IMAGE ANALYSIS AND PRENATAL SCREENING

by

JIAN'AN LUAN

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Mathematics and Statistics
Faculty of Technology

August 1998

REFERENCE ONLY

UNIVERSITY OF PLYMOUTH
Item No. 900 3654417
Date 24 SEP 1998 T
Class No. T 618.3207543 LUA
Contl. No. X703746609
LIBRARY SERVICES

90 0365441 7

Abstract

IMAGE ANALYSIS AND PRENATAL SCREENING

JIAN'AN LUAN

Information obtained from ultrasound images of fetal heads is often used to screen for various types of physical abnormality. In particular, at around 16 to 23 weeks' gestation two-dimensional cross-sections are examined to assess whether a fetus is affected by Neural Tube Defects, a class of disorders that includes Spina Bifida. Unfortunately, ultrasound images are of relatively poor quality and considerable expertise is required to extract meaningful information from them. Developing an ultrasound image recognition method that does not rely upon an experienced sonographer is of interest. In the course of this work we review standard statistical image analysis techniques, and explain why they are not appropriate for the ultrasound image data that we have. A new iterative method for edge detection based on a kernel function is developed and discussed. We then consider ways of improving existing techniques that have been applied to ultrasound images. Storvik (1994)'s algorithm is based on the minimisation of a certain energy function by simulated annealing. We apply a cascade type blocking method to speed up this minimisation and to improve the performance of the algorithm when the noise level is high. Kass, Witkin and Terzopoulos (1988)'s method is based on an active contour or 'snake' which is deformed in such a way as to minimise a certain energy function. We suggest modifications to this energy function and use simulated annealing plus iterated conditional modes to perform the associated minimisation. We demonstrate the effectiveness of the new edge detection method, and of the improvements to the existing techniques by means of simulation studies.

Contents

Abstract i

Contents ii

List of tables and illustrations v

Acknowledgement viii

Author's declaration ix

1 Introduction 1

1.1 What is the problem? 1

1.2 Initial idea 3

1.3 Traditional image restoration methods 6

  1.3.1 Filters 8

  1.3.2 Bayesian approach 10

  1.3.3 Gibbs sampler and Markov chain Monte Carlo 15

  1.3.4 Optimisation techniques: simulated annealing and iterated conditional modes 21

  1.3.5 Reconstruction experiment for the noisy image z shown in Figure 1.3 24

1.4 Structure of the thesis 28

2 Object recognition using simulated annealing and ICM with cascade-type blocking 32

2.1 Introduction 32

2.2 Storvik's approach and object recognition 36

2.3 A cascade-type blocking approach and simulated annealing 39

2.4 Simulation study 48
2.5 Application to real data 65
2.6 Discussion 67

3 Edge detection using kernel functions 69

3.1 Definition of the kernel function 69
3.2 The kernel function 72
3.3 The kernel shape detection algorithm 76
3.4 A simulation study 80
3.5 Modification of the kernel function and refinement of the algorithm 83
3.6 Application to real data 90
3.7 Discussion about the use of the kernel algorithm for detecting shapes defined only by their edges 94
3.8 Shape detection using kernel algorithm 97
  3.8.1 The idea 99
  3.8.2 Description of the kernel algorithm applied to shape detection 102
  3.8.3 An attenuated detection function for very noisy images 106
  3.8.4 A simulation study 112
  3.8.5 Simulation study comparing the performance of the modified kernel algorithm with that of the cascade based simulated annealing algorithm study 117
  3.8.6 Application to real data 118
3.9 Further works 120

4 Study of the effect of the kernel parameters 123

4.1 Introduction 123
4.2 The effect of the kernel parameters 125
4.3 A probability study 133
4.4 Conclusion 140

5 Active contour models: the snake model 141

5.1 The description of the active contour model of Kass, Witkin and Terzopoulas (1988) 142
5.2 The behaviour of the snake model 149
5.3 Discussion about snake models 154
  5.3.1 Dynamic programming active contour model 155
  5.3.2 Balloon model 158
5.4 Simulated annealing and ICM applied to the snake model 159
5.5 Simulation study: comparison between the kernel algorithm, the snake algorithm and the simulated annealing plus ICM algorithm 164
5.6 Conclusions 168

Conclusions and suggestions for further work 169

References 171

LIST OF TABLES AND ILLUSTRATIONS

Tables

2.1 Results of applying algorithms with and without cascade and with different cascade temperature schedules to a noisy image 61
2.2 Results of a simulation study for comparing algorithms with and without cascade 63
2.3 Results of a simulation study of edge blocking visiting schedules 64
3.1 Numerical summary of the error made by the estimate using peanut shaped kernel ... 82
3.2 Numerical summary of the error made by the estimate using banana shaped kernel ... 85
3.3 Numerical summary of the error made by the estimate using dynamic parameter θ ... 86
3.4 Numerical summary of the error made by the estimate using dynamic parameters θ, σ_1 and σ_2 90
3.5 Result of the simulation study for the kernel function algorithm 95
3.6 Errors for images with different noise levels and blurring bands 111
3.7 Results of simulation study for the adjust kernel algorithm 114
3.8 Results of simulation study for comparing the performance of algorithms 118
5.1 Final energies of the estimates obtained from the snake algorithm and the simulated annealing plus ICM algorithm 163
5.2 Numerical summary of the number of differing pixels from different algorithms 166
5.3 Mean of difference between algorithms 167

Figures

1.1 Two ultrasound images of cross-section of fetal heads 3
1.2 Ultrasound image of a cross-section of a fetal head and results of applying filters 4
1.3 Schematic representation of a recording and display system 7
1.4 Part of a pixel grid showing the neighbourhood systems 11
1.5 The possible pixel cliques 12
1.6 Reconstruction experiment for a noisy image 25
1.7 Bar plot of the final energies 27
1.8 Ultrasound image of a human ovarian cyst 29

2.1 Storvik's potential function $V_i(x)$ of the gradient function 37
2.2 Illustration of legal blocks in part of a shape 42
2.3 Three cascade temperature schedules 47
2.4 A binary image of head shape and its noisy version corrupted by $N(0, 1)$ 49
2.5 Edge estimate by using the three stages cascade based algorithm 50
2.6 Estimated edge and smoothed estimated edge together with the true edge 52
2.7 Plot of the value of the energy function against update number for Figure 2.5 53
2.8 Edge estimate by using the non-cascade algorithm 55
2.9 Plot of the value of the energy function against update number for Figure 2.8 56
2.10 Result of employing the monotonically decreasing linear temperature schedule 57
2.11 Plot of the value of the energy function against update number for Figure 2.10 58
2.12 Result of employing the cascade stage linear temperature schedule 59
2.13 Plot of the value of the energy function against update number for Figure 2.12 60
2.14 Bar plots of the energies and number of differing pixels 62
2.15 Estimated edge of the human ovarian cyst in ultrasound image 66
3.1 Ultrasound image of a cross-section of a fetal head 70
3.2 An example of a kernel function 74
3.3 Four kernel functions with different parameters 75
3.4 The choice of candidate points 79
3.5 The search procedure 80
3.6 A simulation study performed on a binary image of an ellipse 81
3.7 The choice of the kernel function in relation to the underlying shape 83
3.8 Estimate using banana shaped kernel function 84
3.9 Estimate using banana shaped kernel function and dynamic parameter θ 86
3.10 An example of the effect of the parameters 88
3.11 Estimate using dynamic parameters θ, σ_1 and σ_2 89
3.12 Extracted edge in two real examples 92, 93
3.13 A binary image of head shape and its noisy version corrupted by $N(0, 1)$ 98
3.14 Illustration of the break point detection method in 1-D 99
3.15 Illustration of the change point detection method for noisy function in 1-D 101
3.16 Illustration of the detection function D 103
3.17 Five detectors 103

3.18 Estimate from noisy image shown in Figure 3.13 105
3.19 A noisy image corrupted by $N(0, 3)$ and the estimated edge 105
3.20 Illustration of the blurring detection function in 1–D 107
3.21 Estimate from noisy image shown in Figure 3.19 by using blurring technique 109
3.22 Estimates from noisy image shown in Figure 3.13 by using blurring technique 110
3.23 A binary image of an ellipse used for simulation study 112
3.24 Error plot for ten simulations against the width of blurring band 115
3.25 Estimated edge of the human ovarian cyst 119
3.26 A duck image 120
3.27 Examples of the bent kernel function 122
4.1 A circle image 124
4.2 Boxplots of the number of differing pixels for the different noise levels 126
4.3 Boxplots of the standard number of differing pixels for the different radii of the true circles 126
4.4 Boxplots of the number of differing pixels for different values of σ_0 129
4.5 Boxplots of the number of differing pixels for different values of σ_1 129
4.6 Boxplots of the number of differing pixels for different values of σ_2 131
4.7 Boxplots of the number of differing pixels for different values of σ_2 131
4.8 Plots of the probabilities of correct detection for different values of κ and σ_0 135
4.9 Plots of the probabilities of correct detection for different values of κ and σ_1 137
4.10 Plots of the probabilities of correct detection for different values of κ and σ_2 139
5.1 The results obtained by applying the snake algorithm to the ultrasound image of a fetal head shown in Figure 3.1 151
5.2 The results obtained by calculating the edge energy using the smoothed version of the ultrasound head image 153
5.3 Results from the simulated annealing plus ICM algorithm 162
5.4 An ellipse shape defined by an edge that is two pixels thick and used for a simulation study 165

ACKNOWLEDGEMENT

I am deeply indebted to Dr Julian Stander for his kindly direction, encouragement, assistance and advice. I am particularly grateful to him for his patience in reading through and commenting so helpfully on earlier versions of this manuscript. My sincere thanks also go to Dr David Wright for bouncing ideas about edge detection off me, and for his unfailing interest, help and encouragement.

I also thank Dr David Graham and Dr David Wilton for their help with some of the mathematics.

My special thanks go to my wife Wenting for her love, and to her, my parents and my brothers for their encouragement and invaluable support throughout my studies.

Finally, I would like to thank the School of Mathematics and Statistics for providing a good environment for research and the University of Plymouth for financial support.

AUTHOR'S DECLARATION

At no time during the registration for degree of Doctor of Philosophy has the author been registered for any other University award.

This study was financed with the aid of a studentship from the University of Plymouth.

A programme of advanced study was undertaken, which included a short course on Computer-Intensive Methods in Statistics organised by the University of Glasgow in 1996, an Introduction to Markov Chain Monte Carlo Methods organised by the Royal Statistical Society in 1996, and a course on Teaching Skills for Graduate Teaching Assistants organised by the University of Plymouth in 1997.

Presentations and publications:

Luan, J., Stander, J. and Wright, D. (1996) The Detection of Shape in Ultrasound Images. Poster in the Royal Statistical Society International Conference, Surrey.

Luan, J., Stander, J. and Wright, D. (1998) On shape detection in noisy images with particular reference to ultrasonography. Accepted for publication in *Statistics and Computing*.

Conferences attended:

The Royal Statistical Society International Conference, Surrey, UK, 4-6 September 1996.

