in a controlled manner and we hope that this will remain an active research field for the future.

1.2 Bayesian formulation of the image reconstruction

1.2.1 Definition of the problem

In this section we review the techniques of Bayesian image restoration whereby the parameters controlling the quality of the reconstruction process are assumed known. The more general case of dealing with uncertainty in the restoration parameters is the main emphasis of this thesis and is outlined in Section 1.3.

We consider the restoration of a degraded binary (0/1) image D described by a set of n pixels

$\{D_1, \ldots, D_n\}$ where $n = m \times m$ and m is the size of the square image. The objective is to obtain a restored image S defined by a set of n pixels $\{S_1, \ldots, S_n\}$ which must look similar to the true image T, with $T = \{T_1, \ldots, T_n\}$. Figure 1.1 shows the 32×32 true, noisy and restored images.

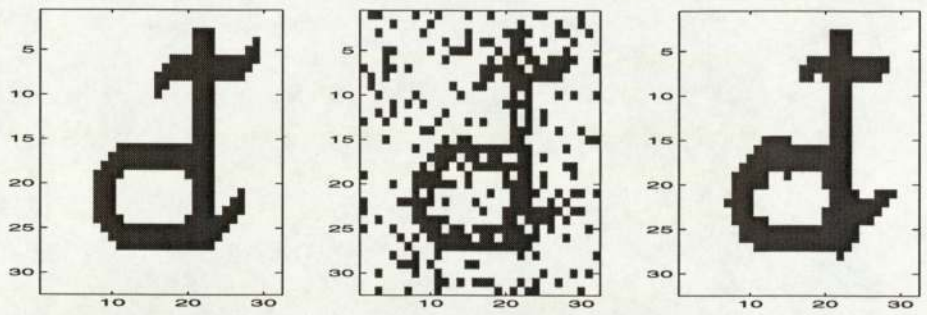


Figure 1.1: True image T (left), noisy image D, restored image S

By using Bayes' theorem, the probability of getting the restored image given the corrupted one is identified as [2]

$$P(S|D) = \frac{P(D|S)P(S)}{P(D)} \tag{1.1}$$

where $P(D|S)$ is the noise likelihood distribution which stands for the corruption (image degradation) process and $P(S)$ is the prior distribution of uncorrupted images. In order that (1.1) represents a distribution, the denominator of the right-hand side is defined as follows

$$P(D) = \sum_S P(D|S)P(S) \tag{1.2}$$

We will see in Section 1.3 that this normalising constant will play an important role as it depends on some implicit parameters which determine the quality of reconstructed images.

1.2.2 Restoring binary images

Let us now define the forms of the prior $P(S)$ and the corruption phenomenon $P(D|S)$ that together specify the process by which the degraded image will be restored.

Consider first the noise likelihood function $P(D|S)$. We start by considering that the noisy image is generated by flipping each of its pixels with respect to the one in the true image with probability q, which we may think of as expressing some "noise level" [8]. The probability of observing a noisy pixel, D_i, given an uncorrupted pixel, S_i, is then

$$P(D_i|S_i) = (1-q)I[D_i = S_i] + qI[D_i \neq S_i]$$

where

$$I[\psi] = \begin{cases} 1 & \text{if } \psi \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

For convenience, we define κ such that

$$q = \frac{e^{-\frac{\kappa}{2}}}{1 + e^{-\frac{\kappa}{2}}} \tag{1.3}$$

so that, by using $D_i, S_i \in \{0, 1\}$, we obtain

$$P(D_i|S_i) = \frac{\exp\left\{-\frac{\kappa}{2}(D_i - S_i)^2\right\}}{1 + e^{-\frac{\kappa}{2}}}$$

We assume that the pixels are drawn independently from this distribution, so that the noise likelihood function follows as [3]

$$P(D|S) = \frac{1}{Z_l(\kappa)} \exp\left\{-\frac{\kappa}{2}\sum_i (D_i - S_i)^2\right\} \tag{1.4}$$

where $Z_l(\kappa)$ is determined by the normalising condition

$$Z_l(\kappa) = \left(1 + e^{-\frac{\kappa}{2}}\right)^n = \sum_D \exp\left\{-\frac{\kappa}{2}\sum_i (D_i - S_i)^2\right\} \tag{1.5}$$

Now consider the prior distribution of uncorrupted images $P(S)$. The concept of a Markov random field, at least in regard to image analysis, is that the conditional distribution of an individual pixel depends only on a neighboured set, much smaller than the image itself. For simplicity, we take the neighbourhood set to contain only the nearest four neighbours of any given pixel (in the interior of the image). In other words, if $\mathcal{N}$ represents the set of vertical and horizontal neighbours of S_i, as shown in Figure 1.2, then

$$P(S_i|\{S_j, j \neq i, j = 1, \ldots, n\}) = P(S_i|\{S_j, S_j \in \mathcal{N}\})$$

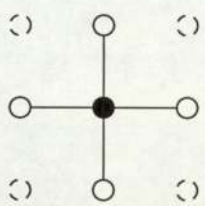


Figure 1.2: Illustration of the neighbourhood configuration. The dotted nodes are not considered as the neighbours of the black one which is connected to its vertical and horizontal neighbours

Referring to the Ising model, Geman and Geman [2] and Besag [3] showed that the above conditional probability establishes a model that takes into account the pairwise interactions between pixels. Basically, it enables us to have some knowledge according to which some pairs of pixels, at neighbouring sites i and j, tend to have the same colour. The pairwise interaction model can then be written in the form [3]

$$P(S) = \frac{1}{Z_p(\beta)} \exp\left\{\beta \sum_{i \backsim j} I[S_i = S_j]\right\} \tag{1.6}$$

where $\sum_{i \backsim j}$ denotes a sum over all pairs of vertical and horizontal neighbouring sites on the pixel lattice. The prior $P(S)$ is called the Markov random field prior [2, 3].

The normalisation factor for this distribution is

$$Z_p(\beta) = \sum_S \exp\left\{\beta \sum_{i \backsim j} I[S_i = S_j]\right\} \tag{1.7}$$

Equations (1.1), (1.4) and (1.6) together define the posterior distribution for restored images

$$P(S|D) = \frac{1}{Z_r(\kappa,\beta)} \exp\left\{-\frac{\kappa}{2}\sum_i (D_i - S_i)^2 + \beta \sum_{i\backsim j} I[S_i = S_j]\right\} \tag{1.8}$$

where the partition function Z_r is given by

$$Z_r(\kappa,\beta) = \sum_S \exp\left\{-\frac{\kappa}{2}\sum_i (D_i - S_i)^2 + \beta \sum_{i\backsim j} I[S_i = S_j]\right\} \tag{1.9}$$

Equation (1.8) contains two terms which model the tendency of images to contain large black or large white areas. The term controlled by κ ($\kappa \geq 0$) represents the binding of the restored configuration S to the noisy one D. The term depending on β ($\beta \geq 0$) controls the degree of correlation between neighbouring pixels and hence the smoothness of the reconstruction. Essentially β quantifies the belief that the restored image could encompass large homogeneous clusters. The quality of the reconstructed image relies on the competition between these two restoration parameters. The primary aim of this thesis is to address the problem of finding the right values for κ and β so that the *optimal* reconstructed image is obtained. As an integral part of the solution to this problem, and also to actually perform image restoration, we will need to compute with the posterior $P(S|D)$ in (1.8) for fixed values of κ and β. This in itself is a highly non-trivial problem to which we devote the following chapter. The task of finding the optimal κ and β will be examined in Chapter 5.

1.3 The Bayesian framework for restoration parameters

Throughout this project, our objective is to estimate the parameters implicit in the noise likelihood function $P(D|S,\kappa)$ in (1.4) and in the Markov random field prior $P(S|\beta)$ in (1.6). The key technique to computing these parameters takes the form of maximising the likelihood $P(D|\kappa,\beta)$ in (1.2) and determining the effective parameter prior $P(\kappa,\beta)$.

Within the Bayesian context, the correct posterior distribution $P(S|D)$ is obtained by averaging the conditional posterior distribution $P(S|D,\kappa,\beta)$ over the parameter posterior $P(\kappa,\beta|D)$

$$\begin{aligned} P(S|D) &= \int\int P(S,\kappa,\beta|D)d\kappa d\beta \\ &= \int\int P(S|\kappa,\beta,D)P(\kappa,\beta|D)d\kappa d\beta \end{aligned} \tag{1.10}$$

Note that, in the previous section, we have simply assumed that $P(\kappa,\beta|D) = \delta(\kappa,\kappa^0)\delta(\beta,\beta^0)$ for chosen values of κ^0 and β^0, where δ is the delta function. The central aim is to deal with $P(\kappa,\beta|D)$.

Let us now suppose that the posterior probability distribution $P(\kappa,\beta|D)$ in (1.10) is sharply peaked around their most probable values κ_{MP} and β_{MP}. Then (1.10) can be written [9]

$$\begin{aligned} P(S|D) &\simeq P(S|\kappa_{MP},\beta_{MP},D)\int\int P(\kappa,\beta|D)d\kappa d\beta \\ &= P(S|\kappa_{MP},\beta_{MP},D) \end{aligned} \tag{1.11}$$

In order to find κ_{MP} and β_{MP}, we evaluate the posterior distribution of κ and β. This is given by

$$P(\kappa, \beta|D) = \frac{P(D|\kappa, \beta)P(\kappa, \beta)}{P(D)} \tag{1.12}$$

which requires a choice for the prior $P(\kappa, \beta)$. We have some prior beliefs about $P(\kappa)$ and $P(\beta)$ for we believe there exists an interval of parameter values that lead to reasonable restored images. We shall discuss more formally how to choose these priors later. Since the denominator in (1.12) is independent of κ and β, we see that the maximum-posterior values (MAP) for these parameters are found by maximising the product of the likelihood term $P(D|\kappa, \beta)$ and the prior $P(\kappa, \beta)$. The term $P(D|\kappa, \beta)$ is called the *evidence*[1] for κ and β.

Note that the Bayesian technique for the image restoration problem proceeds in two levels. The first one consists of determining the probability of restored images and the second one involves the distribution of parameter values. The evidence $P(D|\kappa, \beta)$ at this level is given by the denominator in Bayes' theorem (1.1) from the previous level.

Retaining the form (1.2), and making the dependences on κ and β explicit, we can write

$$P(D|\kappa, \beta) = \sum_S P(D|S, \kappa)P(S|\beta)$$

Appealing back to the expressions (1.4) and (1.6), we find that

$$P(D|\kappa, \beta) = \sum_S \frac{\exp\left\{-\frac{\kappa}{2}\sum_i (D_i - S_i)^2\right\}}{Z_l(\kappa)} \frac{\exp\left\{\beta \sum_{i \backsim j} I[S_i = S_j]\right\}}{Z_p(\beta)}$$

Using the formula (1.9), we can then write

$$P(D|\kappa, \beta) = \frac{Z_r(\kappa, \beta)}{Z_l(\kappa)Z_p(\beta)} \tag{1.13}$$

The evidence relies on the normalising constants Z_r and Z_p whose exact computation is intractable because it requires summing over a set S of m^2 pixels and consequently involves 2^{m^2} calculations, where m is the size of the image. Therefore, the idea is to approximate it by using recent methods [10, 11, 16, 17, 18] we shall see in Chapters 3 and 4. Note, however, that Z_l can be computed exactly in (1.5).

In the next chapter, we use Monte Carlo techniques such as the Gibbs sampling, Metropolis and Swendsen-Wang algorithms to focus on the quality of restored images with respect to the restoration parameters. Chapters 3 and 4 outline the methods of bounding the evidence from both above and below. In Chapter 5, we consider the question of how best to choose the parameters in order to obtain an optimal restored image.

[1] The notion of evidence was developed by Gull for estimating free energies in the statistical mechanics context. In the image restoration problem, the task of calculating the evidence for different parameter choices is analogous to that of estimating free energies [8].

Chapter 2

Markov chain Monte Carlo techniques

In this chapter we focus on the quality of restored images with respect to the choice of two restoration parameters κ and β. In order to perform image restoration, we resort to Monte Carlo methods, in particular the Gibbs and Metropolis algorithms and also the much less well known Swendsen-Wang algorithm. We will compare their empirical performance in yielding restored images as well as their computational power.

2.1 Why Monte Carlo methods

For fixed values of κ and β, the optimal reconstructed image in a Bayesian framework is obtained by averaging the restored images over the posterior distribution defined in (1.8)

$$\langle S \rangle = \sum_S P(S|D)S \tag{2.1}$$

However, evaluating this average exactly turns out to be an intractable problem because it is NP-complete[1] [1] to calculate the partition function in (1.9). Many exact methods, which can be used successfully for a small number of variables, are unsuitable for our problem which involves $m \times m$ pixel variables, where m is the size of the image we are considering. Therefore, we resort instead to *Markov chain Monte Carlo* techniques for estimating the corresponding average directly.

Historically, Markov chain Monte Carlo (MCMC) methods were first developed for performing calculations in statistical physics. As recognised by Geman and Geman [2], they are also useful for image restoration problems and have been extensively applied in this context. In our problem, in order to obtain the posterior average image, we use MCMC methods to sample from the distribution defined in (1.8) because they provide an easily realisable way of implementing this sampling. Basically, the

[1] There are an exponential number of pixel configurations to sum over.

idea is as follows: consider evaluating a sum in a n-dimensional space

$$Y = \sum_{\mathbf{x}} F(\mathbf{x}) p(\mathbf{x})$$

where $\mathbf{x}$ is a n-dimensional vector, $p(\mathbf{x})$ represents some distribution and $F(\mathbf{x})$ some function. Note that, in the case of continuous variables, we integrate the above expression instead of summing it. Monte Carlo methods then approximate this equation with the finite sum

$$Y \simeq \frac{1}{L} \sum_{i=1}^{L} F(\mathbf{x}_i)$$

where $\{\mathbf{x}_i, i = 1, \ldots, L\}$ represents a sample of vectors generated from the probability $p(\mathbf{x})$. The main difficulty is to generate a sequence of vectors from the required distribution $p(\mathbf{x})$. In order to achieve this, we shall set up a Markov chain[2] whose invariant distribution is the required distribution $p(\mathbf{x})$.

2.2 The Gibbs sampling algorithm

This algorithm is one of the simplest amid the MCMC techniques and has been advocated by Geman and Geman [2] in the context of Bayesian image analysis. It is widely applied to problems where the variables have conditional distributions of a parametric form that can easily be sampled from, which is the case in our problem.

The theoretical aspect is as follows [7]: suppose we wish to sample from the joint distribution for $X = \{X_1, \ldots, X_n\}$ given by $P(x_1, \ldots, x_n)$, where X_i may be either continuous or discrete. The Gibbs sampler does this by repeatedly replacing each component X_i with a value picked from its distribution $P(x_i | \{x_j : j \neq i\})$ conditional on the current values of all other components. The procedure for generating state $X^{(t+1)}$ at time $t+1$ from $X^{(t)}$ at time t can be expressed as follows

Pick $x_1^{(t+1)}$ from the conditional distribution for X_1 given $x_2^{(t)}, x_3^{(t)}, \ldots, x_n^{(t)}$.
Pick $x_2^{(t+1)}$ from the conditional distribution for X_2 given $x_1^{(t+1)}, x_3^{(t)}, \ldots, x_n^{(t)}$.
$\vdots$
Pick $x_i^{(t+1)}$ from the conditional distribution for X_i given $x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}$.
Pick $x_n^{(t+1)}$ from the conditional distribution for X_n given $x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{n-1}^{(t+1)}$.

Note that the new value for X_{i-1} is used immediately when picking the next value for X_i.

Before using this algorithm to tackle the image restoration problem, for convenience we translate the binary (0/1) image into a (−1/1) image since the form of the conditional distribution with the (−1/1) model will be simpler to use than the one with the (0/1) model.

[2] In the theory of Markov chains, for a series of random variables $X^{(0)}, \ldots, X^{(n)}$, the conditional distribution of $X^{(n+1)}$ depends only on $X^{(n)}$. More formally,

$$P\left(x^{(n+1)} | \left\{x^{(t)}, t = 1, \ldots, n\right\}\right) = P\left(x^{(n+1)} | x^{(n)}\right)$$

Retaining the formula (1.8), by expanding it and by noticing that for $S_i = \{0,1\}$, $S_j = \{0,1\}$, $I[S_i = S_j] = 1 + 2S_iS_j - S_i - S_j$, we get

$$P(S|D) \propto \exp\left\{-\frac{\kappa}{2}\sum_i (D_i^2 + S_i^2 - 2D_iS_i) + \beta\sum_{i\backsim j}(1 + 2S_iS_j - S_i - S_j)\right\}$$

As $D_i, S_i \in \{0,1\}$, then $D_i^2 = D_i$, and likewise $S_i^2 = S_i$, therefore

$$\begin{aligned}
P(S|D) &\propto \exp\left\{-\frac{\kappa}{2}\frac{1}{2}\sum_i (2D_i + 2S_i - 4D_iS_i) + \frac{\beta}{2}\sum_{i\backsim j}(2 + 4S_iS_j - 2S_i - 2S_j)\right\} \\
P(S|D) &\propto \underbrace{\exp\left\{-\frac{\kappa}{4}\sum_i 1 + \frac{\beta}{2}\sum_{i\backsim j} 1\right\}}_{\text{constant}} \\
&\quad \times \exp\left\{-\frac{\kappa}{2}\frac{1}{2}\sum_i (-1 + 2D_i + 2S_i - 4D_iS_i) + \frac{\beta}{2}\sum_{i\backsim j}(1 + 4S_iS_j - 2S_i - 2S_j)\right\} \\
P(S|D) &\propto \exp\left\{\frac{\kappa}{4}\sum_i (2D_i - 1)(2S_i - 1) + \frac{\beta}{2}\sum_{i\backsim j}(2S_i - 1)(2S_j - 1)\right\}
\end{aligned}$$

Finally, let $S_i \leftarrow 2S_i - 1$ and $D_i \leftarrow 2D_i - 1$, $\forall i = 1 \ldots n$. Then from now, S_i and D_i take on values -1 and $+1$ and we obtain a new form for the distribution $P(S|D)$ which we simply denote by $Q(S)$. Thus

$$Q(S) \propto \exp\left\{\frac{\kappa}{4}\sum_i D_iS_i + \frac{\beta}{2}\sum_{i\backsim j} S_iS_j\right\} \tag{2.2}$$

Now, let us return to our restoration problem. Starting with the noisy image, we first select the values for the pixels S_i. An obvious choice [7] is to independently set each node to -1 or $+1$ with equal probability. We then proceed through the various pixels repeatedly in some predefined order. For example, we can start with the first pixel in the top left-hand corner of the image and end up with the one in the bottom right-hand corner. When a node S_i is visited, a new value for it is chosen from the conditional distribution defined by other nodes. This conditional distribution is derived from the canonical distribution in (2.2) as follows

$$\begin{aligned}
Q(S_i|\{S_j : j \neq i\}) &= \frac{Q(S_1, \ldots, S_i, \ldots, S_n)}{\sum_{S_i=\{-1,1\}} Q(S_1, \ldots, S_i, \ldots, S_n)} \\
&= \frac{\exp\left\{\frac{\kappa}{4}D_iS_i + \frac{\beta}{2}S_i\sum_{i\backsim j} S_j\right\}}{\exp\left\{\frac{\kappa}{4}D_i + \frac{\beta}{2}\sum_{i\backsim j} S_j\right\} + \exp\left\{-\frac{\kappa}{4}D_i - \frac{\beta}{2}\sum_{i\backsim j} S_j\right\}}
\end{aligned}$$

If we choose $+1$ for S_i and set the other S_j to their current values, then the conditional distribution is

$$Q(S_i = 1|\{S_j : j \neq i\}) = \sigma\left(\frac{\kappa}{2}D_i + \beta\sum_{i\backsim j} S_j\right) \tag{2.3}$$

where $\sigma(z) = 1/(1+\exp(-z))$. Since each pixel has only at most four neighbours, the above probability can easily be computed in time that is independent of the total number of nodes. The choice for S_i at

the next state can then be made by drawing a random number uniformly from the interval $[0, 1]$ and setting S_i to $+1$ if this number is less than the calculated probability in (2.3), and to -1 otherwise [7].

2.3 The Metropolis algorithm

The Metropolis algorithm was first proposed by Metropolis, Rosenbluth and Teller in 1953 and has been widely used in statistical physics problems. It shares many of the characteristics of the Gibbs sampling method, but it is more generally applicable to a wide range of problems, because it avoids any need to sample from difficult distributions. Indeed, the Gibbs algorithm can be shown to be equivalent to a suitably chosen Metropolis procedure.

The principle can be described as follows [7]: suppose we wish to sample from the joint distribution for $X = \{X_1, \ldots, X_n\}$ given by $P(x_1, \ldots, x_n)$, where X_i may be either continuous or discrete. The Metropolis algorithm does this by repeatedly considering randomly generated changes to the components of X, accepting or rejecting these changes based on how they affect the probability of the state. The procedure for generating a new state X' from the old state X is

- Select a *candidate state*, X^*, in which all the components other than the i'th element are the same as the ones in X, while x_i^* is picked randomly from a *proposal distribution* which may depend on X, given the probabilities $R_i(X, x_i^*)$.
- *Accept* this candidate state with probability $A(X, X^*)$ otherwise, *reject* it, and retain the current state which will be considered as a new state. In detail, this can be done by generating a random number, r, from the uniform distribution on $[0, 1]$, and then setting the next state as follows

$$X' = \begin{cases} X^* & \text{if } r < A(X, X^*) \\ X & \text{otherwise} \end{cases}$$

The probabilities for the proposal distribution, $R_i(X, x_i^*)$, must, of course, be non-negative and must satisfy $\sum_{x_k} R_k(X, x_k) = 1$. The acceptance probability, $A(X, X^*)$, can have various forms, the most common of which is defined as the *Metropolis* acceptance

$$A(X, X') = \min\left(1, Q(X')/Q(X)\right) \tag{2.4}$$

where $Q(X)$ is the canonical distribution.