The Art and Science of Bayesian Image Analysis International Conference, Leeds, UK, 30 June - 2 July 1997.

Signed ....

Date 01 September 1998

Chapter 1

Introduction

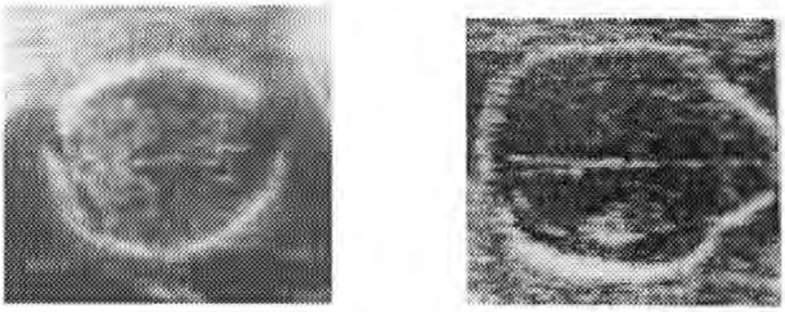
1.1 *What is the problem?*

In medicine, information on the shape and size of certain organs is sometimes of great interest to clinicians. Such information is often obtained by means of ultrasound imaging, X-ray, X-ray computer tomography (CT), magnetic resonance imaging (MRI) and positron emission tomography (PET), for example.

Neural tube defects (NTD) is a class of disorders that includes *spina bifida*, a major physical handicap. Spina bifida refers to a developmental defect of the spinal column in which the arches of one or more of the spinal vertebrae have failed to fuse together so that the spine is 'bifid'. The incidence of this condition is approximately 1 in 1,000 births. The legs of a spina bifida child may be paralysed so that he/she has to rely on crutches and callipers or a wheelchair for getting around. Associated brain damage may affect his/her appearance and speech markedly. Intellectual impairment coupled with the severity of the physical handicap gives rise to problems connected with the child's education and later may limit opportunities for further education and employment. A very good reference to spina bifida and its effects on children, families and society is Anderson and Spain (1977).

Screening is the identification of individuals who may be affected by a certain disease or anomaly. Screening for NTD is routinely undertaken at 15-23 weeks of gestation. By definition, screening is not a diagnostic procedure, but one whose results must be confirmed by an additional investigation. Accordingly, a screen-positive result changes an individual from being classified as healthy into one requiring further examination. One method of screening for NTD is biochemical in nature and is based on the level of alpha-fetoprotein in the blood; see Brock and Sutcliffe (1972) and the UK Collaborative Study on Alpha-fetoprotein in Relation to Neural Tube Defects (1977). The associated test takes place between 16-18 weeks of gestation.

Another method of screening is based on an ultrasound examination of the fetus. The cross-sectional shape of the fetal head in an affected case is often similar to that of a lemon, whereas in a normal case it tends to be roughly ellipsoidal. When the fetal head is lemon shaped, the lemon sign is said to be present. Figure 1.1 shows ultrasound images of two fetal heads viewed from above. The front of the head is towards the right and the right side of the head is towards the bottom of the image. The image in panel (a) is from a normal fetal head and is roughly ellipsoidal, whilst that in panel (b) is from a fetus affected by NTD and exhibits the lemon sign.



(a) (b)

Figure 1.1 Ultrasound images of two fetal heads when viewed from above: (a) a normal head and (b) a head showing the lemon sign (Nicolaides *et al.*, 1986).

The lemon sign is considered to be the earliest and most easily recognisable marker of spina bifida. Accordingly, ultrasonographic examination, which is also used to provide an accurate estimate of gestation age, provides another good discriminator for spina bifida at 16-23 weeks of gestation. For more details, see Nicolaides, Campbell, Gabbe and Guidetti (1986). In practice, decisions about whether to classify an individual as screen positive for NTD are based not only on biochemical marks and the cross-sectional shape of the fetal head, but also on other information obtained during the ultrasound examination and on maternal history.

1.2 *Initial idea*

Because ultrasonography is a cheap, quick, direct and probably safe screening method, it is considered by some to be the best early method for the detection of NTD; see

Nicolaides *et al.* (1986). However, it is well known that ultrasound images are of relatively poor quality. Multiplicative and additive high frequency noise, distortions in regions that are adjacent to the transducer, blurring of spatial information perpendicular to the direction of sonic wave propagation, and speckle noise are typical image degradation in ultrasound imaging.

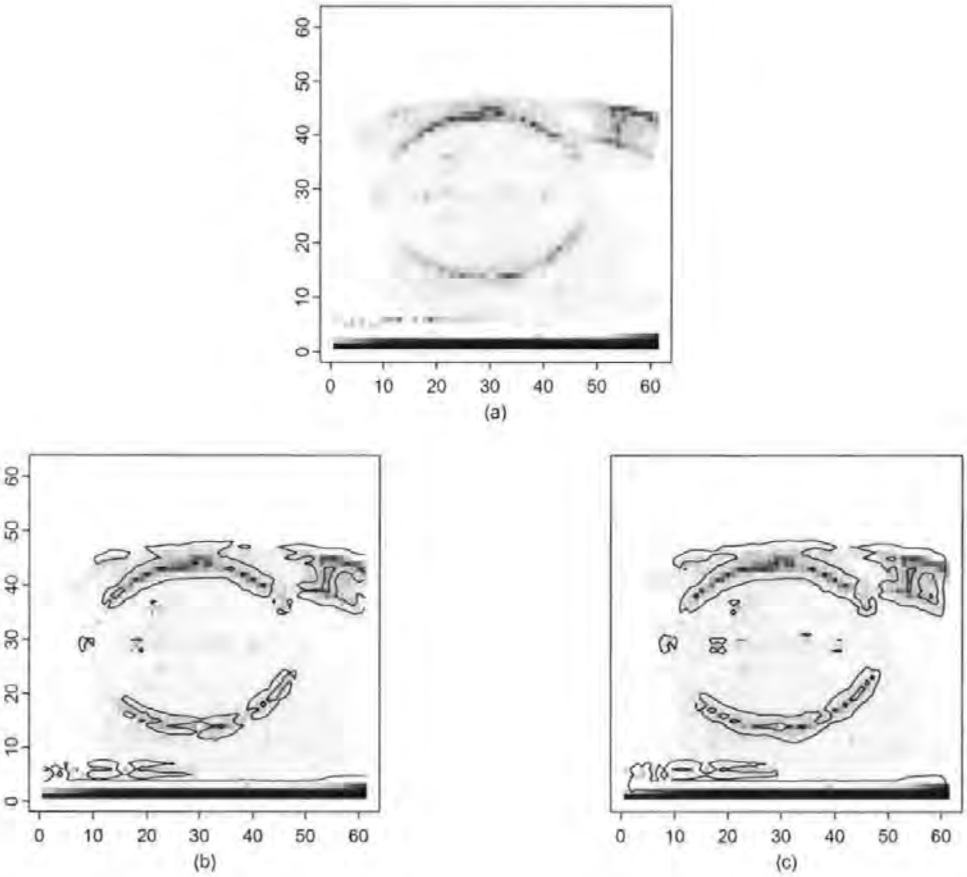


Figure 1.2 (a) A real ultrasound image of a fetal head. (b) Result from Prewitt's filter and (c) result from the Canny filter.

A typical example of an ultrasound image of a fetal head is shown in Figure 1.2(a). This image has been converted to digital format for further image analysis. Such a digital image is made up of picture elements or *pixels*. These are the smallest component of an image.

Associated with every pixel is a numerical value that indicates the grey-level at that pixel. In order to reduce the dark area in the original ultrasound image we have chosen to present and work with its inverse version in which black becomes white, white becomes black and intermediate grey-levels are mapped accordingly. We can see that in this ultrasound image the right and left sides of the cross-section of the fetal head are almost obliterated. Hence, successful ultrasound screening for spina bifida requires experienced sonographers. There is, however, much inter- and intra- operator variability even when the sonographers are experienced, as Roberts, Hibbard, Roberts, Evans, Laurence and Robertson (1983) discuss.

For these reasons, the development of an ultrasound screening recognition method that does not rely upon an experienced sonographer is of interest and many scientists including statisticians have recently begun research into this. Karaman, Kutay and Bozdagi (1995) propose an adaptive smoothing filter to reduce speckle noise. Matsopoulos and Marshall (1994) and Thomas, Peters II and Jeanty (1991) try to locate automatically the fetal head or the fetal femur using mathematical morphology. Kass, Witkin and Terzopoulos (1988) present an active contour algorithm, known as the snake algorithm, for finding an optimal contour in a neighbourhood of an initial guess of the solution. A contour is a closed curve that represents the outline of the cross-section of an object. Although Kass *et al.*'s model is not applied to ultrasound images in their paper, it can be used as a method for edge detection in ultrasound images. Cohen (1991) and Chalana, Winter III, Cyr, Haynor and Kim (1996) use the snake model for ultrasound images. Storvik (1994) applies an approach based on the minimisation of an energy function by means of simulated annealing to find ventricle boundaries in ultrasound images. This simulated annealing methodology was developed in a traditional image analysis context by Geman and Geman (1984). We will discuss other related work in later chapters.

If we can extract the edge of the cross-section of the fetal head and thus its shape directly from an ultrasound image, then we need a discrimination method to classify the head as either lemon shaped (possibly affected) or ellipsoidal (possibly unaffected). Wright, Stander and Nicolaides (1997) present a methodology, based on non-parametric density estimation, for performing such a discrimination from images of shapes.

1.3 *Traditional image restoration methods*

In this section, traditional image restoration methods will be briefly reviewed. These methods exist because a degraded version of an image is usually observed.

Often a simple model is adopted for the degradation process. Let n be the number of pixels in the true but unknown image x and let $x_i, i = 1, \ldots, n$, be the grey-level at pixel i so that $x = (x_1, \ldots, x_n)$. Let the data $z = (z_1, \ldots, z_n)$ be a degraded version of x that we observe. Then a simple model connecting z with x is

$$z_i = x_i + \varepsilon_i, \qquad i = 1, \ldots, n$$

where the noise ε_i at pixel i is distributed according to a $N(0, \kappa)$ distribution and $\varepsilon_1, \ldots, \varepsilon_n$ are independent.

If $x_i \in \{0, 1\}$, the image x can be thought of as being a binary image (black and white), while if $x_i \in \{0, 1, \ldots, g-1\}$ for $g > 2$, the image x can be thought of as having g grey levels, where often $g = 256$.