Now, let us apply this algorithm to the problem of image restoration. It seems natural to treat each node as a separate component. Usually, the proposal distribution [7] for a change to node i is $R_i\left(S, S_i'\right) = \delta\left(-S_i, S_i'\right)$ where S represents the set of pixels $\{S_1, \ldots, S_n\}$ or the image at time t and S' is the image at time $t + 1$. This means that the candidate state S' is always obtained by flipping node i to its opposite value while the other nodes of S' (except node i) remain the same as the ones in the old state, i.e. $S_i' = -S_i$ and $\forall j \neq i, S_j' = S_j$. The probability of accepting S' is found from

equations (2.4) and (2.2). Let us first compute $Q(S')/Q(S)$.

$$\begin{aligned} Q(S')/Q(S) &= \exp\left\{\frac{\kappa}{4}\sum_i D_i S_i' + \frac{\beta}{2}\sum_{i\backsim j} S_i' S_j - \left(\frac{\kappa}{4}\sum_i D_i S_i + \frac{\beta}{2}\sum_{i\backsim j} S_i S_j\right)\right\} \\ &= \exp\left\{2S_i'\left(\frac{\kappa}{4}D_i + \frac{\beta}{2}\sum_{i\backsim j} S_j\right)\right\} \end{aligned}$$

Thus the acceptance probability is

$$A(S, S') = \min\left(1, \exp\left\{S_i'\left(\frac{\kappa}{2}D_i + \beta\sum_{i\backsim j} S_j\right)\right\}\right) \tag{2.5}$$

Starting with the noisy image, we update all the pixels from the top left-hand corner of the image to the bottom right-hand corner. Since each pixel has only at most four neighbours, the above probability can easily be computed in time that is independent of the total number of nodes.

2.4 Simulated Annealing

The core problem for Markov chain sampling methods in general is to generate a sample of vectors representative of their distribution. Indeed, consider an example of a system of continuous variables with a distribution represented in Figure 2.1: if the sampling takes place in the regions of relatively low probability or small volume, then the convergence movement through the state space is hampered by this small volume. Therefore, the objective is to sample preferentially regions where the distribution is large. As it stands, such a task is hard to achieve. It is nevertheless possible that a more sophisticated algorithm like the *simulated annealing* method increases the chance of reaching regions that contain relevant samples in order to gain confidence in the accuracy of the results. It suffices to introduce an additional variational parameter T generally referred to as *temperature* [7]. For the Gibbs sampling algorithm, this is achieved by modifying the conditional distribution (2.3) to give

$$p\left(S_i = 1 \,|\, \{S_j : j \neq i\}\right) = \sigma\left(\left(\frac{\kappa}{2}D_i + \beta\sum_{i\backsim j} S_j\right)/T\right) \tag{2.6}$$

For the Metropolis algorithm, by modifying the acceptance probability (2.5), we have

$$A(S, S') = \min\left(1, \exp\left\{S_i'\left(\frac{\kappa}{2}D_i + \beta\sum_{i\backsim j} S_j\right)/T\right\}\right) \tag{2.7}$$

When T is bigger than 1, the system can explore the state space much more freely, and can readily escape from local minima. Simulated annealing involves starting with a large value for T ($T = 3$ in our problem) and then gradually reducing its value during the course of the simulation, giving the system a much better chance to settle into a region of high probability.

In the context of image restoration, the application of the simulated annealing method takes a long time to reach low values of the temperature T which implies a strong dependence between neighbouring

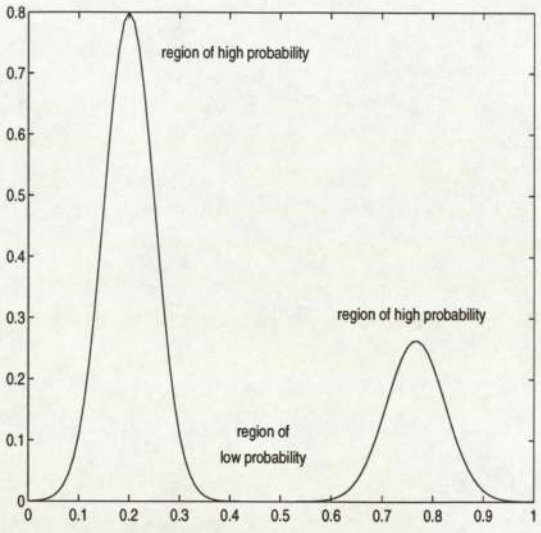


Figure 2.1: Sampling regions of high or low probability

pixels, giving rise to large homogeneous clusters. The main reason is that it does the sampling by systematically updating each pixel locally. Moreover, the major drawback of this method is that it may lead to some bad results because it is incapable of sampling the regions of high probability or, in other words, it cannot escape the local minima representing these results and therefore can spend a long time in the neighbourhood of the latter.

2.5 The Swendsen-Wang algorithm

Swendsen and Wang [4] devised a specialised algorithm that is a spectacular improvement on the previous methods for simulating Ising models near the point of a phase transition, when large clusters of identical pixels are present. So far we have chosen to update each pixel locally by using the Gibbs and Metropolis algorithms and therefore it may take a long time to explore configurations in this situation. The Swendsen-Wang (S-W) algorithm uses *auxiliary bond* variables to provide a simple means of updating large clusters of identical nodes at the same time and it really speeds up the restoration process especially when large like coloured patches are present.

The general idea is as follows: we wish to sample from the distribution of interest $p(x)$ over the state variable x. We introduce an auxiliary variable u and define its conditional distribution $p(u|x)$, obtaining the joint distribution $p(x,u) = p(x)p(u|x)$. Any scheme for sampling from $p(x,u)$ will therefore solve the original problem simply by ignoring the u components of the (x,u) samples. The following algorithm is due to Edwards and Sokal [7, 12].

Assume that $p(x)$ can be written in the form

$$p(x) \propto p_0(x) \prod_{k=1}^{n} b_k(x)$$

where each of the factors $b_k(x)$ is bounded and $p_0(x)$ is a distribution called an *external field*. Edwards

and Sokal express this model by using auxiliary variables $u_1, \ldots, u_n$ and the conditional distribution

$$p(u|x) = \prod_k \frac{1}{b_k(x)} I\left[0 \leq u_k \leq b_k(x)\right] \tag{2.8}$$

where

$$I\left[0 \leq u_k \leq b_k(x)\right] = \begin{cases} 1 & \text{if } 0 \leq u_k \leq b_k(x) \\ 0 & \text{otherwise} \end{cases}$$

Given x, the components of u are independent, with each u_k having a uniform distribution over the interval $[0, b_k(x)]$. Using Bayes' theorem, the distribution of x given u is then

$$\begin{aligned} p(x|u) &\propto p(u|x)p(x) \\ &\propto p_0(x) \prod_k b_k(x) \frac{1}{b_k(x)} I\left[0 \leq u_k \leq b_k(x)\right] \\ &\propto p_0(x) \prod_k I\left[0 \leq u_k \leq b_k(x)\right] \end{aligned} \tag{2.9}$$

In order to sample from the joint distribution $p(x, u)$, we can employ Gibbs sampling, alternately choosing new values for the auxiliary variables u_k given the current x using (2.8), and updating x given the current u_k using (2.9), so that the realizations $x^1, x^2, \ldots$ are distributed according to $p(x)$.

Now consider again tackling the image analysis problem. As usual, let S be an image defined by a set of nodes $\{S_1, \ldots, S_n\}$ where we revert to the $\{0, 1\}$ representation. Appealing back to equation (1.8), we denote $P(S|D)$ by $p(S)$ so that

$$\begin{aligned} p(S) &\propto \exp\left\{-\frac{\kappa}{2} \sum_{i \in S} (D_i - S_i)^2\right\} \times \exp\left\{\beta \sum_{i \backsim j} I\left[S_i = S_j\right]\right\} \\ &\propto \exp\left\{-\frac{\kappa}{2} \sum_{i \in S} (D_i - S_i)^2\right\} \times \prod_{i \backsim j} \exp\left\{\beta I\left[S_i = S_j\right]\right\} \end{aligned}$$

The external field p_0 is identified as

$$p_0(S) = \exp\left\{-\frac{\kappa}{2} \sum_{i \in S} (D_i - S_i)^2\right\}$$

From (2.8), each bond u_{ij} between nodes i and j has the probability

$$p(u_{ij}|S) = \exp\left\{-\beta I\left[S_i = S_j\right]\right\} I\left[0 \leq u_{ij} \leq \exp\left\{\beta I\left[S_i = S_j\right]\right\}\right] \tag{2.10}$$

and using (2.9), we obtain

$$p(S|u) \propto \exp\left\{-\frac{\kappa}{2} \sum_{i \in S} (D_i - S_i)^2\right\} \times \prod_{i \backsim j} I\left[0 \leq u_{ij} \leq \exp\left\{\beta I\left[S_i = S_j\right]\right\}\right] \tag{2.11}$$

From (2.10), given the current image S, we generate bond variables u_{ij} uniformly over the interval $\left[0, e^{\beta I[S_i = S_j]}\right]$. For two adjacent matching pixels, S_i and S_j, we choose to place the bond u_{ij} with the probability $p\left(u_{ij} > 1 | S_i = S_j\right)$. From (2.10), we find that

$$\begin{aligned} p\left(u_{ij} > 1 | S_i = S_j\right) &= 1 - p\left(u_{ij} \leq 1 | S_i = S_j\right) \\ &= 1 - \exp(-\beta) \end{aligned}$$

In other words, Gibbs sampling for the u_{ij} consists of bonding like coloured neighbours, S_i and S_j, with probability $1 - e^{-\beta}$ and omitting bonds between neighbours that differ in value. Thus the bond variable u partitions S into like coloured clusters and strengthens the dependence between identical neighbouring pixels, while completely removing it from other like coloured neighbours. Using (2.11), we update each cluster C by assigning the new colour $k \in \{0, 1\}$ to the pixels of C with probability [12]

$$p(k \in \{0,1\}) = \frac{\exp\left\{-\frac{\kappa}{2}\sum_{i\in C}(D_i - k)^2\right\}}{\exp\left\{-\frac{\kappa}{2}\sum_{i\in C} D_i^2\right\} + \exp\left\{-\frac{\kappa}{2}\sum_{i\in C}(D_i - 1)^2\right\}} \tag{2.12}$$

Updating u essentially grows clusters, and updating S colours the clusters. Figure 2.2 [12] illustrates the S-W algorithm on an 8×8 lattice.

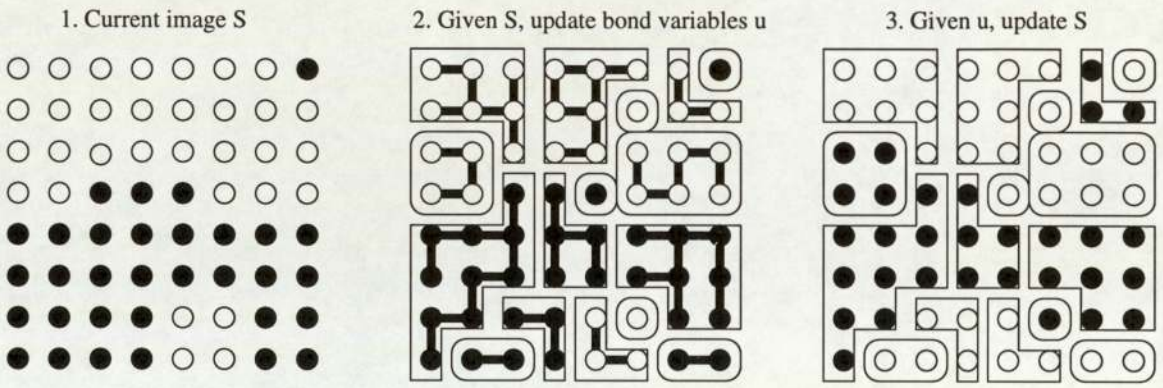


Figure 2.2: Illustration of the Swendsen-Wang algorithm

1. Initial image S and Markov random field graph for $p(S)$.

2. The bond variables u are generated uniformly over the interval $\left[0, e^{\beta I[S_i = S_j]}\right]$, given the current image S. If $u_{ij} > 1$ then S_i and S_j are bonded (marked by the thick lines). Note that this means that some adjacent identically coloured pixels will not be bonded. In fact, the probability of a bond between like coloured pixels is $1 - e^{-\beta}$. These constraints partition the image into clusters of like coloured pixels.

3. Given the bond variables u, S is now an image of independent clusters. Each cluster is recoloured black or white with the probability defined in (2.12). Equation (2.12), which expresses the probability that a cluster is coloured black ($k = 0$) for instance, makes intuitive sense: if we examine the corresponding cluster in the original noisy image, then the probability that we would colour this cluster black is high, if the majority of pixels in this cluster are black.

2.6 Experimental results

2.6.1 Comparison of restored images

Having described the Gibbs, Metropolis and Swendsen-Wang algorithms, we test their empirical performance in yielding restored images as well as their computational power. Figure 2.3 shows the original and the degraded images generated with a "true" noise level $\tilde{q} = 0.2$ which may differ from the assumed noise level q defined in (1.3). Figures 2.4 and 2.5 indicate the reconstructed images using the Swendsen-Wang algorithm and simulated annealing for the Gibbs and Metropolis algorithms for a same amount of the CPU time[3] t, $t = 50$ and $t = 130$ seconds respectively.

The restored image is obtained as follows: we generate N image samples for an equivalent amount of CPU time t and we average over the last half of these images. The reconstructed image is then the binary thresholded version of this mean image.

Figure 2.4 shows that, for $t = 50$ seconds, the images restored by the S-W algorithm look similar to the original ones. Moreover, S-W yields cleaner restored images than the Gibbs and Metropolis algorithms. For Gibbs and Metropolis, we note some remaining small black patches near the edges of the restored images which look almost regular. These small black islands do not totally vanish for $t = 130$ seconds, although the images restored by the three algorithms look most similar to the original ones, as shown in Figure 2.5. We find that the S-W algorithm is a much faster sampling method than the Gibbs and Metropolis algorithms. It gives almost the same relevant restored images for $t = 50$ seconds as it does for $t = 130$ seconds. Indeed, it proves to be at least an order of magnitude more efficient at moving quickly through the state space by means of auxiliary bond variables that update large clusters of identical pixels at the same time near the point of the phase transition. It is exactly in this phase transition where S-W is most effective and single site update algorithms are least efficient. Indeed, Gibbs and Metropolis update each pixel locally and consequently it takes a long time to explore configurations in this situation. That is the reason why Gibbs and Metropolis yield more regular and smoother images for $t = 130$ seconds than they do for $t = 50$ seconds. However, for $t = 130$ seconds, we note that the Metropolis algorithm does not restore the small details of the image while Gibbs sampling does. Maybe a change of the acceptance probability could restore these small details. For example, we can use the Boltzmann acceptance function[4] which makes the Metropolis algorithm identical to Gibbs sampling, assuming that candidate states are selected by flipping only the value of one component. However, we note that the tendency of the S-W algorithm to bind together and flip large like coloured clusters at the same time can be disadvantageous in restoring the fine details of the image. Perhaps a hybrid approach in which a sequence of S-W updates followed by a sequence of Gibbs updates would yield a good quality of restored images.

[3]The CPU time means the time in seconds that has been used by the MATLAB process to run some operation since MATLAB started. The machine used is Silicon Graphics Challenge L (64 bit).

[4]For two states X and X', the Boltzmann acceptance function, $A(X, X')$, is $Q(X')/(Q(X) + Q(X'))$, where Q is the canonical distribution. It has been shown that the Boltzmann and the Metropolis acceptance functions are neither inferior nor superior to each other in all contexts [7].

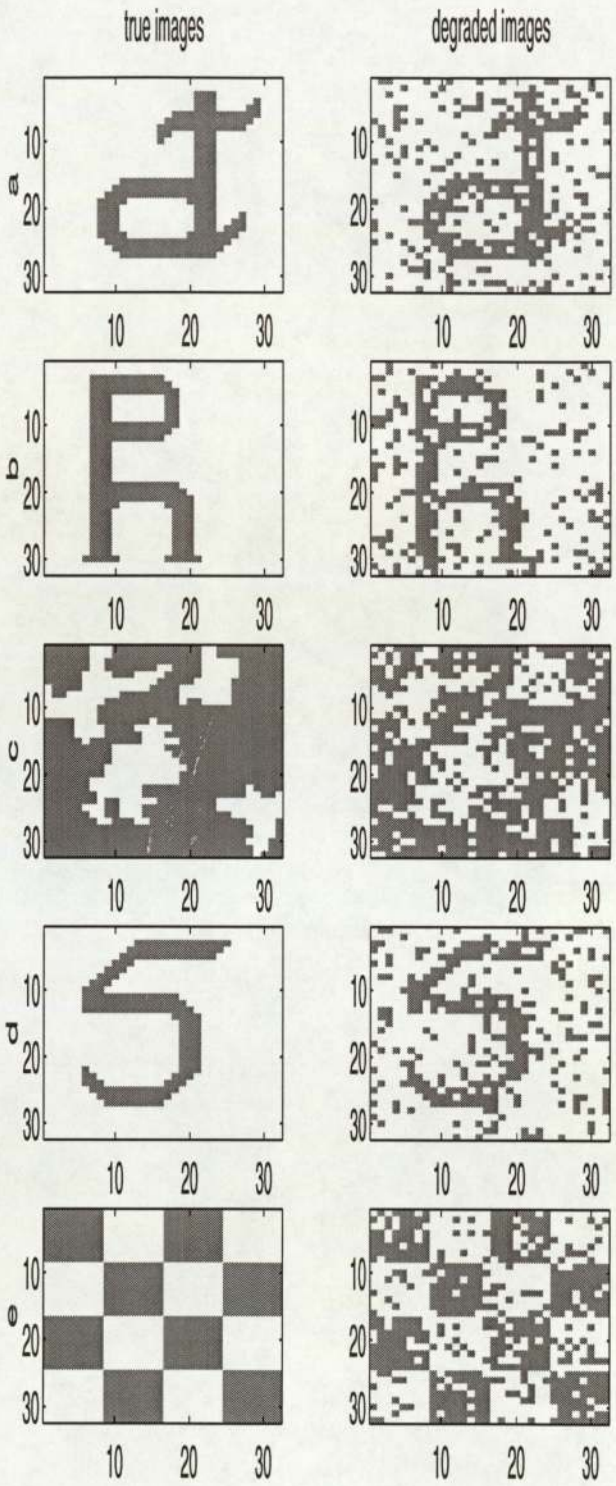


Figure 2.3: True images (left) and corrupted images generated with a "true" noise level $\tilde{q} = 0.2$ which may differ from the assumed noise level q

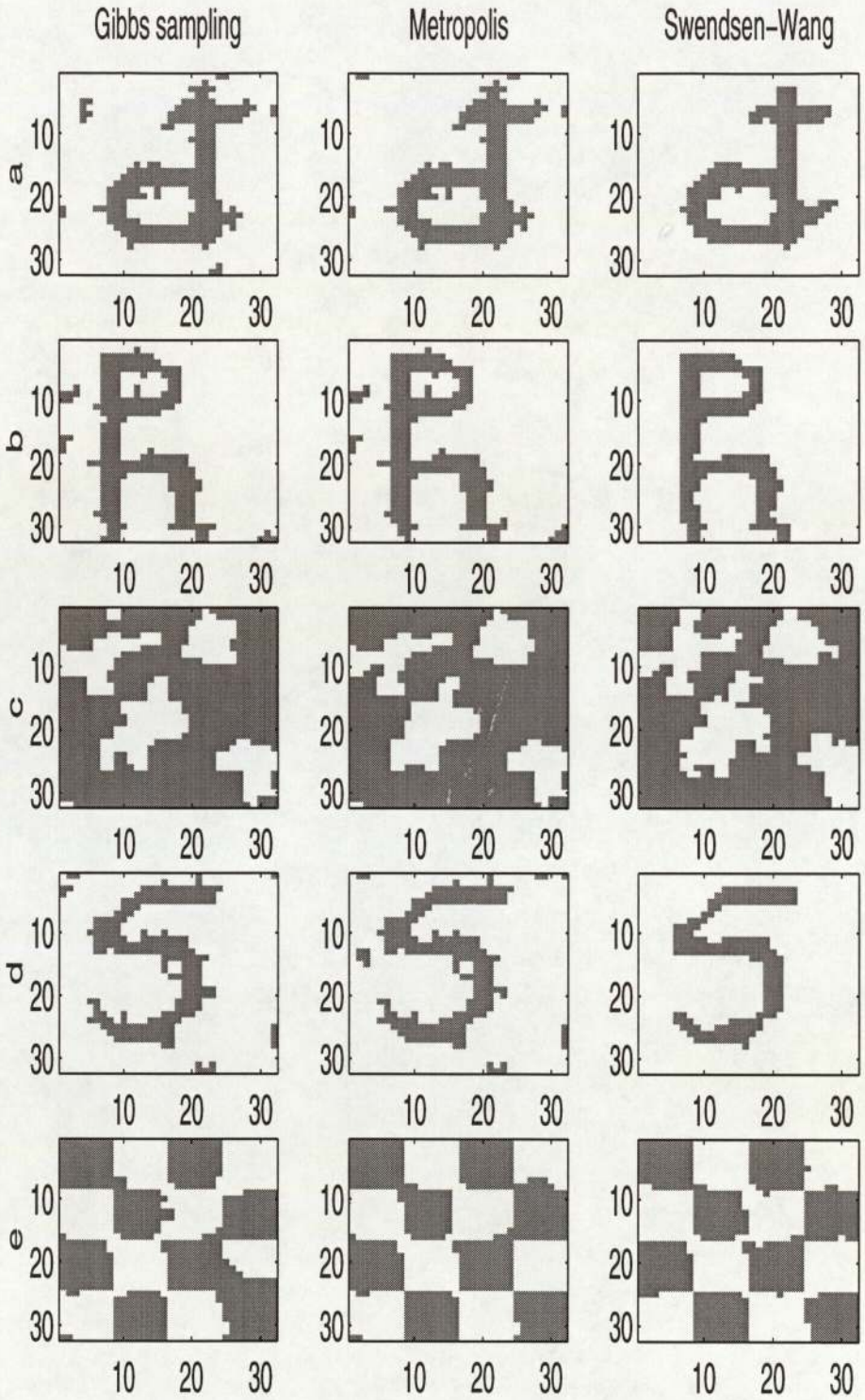


Figure 2.4: Images restored by simulated annealing (Gibbs (left), Metropolis (centre)) and the Swendsen-Wang algorithm (right) for a same amount of CPU time $t = 50$ seconds. The restoration parameters β and κ (corresponding to the noise level q) are: in row (a): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.58$; in row (b): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.59$; in row (c): $\kappa = 2.0$ ($q = 0.27$),$\beta = 0.59$; in row (d): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.57$; in row(e): $\kappa = 1.9$ ($q = 0.28$),$\beta = 0.61$. The temperature T used for simulated annealing is equal to 3 for all the images

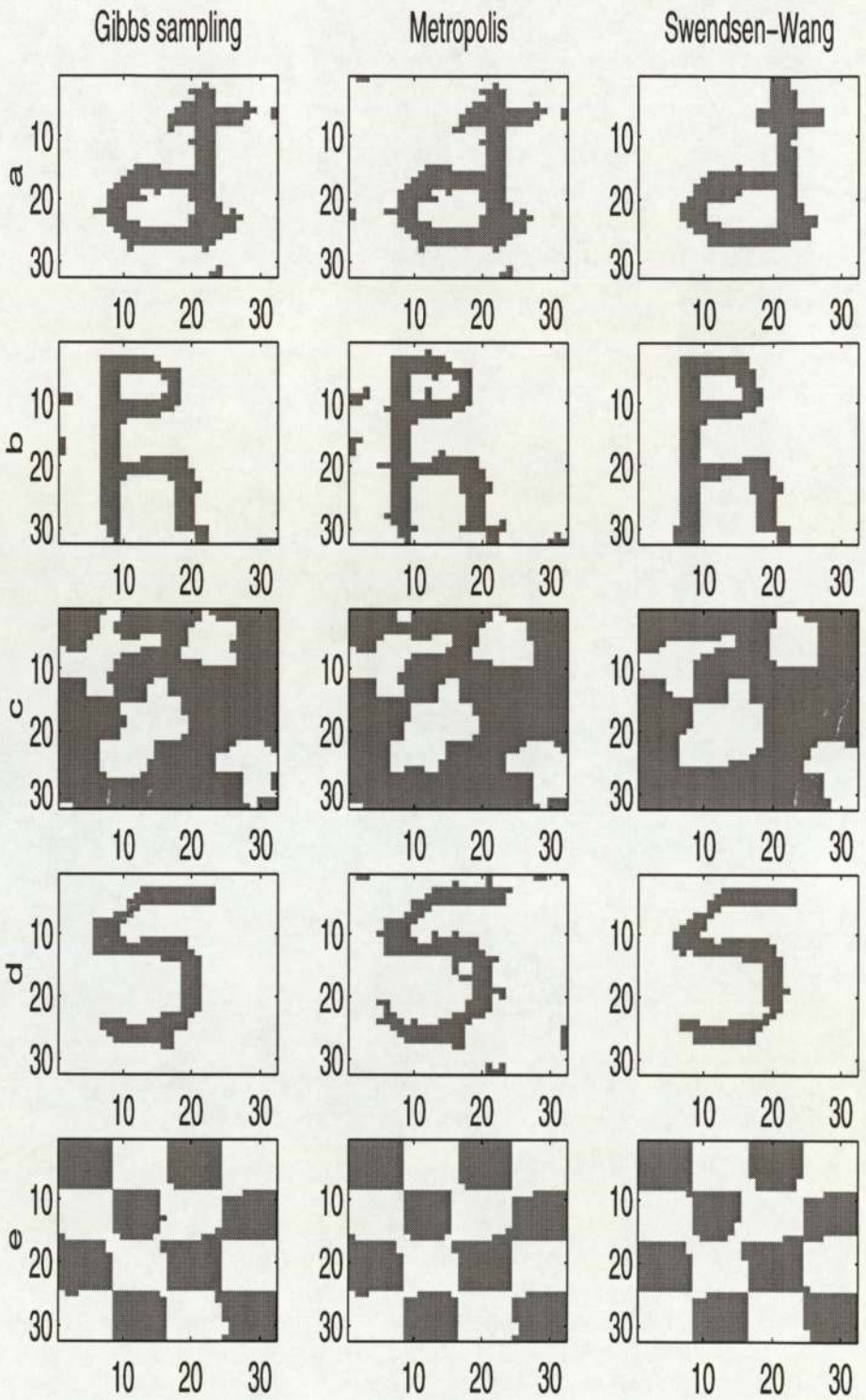


Figure 2.5: Images restored by simulated annealing (Gibbs (left), Metropolis (centre)) and the Swendsen-Wang algorithm (right) for an equivalent amount of CPU time $t = 130$ seconds. The restoration parameters β and κ (corresponding to the noise level q) are: in row (a): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.58$; in row (b): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.59$; in row (c): $\kappa = 2.0$ ($q = 0.27$),$\beta = 0.59$; in row (d): $\kappa = 2.4$ ($q = 0.23$),$\beta = 0.57$; in row(e): $\kappa = 1.9$ ($q = 0.28$),$\beta = 0.61$. The temperature T used for simulated annealing is equal to 3 for all the images

In contrast to the single site update algorithms, the S-W algorithm can cause major configuration changes for a small amount of CPU time especially when large patches are present. This is due to the fact that it can remove the strong dependence between like coloured clusters while it strengthens the dependence between neighbouring nodes within a cluster, thus enabling the update of the whole cluster at the same time. Figure 2.6 shows five successive realizations from the S-W and Gibbs algorithms after an equivalent amount of CPU time t (5 seconds), for increasing t, for $\kappa = 0.1$ and $\beta = 0.5$. We note that the images alter drastically in the case of S-W whereas there is nearly no modification for Gibbs.

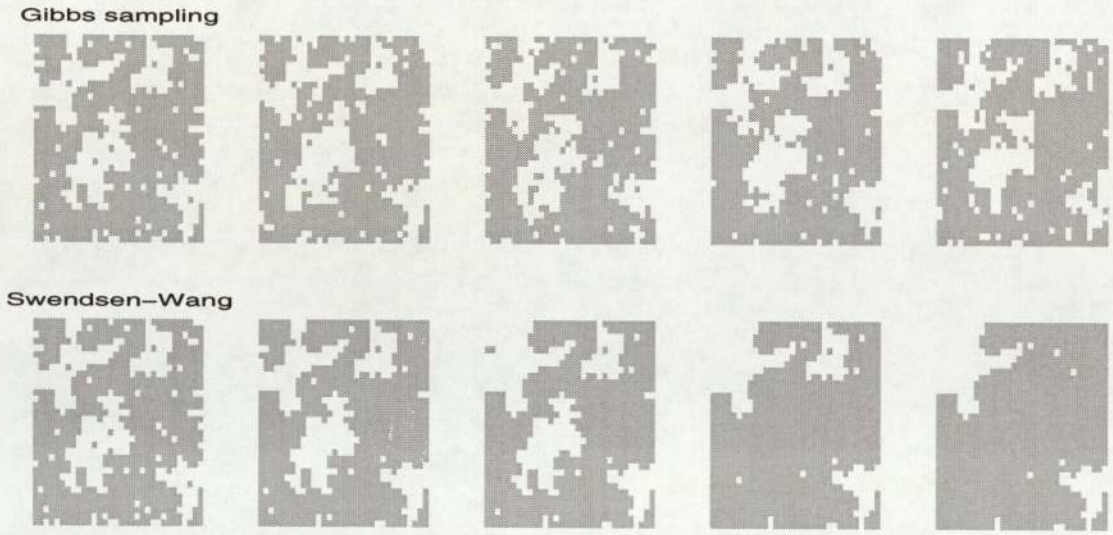


Figure 2.6: Successive images from the Gibbs (above) and Swendsen-Wang algorithms after an equivalent amount of CPU time t (5 seconds), for increasing t, for $\kappa = 0.1, \beta = 0.5$

We can illustrate the image modifications after each CPU second by using the normalised Hamming distance criterion which states that, for a set of images $\{S^{(1)}, \ldots, S^{(n)}\}$, the distance between $S^{(i+1)}$ and $S^{(i)}$ is

$$Hd = \frac{\left\| S^{(i+1)} - S^{(i)} \right\|}{\sum_{i=1}^{n-1} \left\| S^{(i+1)} - S^{(i)} \right\|}$$

Figure 2.7 plots the normalised Hamming distances for two series of the above images in Figure 2.6 for a same amount of CPU time. The nearly constant curve for the Gibbs algorithm demonstrates that Gibbs sampling, in this case, is much slower than S-W sampling.

2.6.2 Influence of κ and β

So far we have assumed that the true image is known so that it is possible to estimate the relevant prior parameters κ and β which control the quality of the restoration. In practice this is not the case and it is therefore necessary to select reliable parameter ranges which give reasonable reconstructions. Our aim is to explore now the sensitivity of restored images to different parameter choices. We recall

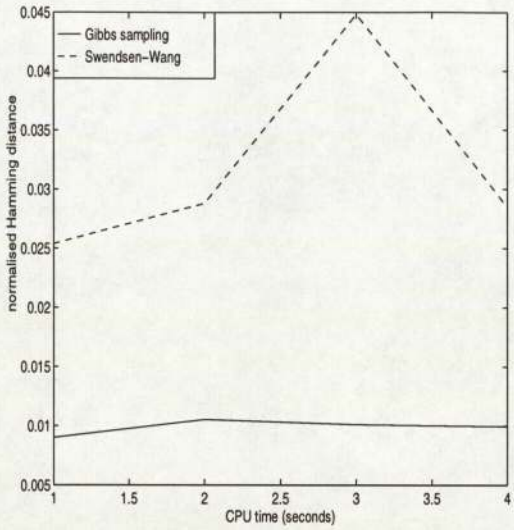


Figure 2.7: Comparison of normalised Hamming distances for the images in Figure 2.6 from the Gibbs and Swendsen-Wang algorithms. The nearly constant curve for the Gibbs algorithm shows that Gibbs sampling is much slower than S-W sampling

that κ controls the binding of the restored configuration to the noisy one and β controls how strong a tendency there is for neighbouring pixels to be the same. We are mainly interested in the behaviour of the images when small or large values of either κ or β are used. Basically, this approach enables us to have some helpful insights about prior knowledge for κ and β that permits improved restorations. We shall return to this more formally in Chapter 5.

Let us first examine β. We consider a set of five values $\{0.15, 0.25, 0.35, 0.95, 1.2\}$ and we assign the value 2.4 to κ corresponding to a noise level $q = 0.23$. We restore the second image in row (b) in Figure 2.3.

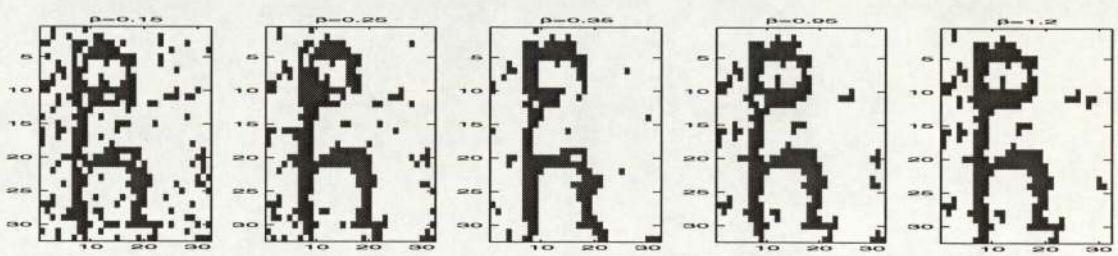


Figure 2.8: Restored images for $\kappa = 2.4$ and for different values of β: $\beta = 0.15$ for the first image from the left, $\beta = 0.25$ for the second image, $\beta = 0.35$ for the third one, $\beta = 0.95$ for the fourth one, $\beta = 1.2$ for the fifth one

Figure 2.8 shows that, for small values (0.15, 0.25), we obtain speckled images while we get a regular patchy configuration for $\beta = 0.35$. Large values (0.95, 1.2) favour images with large single coloured patches: we note that the large black islands are not cleaned up.

Let us now examine κ. Figure 2.9 shows the reconstructed images for $\kappa = \{3.80, 3.03, 1.51, 0.81\}$

which correspond to the values of the noise level $q = \{0.13, 0.18, 0.32, 0.40\}$. The value of β is fixed at 0.6.

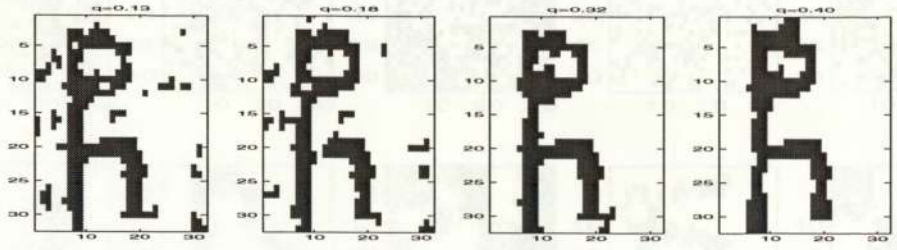


Figure 2.9: Restored images for $\beta = 0.6$ and for different values of q: $q = 0.13$ for the first image from the left, $q = 0.18$ for the second image, $q = 0.32$ for the third one, $q = 0.40$ for the fourth one

Figure 2.9 shows that, for $q = \{0.13, 0.18\}$, the images are speckled while for $q = \{0.32, 0.40\}$, there are no more black islands. By using $q = \{0.13, 0.18\}$, we consider restoring images that are not *really* degraded so that some black islands are not regarded as noisy and hence are not cleaned up. Conversely, for large values of $q = \{0.32, 0.40\}$, we consider restoring very degraded images and consequently all black patches are cleaned up. Therefore, the images are likely to contain large black and white regions. However, if the original image has some black patches, then we have to discard overly large values of q. Note that the shape of the restored images does not alter enormously with respect to different values of q for an appropriate β while it exhibits drastic changes when β takes extreme values, as shown in Figure 2.8. Therefore, it turns out to be necessary to take account of β much more than κ for it has a stronger influence on the image treatment. For further study and research, we will limit κ in the interval $[1.0, 3.0]$ and β in the range $[0.15, 0.95]$.

2.6.3 Restoration of very noisy images

It turns out to be much harder to restore images that are more degraded than the ones in Figure 2.3. As we might expect, noisier images lead to non-regular restorations with large patches of black and white, as shown in Figure 2.10 where the images have a noise level equal to 0.3 and are restored using the S-W algorithm.

2.7 Discussion

As an integral part of this project, we have evaluated some Monte Carlo techniques, in particular the Gibbs and Metropolis algorithms. It is also our aim to evaluate the much less well known Swendsen Wang algorithm which has been shown to be faster than the two single site update algorithms for sampling the kinds of distribution typical in image restoration problems. However, we have found that the tendency of the S-W method to update large clusters of identical pixels can be disadvantageous

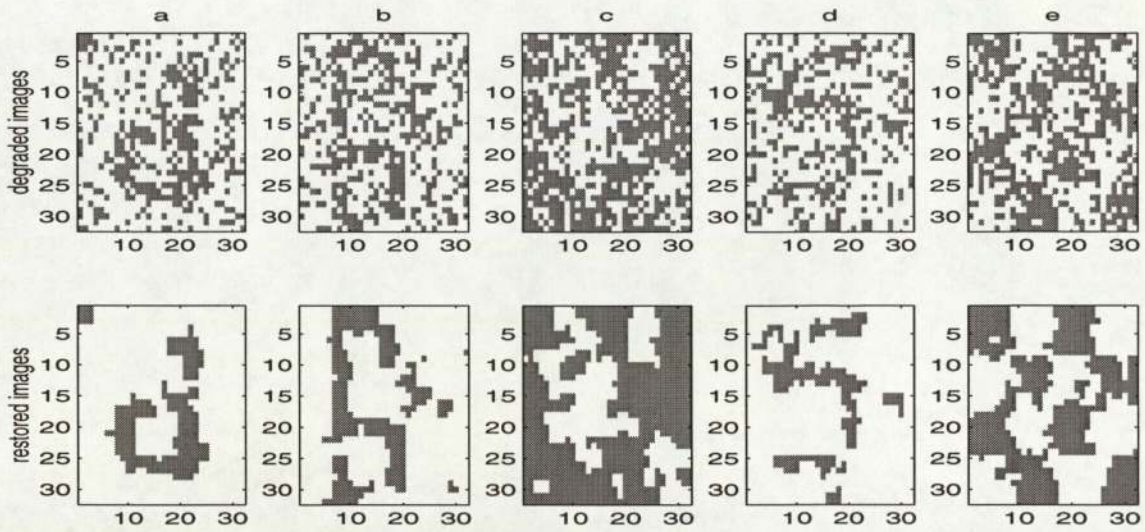


Figure 2.10: Restoration of degraded images with a noise level equal to 0.3. The restoration parameters β and κ (corresponding to the noise level q) are: in column (a): $\kappa = 1.2$ ($q = 0.35$),$\beta = 0.58$; in column (b): $\kappa = 1.16$ ($q = 0.36$),$\beta = 0.6$;in column (c): $\kappa = 1.32$ ($q = 0.34$),$\beta = 0.63$; in column (d): $\kappa = 1.2$ ($q = 0.35$),$\beta = 0.58$;in column (d): $\kappa = 1.0$ ($q = 0.38$),$\beta = 0.63$;

in restoring the fine details of the image. Perhaps a hybrid approach in which a sequence of S-W updates followed by a sequence of Gibbs or Metropolis updates would yield better restored images.

We have found that the prior parameters κ and β have a strong influence on the quality of restored images. We have tried several values for these two parameters and have determined relevant parameter ranges. The simulation process enables us to draw two concluding remarks. The first one is that β has a stronger influence and exerts more effects than κ in analysing the restorations. For small or large values of β, the images are modified more drastically than they are for extreme values of κ. The second remark is that small values of β ($\beta \leq 0.25$) yield speckled restorations while very large values ($\beta \geq 0.95$) lead to patchy images, with large single coloured patches. We have also found that small values of the noise level q ($q < 0.18$) or large values of κ ($\kappa > 0.3$) favour speckled images while large values of q ($q \geq 0.32$) or small values of κ ($\kappa < 1.5$) lead to large single coloured regions. This last conclusion is very important for the subsequent study in Chapter 5 because it lends insight to determining suitable priors for these parameters. In effect, all restored images depend strongly on the priors over κ and β which govern the efficiency of the restoration scheme. However, dealing with uncertainty in these two parameters poses some additional problems and can be addressed by some techniques which have recently been developed for approximating large probabilistic networks [10, 11, 16, 17, 18].

Chapter 3

Bounding techniques

Throughout this project, the key technique to computing estimates for κ and β is the problem of calculating the evidence $P(D|\kappa,\beta)$ defined in (1.13). However, even for a 32×32 image, computing this evidence exactly is intractable because it requires summing over a set S of all nodes, which means we have to take into account $2^{32\times32}$ possible pixel configurations. We therefore apply some other efficient methods which have recently been developed for approximating large probabilistic networks, as we shall see in this chapter and in the next chapter. This chapter deals essentially with the problem of bounding from both above and below the evidence $P(D|\kappa,\beta)$ and therefore the partition functions $Z_p(\beta)$ and $Z_r(\kappa,\beta)$ defined in (1.7) and (1.9). Note that, from (1.5), $Z_l(\kappa)$ can be computed exactly. If we consider $\underline{Z_p(\beta)}$ and $\underline{Z_r(\kappa,\beta)}$ as the lower bounds, $\overline{Z_p(\beta)}$ and $\overline{Z_r(\kappa,\beta)}$ as the upper bounds of $Z_p(\beta)$ and $Z_r(\kappa,\beta)$ respectively, then from (1.13), the lower bound $\underline{P(D|\kappa,\beta)}$ of $P(D|\kappa,\beta)$ is

$$\underline{P(D|\kappa,\beta)} = \frac{\underline{Z_r(\kappa,\beta)}}{Z_l(\kappa)\overline{Z_p(\beta)}} \tag{3.1}$$

Likewise, the upper bound $\overline{P(D|\kappa,\beta)}$ of $P(D|\kappa,\beta)$ is

$$\overline{P(D|\kappa,\beta)} = \frac{\overline{Z_r(\kappa,\beta)}}{Z_l(\kappa)\underline{Z_p(\beta)}} \tag{3.2}$$

Before attempting to bound the evidence, we begin by reducing the expressions of the two partition functions Z_p and Z_r defined in (1.7) and (1.9) to the Boltzmann machine form whose properties enable us to tackle our problem more easily.

3.1 Boltzmann machines

A Boltzmann machine distribution with binary (0/1) variables $S = \{S_1, ..., S_n\}$, where $n = m \times m$ and m is the size of the image, is defined as follows

$$P_n(S|h,J) = \frac{1}{Z_n(h,J)} B_n(S|h,J)$$

where h is a n-dimensional vector called the *bias* vector and J a $n \times n$ symmetric matrix which is referred to as the *connection* matrix between the pixels. The Boltzmann factor B_n has the form

$$B_n(S|h,J) = \exp\left\{\sum_{j=1}^{n} h_j S_j + \frac{1}{2}\sum_{i,j=1}^{n} J_{ij} S_i S_j\right\}$$

The partition function Z_n normalises the distribution

$$Z_n(h,J) = \sum_{S} B_n(S|h,J) \tag{3.3}$$

3.1.1 Transformation of the partition function of the distribution for restored images

The partition function Z_r of the distribution for restored images in (1.9) can be written in the form (3.3)

$$Z_r(\kappa,\beta) = \sqrt{C_r} Z_n(h(\kappa,\beta), J(\beta))$$

where $\sqrt{C_r}$ is a constant.