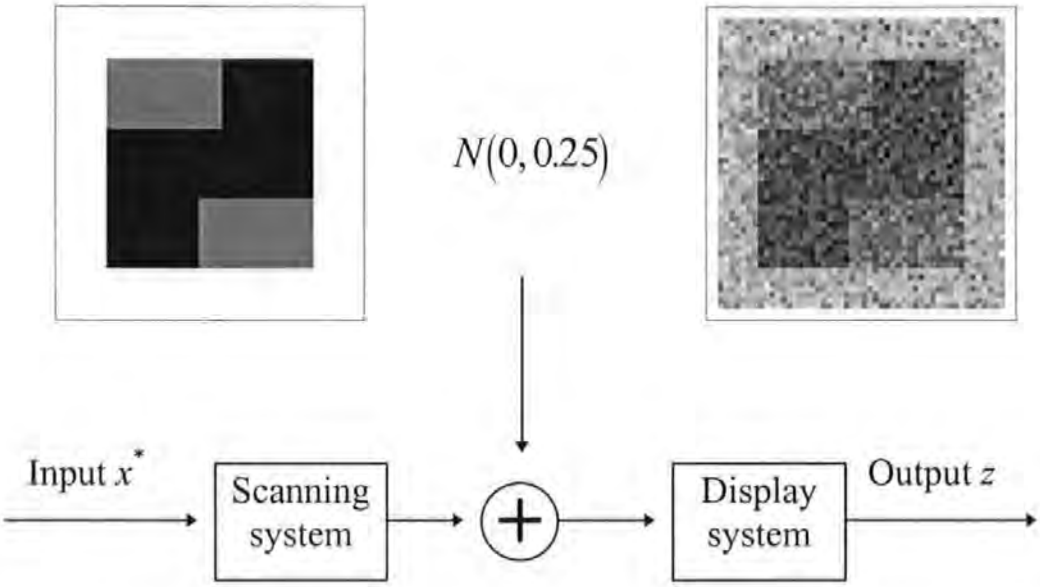


Figure 1.3 Schematic representation of the recording and display system, where the true scene is an image that comprises 50×50 pixels and has $g = 3$ grey levels.

Figure 1.3 shows in schematic form a recording and display system. The true input image x^* is shown on the left of the figure. This image comprises 50×50 pixels and has $g = 3$ grey levels: black (with corresponding value 2), grey (with corresponding value 1) and white (with corresponding value 0). The input is recorded by some form of scanner. This information is then transferred to a display system, but this transfer has degradation associated with it. Here we assume that this degradation takes the form of the addition of independent $N\left(0, 0.5^2\right)$ noise at each pixel. The output from the display system is a noisy image z, as shown on the right of Figure 1.3. We will consider this image again in a reconstruction experiment in Section 1.3.5. One task of image analysis is to find an estimate x of the true image x^* by attempting to remove the noise from z.

1.3.1 *Filters*

Image filters, such as those summarised in Chapter 3 and 4 of Glasbey and Horgan (1995), are often used in an attempt to remove the noise from z.

Thresholding is a method that first classifies all pixels and then divides an image into regions or categories, which hopefully correspond to different parts of object or different objects. Smoothing filters attempt to enhance an image by applying transformations based on groups of pixels. A median filter, for example, smooths an image by replacing each pixel value with the median of the values in a specified local region about that pixel, while a Gaussian filter smooths an image by replacing each pixel value with a Gaussian weighted mean of the values at nearby pixels.

Let $f_{i,j}$ denote the pixel values in an $m \times m$ image, where (i, j) denotes the pixel located at row index i and column index j, and let $g_{i,j}$ denote the value at pixel (i, j) after a filter has been applied. Then the output at pixel (i, j) from the median smoothing filter of size $(2s+1) \times (2s+1)$ is

$$g_{i,j} = \text{median}\left\{f_{i+k,\,j+l} : k, l = -s, \ldots, s\right\}, \text{ for } i, j = (1+s), \ldots, (m-s).$$

The weights that define a Gaussian filter are specified in terms of a Gaussian probability density function:

$$w_{k,l} = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{k^2 + l^2}{2\sigma^2}\right\}, \text{ for } k, l = -[3\sigma], \ldots, [3\sigma],$$

where $[a]$ means the integer part of a. If we let $s = [3\sigma]$, then the output from the Gaussian smoothing filter of size $(2s+1) \times (2s+1)$ is

$$g_{i,j} = \sum_{k=-s}^{s} \sum_{l=-s}^{s} w_{k,l} f_{i+k,\, j+l} \text{ for } i, j = (1+s), \ldots, (m-s).$$

The thresholding and smoothing methods can reduce noise levels in some images so emphasising objects or object edges. However, these methods usually only work well for images corrupted by low levels of noise. In our experience they do not perform well for ultrasound images.

Moreover, most edge detection filters are not very helpful for edge detection in ultrasound images because of the special nature of the degradation process as described in Section 1.2. We have applied the Laplacian filter (Glasbey and Horgan, 1995, p. 58), the range filter (*ibidem*, p. 85), Prewitt's filter (*ibidem*, p. 87), the Canny filter (*ibidem*, p. 89) and the thinning filter (*ibidem*, pp. 139–140) with little success. Figures 1.2(b) and (c) present the results of applying Prewitt's filter and the Canny filter; similar outcomes were obtained when other filters were applied. Rohling, Gee and Berman (1996) discuss problems associated with using the Canny filter on ultrasound images.

Morphology is an approach to image analysis that is based on the assumption that an image consists of structures that may be handled by set theory. Erosion, dilation, opening and closing are the four basic morphological operations and they are most often used for binary images. Morphological operations can be thought of as being filters and are often successfully applied to images with low levels of degradation. The opening and closing operators are often used for smoothing in binary images. It is sometimes useful to apply morphological operators to smooth the binary results obtained from edge detection algorithms. For more details of morphological techniques, see Chapter 5 of Glasbey and Horgan (1995). Examples of applying morphological methods to ultrasound images are presented by Matsopoulos and Marshall (1994) and Thomas, Peters II and Jeanty (1991).

1.3.2 *Bayesian approach*

The Bayesian approach to image analysis can be traced back to Geman and Geman (1984) and Besag (1986). A commonly held belief about images is that they are locally homogeneous, that is, 'nearby' pixels are expected to take similar values. We formalise the notion of 'nearby' by introducing the concept of neighbourhood.

Let $\mathcal{S} = \{1, \ldots, n\}$ represent the pixels of an image x. The set $\{\partial(i) : \partial(i) \subseteq \mathcal{S}\}$ indexed by $i \in \mathcal{S}$ is said to be a *neighbourhood system* if $i \notin \partial(i)$ and $i \in \partial(j) \Leftrightarrow j \in \partial(i)$ for all i and $j \in \mathcal{S}$. In this case $\partial(i)$ is thought of as the set of neighbouring pixels of pixel i. Here we are interested in a neighbourhood system with $\partial(i)$s of the form

$$\partial(i) = \left\{ j : \ j \in \mathcal{S} \text{ and } 1 \leq \|i - j\|^2 \leq d \right\}$$

where $d \geq 1$ and $\|i - j\|$ is Euclidean distance between the centres of the two pixels i and j. Note that pixels at or near the boundaries have fewer neighbours than interior ones. When $d = 1$ the above neighbourhood system is said to be *first-order*, that is, the first-order neighbours of a pixel are its four adjacent horizontal and vertical pixels (with appropriate modifications at the boundaries). When $d = 2$ the neighbourhood system is said to be *second-order*, that is, the second-order neighbours of a pixel are its eight adjacent horizontal, vertical and diagonal pixels (with appropriate modifications at the boundaries). Figure 1.4 illustrates both the first-order neighbourhood system (in the top left corner) and the second-order neighbourhood system (in the bottom right corner).

Given a neighbourhood system a subset c of $\mathcal{S}$ is called a *clique* if any two different elements of c are always neighbours. We use $\mathcal{C}$ to denote the set of all cliques. In Figure 1.5 we illustrate all the possible cliques corresponding to the neighbourhood systems illustrated in Figure 1.4.

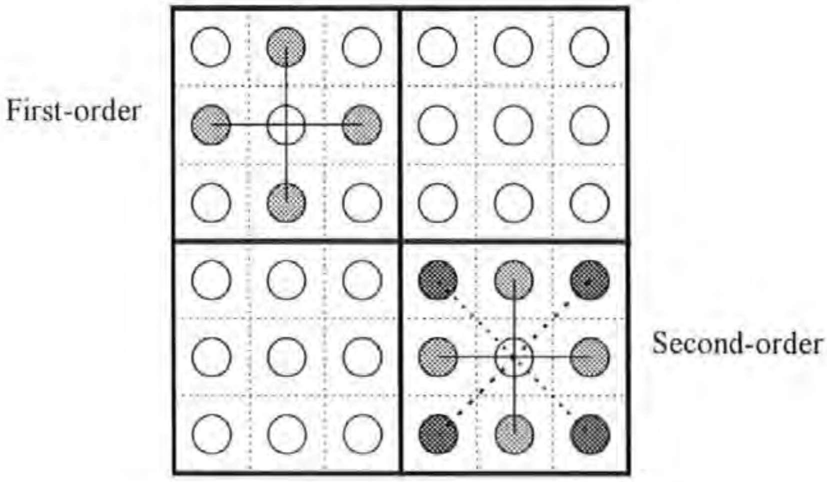


Figure 1.4 Part of a pixel grid showing the neighbourhood systems (Stander, 1992).

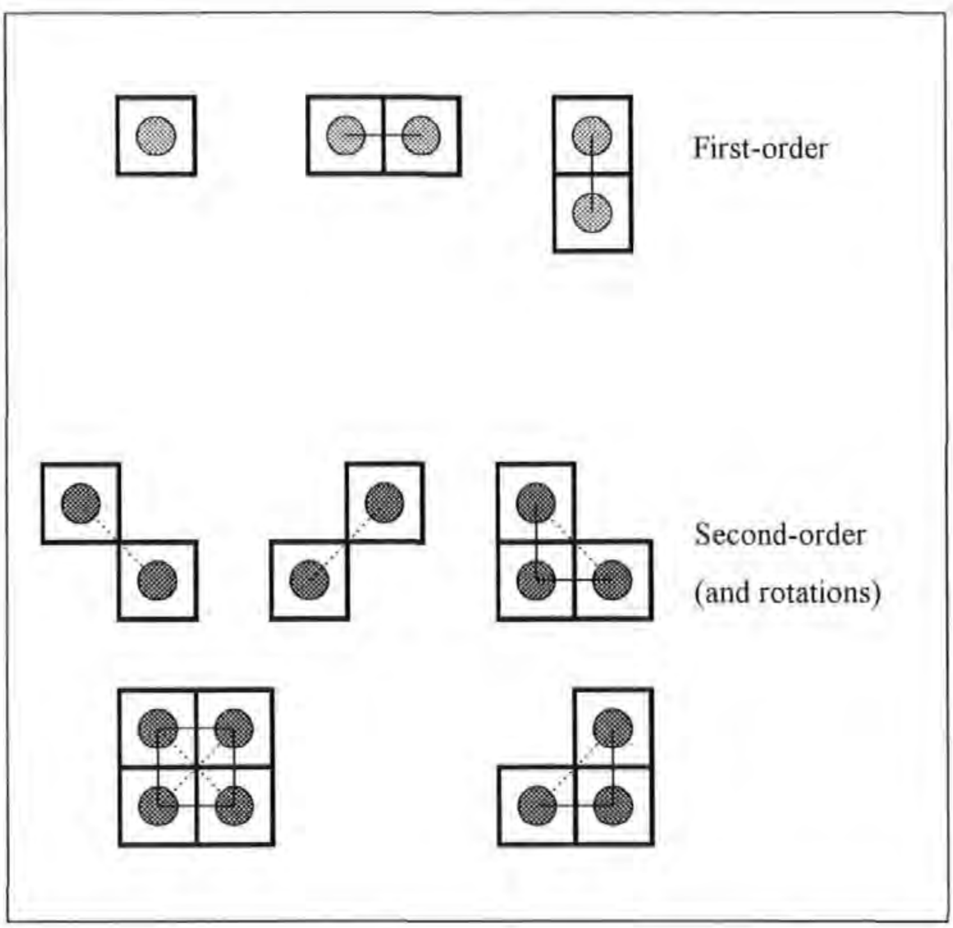


Figure 1.5 The possible pixel cliques in the first-order and second-order neighbourhood systems (Stander, 1992).