We reduce the following expression

$$B_r(S|\kappa,\beta) = \exp\left\{-\frac{\kappa}{2}\sum_{i=1}^{n}(D_i - S_i)^2 + \beta\sum_{i\backsim j} I[S_i = S_j]\right\}$$

to the Boltzmann machines form

$$B_n(S|h,J) = \sqrt{C_r}\exp\left\{\sum_{i=1}^{n} h_i S_i + \frac{1}{2}\sum_{i,j=1}^{n} J_{ij} S_i S_j\right\} \tag{3.4}$$

By noting that $I[S_i = S_j] = 1 + 2S_iS_j - S_i - S_j$ and $S_j^2 = S_j$, we can write

$$\begin{aligned} B_r &= \exp\left\{-\frac{\kappa}{2}\sum_i (D_i^2 - 2D_iS_i + S_i^2) + \beta\sum_{i\backsim j}(1 + 2S_iS_j - S_i - S_j)\right\} \\ &= \exp\left\{\kappa\sum_i D_iS_i - \frac{\kappa}{2}\sum_i S_i - \beta\sum_{i\backsim j}(S_i + S_j) + 2\beta\sum_{i\backsim j} S_iS_j\right\} \\ &\quad \times \exp\left\{-\frac{\kappa}{2}\sum_i D_i^2 + \beta\sum_{i\backsim j} 1\right\} \end{aligned} \tag{3.5}$$

By identifying the constant, the linear and the quadratic terms from (3.4) and (3.5), we obtain

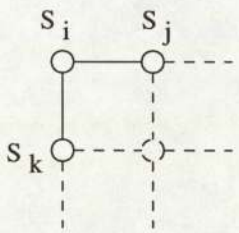
$$\sqrt{C_r} = \exp\left\{-\frac{\kappa}{2}\sum_i D_i^2 + \beta\sum_{i\backsim j} 1\right\} \tag{3.6}$$

$$\sum_i^n h_iS_i = \kappa\sum_i D_iS_i - \frac{\kappa}{2}\sum_i S_i - \beta\sum_{i\backsim j}(S_i + S_j) \tag{3.7}$$

$$\frac{1}{2}\sum_{i,j=1}^{n} J_{ij}S_iS_j = 2\beta\sum_{i\backsim j} S_iS_j \tag{3.8}$$

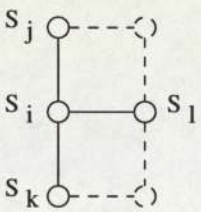
Now, let us compute $\sum_{i\backsim j}(S_i + S_j)$. As $i \backsim j$ means the node S_j corresponding to the index j is the nearest vertical or horizontal neighbour of the node S_i, there are therefore three different cases for the position of S_i in the image:

- S_i has two neighbours



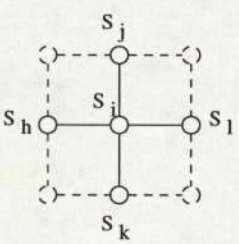
$$\sum_{l\backsim j}(S_l + S_j) = 4S_i + \sum_{i\backsim j} S_j + \sum_{l\backsim j, l\neq i, j\neq i}(S_l + S_j) \tag{3.9}$$

- S_i has three neighbours



$$\sum_{l\backsim j}(S_l + S_j) = 6S_i + \sum_{i\backsim j} S_j + \sum_{l\backsim j, l\neq i, j\neq i}(S_l + S_j) \tag{3.10}$$

- S_i has four neighbours



$$\sum_{l\backsim j}(S_l + S_j) = 8S_i + \sum_{i\backsim j} S_j + \sum_{l\backsim j, l\neq i, j\neq i}(S_l + S_j) \tag{3.11}$$

Thus, from (3.7), (3.9), (3.10) and (3.11), the structure of the h vector is

$$h_i = \begin{cases} \kappa\left(D_i - \frac{1}{2}\right) - 4\beta, & \text{if } S_i \text{ has two neighbours} \\ \kappa\left(D_i - \frac{1}{2}\right) - 6\beta, & \text{if } S_i \text{ has three neighbours} \\ \kappa\left(D_i - \frac{1}{2}\right) - 8\beta, & \text{if } S_i \text{ has four neighbours} \end{cases} \tag{3.12}$$

Let us now compute $\sum_{i\backsim j} S_i S_j$. For any position of any pixel in the lattice, we find that

$$\sum_{i\backsim j} S_i S_j = \sum_{i,j=1}^{n} Y_{ij} S_i S_j$$

with

$$Y_{ij} = \begin{cases} 1, & \text{if } S_j \text{ is a neighbour of } S_i \\ 0, & \text{otherwise} \end{cases}$$

Therefore, from (3.8), the structure of the J matrix is

$$J_{ij} = \begin{cases} 4\beta, & \text{if } S_j \text{ is a neighbour of } S_i \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

3.1.2 Transformation of the partition function of the Markov random field prior

Likewise, the partition function of the Markov random field prior in (1.7) can be reduced to the form (3.3)

$$Z_p(\beta) = \sqrt{C_p} Z_n(h(\beta), J(\beta))$$

with

$$\sqrt{C_p} = \exp\left\{\beta \sum_{i \backsim j} 1\right\}$$

As before, the structure of the h vector is

$$h_i = \begin{cases} -4\beta, & \text{when } S_i \text{ has two neighbours} \\ -6\beta, & \text{when } S_i \text{ has three neighbours} \\ -8\beta, & \text{when } S_i \text{ has four neighbours} \end{cases} \tag{3.14}$$

and the structure of the J matrix is

$$J_{ij} = \begin{cases} 4\beta, & \text{if } S_j \text{ is a neighbour of } S_i \\ 0, & \text{otherwise} \end{cases}$$

Note that the connection matrix J for Z_r and Z_p is symmetric and all its elements have the same value.

We have therefore shown that the calculation of $P(D|\kappa, \beta)$ defined in (1.13) can be reduced to finding the partition function $Z_n(h, J)$. As the image is regarded as a non-trivially connected graph, the task of computing $Z_n(h, J)$ exactly is not feasible. Therefore, the only way of efficiently estimating it is to use approximate methods in conjunction with exact calculations whenever possible. The technique known as recursive node-elimination [16] aims at obtaining variational bounds that allow a recursive formula of the form

$$Z_n(h, J) \gtrless C(h, J) Z_{n-1}\left(\tilde{h}, \tilde{J}\right) \tag{3.15}$$

This formula has three main advantages. First, several nodes at the same time can be removed by merely transforming the model parameters h and J. Second, the approximations involved in the elimination should yield at each recursive step upper and lower bounds which are optimised in order to be as tight as possible. Note that it is necessary to have rigorous upper and lower bounds in order to be able to know approximately where the exact result lies. Finally, and most importantly, if the remaining and simplified partition function $Z_{n-1}\left(\tilde{h}, \tilde{J}\right)$ permits the use of exact methods or decimation techniques, then a considerable gain in accuracy and rigorous estimates of confidence should be obtained.

3.2 Recursive node-elimination

3.2.1 The lower bound

The first approach for the lower bound recursion is developed by Jaakkola and Jordan [16]. Consider eliminating the pixel S_k

$$\begin{aligned} Z_n(h,J) &= \sum_S B_n(S|h,J) \\ &= \sum_{S\backslash S_k}\sum_{S_k} B_n(S|h,J) \\ &= \sum_{S\backslash S_k}\left(1+e^{h_k+\sum_j J_{kj}S_j}\right) B_{n-1}\left(S\backslash S_k|h,J\right) \end{aligned} \tag{3.16}$$

Let x be

$$x = h_k + \sum_j J_{kj}S_j \tag{3.17}$$

The lower bound methodology then consists of bounding the negative logarithm of the term of $1+e^x$. Consider the function $g(x) = -\log(1+e^x)$, we can write [15]

$$\begin{aligned} g(x) &= -\log\left(\sum_{m\in\{0,1\}} e^{mx}\right) \\ &= -\log\left(\sum_{m\in\{0,1\}} \xi^m(1-\xi)^{1-m}\frac{e^{mx}}{\xi^m(1-\xi)^{1-m}}\right) \end{aligned}$$

where ξ is in $[0,1]$. As $-\log(x)$ is a convex function and as $\sum_{m\in\{0,1\}}\xi^m(1-\xi)^{1-m}=1$, we can then apply Jensen's inequality

$$\begin{aligned} g(x) &\leq -\sum_{m\in\{0,1\}} \xi^m(1-\xi)^{1-m}\log\frac{e^{mx}}{\xi^m(1-\xi)^{1-m}} \\ &= -\xi x + \xi\log\xi + (1-\xi)\log(1-\xi) \\ &= -\xi x - H(\xi) \end{aligned} \tag{3.18}$$

where $H(.)$ is the binary entropy function, $H(\xi) = -\xi\log\xi - (1-\xi)\log(1-\xi)$. Therefore, it follows from the above equation that

$$e^{\xi x + H(\xi)} \leq 1 + e^x \tag{3.19}$$

Performing the optimisation over ξ gives $\xi^* = e^x/(1+e^x)$ and leads to an equality instead of a bound. We can use optimisation algorithms such as the scaled conjugate gradient method or the simplex search routine[1] to optimise ξ in order to tighten the bound. The geometry of the bound when ξ is kept fixed for all x is illustrated in Figure 3.1. The value of x for which ξ is optimal is the point where the bound is exact. The solid curve represents $1+e^x$ and the dashed curve plots $e^{\xi x+H(\xi)}$ for a fixed $\xi = e^{0.7}/\left(1+e^{0.7}\right)$.

[1]The scaled conjugate gradient method generally yields better results than the simplex search routine but needs to take into account gradient information (which is complicated to program in our problem) while the simplex technique does not.

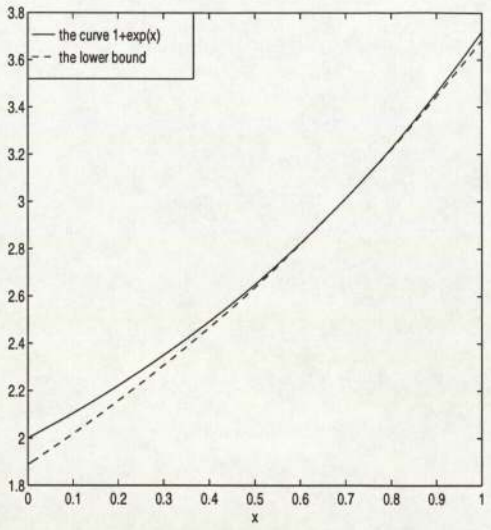


Figure 3.1: Geometry of the lower bound

From (3.19), there exist variational parameters ξ_k in $[0,1]$ such that (3.16) can be written

$$\begin{aligned} Z_n(h,J) &\geq \sum_{S\backslash S_k} e^{\xi_k\left(h_k+\sum_j J_{kj}S_j\right)+H(\xi_k)} B_{n-1}\left(S\backslash S_k|h,J\right) \\ &= e^{\xi_k h_k+H(\xi_k)} \sum_{S\backslash S_k} B_{n-1}\left(S\backslash S_k|\tilde{h},J\right) \\ &= e^{\xi_k h_k+H(\xi_k)} Z_{n-1}\left(\tilde{h},J\right) \end{aligned} \tag{3.20}$$

The bias vector h is transformed to $\tilde{h}_j = h_j + \xi_k J_{kj}$, which means that only the biases of the neighbours of the deleted nodes are modified. However, the structure of the connection matrix J is unchanged. This amounts to saying that if a node is removed in the image, then the structure of J for the $(n-1)$ remaining other nodes is the same as the one for n nodes. In other words, there is no additional connection between the pixels adjacent to the eliminated one, as illustrated in Figure 3.2. Thus, graphically, the operation translates into merely suppressing the node.

However, each node elimination involves an additional bound and therefore the approximation of Z_n becomes more and more inaccurate as the nodes are removed. It is desirable to apply the recursion only to the extent that the remaining and simplified graph permits the use of exact methods or decimation techniques. Consequently, the problem is transformed to that of finding the pixels whose suppression would render the rest of the graph tractable. Figure 3.2 [16] shows that the elimination of the dotted nodes reveals a chain graph whose structure of the connection matrix permits fast and exact methods, as we shall see in Chapter 4.

For a 32×32 image, we will see in Chapter 4 that removing 325 pixels among 1024 yields a tractable subgraph. This means that we have to optimise a 325-dimensional vector ξ introduced in (3.20), which will certainly take a long time. We will see in Section 3.2.2 that, with 75 nodes deleted on a total number of 256, it takes about an hour to get the relevant lower bound if we use the simplex search

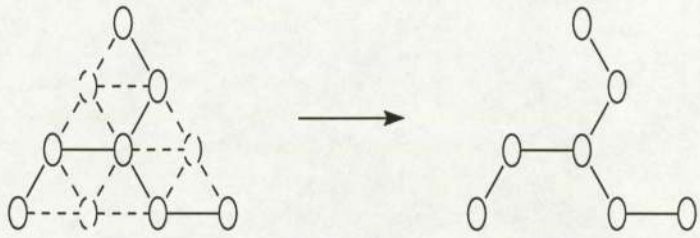


Figure 3.2: Enforcing tractable networks

routine. Moreover, the optimisation procedure does not always lead successfully to the desired result. Indeed, one of the main difficulties which still remains is the tendency for such search algorithms to spend a long time in the neighbourhood of poor local minima and failing to discover good minima which make a much more significant contribution to improving the accuracy of the bound. Thus, potentially, a large number of steps are needed before obtaining good result.

3.2.2 Alternative interpretation of the lower bound

As the computational effort seems prohibitive, we examine a similar method which reduces the optimisation required. The above methodology of the lower bound is actually equivalent to searching for the tangent of the curve $f(x) = \log(1 + e^x)$ that allows to approximate $f(x)$ accurately in the region $x = h_k + \sum_j J_{kj} S_j$. A simplified idea is to find a tangent point x^* that is close to x and is as follows: although x cannot be known because S_j takes two possible values 0 or 1, it is easy to bound it by

$$h_k \leq x \leq h_k + \sum_j J_{kj} \tag{3.21}$$

Moreover, $f(x)$ is a convex function of x and since any tangent line for a convex function always remains below the function, it also serves as a lower bound. This can be illustrated in Figure 3.3, where we assume that the eliminated node S_k has four neighbours: from (3.13) and (3.21), we have $h_k \leq x \leq h_k + 16\beta$, then we consider $h_k = \kappa\left(D_i - \frac{1}{2}\right) - 8\beta$ in (3.12), and we take for demonstration $D_i = 1$, $\kappa = 2$ and $\beta = 0.85$. The solid curve stands for $\log(1 + e^x)$ and the dashed curve plots the tangent at $x^* = 3$.

By taking the tangent to the curve at x^*, we can therefore write

$$\begin{aligned} f(x) &\geq \frac{\partial f(x^*)}{\partial x}(x - x^*) + f(x^*) \\ &= \frac{e^{x^*}}{1 + e^{x^*}} x + f(x^*) - \frac{e^{x^*}}{1 + e^{x^*}} x^* \end{aligned}$$

By denoting $\lambda = e^{x^*} / \left(1 + e^{x^*}\right)$ and $H(\lambda) = -\lambda \log(\lambda) - (1 - \lambda)\log(1 - \lambda)$, we get an expression which is similar to (3.19)

$$e^{\lambda x + H(\lambda)} \leq 1 + e^x \tag{3.22}$$

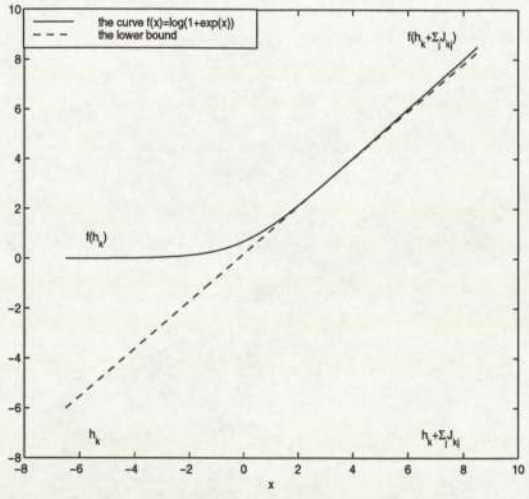


Figure 3.3: Illustration of the tangent bound

From (3.22), there exist variational parameters λ_k such that

$$\begin{aligned} Z_n(h,J) &\geq \sum_{S\backslash S_k} e^{\lambda_k\left(h_k+\sum_j J_{kj}S_j\right)+H(\lambda_k)} B_{n-1}\left(S\backslash S_k|h,J\right) \\ &= e^{\lambda_k h_k+H(\lambda_k)} \sum_{S\backslash S_k} B_{n-1}\left(S\backslash S_k|\tilde{h},J\right) \\ &= e^{\lambda_k h_k+H(\lambda_k)} Z_{n-1}\left(\tilde{h},J\right) \end{aligned} \tag{3.23}$$

where the bias vector h is modified and is transformed to $\tilde{h}_j = h_j + \lambda_k J_{kj}$ and the structure of the J matrix remains unchanged as it does in the previous section. We note that this method is equivalent to the one described in Section 3.2.1: we need also to optimise the λ vector in (3.23) so as to have a rigorous bound. Nevertheless, we have to stress that, in contrary to the previous method where the ξ vector in (3.20) does not give a clue of what would be its optimal value, this method enables us to understand better the way of finding the relevant λ vector because the latter is a function of the tangent point x^* which has to be close to x, otherwise the accuracy of the bound is compromised. For instance, as shown in Figure 3.3, the gap between the curve and the tangent line is important if the value of x is less than 0.

Our main concern as we have seen is to avoid optimisation over large dimensional spaces. We therefore resort to searching for a suboptimal solution in a lower dimensional space of variational parameters. Indeed, we choose to find the best one-dimensional optimal parameter set that gives the lower bound in order to reduce the computation time.

We have tested[2] the accuracy of the lower bounds on the partition function Z_n, in the case of 16 nodes, where 4 nodes are removed, and in the case of 256 nodes, where 75 nodes are suppressed, for two different approaches: the first approach involves optimisation over a y-dimensional space of

[2] All the results in this section are calculated on the partition function Z_n of the distribution for restored images, in the case of the noisy image presented in row (b) in Figure 2.3.

variational parameters, where y is the number of nodes removed and the second approach involves optimisation over a one-dimensional space of variational parameters. For 16 nodes, it is quite fast to calculate the exact value. The small number of nodes is chosen to facilitate comparisons with the exact result. Figure 3.4 plots the estimates of the log-lower bounds obtained using these two approaches with respect to β and for $\kappa = 1.4$. Figure 3.5 plots, for 16 nodes, the log-exact value and the log-lower bound estimates obtained using the second approach with respect to β and for a fixed value of $\kappa = 1.4$.

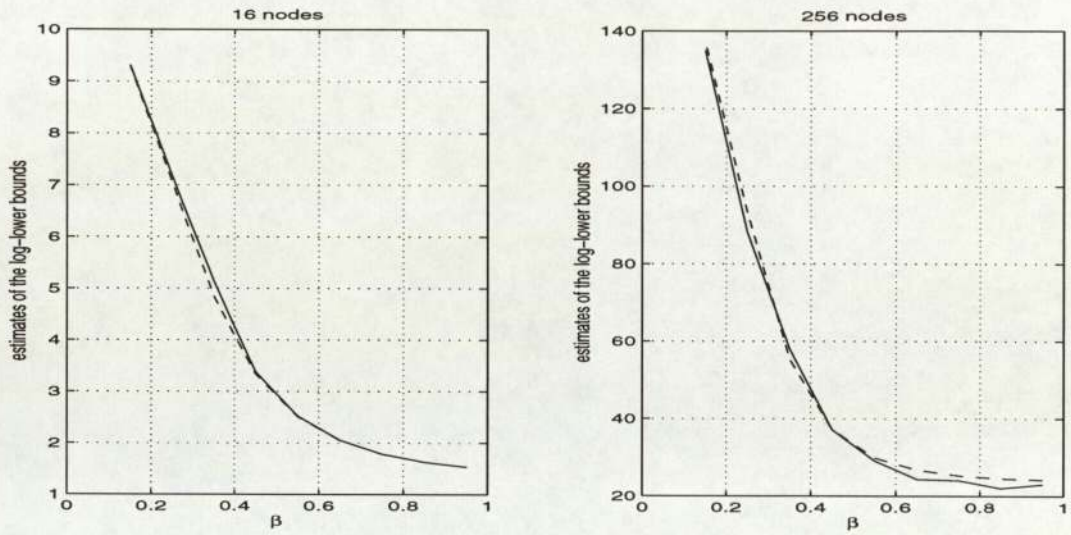


Figure 3.4: Comparison between the log-lower bound estimates (solid curve) using the first approach and the log-lower bound estimates (dashed curve) using the second approach for 16 and 256 nodes. The first approach involves optimisation over a y-dimensional space of variational parameters, where y is the number of eliminated nodes. For 16 nodes: $y = 4$; for 256 nodes: $y = 75$. The second approach involves optimisation over a one-dimensional space of variational parameters. The value of κ is 1.4. The mean error (averaged over $\beta = 0.15..0.95$) is about 0.053 for 16 nodes and the mean absolute error is about 1.25 for 256 nodes

Figure 3.4 reveals, in the case of 16 nodes, where we remove 4 nodes, how the small error ($\simeq 0.053$) (averaged over $\beta = 0.15..0.95$) between the log-lower bound estimates enables us to trust the second approach which presents a good approximation to the log-exact value. Indeed, the mean error between the log-lower bound and the log-exact value is about 0.55, as shown in Figure 3.5. In the case of 256 nodes, we have removed 75 nodes and therefore we have optimised a 75-dimensional vector ξ introduced in (3.20). The bound calculated using the second approach also yields a good approximation compared to the one obtained using the first approach due to the small absolute error ($\simeq 1.25$) in the estimates of the log-lower bounds. However, Figure 3.4 shows that the first approach yields some worse lower bound estimates than those obtained by the second approach. The main reason as we have seen is that the presence of local minima hinders the search for good minima or simply that the number of iterations required during the optimisation process for having the relevant results is not sufficient. We have set the number of iterations to 7000 and it has taken more than an hour to obtain an acceptable

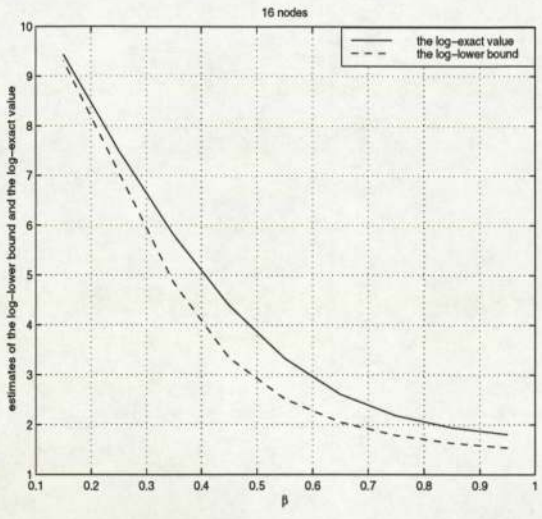


Figure 3.5: Comparison, for 16 nodes, between the log-exact value and the log-lower bound obtained by the second approach involving optimisation over a one-dimensional space of variational parameters. The value of κ is 1.4. The mean error is about 0.55

lower bound whereas it has taken about one minute to get the lower bound using the second approach.

The approach involving optimisation over a one-dimensional space of variational parameters enables us to obtain a rigorous lower bound on the partition function whose value cannot be calculated exactly.

3.2.3 The upper bound

There are some techniques for obtaining an accurate and rigorous upper bound on the partition function Z_n defined in (3.3). One of these [16] is to bound $f(x) = \log(1 + e^x)$ by a parabolic function, where $x = h_k + \sum_j J_{kj}$. To derive the quadratic upper bound we can write

$$f(x) = \log e^{x/2} + \log\left(e^{-x/2} + e^{x/2}\right) \tag{3.24}$$

Now, $\phi(x) = \log\left(e^{-x/2} + e^{x/2}\right)$ is a symmetric function of x and also a concave function of x^2. Any tangent line for a concave function always remains above the function and so it also serves as an upper bound. Therefore we can bound $\phi(x)$ by the tangents of $\phi(\sqrt{w})$ (due to the concavity in x^2). Thus

$$\begin{aligned} \phi(x) &\leq \frac{\partial\phi(\sqrt{w})}{\partial w}(x^2 - w) + \phi(\sqrt{w}) \\ &= \eta(w)x^2 - F(\eta, w) \end{aligned} \tag{3.25}$$

where

$$\begin{aligned} \eta(w) &= \frac{\partial\phi(\sqrt{w})}{\partial w} \\ F(\eta, w) &= \eta(w)w - \phi(\sqrt{w}) \end{aligned}$$

The desired result follows the change of variables: $w = x_k^2$. From (3.25), we note that the bound is exact whenever $x_k = x$. Therefore, it follows from (3.24) and (3.25) that

$$f(x) \leq \frac{x}{2} + \eta(x_k)x^2 - F(\eta, x_k) \tag{3.26}$$

The geometry of the quadratic upper bound when x_k is kept fixed for all x is illustrated in Figure 3.6, where we assume that the eliminated node S_k has four neighbours: from (3.13) and (3.21), we have $h_k \leq x \leq h_k + 16\beta$, then we consider $h_k = \kappa\left(D_i - \frac{1}{2}\right) - 8\beta$ in (3.12), and we take for demonstration $D_i = 1$, $\kappa = 2$ and $\beta = 0.85$. The solid curve stands for $\log(1 + e^x)$ and the dashed curve plots $x/2 + \eta(x_k)x^2 - F(\eta, x_k)$ for a fixed $x_k = 1$.

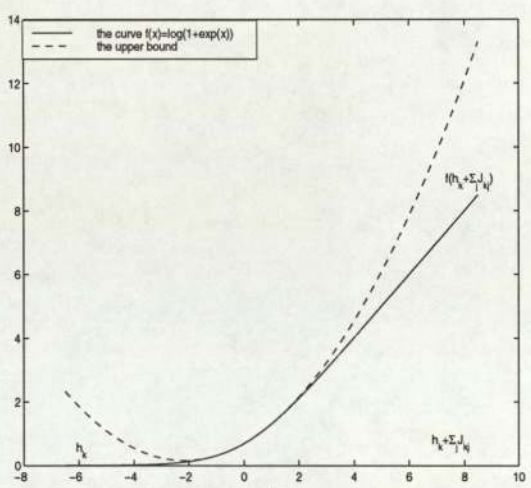


Figure 3.6: Geometry of the quadratic upper bound

From (3.26), there exist variational parameters x_k such that (3.16) can be written

$$\begin{aligned} Z_n(h, J) &\leq \sum_{S \setminus S_k} e^{x/2 + \eta(x_k)x^2 - F(\eta, x_k)} B_{n-1}(S \setminus S_k | h, J) \\ &= e^{h_k/2 + \eta(x_k)h_k^2 - F(\eta, x_k)} Z_{n-1}\left(\tilde{h}, \tilde{J}\right) \end{aligned} \tag{3.27}$$

where the bias vector h is transformed to

$$\tilde{h}_j = h_j + 2h_k\eta(x_k)J_{kj} + \frac{J_{kj}}{2} + \eta(x_k)J_{kj}^2$$

and the connection matrix J is transformed to

$$\tilde{J}_{ij} = J_{ij} + 2\eta(x_k)J_{ik}J_{kj} \tag{3.28}$$

for $i \neq j \neq k$. It means that the nodes S_i and S_j adjacent to the eliminated node S_k become connected. In other words, if $J_{ik} \neq 0$ and $J_{kj} \neq 0$ then $J_{ij} \neq 0$ after the recursion.

The main drawback of this method is that it leads unavoidably to the structural changes of the pixel graph after each recursion in such a way that the remaining subgraph becomes densely connected

so that the use of exact methods for computing the partition function Z_n is prohibitive, as illustrated in Figure 3.7. A naive idea consists of removing a very large number of nodes, but then the bound will certainly be poor and moreover, it is not possible to avoid the optimisation process which we have seen takes a long time and does not successfully give a good result in a high dimensional space.

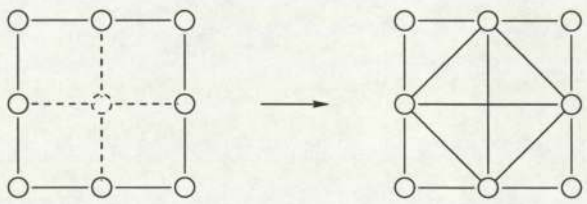


Figure 3.7: Graphical modifications following the recursive quadratic method. The nodes adjacent to the eliminated dotted node become connected: the resulting configuration is more complicated than the initial one

An alternative method [16], which yields less accurate upper bounds than the previous method but preserves the structure of the connection matrix, is to use an application of Jensen's inequality due to the convexity of the function $f(x) = \log(1 + e^x)$, where $x = h_k + \sum_j J_{kj} S_j$. The idea is as follows: let $f_k(u) = f(u + h_k)$, where $u = \sum_j J_{kj} S_j$ and note that $f_k(u)$ has the same convexity properties as f. For any convex function f_k then we have (by Jensen's inequality)

$$\begin{aligned} f_k\left(\sum_j J_{kj} S_j\right) &= f_k\left(\sum_j \mu_j \frac{J_{kj} S_j}{\mu_j}\right) \\ &\leq \sum_j \mu_j f_k\left(\frac{J_{kj} S_j}{\mu_j}\right) \end{aligned} \tag{3.29}$$

As $S_j \in \{0, 1\}$, we can write

$$f_k\left(\frac{J_{kj} S_j}{\mu_j}\right) = \left[f_k\left(\frac{J_{kj}}{\mu_j}\right) - f_k(0)\right] S_j + f_k(0) \tag{3.30}$$

From (3.29) and (3.30), we have

$$f(x) \leq \hat{h}_0 + \sum_j \hat{h}_j S_j \tag{3.31}$$

where

$$\hat{h}_j = \mu_j \left[f\left(\frac{J_{kj}}{\mu_j} + h_k\right) - f(h_k)\right] \tag{3.32}$$

and $\hat{h}_0 = f(h_k)$. μ_j are variational parameters such that

$$\sum_j \mu_j = 1 \tag{3.33}$$

From (3.31), (3.32), (3.33), the upper bound of the partition function in (3.16) can be written

$$\begin{aligned} Z_n(h, J) &\leq \sum_{S \backslash S_k} e^{\hat{h}_0 + \sum_j \hat{h}_j S_j} B_{n-1}(S \backslash S_k | h, J) \\ &= e^{\hat{h}_0} Z_{n-1}\left(\tilde{h}, J\right) \end{aligned} \tag{3.34}$$

The bias vector is transformed to $\tilde{h}_j = h_j + \hat{h}_j$ while the connection matrix J remains unchanged. However, the main drawback of this method is that it is very computation-intensive because of a huge number of variational parameters μ_j that need to be optimised. Indeed, the number of variational parameters is the total number of neighbours of the deleted nodes. As the main objective is to remove enough nodes so as to be able to apply exact techniques, the computational effort is then prohibitive. We will see in Chapter 4 that, for 32×32 image, removing 325 pixels among 1024 leads to a tractable subgraph: this means that we have to optimise a 1275-dimensional vector μ introduced in (3.33).

We prefer to examine another method that involves no optimisation procedure and preserves the structure of the connection matrix. This is achieved by taking account of the convexity of the function $f(x)$ in the interval $\left[h_k, h_k + \sum_j J_{kj}\right]$, then a trivial way is to bound it directly by a line going through h_k and $h_k + \sum_j J_{kj}$, as illustrated in Figure 3.8, where we assume that the eliminated node S_k has four neighbours: from (3.13) and (3.21), we have $h_k \leq x \leq h_k + 16\beta$, then we consider $h_k = \kappa\left(D_i - \frac{1}{2}\right) - 8\beta$ in (3.12), and we take for demonstration $D_i = 1$, $\kappa = 2$ and $\beta = 0.85$. The solid curve stands for $\log(1 + e^x)$ and the dashed curve plots the line going through h_k and $h_k + 16\beta$.

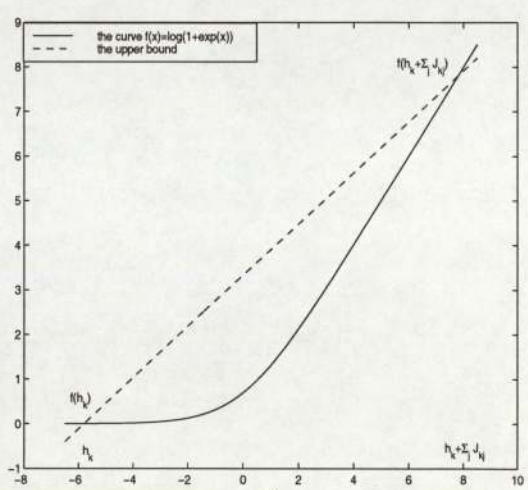


Figure 3.8: Geometry of the linear upper bound

Thus, we can write

$$f(x) \leq cx + d$$

where

$$c = \frac{f\left(h_k + \sum_k J_{kj}\right) - f\left(h_k\right)}{\sum_k J_{kj}}$$

and

$$d = f(h_k) - \frac{f\left(h_k + \sum_k J_{kj}\right) - f\left(h_k\right)}{\sum_k J_{kj}} h_k$$

Hence, the upper bound of the partition in (3.16) can be written

$$\begin{aligned} Z_n(h,J) &\leq \sum_{S\setminus S_k} e^{c\left(h_k+\sum_j J_{kj}S_j\right)+d} B_{n-1}\left(S\setminus S_k|h,J\right) \\ &= e^{ch_k+d} Z_{n-1}\left(\tilde{h},J\right) \end{aligned} \tag{3.35}$$

where the bias vector is transformed to $\tilde{h}_j = h_j + cJ_{kj}$ and the structure of the J matrix remains unchanged.

Figure 3.8 suggests that this upper bound methodology may yield bad results due to the large gap between the curve and the line if $x = h_k + \sum_j J_{kj}S_j$ is not close to the two extreme limits h_k and $h_k + \sum_j J_{kj}$. Another idea is to attempt to partition the curve in several parts and to bound them by a line. Unfortunately, this cannot be achieved because it is not possible to determine the value of x. However, the greater β, the more inaccurate the upper bound is, the smaller β, the tighter the upper bound. Figure 3.9 shows the configuration of the upper bounds for two values of $\beta = \{0.85, 0.35\}$, $D_i = 1$, $\kappa = 2.0$ for the case $h_k = \kappa\left(D_i - \frac{1}{2}\right) - 8\beta$.

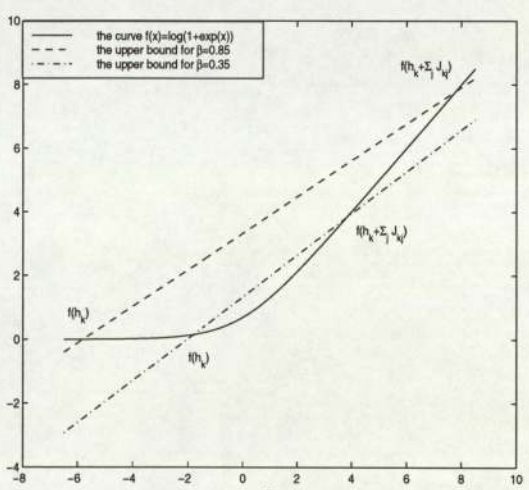


Figure 3.9: Configurations of the upper bound for different values of β, for $\kappa = 1.4$

We have tested the accuracy of the upper bounds on the partition function Z_n, in the case of 16 nodes, where 4 nodes are removed, and in the case of 256 nodes, where 75 nodes are suppressed. The case of 16 nodes is again chosen in order to facilitate comparisons between the upper bound and the exact value of the partition function. Figure 3.10 plots the log-lower bound and the log-upper bound estimates as a function of β for $\kappa = 1.4$. We note that the mean error between the log-lower bound and the log-upper bound estimates is about 35 in the case of 256 nodes and about 2.2 in the case of 16 nodes.

In a high dimensional space, it is unlikely that we get a good approximation of the partition function, so it would be interesting to know precisely whether the exact value is closer to the lower

bound or to the upper bound. We consider the case of 16 nodes where we plot the difference between the log-lower, the log-upper bounds and the log-exact value with respect to β, for $\kappa = 1.4$. Figure 3.11 enables us to state that, provided we optimise the lower bound accurately, then the exact value is closer to the lower bound than to the upper bound. This result can be expected for the upper bound technique requires no optimisation[3].

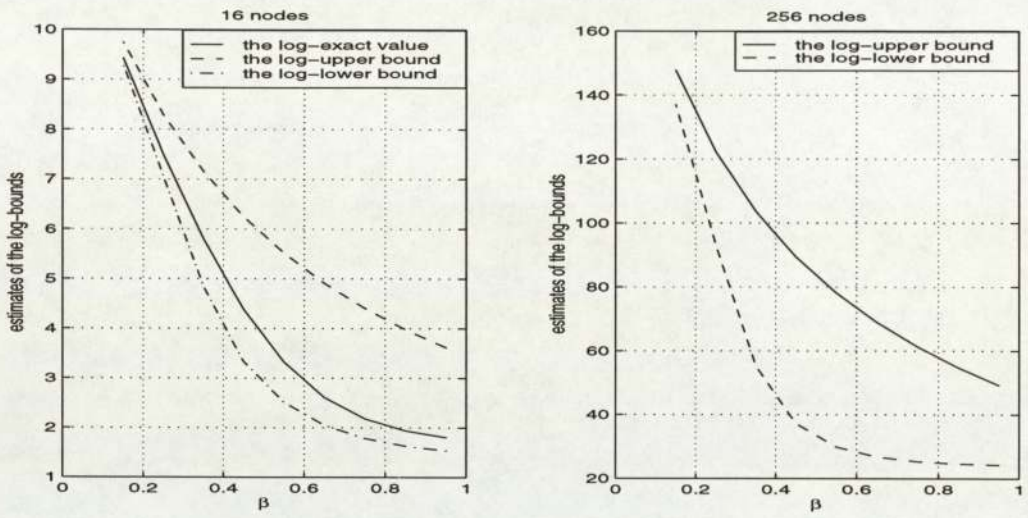


Figure 3.10: Comparison between the log-upper bound and the log-lower bound estimates for $\kappa = 1.4$ in the case of 16 nodes (left), where 4 nodes are removed and in the case of 256 nodes, where 75 nodes are suppressed. The mean error between the log-lower bound and the log-upper bound estimates is about 2.2 for 16 nodes and about 35 for 256 nodes

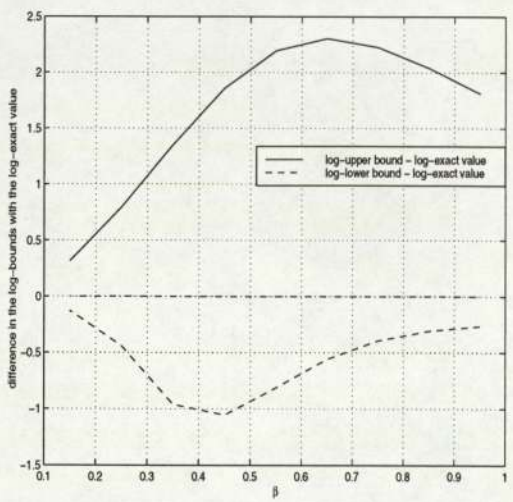


Figure 3.11: Difference in the log-lower, log-upper bounds with the log-exact value for 16 nodes

[3] We explored a method which consists of removing the connections between the nodes instead of the nodes. Unfortunately, our attempt to find a consistent upper bound technique was unsuccessful.

Chapter 4

Decimation

So far we have developed a recursive node-elimination method for efficiently approximating the intractable partition function Z_n defined in (3.3). The main objective is not to remove all nodes in the image, otherwise the bounds will be poor, rather enough to be able to apply exact methods. The exact method we consider here is referred to as a decimation technique, which consists of transforming complex networks into reduced ones without changing their properties. In other words, within the context of Boltzmann machines, the simpler networks retain the same Boltzmann distribution as the complicated ones. The main advantage of the decimation technique [10, 11, 17, 18] is that it permits exact calculations in polynomial computation time in the number of pixels and not in exponential time (we remember that, in general, computing the partition function of a Boltzmann machine requires summing over a set S of m^2 nodes and consequently involves 2^{m^2} calculations, where m is the size of the image).

4.1 Decimation of one node connected to two nodes

The idea [10, 11] about decimation can be illustrated by this following rule. Consider three nodes connected in series, as shown in Figure 4.1. On the left hand side, the extreme pixels S_3 and S_4 are connected with the rest of the image. Though they have no links between them, they have an effective interaction mediated by the intermediate node S_1. The combination of the two connections J_{13} and J_{14} in series leads to a single equivalent connection J'_{34} and the biases h_3 and h_4 are modified to give the new ones h'_3 and h'_4 : in this case the pixel S_1 is said to be "decimated".

An expression for J'_{34}, h'_3 and h'_4 can be obtained by requiring that the nodes S_3 and S_4 in both graphs obey the same Boltzmann distribution. Thus

$$\sum_{S_1=\{0,1\}} e^{h_3S_3+h_1S_1+h_4S_4+J_{13}S_1S_3+J_{14}S_1S_4} = \sqrt{C}e^{h'_3S_3+h'_4S_4+J'_{34}S_3S_4} \tag{4.1}$$

where $\sqrt{C}$ is a constant, independent of S_3 and S_4.

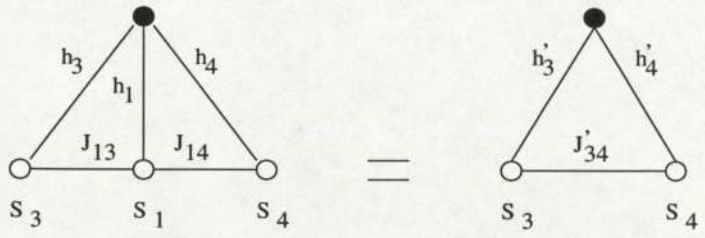


Figure 4.1: Combining connections in series: the graph on the right is obtained by decimating out S_1. The black node represents the bias

By enumerating the possible values of $S_3 = \{0, 1\}$, we obtain the constraints

$$\begin{aligned} e^{h_4 S_4} + e^{h_1 + h_4 S_4 + J_{14} S_4} &= \sqrt{C} e^{h_4' S_4} \\ e^{h_3 + h_4 S_4} + e^{h_3 + h_1 + h_4 S_4 + J_{13} + J_{14} S_4} &= \sqrt{C} e^{h_3' + h_4' S_4 + J_{34}' S_4} \end{aligned}$$

For the possible values of $S_4 = \{0, 1\}$, we have

$$\begin{aligned} 1 + e^{h_1} &= \sqrt{C} \\ e^{h_3} + e^{h_3 + h_1 + J_{13}} &= \sqrt{C} e^{h_3'} \\ e^{h_4} + e^{h_1 + h_4 + J_{14}} &= \sqrt{C} e^{h_4'} \\ e^{h_3 + h_4} + e^{h_3 + h_1 + h_4 + J_{13} + J_{14}} &= \sqrt{C} e^{h_3' + h_4' + J_{34}'} \end{aligned}$$

Finally, from these equations, we obtain

$$\begin{aligned} \sqrt{C} &= 1 + e^{h_1} \\ h_3' &= h_3 + \log \frac{1 + e^{h_1 + J_{13}}}{\sqrt{C}} \\ h_4' &= h_4 + \log \frac{1 + e^{h_1 + J_{14}}}{\sqrt{C}} \\ J_{34}' &= \log \sqrt{C} \frac{1 + e^{h_1 + J_{13} + J_{14}}}{(1 + e^{h_1 + J_{13}})(1 + e^{h_1 + J_{14}})} \end{aligned}$$

Note that the decimation technique can be applied if the node to be decimated has an effective interaction with the other nodes, or in other words, is connected to at least one node.

In the problem of image restoration, the challenge is how to eliminate the pixels so that the configuration of the resulting reduced image permits the use of the decimation technique. The aim consists then of removing the smallest possible number of nodes in order to rigorously bound the partition function in such a way that it is easy to implement the decimation operations. This first rule alone, in conjunction with the recursive methods, does not suffice to yield a good approximation of the desired quantity and we need three more rules so as to be able to decimate out a large number of nodes remaining in the reduced image.

4.2 Decimation of one node connected to one node

In this rule, we decimate a node which is only connected to one node, as shown in Figure 4.2. Note that, henceforth, for more clarity on the figures, the biases are not presented any longer.