It is now possible to construct a prior distribution $p(x)$ on the set of all images as a *Gibbs distribution*:

$$P(X = x) = p(x) = \frac{1}{Z}\exp\left\{-\sum_{c \in \mathcal{C}} V_c(x)\right\} \tag{1.1}$$

where Z is a normalising constant, and the family $\{V_c(x), c \in \mathcal{C}\}$ is referred to as a *potential* and is such that each clique potential $V_c(x)$ only depends on those x_i with $i \in c$. The term $\sum_{c \in \mathcal{C}} V_c(x)$ is called the *energy function*. The probability distribution (1.1) is a

Markov random field with respect to the neighbourhood system $\{\partial(i): \partial(i) \subseteq \mathcal{S}\}$. This means that for all images x that belong to state space $\mathcal{X}$ the conditional probability $P(X_i = x_i | X_j = x_j, j \neq i)$ depends only on x_j for $j \in \partial(i)$, so that

$$P(X_i = x_i | X_j = x_j, j \neq i) = P(X_i = x_i | X_j = x_j, j \in \partial(i)).$$

An example of a prior distribution of the form specified by (1.1) is

$$p(x) \propto \exp\left[-\beta\left\{\sum_{[i,j]}\phi_{\alpha,\gamma}\left(|x_i - x_j|\right) + D\sum_{\langle i,j\rangle}\phi_{\alpha,\gamma}\left(|x_i - x_j|\right)\right\}\right] \tag{1.2}$$

where $\sum_{[i,j]}$ indicates summation over first-order neighbours, and $\sum_{\langle i,j\rangle}$ indicates summation over second-order diagonal neighbours. The function $\phi_{\alpha,\gamma}(u)$ belongs to the general family

$$\phi_{\alpha,\gamma}(u) = \frac{1}{1 + (\alpha u^{\gamma})^{-1}}\ , \quad u \geq 0,$$

suggested by Geman and McClure (1987), indexed by two parameters $\alpha > 0$ and $\gamma > 0$. The parameters β and D satisfy $\beta > 0$ and $D \geq 0$.

From knowledge of the noise distribution, we can write down the likelihood $l(z|x)$ of the observed image z given the true scene x, for example, the likelihood takes the form

$$l(z|x) = \frac{1}{(2\pi\kappa)^{n/2}}\exp\left\{-\frac{1}{2\kappa}\sum_{i=1}^{n}(z_i - x_i)^2\right\}. \tag{1.3}$$

Bayes theorem then provides the posterior distribution of an image x given the observed z:

$$p(x|z) \propto l(z|x)\, p(x). \tag{1.4}$$

With the prior given by (1.2) and the likelihood given by (1.3), the posterior distribution $p(x|z)$ is a Gibbs distribution based on the same neighbourhood system. We shall write

$$p(x|z) \propto \exp\{-U(x)\}$$

where $U(x)$ is the (posterior) energy function taking the form:

$$U(x) = \frac{1}{2\kappa}\sum_{i=1}^{n}(z_i - x_i)^2 + \beta\left\{\sum_{[i,j]}\phi_{\alpha,\gamma}\left(\left|x_i - x_j\right|\right) + D\sum_{\langle i,j\rangle}\phi_{\alpha,\gamma}\left(\left|x_i - x_j\right|\right)\right\}. \qquad (1.5)$$

We may think of β as a smoothing parameter (or smoothness constant). The value of the parameter D is usually chosen in the interval $[0, 1]$ so that the contribution to the energy from diagonal neighbours is not greater than that from first order neighbours. Often D is taken to be $\frac{1}{\sqrt{2}}$ in order to reduce rotational variability; see Silverman, Jennison, Stander and Brown (1990). If we set $D = 0$, the diagonal neighbours do not contribute to the energy, and so, when $D = 0$ we refer to U as the first-order energy model, otherwise we refer to it as the second-order energy model.

Finding the maximum *a posteriori* (MAP) estimate is one method of making inference about x. The MAP estimate corresponds to the global minimum of the energy function $U(x)$ over all images x. If the smoothing parameter $\beta = 0$, then the second term of (1.5) makes no contribution to U and the minimising image is the one in which the value of the grey level at pixel i is closest to its record z_i. We call this reconstruction the *maximum likelihood estimate* (MLE). On the other hand, if β is very large, the contribution to U of the first term in (1.5) is unimportant and so the minimising image is such that the value at every pixel is the same. The parameter β can be chosen by eye so as to give a reconstruction that appears good. Further discussion about methods for estimating β when the true scene comprises only two colours (binary image) is presented in Chapter 3 of Stander (1992). Thompson, Brown, Kay and Titterington (1991) present a study of methods of choosing the smoothing parameter for general grey level images. The fully Bayesian approach assumes that the prior parameter β is from a hyperprior, and updates β as well as the image x. This approach was considered in the image analysis context by Heikkinen and Högmander (1994) working with biogeographic data and later by Weir (1997) and Higdon, Bowsher, Johnson, Turkington, Gilland and Jaszczak (1997) working with SPECT data.

1.3.3 *Gibbs sampler and Markov chain Monte Carlo*

If the image x comprises n pixels and if each pixel can take one grey level from g possible values, then the size of the state space $\mathcal{X}$ is g^n. However, even when there are $g = 2$ grey levels and $n = 10 \times 10$ pixels, the size of $\mathcal{X}$ is still $2^{100} > 1.26 \times 10^{30}$, with the consequence that it is impossible in practice to search over all the images to find the global minimum of the energy function U. Simulated annealing and iterated conditional modes (ICM) are two optimisation methods that have been applied in statistical image analysis since they were proposed in the 1980's. In this section we introduce the Gibbs sampler and Markov chain Monte Carlo methods. We then describe these related optimisation techniques in next section.

The Gibbs sampler was first used in the image analysis context by Geman and Geman (1984). We now describe the Gibbs sampler in detail.

Suppose we wish to generate a sample from a multivariate distribution

$$\pi(x) = \pi(x_1, \ldots, x_n), \; x_i \in \Re,$$

but cannot do so directly. Let $\pi(x_i | x_{\mathcal{S} \setminus i})$ denote the conditional density for the component x_i, given the values of the other components $x_{\mathcal{S} \setminus i}$, where $\mathcal{S} \setminus i = \{j: j \in \mathcal{S} \text{ and } j \neq i\}$. The Gibbs sampler algorithm proceeds as follows. First select arbitrary starting values $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$. Then successively draw random samples from the conditional distribution $\pi(x_i | x_{\mathcal{S} \setminus i})$, $i = 1, \ldots, n$, according to the following scheme:

$$x_1^{(1)} \text{ from } \pi\left(x_1 \middle| x_{\mathcal{S} \setminus 1}^{(0)}\right)$$

$$x_2^{(1)} \text{ from } \pi\left(x_2 \middle| x_1^{(1)}, x_3^{(0)}, \ldots, x_n^{(0)}\right)$$

$$x_3^{(1)} \text{ from } \pi\left(x_3 \middle| x_1^{(1)}, x_2^{(1)}, x_4^{(0)}, \ldots, x_n^{(0)}\right)$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$x_n^{(1)} \text{ from } \pi\left(x_n \middle| x_{\mathcal{S} \setminus n}^{(1)}\right).$$

This completes a transition from $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$ to $x^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)})$. We will refer to the n updates that take $x^{(0)}$ to $x^{(1)}$ as an iteration. This cycle of updating one component at a time is repeated many times producing a sequence $x^{(0)}, x^{(1)}, \ldots, x^{(t)}, \ldots$, which are realisations of a Markov chain, with transition probability from $x^{(t)}$ to $x^{(t+1)}$ given by

$$p\left(x^{(t)}, x^{(t+1)}\right) = \prod_{i=1}^{n} \pi\left(x_i^{(t+1)} \middle| x_j^{(t+1)}, x_l^{(t)}, j < i, l > i\right).$$

The methodology of constructing a Markov chain to generate samples from a complicated distribution π, often known up to a constant of proportionality, is referred to as *Markov chain Monte Carlo* (MCMC). There exists other algorithms that are based on the MCMC principle, such as the Metropolis algorithm and Metropolis-Hastings algorithm proposed by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953) and Hastings (1970). For a general introduction to MCMC, see Gilks, Richardson and Spiegelhalter (1995).

MCMC methods have been used extensively in statistical physics over the last forty years, in spatial statistics over the last twenty years and in Bayesian image analysis over the last decade. Recently the literature on this topic has increased rapidly; see Gelman and Rubin (1992), Geyer (1992), Besag and Green (1993), Smith and Roberts (1993), Besag, Green, Higdon and Mengerson (1995), Robert (1995), Green (1995) and Cowles and Carlin (1996) for important contributions. Brooks (1998) provides a tutorial review of some of the most common areas of research in MCMC. Frigessi, Martinelli and Stander (1997) give bounds on the convergence time of the Gibbs sampler used in certain Bayesian image reconstruction situations that are of order $n \log n$, where n is the number of pixels. Gibbs (1998) provides bounds of order n^2, but with a proportionality constant that is easy to compute.

The basic idea behind Markov chain Monte Carlo is that if it is impossible to find a way to simulate independent realisations of some complicated distribution $\pi(x)$, it is almost as useful to simulate dependent realisations, say $x^{(1)}, x^{(2)}, \ldots$, to form an irreducible aperiodic Markov chain having $\pi(x)$ as its stationary distribution. The Metropolis-Hastings algorithm (Metropolis *et al.*, 1953; Hastings, 1970) provides such a chain. Because of the dependence, we need larger samples than would be required if independent sampling were possible.

We now present the Metropolis-Hastings algorithm and the Metropolis algorithm.

Metropolis-Hastings algorithm:

Given current state $x^{(t)}$, obtain the next state $x^{(t+1)}$ by means of the following two steps:

Step 1 Sample a candidate point x' from a *proposal* distribution $q\left(x \middle| x^{(t)}\right)$.