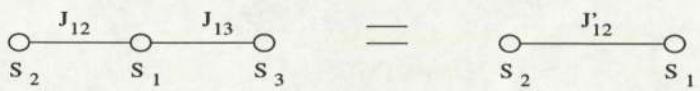


Figure 4.2: Decimation of one node

In order for the two systems to obey the same Boltzmann distribution, the following equations must hold

$$\sum_{S_3=\{0,1\}} e^{h_1 S_1 + h_2 S_2 + h_3 S_3 + J_{12} S_1 S_2 + J_{13} S_1 S_3} = \sqrt{C} e^{h_1' S_1 + h_2' S_2 + J_{12}' S_1 S_2} \tag{4.2}$$

By enumerating the possible values for $S_1 = \{0, 1\}$ and $S_2 = \{0, 1\}$, we find

$$\begin{aligned} \sqrt{C} &= 1 + e^{h_3} \\ h_1' &= h_1 + \log \frac{1 + e^{h_3 + J_{13}}}{\sqrt{C}} \\ h_2' &= h_2 \\ J_{12}' &= J_{12} \end{aligned}$$

Note that the connection J_{12} and the bias h_2 are not modified, only the bias for S_1 alters.

4.3 Decimation of two nodes

Depicted in Figure 4.3, this rule involves decimating out two nodes, first S_2 and then S_1.

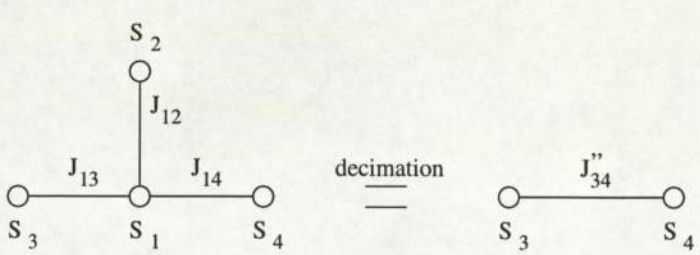
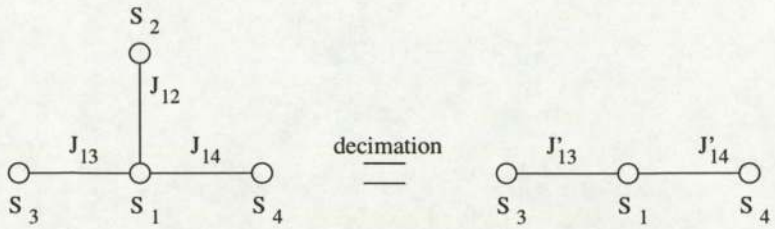


Figure 4.3: Decimation of two nodes

As before, the two systems are required to obey the same Boltzmann distribution

$$\sum_{S_2, S_1} e^{\sum_{i=1}^{4} h_i S_i + \sum_{i=2}^{4} J_{1i} S_1 S_i} = \sqrt{C} e^{h_3'' S_3 + h_4'' S_4 + J_{34}'' S_3 S_4} \tag{4.3}$$

This decimation is a two-step process. First, we decimate S_2.



$$\begin{aligned}\sqrt{C'}e^{h_1'S_1+h_3'S_3+h_4'S_4+J_{13}'S_1S_3+J_{14}'S_1S_4} &= \sum_{S_2=\{0,1\}} e^{\sum_{i=1}^{4}h_iS_i+\sum_{i=2}^{4}J_{1i}S_1S_i} \\ &= e^{h_1S_1+h_3S_3+h_4S_4+J_{13}S_1S_3+J_{14}S_1S_4} \\ &\quad + e^{h_1S_1+h_2+h_3S_3+h_4S_4+J_{12}S_1+J_{13}S_1S_3+J_{14}S_1S_4}\end{aligned}$$

By enumerating the possible values for $S_1 = \{0,1\}$, $S_3 = \{0,1\}$ and $S_4 = \{0,1\}$, we get these following constraints

$$\begin{aligned}\sqrt{C'} &= 1+e^{h_2} \\ h_1' &= h_1 + \log\frac{1+e^{h_2+J_{12}}}{\sqrt{C'}} \\ h_3' &= h_3 \\ h_4' &= h_4 \\ J_{13}' &= J_{13} \\ J_{14}' &= J_{14}\end{aligned}$$

Now, we sum over S_1 in order to have the desired decimation rule. Using the decimation rule (4.1), we obtain

$$\begin{aligned}\sqrt{C} &= \left(1+e^{h_2}\right)\left(1+e^{h_1}\frac{1+e^{h_2+J_{12}}}{1+e^{h_2}}\right) \\ h_3'' &= h_3 + \log\frac{1+e^{h_2}+e^{h_1+J_{13}}\left(1+e^{h_2+J_{12}}\right)}{1+e^{h_2}+e^{h_1}\left(1+e^{h_2+J_{12}}\right)} \\ h_4'' &= h_4 + \log\frac{1+e^{h_2}+e^{h_1+J_{14}}\left(1+e^{h_2+J_{12}}\right)}{1+e^{h_2}+e^{h_1}\left(1+e^{h_2+J_{12}}\right)} \\ J_{34}'' &= \log\frac{\left[1+e^{h_2}+e^{h_1}\left(1+e^{h_2+J_{12}}\right)\right]\left[1+e^{h_2}+e^{h_1+J_{13}+J_{14}}\left(1+e^{h_2+J_{12}}\right)\right]}{\left[1+e^{h_2}+e^{h_1+J_{13}}\left(1+e^{h_2+J_{12}}\right)\right]\left[1+e^{h_2}+e^{h_1+J_{14}}\left(1+e^{h_2+J_{12}}\right)\right]}\end{aligned}$$

4.4 Decimation of three nodes

This rule involves decimating out three nodes, first S_5, then S_2 and finally S_1, as shown in Figure 4.4. In order for the two systems to obey the same Boltzmann distribution, the following equations must hold

$$\sum_{S_5,S_2,S_1} e^{\sum_{i=1}^{5}h_iS_i+\sum_{i=2}^{4}J_{1i}S_1S_i+J_{35}S_3S_5+J_{45}S_4S_5} = \sqrt{C_1}\sqrt{C_2}\sqrt{C_3}e^{h_3''S_3+h_4''S_4+J_{34}''S_3S_4} \qquad (4.4)$$

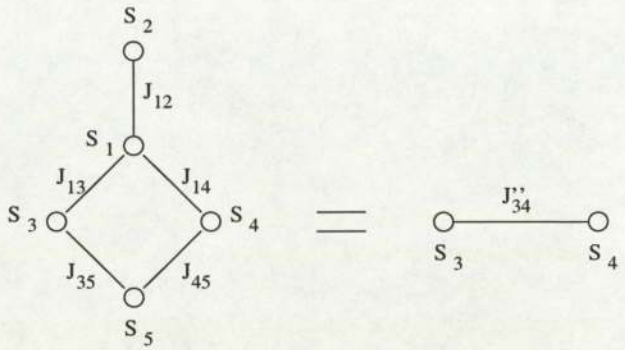
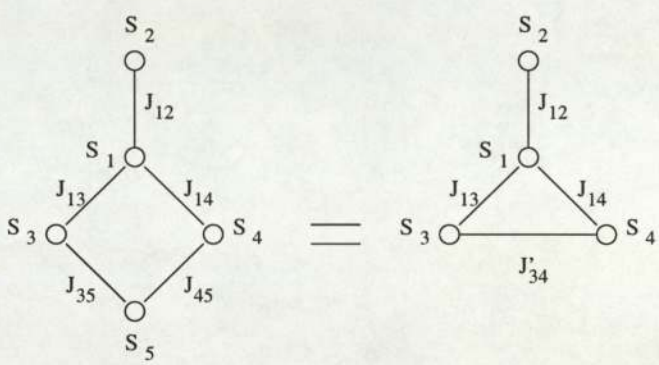


Figure 4.4: Decimation of three nodes

This decimation is a three-step process. First, we start summing over S_5. Using the decimation rule (4.1), we know that the decimation of S_5 does not affect neither the biases h_1 and h_2 nor the connections J_{12}, J_{13} and J_{14}. However, the biases h_3 and h_4 change and a new connection appears between S_3 and S_4.



These networks are required to have the same Boltzmann distribution

$$\sum_{S_5} e^{\sum_{i=1}^{5} h_i S_i + \sum_{i=2}^{4} J_{1i} S_1 S_i + J_{35} S_3 S_5 + J_{45} S_4 S_5} = \sqrt{C_1} e^{\sum_{i=1}^{2} h_i S_i + h_3' S_3 + h_4' S_4 + \sum_{i=2}^{4} J_{1i} S_1 S_i + J_{34}' S_3 S_4}$$

We obtain the following expressions for the constant $\sqrt{C_1}$, the new biases h_3', h_4' and the new connection J_{34}'

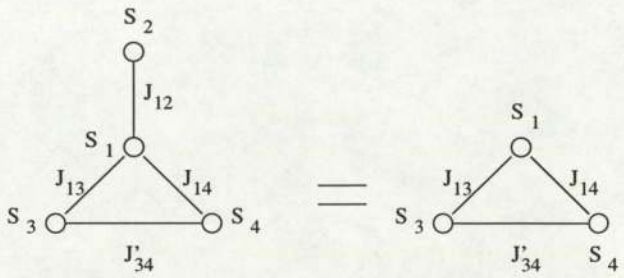
$$\sqrt{C_1} = 1 + e^{h_5} \tag{4.5}$$

$$h_3' = h_3 + \log \frac{1 + e^{h_5 + J_{35}}}{\sqrt{C_1}} \tag{4.6}$$

$$h_4' = h_4 + \log \frac{1 + e^{h_5 + J_{45}}}{\sqrt{C_1}} \tag{4.7}$$

$$J_{34}' = \log \sqrt{C_1} \frac{1 + e^{h_5 + J_{35} + J_{45}}}{(1 + e^{h_5 + J_{35}})(1 + e^{h_5 + J_{45}})} \tag{4.8}$$

Now, let us decimate the node S_2



The two systems are required to obey the same Boltzmann distribution

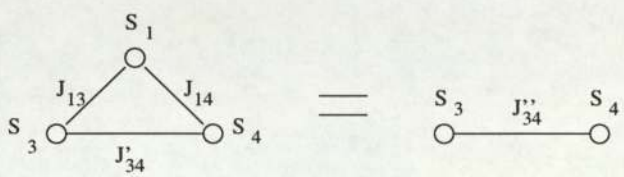
$$\sum_{S_2} e^{\sum_{i=1}^{2} h_i S_i + h_3^{'} S_3 + h_4^{'} S_4 + \sum_{i=2}^{4} J_{1i} S_1 S_i + J_{34}^{'} S_3 S_4} = \sqrt{C_2} e^{h_1^{'} S_1 + h_3^{'} S_3 + h_4^{'} S_4 + J_{13} S_1 S_3 + J_{14} S_1 S_4 + J_{34}^{'} S_3 S_4}$$

Using the decimation rule (4.2), we know that the bias of S_1 will alter, the other biases and connections will remain unchanged. Thus the constant $\sqrt{C_2}$ and the new bias for S_1 are

$$\sqrt{C_2} = 1 + e^{h_2} \tag{4.9}$$

$$h_1^{'} = h_1 + \log \frac{1 + e^{h_2 + J_{12}}}{\sqrt{C_2}} \tag{4.10}$$

We decimate S_1 in order to get the desired decimation rule



In order for the two systems to obey the same Boltzmann distribution, the following equations must hold

$$\sum_{S_1} e^{h_1^{'} S_1 + h_3^{'} S_3 + h_4^{'} S_4 + J_{13} S_1 S_3 + J_{14} S_1 S_4 + J_{34}^{'} S_3 S_4} = \sqrt{C_3} e^{h_3^{''} S_3 + h_4^{''} S_4 + J_{34}^{''} S_3 S_4}$$

By taking into account the possible values for $S_3 = \{0, 1\}$ and $S_4 = \{0, 1\}$, we obtain

$$\sqrt{C_3} = 1 + e^{h_1^{'}} \tag{4.11}$$

$$h_3^{''} = h_3^{'} + \log \frac{1 + e^{h_1^{'} + J_{13}}}{\sqrt{C_3}} \tag{4.12}$$

$$h_4^{''} = h_4^{'} + \log \frac{1 + e^{h_1^{'} + J_{14}}}{\sqrt{C_3}} \tag{4.13}$$

$$J_{34}^{''} = J_{34}^{'} + \log \sqrt{C_3} \frac{1 + e^{h_1^{'} + J_{13} + J_{14}}}{\left(1 + e^{h_1^{'} + J_{13}}\right)\left(1 + e^{h_1^{'} + J_{14}}\right)} \tag{4.14}$$

Finally, we have :

from (4.5), $\sqrt{C_1} = 1 + e^{h_5}$

from (4.9), $\sqrt{C_2} = 1 + e^{h_2}$

from (4.10) and (4.11), $\sqrt{C_3} = 1 + e^{h_1} \frac{1 + e^{h_2 + J_{12}}}{1 + e^{h_2}}$

from (4.6), (4.10) and (4.12)

$$h_3'' = h_3 + \log \frac{1 + e^{h_5+J_{35}}}{\sqrt{C_1}} + \log \frac{\sqrt{C_2} + e^{h_1+J_{13}} \left(1 + e^{h_2+J_{12}}\right)}{\sqrt{C_2}\sqrt{C_3}}$$

from (4.7), (4.10) and (4.13)

$$h_4'' = h_4 + \log \frac{1 + e^{h_5+J_{45}}}{\sqrt{C_1}} + \log \frac{\sqrt{C_2} + e^{h_1+J_{14}} \left(1 + e^{h_2+J_{12}}\right)}{\sqrt{C_2}\sqrt{C_3}}$$

from (4.8), (4.10) and (4.14)

$$\begin{aligned} J_{34}'' &= \log \sqrt{C_1} \frac{1 + e^{h_5+J_{35}+J_{45}}}{\left(1 + e^{h_5+J_{35}}\right)\left(1 + e^{h_5+J_{45}}\right)} \\ &\quad + \log \sqrt{C_3}\sqrt{C_2} \frac{\sqrt{C_2} + e^{h_1+J_{13}+J_{14}} \left(1 + e^{h_2+J_{12}}\right)}{\left(\sqrt{C_2} + e^{h_1+J_{13}} \left(1 + e^{h_2+J_{12}}\right)\right)\left(\sqrt{C_2} + e^{h_1+J_{14}} \left(1 + e^{h_2+J_{12}}\right)\right)} \end{aligned}$$

4.5 Heuristic node-removal scheme

For the recursive node-elimination and decimation methods to be useful in practice, it is necessary to find a way of removing the nodes such that the densely connected configuration of the original image is reduced to a simple subgraph which can be decimated using the above rules. The partition function of the simplified graph can then be found in polynomial time. Before proposing a method for pruning the nodes, we adopt a simple numbering of pixels by assigning the sequence number $\tau = i + m(j-1)$ to the pixel (i, j) where the image is considered as a square array of pixels $\{(i, j), 1 \leq i, j \leq m\}$ and m is the size of the image. With this scheme counting the nodes column by column from node 1 to node $n = m^2$, starting in the upper left, we then use the following heuristic to prune a 32×32 image, as shown in Figure 4.5

1. For the columns $c = \{2, 5, 8, 11, 14, 17, 20, 23, 26, 29\}$, we suppress the nodes number $m(c-1) + 2t + 1$, where $t = 0, \ldots, 15$ for even columns and $t = 1, \ldots, 15$ for odd columns.

2. For the columns $c = \{3, 6, 9, 12, 15, 18, 21, 24, 27, 30\}$, we eliminate the nodes number $m(c-1)+2t$, where $t = 1, \ldots, 15$ for odd columns and $t = 1, \ldots, 16$ for even columns.

3. for the column $c = 32$, we remove the nodes number $m * (c-1) + 2t + 1$, with $t = 1, \ldots, 15$

The black nodes represent those which are eliminated after the recursive procedure, the dotted lines symbolise the links removed between the deleted nodes and theirs neighbours and the white nodes are those which remain in the image. The number of suppressed nodes is 325 for a total number of 1024. This heuristic is fast and very straightforward to implement: we apply the rules (4.3) and (4.4) so as to turn the reduced grid into a chain graph for which we eventually compute the partition function by using rule (4.2).

We believe that this heuristic is almost optimal in the sense that we have attempted to discover other schemes but have wound up finding that the number of eliminated nodes is always greater than 325, which is less than one third of the total number of nodes (1024). Nijman and Kappen [18]

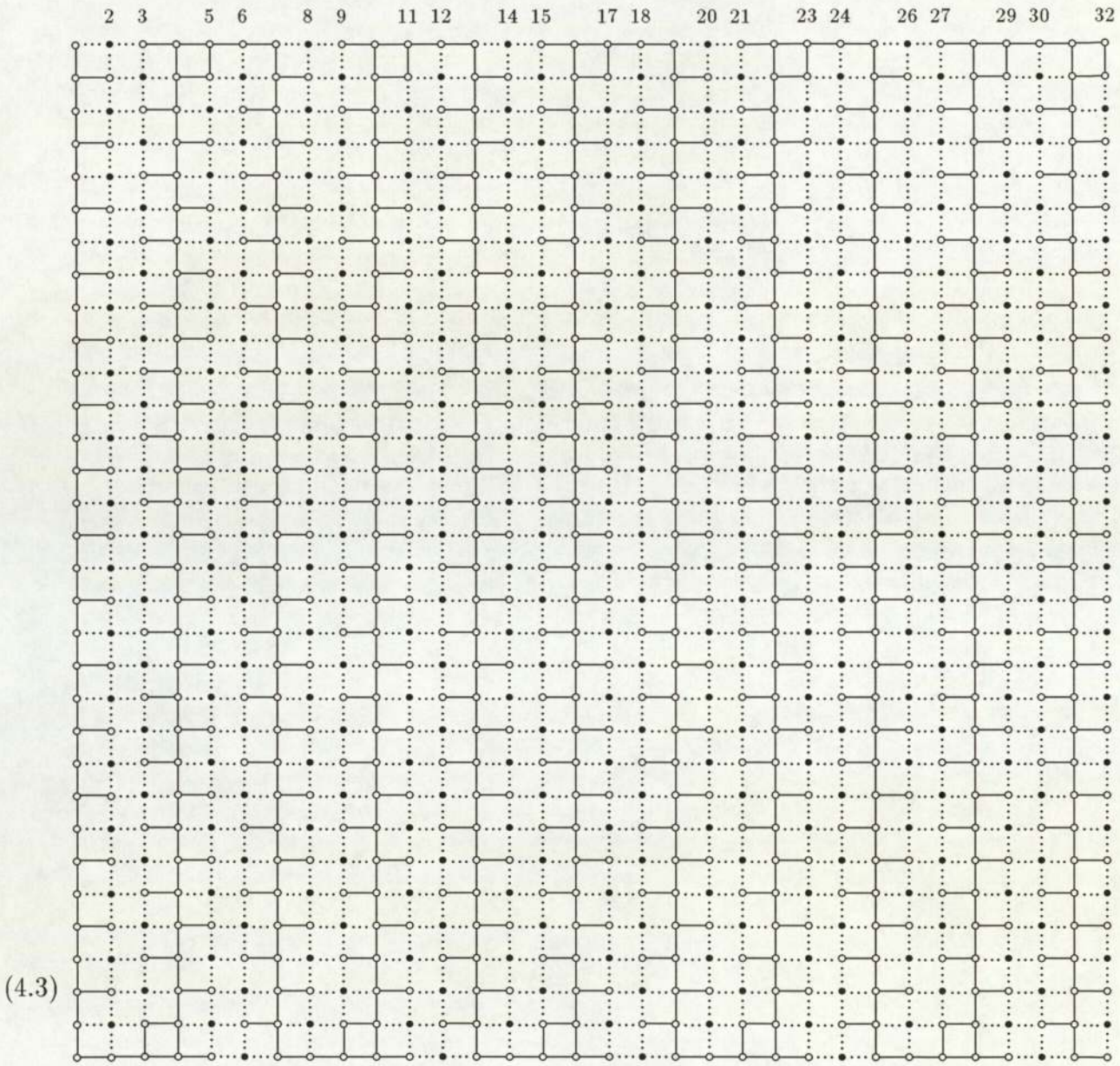


Figure 4.5: Resulting configuration of the image after the recursive procedure. The black nodes represent the eliminated ones, the dotted lines symbolise the links removed between the deleted nodes and theirs neighbours and the white pixels are those which remain in the image. The figures on the top number the columns of the grid. The decimation rules (4.3) and (4.4) are applied in order to obtain a chain graph for which the partition function is calculated using the rule (4.2)

propose, for a 12 by 12 grid, another heuristic that leads to removing from 46 to 48 nodes, which represents about one third of the total number of nodes. Throughout this project, the main question concerns the way the nodes have to be suppressed in order to approximate the partition function accurately. We have faced two approaches: the first one aims at removing the smallest number of nodes, and the second one consists of taking account of the importance of the positions of the pixels we choose to eliminate. This second approach is quite hard to achieve. We have seen so far that, although the connection matrix remains unchanged, the bias vector is modified after each recursion, and consequently, we have to select the "right" nodes to remove. In other words, if we want to maximise the lower bound, then from (3.23), we delete the nodes which have the largest biases, and conversely, if we want to minimise the upper bound, then from (3.35), we eliminate those which have the smallest biases. Nijman and Kappen [18] have attempted to investigate the way of suppressing the right nodes but have not proved whether this leads to a good approximation of the partition function. Furthermore, such a scheme would be quite intricate to program and our main purpose is to eliminate the smallest set of pixels in such a way that the decimation operations are easy and fast to implement. Nevertheless, only the combination of these two approaches is likely to yield a good approximation of the partition function and this issue is left for future study.

Chapter 5

Parameter estimation

So far we have seen that restored images depend strongly on the prior parameters κ and β which control the quality of the restoration scheme. We have also assumed that the corrupted image is generated from the original one which, in reality, we do not know and consequently we may have little idea of appropriate values for κ and β. Thus the problem of parameter estimation turns out to be fundamental to image reconstruction and has been extensively studied over the last few years.

5.1 Estimation using bounding techniques

We have seen in Chapter 1 that the key technique to computing these parameters takes the form of maximising the product of the evidence $P(D|\kappa,\beta)$ and the prior $P(\kappa,\beta)$. As we may have little idea of suitable values for these parameters, we would choose $P(\kappa,\beta)$ to be insensitive to the values of κ and β. In other words, we would consider a prior which in some sense gives equal probability to all possible values. The approach to determining κ and β then amounts to maximising the evidence which we have seen can be approximated using the recursive node-elimination and decimation methods presented in Chapters 3 and 4. Having determined the lower bound in (3.1) and the upper bound in (3.2) of the evidence, we can plot the bounded log-evidence[1] versus κ and β, with κ ranging from 1.0 to 3.0 and β ranging from 0.15 to 0.95, as shown in Figure 5.1. This choice of parameter ranges stems from the results in Chapter 2.

Unfortunately, Figure 5.1 shows that the maximum of the evidence cannot be determined and thus we cannot estimate the MAP values for κ and β given a flat prior. The large gap between the lower and upper bounds stems from the inaccuracy of the upper bound methodology presented in Section 3.2.3. We have seen that a tight upper bound is much more difficult to achieve and we have resorted to a trivial linear upper bound that involves no optimisation. This is potentially the reason for the difficulty in estimating the restoration parameters based on the bounded evidence.

[1] In this chapter, we deal with the noisy image presented in row (b) in Figure 2.3 in Chapter 2.

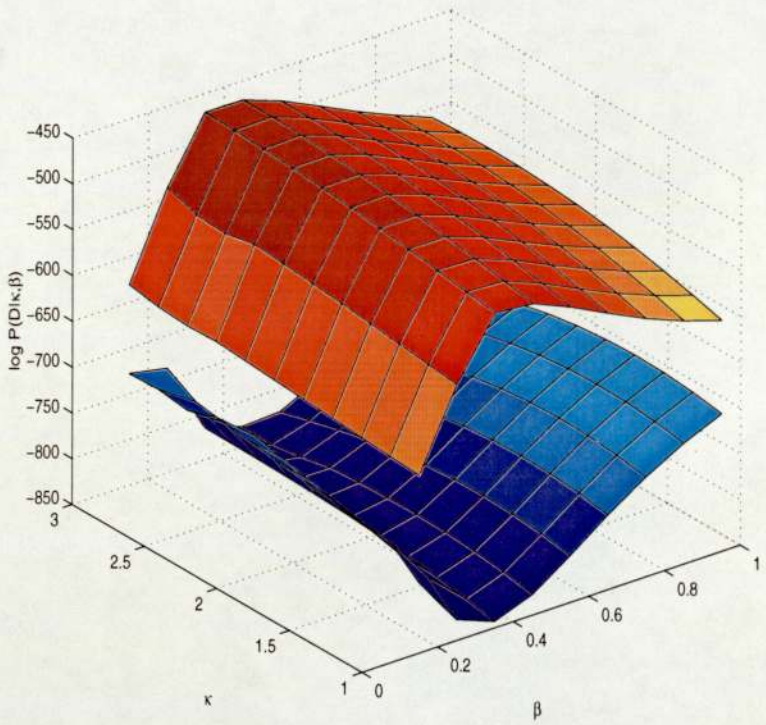


Figure 5.1: The bounded log-$P(D|\kappa,\beta)$ with respect to κ and β

5.2 Estimation using thermodynamic integration methods and mean field approximations

We have developed a method for analytically bounding the evidence $P(D|\kappa,\beta)$. We now present a different method [8] based on Monte Carlo techniques for approximating $P(D|\kappa,\beta)$. Appealing back to the formula (1.13), the evidence is

$$P(D|\kappa,\beta) = \frac{Z_r(\kappa,\beta)}{Z_l(\kappa)Z_p(\beta)}$$

it can also be written

$$\begin{aligned}
\log P(D|\kappa,\beta) &= \int_0^\beta \frac{\partial \log P(D|\kappa,\beta)}{\partial\beta} d\beta + \log P(D|\kappa,0) \\
&= \int_0^\beta \frac{\partial \log Z_r(\kappa,\beta)}{\partial\beta} d\beta - \int_0^\beta \frac{\partial \log Z_l(\kappa)}{\partial\beta} d\beta - \int_0^\beta \frac{\partial \log Z_p(\beta)}{\partial\beta} d\beta \\
&\quad + \log P(D|\kappa,0) \\
&= \int_0^\beta \frac{\partial \log Z_r(\kappa,\beta)}{\partial\beta} d\beta - \int_0^\beta \frac{\partial \log Z_p(\beta)}{\partial\beta} d\beta \\
&\quad + \log Z_r(\kappa,0) - \log Z_l(\kappa) - \log Z_p(0)
\end{aligned}$$

Retaining the forms (1.5), (1.7), (1.9) and noting that $Z_r(\kappa,0) = Z_l(\kappa)$, we obtain

$$\log P(D|\kappa,\beta) = \int_0^\beta d\beta \left\langle \sum_{i\backsim j} I\left[S_i = S_j\right] \right\rangle_{P(S|D,\kappa,\beta)} - \log Z_p(\beta) \tag{5.1}$$

where $\langle . \rangle_{P(S|D,\kappa,\beta)}$ means averaging with respect to the distribution $P(S|D,\kappa,\beta)$ defined in (1.8).

Computing the one-dimensional integral $I(\beta) = \int_0^\beta d\beta \left\langle \sum_{i \backsim j} I[S_i = S_j] \right\rangle_{P(S|D,\kappa,\beta)}$ exactly turns out to be intractable, we therefore resort to using Monte Carlo methods to sample from $P(S|D,\kappa,\beta)$ over a range of values of β. In particular, we use the Swendsen-Wang algorithm to compute this integral for $\beta = \{0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95\}$. We numerically compute $I(\beta)$ by splitting the interval $[0, \ldots, 0.95]$ into 95 equally spaced points. For each point, we generate six image samples for which we estimate $\sum_{i \backsim j} I[S_i = S_j]$ we average to obtain $\left\langle \sum_{i \backsim j} I[S_i = S_j] \right\rangle$. We then perform trapezoidal integration to obtain $I(\beta)$.

Consider now the logarithm of the partition function $Z_p(\beta)$ in (5.1). There are two methods for computing it. The first one is to use the same idea as the one which has enabled us to calculate $\log P(D|\kappa,\beta)$. It consists of integrating $\log Z_p(\beta)$ over β

$$\begin{aligned} \log Z_p(\beta) &= \int_0^\beta \frac{\partial \log Z_p(\beta)}{\partial \beta} d\beta + \log Z_p(0) \\ &= \int_0^\beta d\beta \left\langle \sum_{i \backsim j} I[S_i = S_j] \right\rangle_{P(S|\beta)} + \log Z_p(0) \end{aligned} \tag{5.2}$$

where $\langle . \rangle_{P(S|\beta)}$ means averaging with respect to the distribution $P(S|\beta)$ defined in (1.6). Again we use the same technique which has enabled us to evaluate the integral in (5.1). Figure 5.2 plots $\log Z_p(\beta)$ as well as its lower and upper bounds.

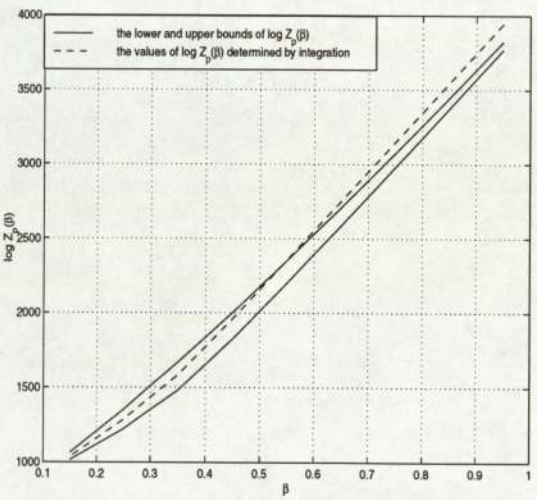


Figure 5.2: The log-$Z_p(\beta)$ obtained by thermodynamic integration methods (dashed curve) with respect to its lower and upper bounds (solid curves)

We note that the values of $\log Z_p(\beta)$ are no longer accurate when β is greater than 0.55. We have performed this calculation several times but have wound up finding the same inaccurate results. The main reason for which these results obtained by Monte Carlo techniques are inaccurate is that it may be difficult to generate a sequence of vectors representative of the distribution $P(S|\beta)$. It is probably

the reason for which Pryce and Bruce [8], who appealed to thermodynamic integration methods for computing the same kind of partition functions as $Z_p(\beta)$, resorted to mean field approximations to calculate $Z_p(\beta)$.

Adopting the idea of mean field approximations proposed by Pryce and Bruce [8], we prefer to use another mean field approach which is equivalent to Pryce and Bruce's approach but is easier to apply. Developed in the graphical models context, this mean field approach [13, 14] is equivalent to the recursive node-elimination methodology if all the nodes are removed. Indeed, consider the Boltzmann form for the Markov random field prior $P(S)$ defined in (1.6), $P(S) = B(S)/Z_p$ where $Z_p(\beta) = \sum_S B(S)$ and the Boltzmann factor $B(S)$ is

$$B(S) = \exp\left\{\sum_{i=1}^{n} h_i S_i + \frac{1}{2}\sum_{i,j=1}^{n} J_{ij} S_i S_j\right\}$$

For any approximating distribution $Q(S)$, we can form the lower bound [13, 14]

$$\begin{aligned} \log Z_p &= \log \sum_S B(S) \\ &= \log \sum_S Q(S)\frac{B(S)}{Q(S)} \\ &\geq \sum_S Q(S) \log \frac{B(S)}{Q(S)} \end{aligned} \tag{5.3}$$

where the last line follows from Jensen's inequality. Note that the difference between the left and right-hand side of (5.3) is the Kullback-Leibler divergence

$$\mathrm{KL}(Q||P) = \sum_S Q(S) \log \frac{Q(S)}{P(S)} \tag{5.4}$$

The bound in (5.3) is valid for any probability distribution $Q(S)$. Consider the factorized distribution [14]

$$Q(S) = \prod_{i=1}^{n} \theta_i^{S_i} (1-\theta_i)^{1-S_i} \tag{5.5}$$

in which the nodes S_i appear as independent Bernoulli variables with adjustable means θ_i. A mean field approximation is obtained by substituting the factorized distribution in (5.5) for the Boltzmann distribution $P(S)$. The best approximation of (5.5) is found by choosing the mean values, $\{\theta_i\}_{1\leq i\leq n}$, that minimise the Kullback-Leibler divergence, $\mathrm{KL}(Q||P)$. This is equivalent to minimising the gap between $\log Z_p$ and the lower bound obtained from mean field theory. From (5.3) and (5.5), the mean field lower bound is

$$\mathcal{L}_{MF} = \sum_i h_i \theta_i + \frac{1}{2}\sum_{i,j} J_{ij}\theta_i\theta_j - \sum_i [\theta_i \log \theta_i + (1-\theta_i)\log(1-\theta_i)] \tag{5.6}$$

As we remember that the connection matrix J is symmetric, we obtain the following mean field equation by setting the gradient of (5.6) equal to zero

$$\theta_i = \sigma\left(h_i + \sum_j J_{ij}\theta_j\right) \tag{5.7}$$

where $\sigma(.)$ is the sigmoid function, $\sigma(z) = 1/(1+e^{-z})$. The mean field bound $\mathcal{L}_{MF}$ is then obtained by iterating the mean field equations in (5.7). However, a tight bound is hard to achieve because we have to optimise a m^2-dimensional vector θ introduced in (5.6), where m is the size of the image. We have seen previously that optimising in a high dimensional space does not always lead to accurate results and that the amount of computational effort is beyond our current means. In fact, the mean field approach is equivalent to the recursive node-elimination technique: each lower bound recursion is a mean field approximation of the eliminated node. Indeed, consider the Boltzmann form for the prior $P(S)$ defined in (1.6), we can write

$$P(S_1, \ldots, S_n) = P(S_k | \{S_j, j \neq k\}) P(S_1, \ldots, S_{k-1}, S_{k+1}, \ldots, S_n)$$

where $P(S_k | \{S_j, j \neq k\})$ is the conditional probability distribution for a node S_k

$$P(S_k | \{S_j, j \neq k\}) = \frac{\exp\left(h_k S_k + \sum_j J_{kj} S_k S_j\right)}{1 + \exp\left(h_k + \sum_j J_{kj} S_j\right)}$$

A mean field approximation is obtained by replacing $P(S_k | \{S_j, j \neq k\})$ (which may be quite difficult to compute) by a distribution $Q_k(S_k)$ (which may be much simpler). We can choose $Q_k(S_k)$ to be

$$Q_k(S_k) = \theta_k^{S_k} (1-\theta_k)^{1-S_k}$$

We then minimise the Kullback-Leibler divergence $\mathrm{KL}(Q_k(S_k) || P(S_k | \{S_j, j \neq k\}))$

$$\begin{aligned}
\mathrm{KL}(Q_k(S_k) || P(S_k | \{S_j, j \neq k\})) &= \sum_{S_k} Q_k(S_k) \log \frac{Q_k(S_k)}{P(S_k | \{S_j, j \neq k\})} \\
&= \theta_k \log \theta_k + (1-\theta_k) \log(1-\theta_k) - \theta_k \left(h_k + \sum_j J_{kj} S_j\right) \\
&\quad + \log\left(1 + e^{h_k + \sum_j J_{kj} S_j}\right)
\end{aligned}$$

This amounts to minimising the difference between the left and right-hand side of the following equation

$$\log\left(1 + e^{h_k + \sum_j J_{kj} S_j}\right) \geq \theta_k \left(h_k + \sum_j J_{kj} S_j\right) - \theta_k \log \theta_k - (1-\theta_k) \log(1-\theta_k) \qquad (5.8)$$

Equation (5.8) is equivalent to (3.18). Thus we have shown that the lower bound recursion is basically the mean field approximation of each node removed.

Let us now return to our problem. We have seen in Chapter 3 that each recursive node elimination translates into an additional bound and consequently the approximation of the bound deteriorates with the number of such recursions. We therefore believe that the mean field bound for Z_p could be worse than the lower bound we have calculated by only suppressing 325 nodes. However, provided we set the parameters θ_i to values which make the mean field bound as accurate as possible, we can choose the mean field bound to be the lower bound of Z_p.

Figure 5.3 shows the log-evidence $P(D|\kappa, \beta)$ as a function of κ and β, computed using the expression in (5.1). The evidence maximum corresponds to $\kappa \simeq 3.0$ and $\beta \simeq 0.35$. Figure 5.4 shows that the log-evidence is between its lower and upper bounds. Note that this is practically an important piece of information as it enables us to be more confidant that the Monte Carlo estimate of the evidence is accurate, and highlights an important potential of the bounding techniques.

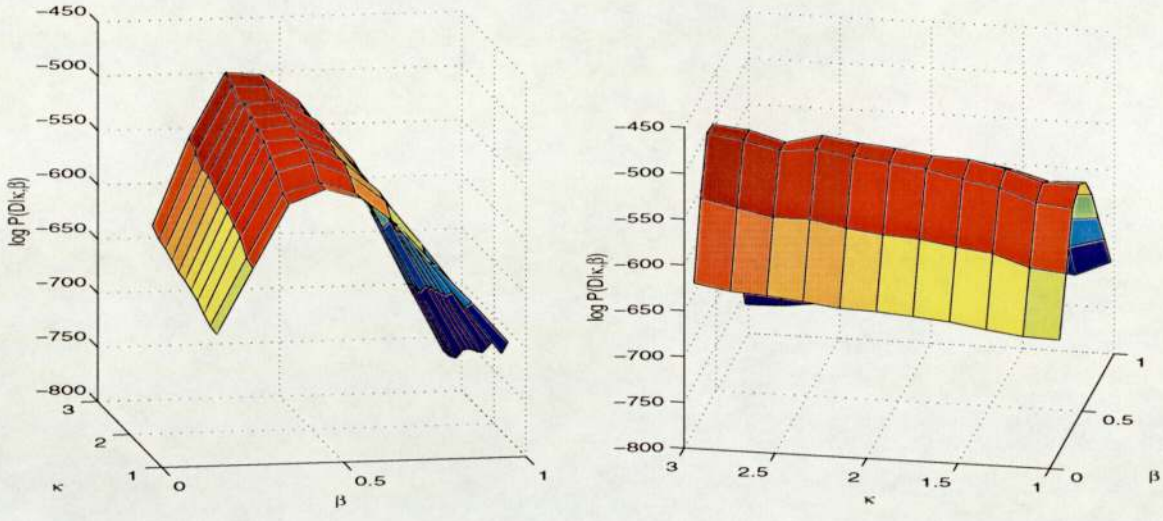


Figure 5.3: The log-$P(D|\kappa, \beta)$ as a function of κ and β, computed using thermodynamic integration methods and mean field approximations, for two different views. The evidence maximum corresponds to $\kappa \simeq 3.0$ and $\beta \simeq 0.35$

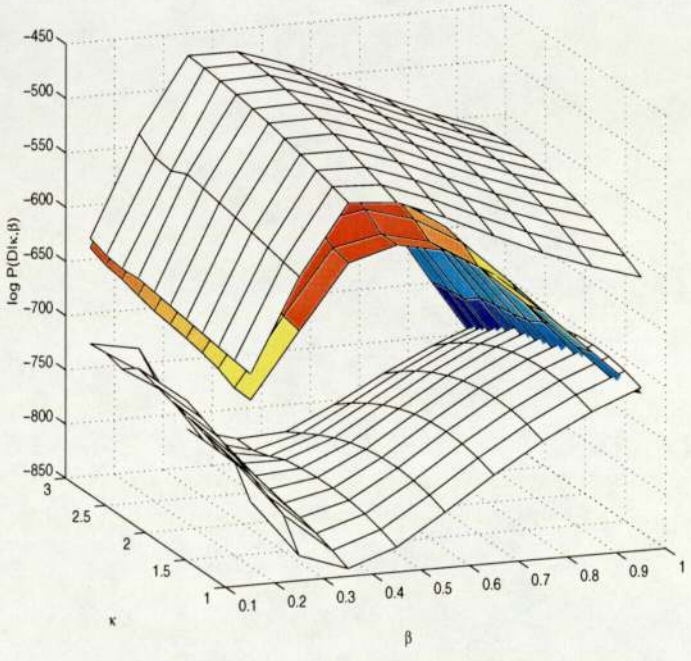


Figure 5.4: The log-$P(D|\kappa, \beta)$ obtained by thermodynamic integration methods and mean field approximations (coloured surface) with respect to its lower and upper bounds (white surfaces)

5.3 Prior determination

We have estimated the restoration parameters by considering that the prior $P(\kappa, \beta)$ gives equal probability to the values of κ and β. We now consider the influence of $P(\kappa, \beta)$ on the reconstructions.

Consider first the prior $P(\beta)$. Higdon [12] estimates the parameter ranges by computing for different values of β the expected energy $\langle V(S) \rangle = \left\langle \sum_{i \backsim j} I[S_i = S_j] \right\rangle$ with respect to the Markov random field prior $P(S|\beta)$ defined in (1.6), where S is an uncorrupted image

$$\langle V(S) \rangle = \sum_{S} \sum_{i \backsim j} I[S_i = S_j] \, P(S|\beta) \tag{5.9}$$

The aim is to determine the parameter range for which the mean energy $\langle V(S) \rangle$ becomes maximal and stable. In other words, we have to determine the values of β which are near the critical region where $\langle V(S) \rangle$ varies drastically. The intuitive meaning of $\langle V(S) \rangle$ is that the image contains large single coloured patches when $\langle V(S) \rangle$ is stable and maximal. Conversely, in the critical region, where $\langle V(S) \rangle$ is unstable, we have a patchy image with speckles. It is only near the critical region that we have large regular patches of black and white. Thus any regular image will favour β near the critical region.

Computing (5.9) is not possible explicitly and therefore requires Monte Carlo samples be obtained from the prior $P(S|\beta)$ over a set of values of β. We use the Swendsen-Wang algorithm so as to compute $\langle V(S) \rangle$ for 22 values of β in the range $[0.15, 0.95]$. For each value, we generate eight image samples for which we compute the energies we then average to obtain the expected energy.

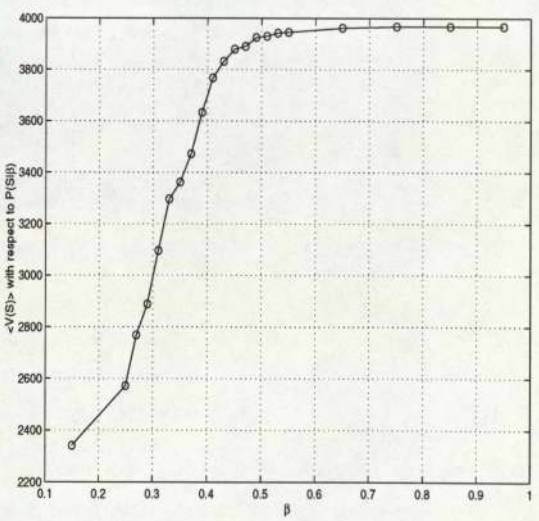


Figure 5.5: The expected energy $\langle V(S) \rangle$ with respect to β. The curve changes drastically in the critical region $[0.25, 0.50]$ where the energy is unstable. It becomes maximal and stable from 0.5 where regular images are likely to be obtained

Figure 5.5 shows a steep slope in the range $[0.25, 0.50]$. It is exactly in this critical region that the images are very sensitive to small changes in β. The energy is only stable from 0.5, which means that

regular and smooth realisations are likely to occur. Besides, this confirms the results in Chapter 2 where we have seen that small values of β lead to speckled restorations. We therefore need to specify a prior that favours the values of β near the critical region and disregards small values and those which lie in the critical region. Moreover, we wish to discard overly large values ($\beta > 0.9$) because the results in Chapter 2 suggest that the images are patchy with large single coloured patches. Thus we choose a gamma distribution[2] $\gamma(\beta, \alpha, \lambda)$ which favours regular and smooth realisations around 0.62. Figure 5.6 plots the gamma distribution for $\alpha = 373$ and $\lambda = 1.689e - 3$.