Step 2 With probability

$$\alpha\left(x^{(t)}, x'\right) = \min\left\{1, \quad \frac{\pi(x')\, q\left(x^{(t)} \middle| x'\right)}{\pi\left(x^{(t)}\right) q\left(x' \middle| x^{(t)}\right)}\right\} \tag{1.6}$$

set

$$x^{(t+1)} = x' \qquad \text{(acceptance)},$$

else set

$$x^{(t+1)} = x^{(t)} \qquad \text{(rejection)}.$$

Note that since π only appears in the ratio $\pi(x')/\pi\left(x^{(t)}\right)$, knowledge of the normalising constant is not required.

Metropolis algorithm:

The Metropolis algorithm is a special case of the Metropolis-Hastings algorithm with a symmetric proposal: $q\left(x \middle| x^{(t)}\right) \equiv q\left(x^{(t)} \middle| x\right)$. This means that the probability in (1.6) reduces to

$$\alpha\left(x^{(t)}, x'\right) = \min\left\{1, \quad \frac{\pi(x')}{\pi\left(x^{(t)}\right)}\right\}.$$

The Gibbs sampler which we have described above is a single-site updating version of the Metropolis-Hastings algorithm with proposal distribution at the i^{th} site of the form

$$q\left(x' \middle| x^{(t,i)}\right) = \pi\left(x_i' \middle| x_{\mathcal{S}\setminus i}^{(t,i)}\right) = \frac{\pi\left(x_i', x_{\mathcal{S}\setminus i}^{(t,i)}\right)}{\int \pi\left(x_i, x_{\mathcal{S}\setminus i}^{(t,i)}\right) dx_i},$$

where

$$x^{(t,i)} = \left(x_j^{(t+1)}, x_l^{(t)}, \; j < i, l \geq i\right),$$

$x_{\mathcal{S}\setminus i}^{(t,i)}$ denotes all the components of $x^{(t,i)}$ except the i^{th}, and we suppose that the candidate state x' differs from $x^{(t,i)}$ only in the i^{th} component. Note that we are only updating the i^{th} component $x'_{\mathcal{S}\setminus i} = x_{\mathcal{S}\setminus i}^{(t,i)}$ with the result that

$$\pi(x') = \pi\left(x_i', x'_{\mathcal{S}\setminus i}\right) = \pi\left(x_i', x_{\mathcal{S}\setminus i}^{(t,i)}\right)$$

$$\pi\left(x^{(t,i)}\right) = \pi\left(x_i^{(t,i)}, x_{\mathcal{S}\setminus i}^{(t,i)}\right) = \pi\left(x_i^{(t,i)}, x'_{\mathcal{S}\setminus i}\right).$$

Using the above formula for the Metropolis-Hastings algorithm, we find that the acceptance probability for the Gibbs sampler at the i^{th} site is given by

$$\alpha_i\left(x^{(t,i)}, x'\right) = \min\left\{1, \quad \frac{\pi(x')\, q\left(x^{(t,i)} \middle| x'\right)}{\pi\left(x^{(t,i)}\right) q\left(x' \middle| x^{(t,i)}\right)}\right\}$$

$$= \min\left\{1, \quad \frac{\pi(x') \dfrac{\pi\left(x_i^{(t,i)}, x'_{\mathcal{S}\setminus i}\right)}{\int \pi\left(x_i, x'_{\mathcal{S}\setminus i}\right) dx_i}}{\pi\left(x^{(t,i)}\right) \dfrac{\pi\left(x'_i, x^{(t,i)}_{\mathcal{S}\setminus i}\right)}{\int \pi\left(x_i, x^{(t,i)}_{\mathcal{S}\setminus i}\right) dx_i}}\right\}$$

$$= \min\left(1, \quad \frac{\pi(x')\, \pi\left(x^{(t,i)}\right)}{\pi\left(x^{(t,i)}\right) \pi(x')}\right)$$

$$= 1.$$

So, with the Gibbs sampler a candidate state $x' = \left(x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x'_i, x_{i+1}^{(t)}, \ldots, x_n^{(t)}\right)$ is never rejected.

In image analysis the Gibbs sampler is used to produce a sequence of dependent images from the posterior distribution $p(x|z)$. The Gibbs sampler starts from an initial image (MLE estimate, for example) and then repeats the following procedure many times: visit each pixel in turn and replace the current value by one sampled from the conditional distribution of the value at that pixel given the current values at all the other pixels and the observed data. At the i^{th} pixel, the above conditional distribution is denoted $p\left(x_i \middle| x_{\mathcal{S}\setminus i}, z\right)$, and for the posterior distribution defined in Section 1.3.2 it can be shown that

$$p\left(x_i \middle| x_{\mathcal{S}\setminus i}, z\right) \propto \exp\left[-\frac{1}{2\kappa}\sum_{i=1}^{n}(z_i - x_i)^2 - \beta\left\{\sum_{j\in\partial_i^{(1)}} \phi_{\alpha,\gamma}\left(\left|x_i - x_j\right|\right) + D\sum_{j\in\partial_i^{(2)}} \phi_{\alpha,\gamma}\left(\left|x_i - x_j\right|\right)\right\}\right]$$

where $\partial_i^{(1)}$ denotes the first-order (horizontal and vertical) neighbours of pixel i and $\partial_i^{(2)}$ denotes the second-order (diagonal) neighbours.

1.3.4 *Optimisation techniques: simulated annealing and iterated conditional modes*

Simulated annealing as described by Geman and Geman (1984) and the iterated conditional modes (ICM) algorithm as proposed by Besag (1986) have been employed in an attempt to minimise the energy function $U(x)$ over $\mathcal{X}$. Both simulated annealing and ICM are iterative algorithms, with simulated annealing usually being based on the Gibbs sampler.

Simulated annealing was considered by Kirkpatrick, Gellat and Vecchi (1983) and later introduced to the imaging context by Geman and Geman (1984). It has ever since been the subject of an enormous literature; see, for example, Laarhoven and Aarts (1987) and Winkler (1995). Among recent contributions relevant to our work are Stander and Silverman (1994) and Hurn and Jennison (1995).

The idea of the simulated annealing algorithm is that instead of using a Markov chain to sample from the posterior distribution $p(x|z) \propto \exp\{-U(x)\}$, we use a Markov chain to sample from a probability distribution defined by

$$p_t(x) \propto \exp\left\{-\frac{U(x)}{\tau_t}\right\},$$

where t is the update number, $\tau_t > 0$ is the temperature at the t^{th} iteration, and $\tau_t \to 0$ as $t \to \infty$. In theory, provided $\tau_t \to 0$ very slowly, the simulated annealing algorithm should reach the global minimum of $U(x)$, that is, the image corresponding to the MAP estimate should result; see Hajek (1988) for details and Geman and Geman (1984) who provide a formal proof of convergence in the image analysis case. Because τ_t has to tend to zero very slowly, in practice the method is computationally very expensive.

The way in which τ_t tends to zero is known as the temperature schedule. Geman and Geman (1984) suggest a temperature schedule that is of the form

$$\tau_t = \frac{C}{\log(1+t)},$$

where C is a constant independent of t. If C is big, a large number of iterations are required for τ_t to approach zero. Geman and Geman (1984) give a value of C that guarantees that realisation of the associated (inhomogeneous) Markov chain will eventually tend to the MAP estimate. As this value is so big that an enormous number of iterations is required for τ_t to approach zero, Geman and Geman (1984) actually used smaller values in their restorations.

Stander and Silverman (1994) considered several other temperature schedules. These schedules are defined in terms of the finite total number of iterations T, the temperature f for the first update and the temperature l for the last update. The values of f and l are chosen to give reasonable results. The details of these schedules are:

- Straight line schedule

$$\tau_t = \frac{l-f}{T-1}(t-1)+f$$

- Geometric schedule

$$f\left(\frac{l}{f}\right)^{(t-1)/(T-1)}$$

- Reciprocal schedule

$$\frac{lf(T-1)}{(lT-f)+(f-l)t}$$

- Logarithmic schedule

$$\frac{lf\left\{\log(T+1)-\log 2\right\}}{\left\{l\log(T+1)-f\log 2\right\}+(f-l)\log(t+1)};$$

see Stander (1992) and Stander and Silverman (1994) for more details of these temperature schedules and the choice of values of f and l.

The ICM method can be thought of as simulated annealing at zero temperature. At each update of ICM an image is produced that does not increase $U(x)$, whereas in simulated annealing, an image that increases $U(x)$ may be produced. The ICM algorithm starts from an initial image (MLE estimate, for example) and then repeats the following procedure: visit each pixel in turn and replace the current value by one that provides the minimum value of $U(x)$ or the maximum of $p(x|z)$. The ICM procedure stops when no further decreases in the energy function occur. The resulting image corresponds to a local minimum of the energy function. The ICM method is computationally inexpensive, usually requiring less then 10 iterations for convergence (see Besag, 1989, for example).

We have seen that the above Bayesian approach to image analysis is based on the posterior distribution $p(x|z) \propto \exp\{-U(x)\}$ where $U(x)$ is given by (1.5). The restoration is often taken to be the image that maximises $p(x|z)$, or equivalently minimises $U(x)$, and can be found (at least approximately) by means of simulated annealing and ICM. The above Bayesian approach is essentially a smoothing method, and so does not work well when applied to ultrasound image of fetal heads, such as the one shown in Figure 1.2(a) as it tends to smooth away the details that remain.

1.3.5 *Reconstruction experiment for the noisy image z shown in Figure 1.3*

In this section we present the results of an experiment that is designed to demonstrate the above Bayesian approach. Our aim is to restore the true image x^* from the noisy image z shown in Figure 1.3.

In panel (a) of Figure 1.6 the MLE is displayed. The image presented in panel (b) of Figure 1.6 is obtained by applying Gaussian smoothing with $\sigma = 1$ to z and then rounding the resulting value at each pixel to the nearest value in $\{0, 1, 2\}$. The image in panel (c) is obtained in an analogous way using median smoothing with $s = 1$. These images look better than the original observed image — the output z shown in Figure 1.3, but are still poor. There are 459 (18.36%), 214 (8.56%) and 127 (5.08%) misclassified pixels, respectively, for the three estimates shown in panels (a), (b) and (c).

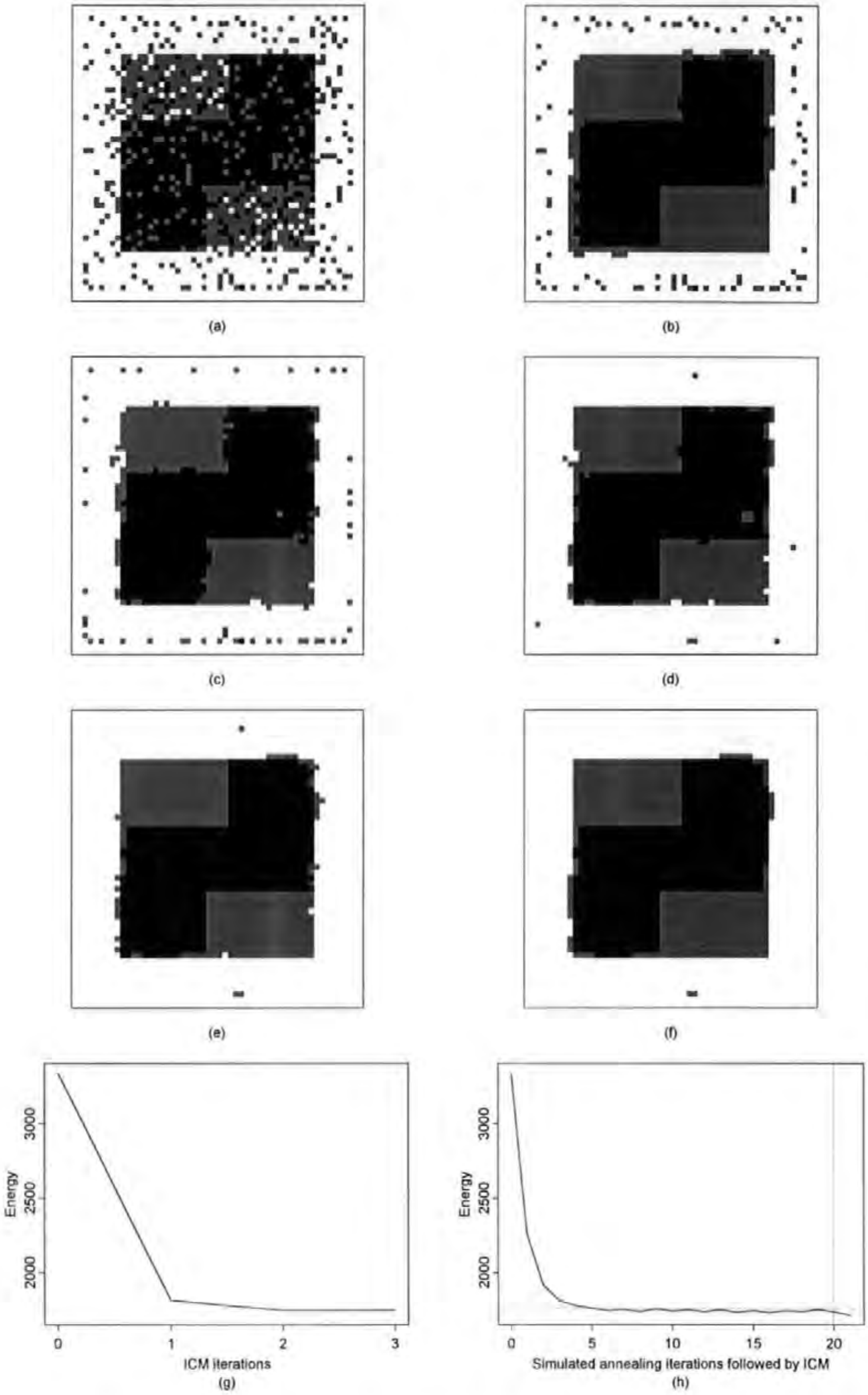


Figure 1.6 Reconstruction experiment for the noisy image z shown in Figure 1.3. Estimates obtained by using (a) maximum likelihood (MLE), (b) the Gaussian smoothing filter, (c) the median smoothing filter, (d) the ICM algorithm, (e) the simulated annealing algorithm, and (f) the simulated annealing followed by ICM algorithm. Panel (g) presents a graph of the energy function plotted against iteration number for ICM algorithm. Panel (h) presents a similar plot for the simulated annealing followed by ICM algorithm.

We then applied the Bayesian approach to the noisy image z with known variance $\kappa = 0.25$. We took $\alpha = 0.1$, $\beta = 15$ and $\gamma = 2$, and considered the first-order energy model (1.4) by setting $D = 0$. The ICM algorithm requires just three iterations for convergence (no pixels are changed on the final iteration) and yields the reconstruction shown in panel (d) with 67 (2.68%) misclassified pixels. There were 426 pixels that changed their grey levels after the first iteration, and 26 pixels after the second. Figure 1.6(e) presents the result from the simulated annealing algorithm after twenty iterations, where the logarithmic temperature schedule with $f = 0.9$ and $l = 0.1$ was employed. Twenty iterations seem to be enough to obtain a satisfactory result in this simple example. In this estimate, there are 51 (2.04%) misclassified pixels. Finally, we applied the ICM algorithm to the result shown in panel (e) that was obtained from the simulated annealing algorithm. Four iterations of ICM were required for convergence, and there were 15 pixels that changed their grey levels after the first iteration, one pixel after the second, and one pixel after the third. The result is presented in panel (f) and there are 52 (2.08%) misclassified pixels. Note that this final ICM stage has provided additional smoothing. The graphs presented in panels (g) and (h) are plots of the energy function (1.5) against iteration number for the ICM algorithm and the simulated annealing followed by ICM algorithm. In the panel (h) the broken vertical line marks the end of simulated annealing and beginning of ICM, whereas the broken horizontal line marks the final energy produced by the ICM algorithm only. From these energy plots we see that simulated annealing allows increases in energy whereas ICM does not.

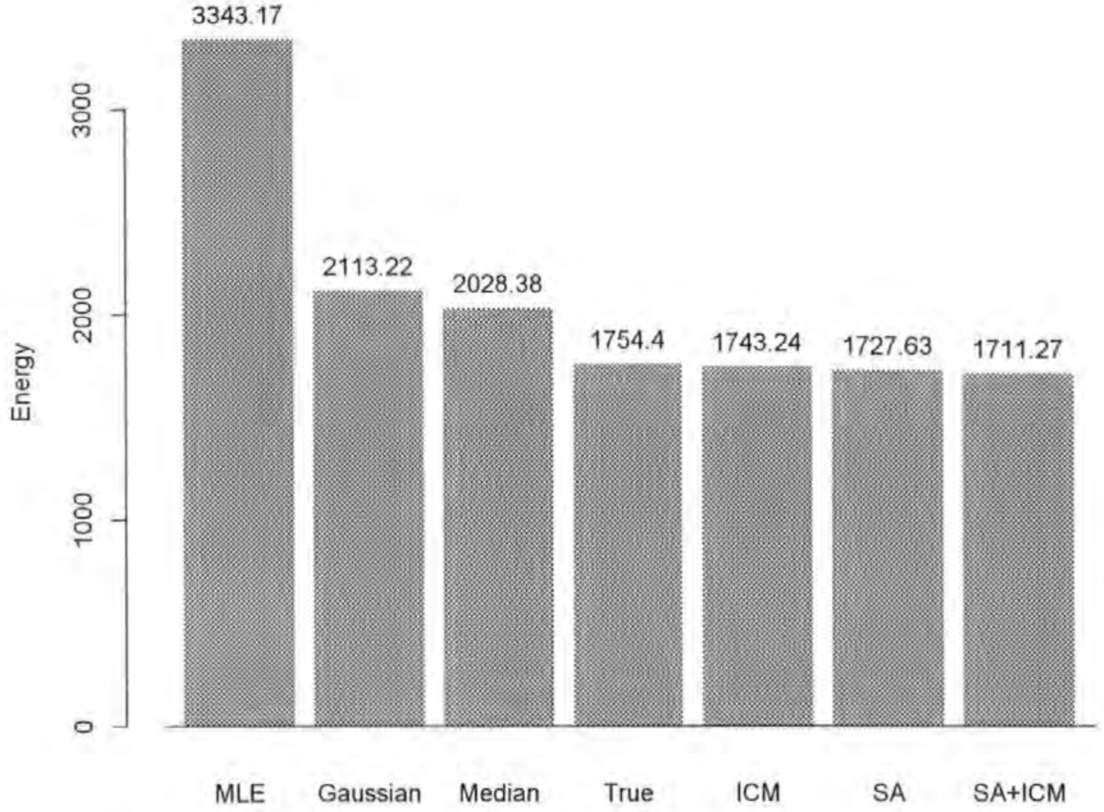


Figure 1.7 Bar plot of the final energies of the results from six image reconstruction methods together with the energy of the true image x^*. The energies are sorted into descending order and the values of the energies are printed on the top of each bar.

We see that the simulated annealing algorithm, the ICM algorithm and the simulated annealing followed by ICM algorithm provide better reconstructions of the true image x^* from the noisy image z than were obtained by applying Gaussian and median smoothing. Both x^* and z are presented in Figure 1.3.

Figure 1.7 presents the energies of the results obtained by using maximum likelihood (MLE), the Gaussian smoothing filter, the median smoothing filter, ICM, simulated annealing (SA), and simulated annealing followed by ICM (SA+ICM), together with the energy of the true image x^*. The energies are sorted into descending order and the values of the energies are printed on the top of each bar.

The energy of the true image x^* is 1754.40. From Figure 1.7 we see that the maximum likelihood estimate (MLE) has the highest energy (3343.17). The Gaussian and the median smoothing filters considerably reduce the energy to 2113.22 and 2028.38, respectively. These energies are close to the energy of the true image. The energies of the results obtained by using ICM, simulated annealing and simulated annealing followed by ICM are lower than the energy of the true image. The simulated annealing algorithm produces an estimate whose energy (1727.63) is lower than that of ICM (1743.24). The result shows that the simulated annealing algorithm allows escapes from local minima of the energy. Simulated annealing followed by ICM yields the image that has the lowest energy (1711.27) in this example.

1.4 *Structure of the thesis*

We consider ways of improving existing techniques that have been applied to ultrasound images and we develop and discuss a new iterative method for edge detection. We demonstrate the effectiveness of the new edge detection method, and of the improvements to the existing techniques by means of simulation studies.

In Chapter 2 we implement and improve the edge detection method of Storvik (1994) for fitting an object contour to an ultrasound image by using simulated annealing. Our improvement is based on a cascade-type blocking method (Jubb and Jennison, 1991; Hurn and Jennison, 1995) that substantially speeds up and improves the performance of the algorithm. The approach proposed in Chapter 2 can be used to detect shape in images where the data are a corrupted version of the shape itself, such as the ultrasound image of a human ovarian cyst shown in Figure 1.8. In this type of image, the shape is defined by its edge and pixels inside and outside the shape have different records. Two simulation studies are performed based on such an image of a head shape. One is designed to compare the performance of the cascade-based algorithm with that of the direct simulated annealing algorithm. The other aims to investigate whether there is a significant difference due to the order in which edge pixels are visited.

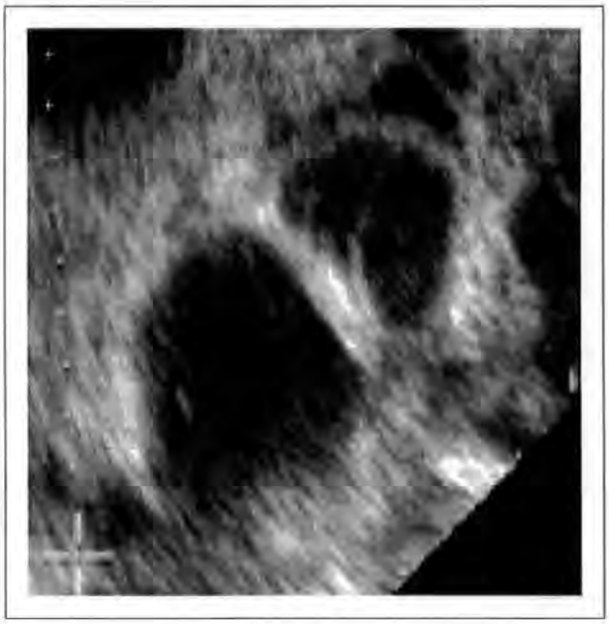


Figure 1.8 Ultrasound image of a human ovarian cyst.