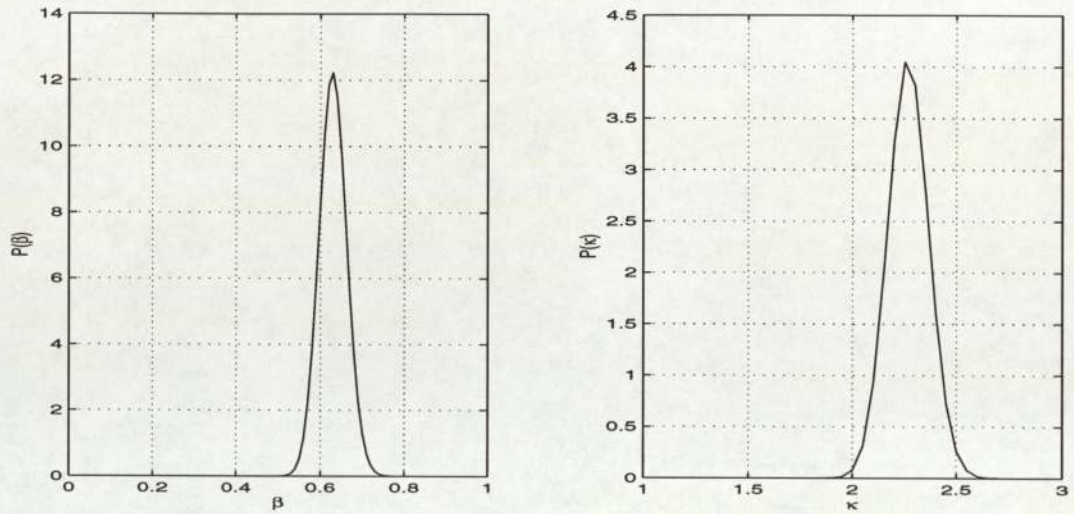


Figure 5.6: The gamma distribution $\gamma(\beta, \alpha, \lambda)$ for β (left), with $\alpha = 373$ and $\lambda = 1.689e - 3$. The gamma distribution $\gamma(\kappa, \alpha, \lambda)$ for κ (right), with $\alpha = 540$ and $\lambda = 0.0042$. $\gamma(\beta, \alpha, \lambda)$ favours regular and smooth realisations around 0.62 and $\gamma(\kappa, \alpha, \lambda)$ favours regular images around 2.3 (corresponding to a noise level $q = 0.24$

Let us tackle now the prior $P(\kappa)$ for the parameter κ. The results in Chapter 2 show that κ exerts less effect on the image restoration than β. A choice of a uniform distribution for $P(\kappa)$ means however that each value of κ has an equal probability on the restorations, which is not the case. Let us embody the qualitative assumptions about $P(\kappa)$ in terms of noise level q defined in (1.3). We believe that, in practice, we do not restore images which are very noisy or a little degraded. In other words, the degraded images may have a noise level ranging from 0.18 to 0.30 (we have seen in Chapter 2 that it is hard to restore noisy images with $q = 0.30$). Assume we deal with noisy images having a noise level from 0.18 to 0.30. In Chapter 2, we have seen that small values of the noise level q ($q < 0.18$) lead to speckled restorations. The main reason is that, by using small values of q, we consider that we are restoring images that are not *really* degraded so that some speckles are not regarded as noisy and hence are not cleaned up. Conversely, for large values of q ($q > 0.3$), we consider that we are restoring

[2]The gamma distribution is defined as follows: for $x \in [a, b]$, α and λ two real numbers,

$$\gamma(x, \alpha, \lambda) = \frac{x^{\alpha-1} e^{-x/\lambda}}{\int_a^b x^{\alpha-1} e^{-x/\lambda} dx}$$

very degraded images. Consequently, all black patches are cleaned up and restored images are likely to contain large black and white regions. However, if the original image has some small patches, then overly large values of q are not suitable for the restoration of this kind of image. Therefore, regular realisations tend to occur in the range $[0.18, 0.30]$ for q or $[1.6, 3.0]$ for κ. Thus we choose $P(\kappa)$ to be a gamma distribution $\gamma(\kappa, \alpha, \lambda)$ which favours regular and smooth images around $\kappa = 2.3$, as shown in Figure 5.6.

We have seen in Chapter 1 that the most probable values for κ and β are found by maximising the posterior distribution $P(\kappa, \beta|D)$ defined in (1.12) or the product of the evidence $P(D|\kappa, \beta)$ and the priors $P(\kappa)$ and $P(\beta)$. Figure 5.7, which plots the bounded log-posterior $P(\kappa, \beta|D)$ with respect to κ and β, shows that, unfortunately, it is not possible to determine the parameter ranges where $P(\kappa, \beta|D)$ is maximal because the large gap between the bounds makes the estimation of the peak impossible.

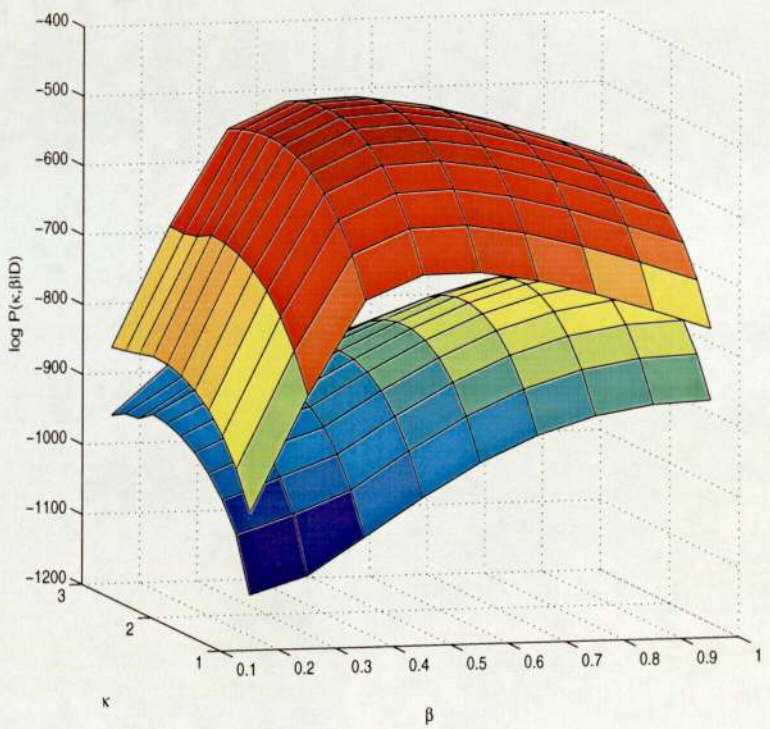


Figure 5.7: The bounded log-posterior $P(\kappa, \beta|D)$ as a function of κ and β

Figure 5.8 plots $\log P(\kappa, \beta|D)$ as a function of κ and β, computed using thermodynamic integration methods and mean field approximations. The maximum occurs for $\kappa \simeq 2.4$ and $\beta \simeq 0.55$, as shown in Figure 5.9. Comparison with Figure 5.3 shows that the most probable values have shifted when the prior $P(\kappa, \beta)$ is taken into account. Figure 5.10 shows that the restored image is speckled when $\kappa = 3.0, \beta = 0.35$. This is not surprising since we have seen that large values of κ and small values of β lead to speckled restorations. However, the image is regular and rather smooth when $\kappa = 2.4, \beta = 0.55$.

In fact, we have seen that regular and smooth images are likely to occur when β is near the critical region and κ around 2.3. When we have determined the priors $P(\kappa)$ and $P(\beta)$, we have favoured consistent images for β around 0.62 and κ around 2.3. We therefore believe that regular and smooth images (restored from the noisy image presented in row (b) in Figure 2.3) occur for β in the range $[0.55, 0.62]$ and $\kappa \simeq 2.4$ corresponding to a noise level $q \simeq 0.23$. Note that, in Chapter 2, we have obtained a relevant restored image for $\kappa = 2.4$ and $\beta = 0.59$.

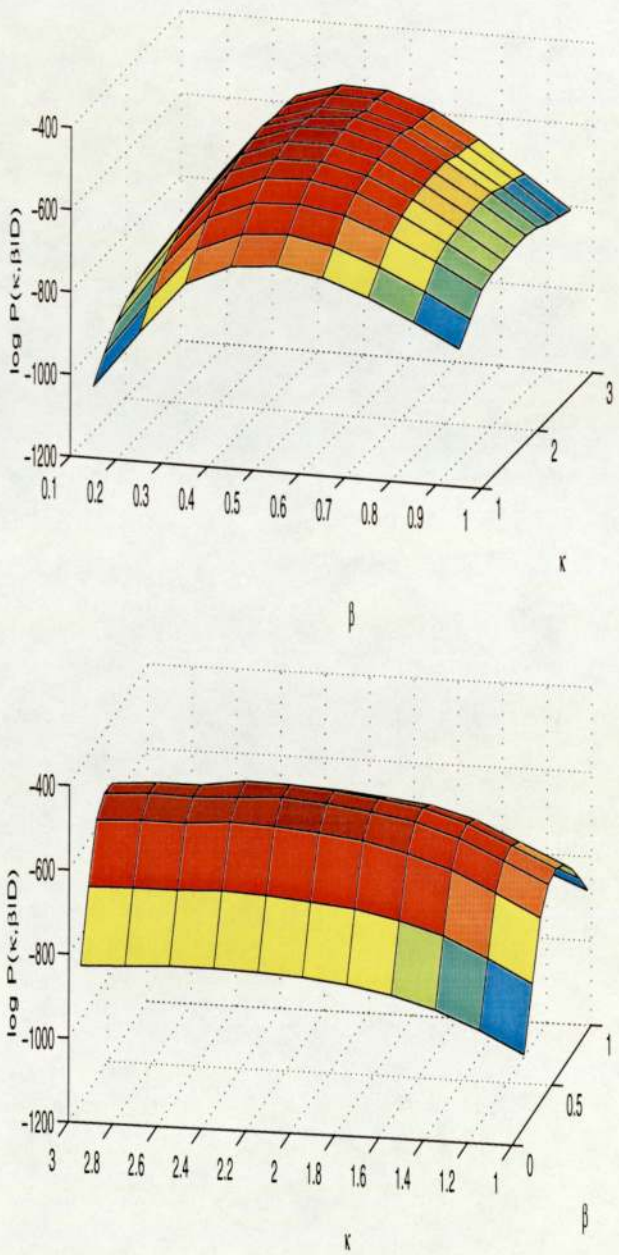


Figure 5.8: The log-posterior $P(\kappa, \beta|D)$ as a function of κ and β, computed using thermodynamic integration methods and mean field approximations, for two different views. The maximum occurs for $\kappa \simeq 2.4$ and $\beta \simeq 0.55$

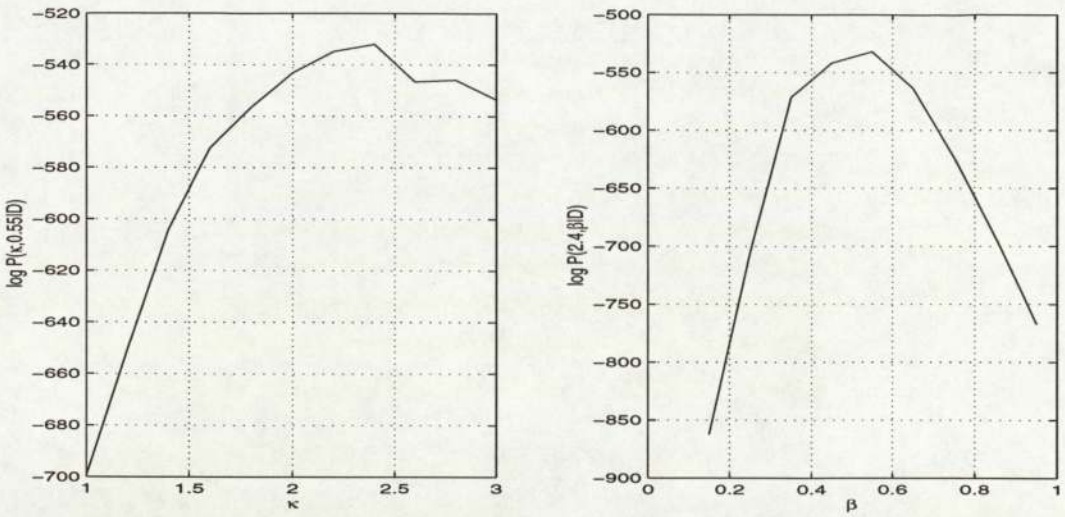


Figure 5.9: This shows the log-posterior $P(\kappa, \beta|D)$ as a function of κ for $\beta = 0.55$ (left), and the log-posterior $P(\kappa, \beta|D)$ as a function of β for $\kappa = 2.4$ (right)

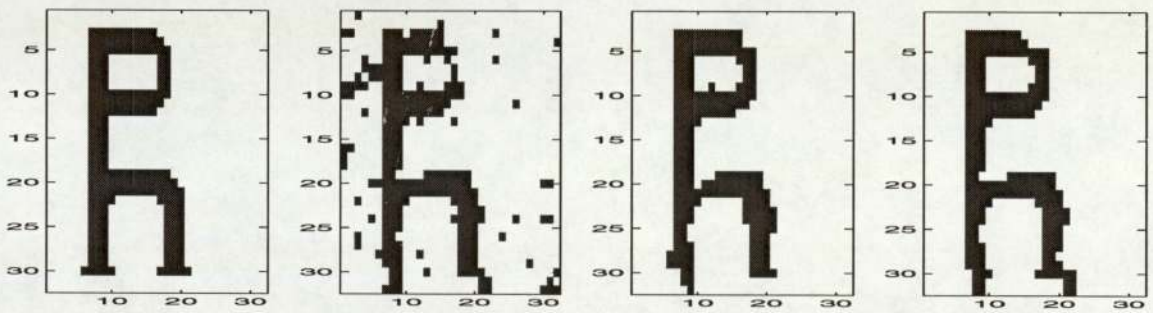


Figure 5.10: The original image (first from the left), the restored image (second) for $\kappa = 3.0, \beta = 0.35$ when $P(\kappa, \beta)$ is disregarded, the restored image (third) for $\kappa = 2.4, \beta = 0.55$ when $P(\kappa, \beta)$ is taken into account, the restored image (fourth) obtained in Chapter 2 for $\kappa = 2.4, \beta = 0.59$

5.4 Comments

The first remark is that the parameter priors have a strong influence on the image restoration process. The information supplied by the evidence does not provide reliable restoration parameters when the prior assumptions about the noise and image generation process are poor. Indeed, when the parameter priors are supposed to exert no effect on the reconstructions, the evidence optimal parameters lead to speckled images, but if they are well matched to reality, consistent images should be obtained.

The second remark is that the bounding techniques do not enable us to determine the restoration parameters. The main reason stems essentially from inaccurate estimates of the upper bound. Indeed, we have resorted to a linear method which involves no optimisation since a consistent upper bound proves to be difficult to achieve. Accurate variational recursive node elimination techniques do not work well as the structure of the pixel graph becomes more complicated with each recursion. This has a detrimental effect on our ability to use exact methods to compute the evidence on the remaining subgraph.

Chapter 6

Conclusion

This project has dealt with the problem of restoring images in a Bayesian framework, in which prior beliefs about the underlying clean image generation process and the noise or corruption process are explicitly assumed. The quality of the resulting restored image depends on the particular prior restoration parameters that are chosen and, typically, these will not be known. It is important, therefore, to have a methodology capable of dealing with this parameter uncertainty. Again, this can be framed within the Bayesian formalism, and requires the computation of normalising constants, or the evidence for some particular restoration parameters.

Unfortunately, the computation of the most likely restoration parameters, by maximising the evidence, is a computationally intractable problem. If this were to be computed in a naive way by simply summing over all the possible image configuration states, then the time complexity of calculating the evidence is exponential in the number of pixels in the image. The main emphasis of this project was to develop and apply novel techniques to be able to estimate this evidence in a controlled manner. Typically, work in the past has used Monte Carlo techniques to approximate this quantity, although this is not highly satisfactory as such approximations are essentially uncontrolled. That is, there is no guarantee that the Monte Carlo estimate of the parameter evidence will be in any way close to the true value.

As an integral part of this project, we have evaluated some of the more common Monte Carlo techniques, in particular the Gibbs and Metropolis algorithms. It was also our aim to evaluate the much less well known Swendsen Wang algorithm which has been shown to be a spectacular improvement on previous algorithms in the statistical physics of Ising models - closely related to the image restoration problem. We have found that the S-W method tends to be a much faster sampling procedure than the two single site update algorithms that we have examined. Indeed, it proves to be at least an order of magnitude more efficient at moving quickly through the state space. However, we have also found that the tendency of the S-W algorithm to bind together and flip clusters of identical pixels can be disadvantageous in restoring the fine details of the image. Perhaps a hybrid approach in which a sequence of SW updates followed by a sequence of Gibbs updates would provide a useful

compromise between efficiency and thoroughness of sampling.

Recently, within the graphical models community, analytic techniques have been applied to solving problems involving the computation of normalising constants. It has been the central aim of this project to see if similar techniques could bear fruit in the image restoration problem. The aim was, rather than to resort to uncontrollable Monte Carlo methods, to rigorously bound the evidence from both above and below. Clearly this is not a trivial problem and it can be shown that even bounding the evidence to a desired accuracy is NP-complete [1]. However, we have successfully implemented a recursive node elimination scheme that preserves both upper and lower bounds on the evidence. The node elimination scheme aims to reduce the pixel graph into a subgraph for which the evidence can be calculated in time polynomial in the number of remaining pixels. For each node removed, the lower bound introduces an extra variational parameter so that, for large images, obtaining the best lower bound potentially involves optimisation over a large dimensional space (of the order of 300 for an image of 32 by 32 pixels). Whilst this is, in principle, an attractive method, we have found that this amount of computational effort is beyond our current means and we have resorted to searching for a suboptimal solution in a lower dimensional space of variational parameters. Indeed, for simplicity, we have chosen to find the best one-dimensional optimal parameter set that gives a lower bound. There is little difficulty for the user to decide, beforehand, roughly the computational effort that is affordable, and then to search for a suboptimal solution for the lower bound that is consistent with these computational requirements.

A tight upper bound proves to be much more difficult to achieve. Variational recursive node elimination techniques do not work well in this case as the structure of the pixel graph changes with each recursion. This has a detrimental effect on our ability to find an exact algorithm to compute the evidence on the remaining subgraph. We therefore resorted to a trivial linear upper bound that involves no optimisation. This is potentially the reason for the later difficulty in estimating the best restoration parameters based on the bounded evidence, since a poor upper bound will make estimating a peak in the evidence impossible. In principle, it is possible that a different approach to bounding the evidence from above which avoids this trivial linear bound could be found. Indeed, a combination of exact and approximate methods may be the way forward. One reason for this is that mean field bounds are expected to work best when the pixel graph is densely connected. For our lower bound, which is essentially a type of mean field bound, the graph is not particularly densely connected, so that there may be improvements possible from more sophisticated bounding techniques. For the upper bound, however, using a variational node elimination technique would increase the connectivity of the remaining subgraph. It should therefore be the case that approximating the remaining evidence on the densely connected subgraph will be well suited to a mean field analysis. The study of such hybrid methods could therefore be very fruitful.

In conclusion, parameter estimation is a hard problem. However, we have shown that the application of bounding techniques to the problem of dealing with uncertainty in the parameters controlling Bayesian image restoration is feasible. A combination of exact and approximate techniques may be

the best way forward since exact bounds from both above and below have proved to be difficult to implement and computationally expensive. Nevertheless, we believe that such bounding approaches are a great improvement of previous estimates based solely on Monte Carlo techniques, and we expect that this will remain an active research area for some time to come.

Bibliography

[1] Karp, R.M. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85-103. Plenum Press, 1972.

[2] Geman, S. and Geman, D. (1984) Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 721-741.

[3] Besag, J. (1986) On the Statistical Analysis of Dirty Pictures, *Journal of the Royal Statistical Society, Series B*, 48: 259-302.

[4] Swendsen, R.H. and Wang, J.S. (1987) Nonuniversal critical dynamics in Monte Carlo simulations, *Physical Review Letters*, 58: 86-88.

[5] Greig, D.M., Porteous, B.T. and Seheult, A.H. (1988) Exact Maximum *A Posteriori* Estimation for Binary Images, *Journal of the Royal Statistical Society, Series B*, 51: 271-279

[6] Dubes, R.C. and Jain, A.K. (1989) Random field models in image analysis, *Journal of Applied Statistics*, Vol. 16, No .2.

[7] Neal, R.M. (1993) Probabilistic Inference Using Markov Chain Monte Carlo Methods, *Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.*

[8] Pryce, J.M. and Bruce, A.D. (1994) Statistical mechanics of image restoration, *Journal of Physics A*, Vol. 28, No. 3.

[9] Bishop, C.M. (1995) Neural Networks for Pattern Recognition, *Oxford University Press*, pp. 406-416.

[10] Saul, L. and Jordan, M.I. (1994) Learning in Boltzmann Trees, *Neural Computation*, 6: 1173-1183.

[11] Saul, L. and Jordan, M.I. (1995) Boltzmann Chains and Hidden Markov Models, *Advances in Neural Information Processing Systems 7. MIT Press.*

[12] Higdon, D.M. (1996) Auxiliary Variable Methods for Markov Chain Monte Carlo with Applications, *MCMC Preprint Service, Department of Mathematics, University of Bristol.*

[13] Saul, L. and Jordan, M.I. (1996) Exploiting Tractable Substructures in Intractable Networks *Advances in Neural Information Processing Systems 8. MIT Press*

[14] Saul, L, Jaakkola, T. and Jordan, M.I. (1996) Mean Field Theory for Sigmoid Belief Networks, *Journal of Artificial Intelligence Research*, 4: 61-76.

[15] Jaakkola, T. and Jordan, M.I. (1996) Computing upper and lower bounds on likelihoods in intractable networks, in *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence.*

[16] Jaakkola, T. and Jordan, M.I. (1996) Recursive algorithms for approximating probabilities in graphical models, *MIT Computational Cognitive Science Technical Report 9604.*

[17] Rüger, S., Weinberger, A. and Wittchen, S. (1996) Decimatable Boltzmann Machines vs. Gibbs Sampling, *Technical Report TR 96-29, Computer Science Department, Technische Universität Berlin.*

[18] Nijman, M.J., and Kappen, H.J. (1996) Efficient Learning in Sparsely Connected Boltzmann Machines, *International Conference on Artificial Neural Networks 1996.*

[19] Titterington, D.M. (1997) Some aspects of Bayesian image analysis, in *Proceedings in The Art and Science of Bayesian Image Analysis, Leeds University Press.*