The ultrasound image of a fetal head shown in Figure 1.1 is a different type of image in that the shape is defined by a thin outline of pixels with records that are different from those at pixels inside and outside the shape. The approach presented in Chapter 2 does not work well on this type of image. In Chapter 3, an algorithm for extracting the head edge and thus the head shape from such ultrasound images is proposed. This algorithm is based on a specially designed kernel function. The algorithm not only provides input for the discrimination algorithm of Wright *et al.* (1997) but also allows automatic fetal head measurements to be made as described in Chalana *et al.* (1996) and Matsopoulos and Marshall (1994). In Sections 3.4 and 3.5 of Chapter 3, simulation studies based on an edge defined image of an ellipse are carried out in order to test the kernel algorithm we developed.

The kernel algorithm is modified in Section 3.8 to detect shape in images where the data are a corrupted version of the shape itself. Although this type of shape may be detected by the method described in Chapter 2, the kernel algorithm is considerably less complicated and computationally expensive. We conduct simulation studies based on the image of a head shape and an image of an ellipse in order to compare the performance of the algorithm described in Chapter 2 and the algorithm we develop in Section 3.8.

A simulation study based on an edge defined circular image is performed in Chapter 4 in order to study the effect of the parameters used in the definition of the kernel function.

Finally, in Chapter 5 we introduce and discuss the snake model proposed by Kass *et al.* (1988). The associated energy function is modified in order to improve the original snake methodology. We also discuss how the simulated annealing and ICM algorithms can be used to minimise the modified energy function. We run a simulation study based on a shape that is defined by an edge that is two pixels thick to compare the performance of the kernel algorithm, snake algorithm and simulated annealing plus ICM algorithm.

All our work is motivated by two real ultrasound images. One of these, the ultrasound image of a fetal head shown in Figure 1.2(a), is considered in Chapter 3, where the proposed kernel algorithm is applied to it, and in Chapter 5 where the snake methodology and associated improvements are used. The other image, an ultrasound image of a human ovarian cyst shown in Figure 1.8, is considered in Chapter 2, where Storvik's algorithm and associated improvements are applied to it, and in Chapter 3, where the modified kernel algorithm is employed. We are grateful to Dr P. Dubbins and Dr T. Reynolds for having supplied these original photographs.

All the approaches described in this thesis have been implemented in S-Plus for Windows Version 3.3 Release 1.

Chapter 2

Object Recognition Using Simulated Annealing and ICM with Cascade-type Blocking

2.1 *Introduction*

Many papers have developed algorithms for object recognition in recent years.

Kass, Witkin and Terzopoulos (1988) present a non-Bayesian active contour model called 'snake' as a general method for edge detection and motion tracking. A snake is an energy-minimising contour that is guided by external constraint forces and influenced by image forces that pull it toward image features such as edges and lines. Snakes are very useful creatures; there are many applications in computer vision in which they are employed, such as the problems of detecting edges, lines and contours. We will discuss the details of the snake model in Chapter 5.

Canny (1986) presents a gradient edge filter method. This local algorithm is a very simple method for edge detection but it is not able to handle complex problems such as those involving highly degraded deformable shapes. Friedland and Adam (1989), and Friedland and Rosenfeld (1992) use a low-level energy function to force smoothness of the edge and sharpness between the object and the background, and a high-level energy function to compare the entire boundary with a library of known compact objects. Mardia and Qian (1995) build up a Bayesian approach to compact object recognition but their method also depends on a library of known objects.

The deformable template model of Grenander, Chow and Keenan (1990) is based upon the deformation of a template to find an optimum fit to the object. The template is a closed polygon with fixed number of sides of variable length representing the outline of a typical object. Grenander and Miller (1994) generalise this to a variable number of objects and use jump-diffusion sampling to explore the state space of deformations. Helterbrand, Cressie and Davidson (1994)'s work is quite similar to the template and snake models. The main difference is that the objects are defined by boundaries that are one pixel thick. Qian, Titterington and Chapman (1996) employ a Bayesian framework for the problem of identifying the irregular boundary of a magnetic domain in a thin multilayer film, using data in the form of an electron micrograph. They use the ICM procedure and find initialisation an especially difficult problem, which they resolve by means of a template-like modelling approach. Qian *et al.* (1996) use a star shaped polygon and their prior distribution is based on the smoothness properties of this shape. A star shaped object is one in which every point of the edge is visible from an interior point. Hurn and Rue (1997) combine the template and marked point process approaches to handle scenes containing variable numbers of objects of different types. Pievatolo and Green (1998) describe a statistical model that allows polygons with any number of sides. Both Hurn and Rue (1997) and

Pievatolo and Green (1997) use reversible jump Markov chain Monte Carlo algorithms introduced by Green (1994, 1995); see also Richardson and Green (1997). All the work in the above papers is built upon the Bayesian approach for object recognition. These authors have shown that a Bayesian approach provides a successful method for finding the outline or contour of an underlying object.

Storvik (1994) presents an approach for edge detection that is applicable when the pixels inside and outside an object take different records. His procedure is reminiscent of the snake algorithm suggested by Kass *et al.* (1988) and adapted by Chalana, Linker, Haynor and Kim (1996) in a recent application to echocardiography. Teles de Figueiredo and Leitão (1992) also used a Bayesian approach to estimate ventricular edges in angiographic images and this paper is related to the work of Friedland and Adam (1989). Storvik's method is somewhat related to a shape analysis approach to compact object recognition discussed by Friedland and Rosenfeld (1992) and Mardia and Qian (1995).

This chapter builds upon Storvik (1994)'s approach for single object recognition from ultrasound images. The main algorithm employed by Storvik (1994) is simulated annealing. This makes Storvik's method very computationally intensive. In this chapter we shall assume that the true grey-levels of pixels inside and outside the contour of the object are different. We present a modification of Storvik's approach based on the cascade-type blocking algorithm discussed by Hurn and Jennison (1995) that considerably reduces the computation required. Part of this work has been accepted for publication; see Luan, Stander and Wright (1998).

Our approach has associated with it the following features:

- It works for images comprising a single object.
- It operates on edge pixels and their first and second order neighbours.
- Gaussian distributions are assumed for the grey levels inside and outside the object, the parameters of these distributions being estimated at the beginning of the algorithm.
- A threshold convex hull initial configuration is defined at the beginning of the algorithm.
- The main optimisation method used is simulated annealing.
- The ICM algorithm is applied to the contour obtained from simulated annealing.
- The algorithm we describe below can be applied to detect any single shape when the pixels inside and outside the object take different records, whereas the algorithms given in Friedland and Rosenfeld (1992) and Mardia and Qian (1995) apply to any star-shaped object.

Storvik (1994)'s algorithm is based on an energy function. This energy function, together with other details of the algorithm, is briefly reviewed in Section 2.2. In Section 2.3 we present a slightly different form of the energy function and introduce our cascade-type blocking modification. Some details of the computer implementation of the resulting approach are given in Section 2.4. In Section 2.5 we present an example of detecting underlying shape in a real ultrasound image. We finish with a discussion in Section 2.6.

2.2 *Storvik's approach and object recognition*

Due to noise and/or blur, the observed image z differs from the true but unknown image. As mentioned in Section 1.3.4, many approaches that are applied in image analysis such as ICM and simulated annealing are based on the minimisation of an energy function $U(x)$, where x is a possible configuration such as an image or contour. Storvik (1994) uses simulated annealing to minimise his energy function and he presents good results for two examples, one based on an ultrasound image of the left ventricle and the other based on a Magnetic Resonance image of the human brain. We will describe Storvik (1994)'s approach in this section.

Due to computational considerations, Storvik (1994) describes contours in terms of nodes, rather than in terms of image pixels. The nodes give the co-ordinates of points on the contour in a clockwise direction. In Storvik's approach, x denotes a contour which has a polygon representation $x = (d_1, d_2, \ldots, d_N)$, where each node d_i on the contour is given by its co-ordinates and N is the number of nodes and may be stochastic. The energy function $U(x)$ takes the form

$$U(x) = \alpha_1 u_1(x) + \alpha_2 u_2(x) + \alpha_3 u_3(x), \tag{2.1}$$

where the three components $u_1(x)$, $u_2(x)$ and $u_3(x)$ will be described shortly, and α_1, α_2 and α_3 are the weights for the three components.

The first component $u_1(x)$ is the energy related to the smoothness of the contour x:

$$u_1(x) = \frac{(\text{length of the outline of the shape defined by } x)^2}{\text{area of shape defined by } x}.$$

The second energy component $u_2(x)$ is the gradient operator for recognising edges in the vertical direction:

$$u_2(x) = \sum_i V_i(x),$$

where $\sum_i$ indicates summation over all nodes on the contour x, and the potential function $V_i(x)$ is given by

$$V_i(x) = I\left(\text{segment } (d_i, d_{i+1}) \text{ horizontal}\right) \times \begin{cases} -(z_o - z_e), & \text{if } z_o > z_e \\ -\dfrac{1}{1 + z_o - z_e}, & \text{otherwise} \end{cases}$$

where $I(\cdot)$ is the indicator function, e and o are the two pixels adjacent to the horizontal segment (d_i, d_{i+1}) inside and outside the contour, and z_e and z_o are the observed values at pixel e and o.

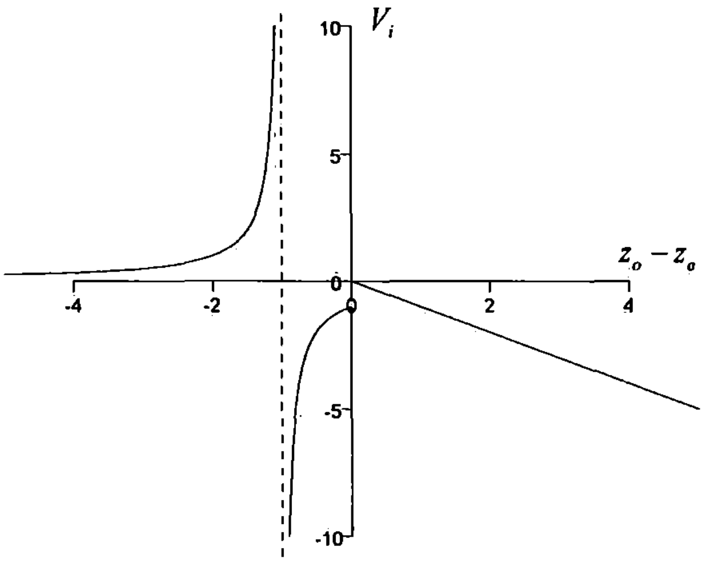


Figure 2.1 Storvik's potential function $V_i(x)$ of the gradient operator function u_2 which is employed only in the vertical direction when edge pixel e and outside pixel o are adjacent to horizontal sengment (d_i, d_{i+1}). It takes a strange form.

Figure 2.1 shows a plot of the gradient operator $V_i(x)$ as a function of $z_o - z_e$, when the edge pixel e and outside pixel o are adjacent to horizontal segment (d_i, d_{i+1}). It has a strange form when $z_o - z_e \leq 0$. Moreover, in general applications it is impossible to know which direction is more important, and so we believe that we should consider both the vertical and horizontal directions in the gradient operator.

The third component $u_3(x)$ is defined as the negative of the logarithm of the likelihood:

$$u_3(x) = -\log\{f(z|x)\} ,$$

where $f(z|x)$ is the likelihood of observed data z given x. In this case, the likelihood is chosen to measure the difference of the grey-levels inside and outside the shape:

$$f(z|x) = \prod_{j \in S} f_1(z_j) \cdot \prod_{j \notin S} f_2(z_j) ,$$

where S is the set of pixels inside the shape defined by contour x, z_j is the observed value of z at pixel j, and f_i is a $N(\mu_i, \sigma^2)$ density, $i = 1, 2$.

The weight α_3 of $u_3(x)$ in Storvik's energy function (2.1) is unnecessary. We can let $\alpha_3 \equiv 1$ so the energy function can be a simple form

$$U(x) = \alpha_1 u_1(x) + \alpha_2 u_2(x) + u_3(x), \tag{2.2}$$

Storvik's method needs sufficient nodes to achieve the necessary accuracy, but it is not clear how to choose the number of nodes. Storvik (1994) minimises his energy function by means of the stochastic simulated annealing algorithm. Storvik (1994) employs an enormous number (up to 100,000,000) of simulated annealing updates in his attempt to minimise $U(x)$. The reason for this is that each simulated annealing update makes only small changes to the current configuration. Hence a badly placed initial configuration requires an enormous number of these small changes before it can estimate well the true edge. Qian *et al.* (1996) experienced similar problems. In the next section, we modify the form of the energy function and develop this method based on the cascade algorithm discussed by Hurn and Jennison (1995).

2.3 *A cascade-type blocking approach and simulated annealing*

In this section we propose using the cascade algorithm in order to reduce the number of simulated annealing updates needed by Storvik's algorithm. The cascade algorithm was first introduced by Jubb and Jennison (1991) and developed further by Hurn and Jennison (1995). It has similarities with the renormalisation approach of Gidas (1989).

We introduce our cascade-type blocking approach by means of an example. A $2^8 \times 2^8$ pixel image can be divided up into $2^6 \times 2^6$ blocks of 4×4 pixels. Starting from an initial configuration x_0, a simulated annealing run can be used to find a configuration x_1 that approximately minimises the original energy function U, given in (2.2), over edges that are restricted to lie along the boundaries of the 4×4 coarser blocks. In this way large changes in x_0 can be effected relatively quickly. Next $2^7 \times 2^7$ blocks comprising 2×2 pixels are considered. The configuration x_1 from the previous simulated annealing run can be used as the initial configuration for a simulated annealing run that finds a configuration x_2 that approximately minimises the original energy function U over edges that are restricted to lie along the boundaries of the 2×2 blocks. Finally, the configuration x_2 can be used as the initial configuration for a simulated annealing run that attempts to minimise U over unrestricted edges.

The advantage of the cascade-type blocking algorithm is that the individual restricted optimisations, which allow large changes in the edge to be made, can be performed relatively quickly. The result of this is that even when programming overheads are considered the total time for the cascade-type blocking approach is usually less than the time required to minimise U directly. Hurn and Jennison (1995) consider other blocking choices including blocks that are defined adaptively according to the data. Since the standard blocking approach already offers us considerable advantages, we do not consider such sophistication here.

The energy of any configuration at any blocking level is defined to be the energy calculated on the original resolution using the original unaveraged grey levels.

We define the object contour in terms of edge pixels. Also, we re-define the gradient energy function $u_2(x)$ so that the gradient operator for recognising edges is considered in both the vertical and horizontal directions:

$$u_2(x) = \sum_e \sum_o V_{eo}(x),$$

where the first sum is over all edge pixels and the second sum is over all exterior first-order neighbours of the edge pixel e. We construct the gradient potential function as $V_{eo}(x) = -|z_o - z_e|$ which means that the greater is the difference between the records at two adjacent pixels separated by part of the edge, the more negative is the contribution to the energy U.

Our algorithm proceeds as follows. First, we propose a configuration as the initial contour, and we call this current contour x. We then record the length of the contour, its edge pixels and its interior and exterior, where edge pixels are included in the interior. We suppose here that each pixel is a unit square so that the area of the object is the number of interior pixels and the length of the contour is the number of edge pixel sides which are adjacent to an exterior pixel. The means μ_1 and μ_2 and the common variance σ^2 of the Gaussian distributions that define the likelihood are estimated at the beginning of the process, by taking $\hat{\mu}_1$ to be the mean of the records at pixels in a chosen clear interior region, $\hat{\mu}_2$ to be the mean of the records at pixels in a chosen clear exterior region, and $\hat{\sigma}^2$ to be the unbiased pooled variance based on the records in these two areas:

$$\hat{\sigma}^2 = \frac{(n_1 - 1) \times (\text{variance of chosen interior}) + (n_2 - 1) \times (\text{variance of chosen exterior})}{n_1 + n_2 - 2},$$

where n_1 and n_2 are the number of pixels in the chosen interior and exterior area respectively.

We now give a description of the sampling procedure employed at each cascade stage. For this we need to state what we mean by an edge block and by a legal block. A block is called an edge block if it is part of the shape and if it is touched by an edge. A legal block is one of the edge blocks that can be removed from the shape, or one of the external first order blocks to the edge blocks that can be added to the shape without destroying the closed nature of the edge. The idea of legal blocks is illustrated in Figure 2.2.

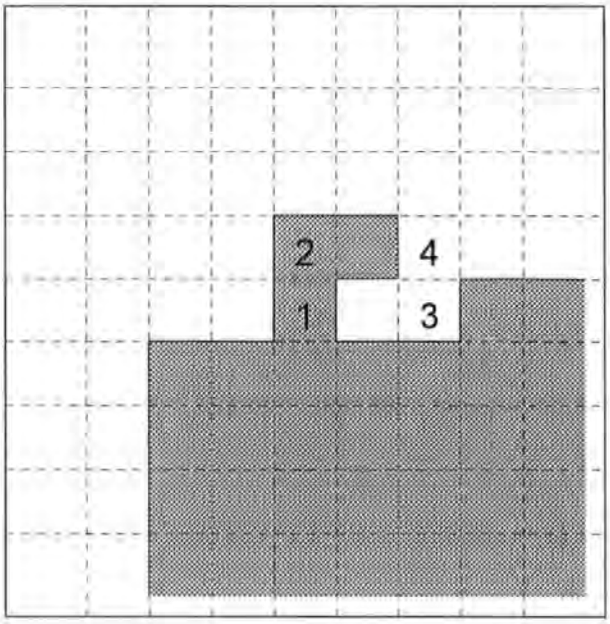


Figure 2.2 Legal blocks in part of a shape S shaded grey. Each square represents a block. The numbered blocks are not legal blocks since, if edge blocks 1 or 2 were removed from S or if blocks 3 or 4 were added to S, the edge would no longer be closed. The remaining edge blocks and their external first order neighbours are legal blocks.

In each stage of the cascade algorithm a fixed number of sweeps of the stochastic simulated annealing is performed, where a sweep is defined as an updating of all the edge blocks defined by the current configuration at the beginning of the sweep. The deterministic monotonically decreasing ICM algorithm of Besag (1986) is performed at the end of each cascade stage.

For both simulated annealing and ICM, we adopt the most common pixel updating schedule in which each edge pixel is considered in turn in a fixed order. Alternatively, each edge pixel could be considered at random in simulated annealing. The drawback of fixed order updating is that certain effects may occur because the next search pixel always follows the same pixel, while the drawback of random updating is that we cannot balance the visiting times to each pixel in a finite number of iterations. The updating schedule can also be defined for our problem as following: all edge blocks are recorded and randomly re-ordered at the start of every sweep, each edge block is then visited according to this random order in the sweep. However, in the simulation study presented in Section 2.4, we found out that there was no significant difference between fixed order updating and random order updating.

Let x be the current estimate of the edge, let

$$p_\tau(x) \propto \exp\left\{-\frac{U(x)}{\tau}\right\},$$

let `no.stages` be the number of blocking stages of the cascade algorithm and let `no.sweeps` be the number of sweeps of simulated annealing in each stage. We now describe the proposed algorithm for obtaining an estimate of the configuration that minimises $U(x)$.

```
Find an initial contour and set $x$ equal to it
For stage = 1,...,no.stages
    Start of simulated annealing
    For sweep = 1,...,no.sweeps
        Let $N_{sweep}$ be the number of edge blocks in $x$
        Label the edge blocks $1,\dots,N_{sweep}$
        For $\mathrm{b} = 1,\ldots,N_{\mathrm{sweep}}$
            Set the temperature $\tau$
            Find the number $n_{\mathrm{b}}$ of legal blocks contiguous to block $\mathrm{b}$ and label these blocks $1,\ldots,n_{\mathrm{b}}$
            Let the new configurations obtained by removing or adding the legal blocks as appropriate be $x_1', x_2', \ldots, x_{n_{\mathrm{b}}}'$
            Sample one of the configurations $x_1', x_2', \ldots, x_{n_{\mathrm{b}}}'$, $x$ with probabilities proportional to $p_\tau(x_1'), p_\tau(x_2'), \ldots, p_\tau(x_{n_{\mathrm{b}}}'), p_\tau(x)$
            Set $x$ to the sampled configuration
        End for
        End of one sweep
    End for
    End of simulated annealing part of one cascade blocking stage
    Start of ICM
    Let $N_{\mathrm{icm}}$ be the number of edge blocks in $x$
    Label the edge blocks $1,\ldots,N_{\mathrm{icm}}$
    Set change $= 0$
    For $\mathrm{b} = 1,\ldots,N_{\mathrm{icm}}$
        Find the number $n_{\mathrm{b}}$ of legal blocks contiguous to block $\mathrm{b}$ and label these blocks $1,\ldots,n_{\mathrm{b}}$
        Let the new configurations obtained by removing or adding the legal blocks as appropriate be $x_1', x_2', \ldots, x_{n_{\mathrm{b}}}'$
        Calculate the energy $U(x_1'), U(x_2'), \ldots, U(x_{n_{\mathrm{b}}}'), U(x)$
        Set $x$ to be the new current configuration with the lowest energy
```

```
            If there has been a change in $x$, then $\texttt{change} = \texttt{change} + 1$
        End for
        If $\texttt{change} > 0$, then go to the start of ICM
        **End of ICM**
        **End of one cascade blocking stage**
End for
**End of algorithm**
```

We shall discuss how the initial contour is found in Section 2.4. Convergence is guaranteed for ICM and this occurs when $\texttt{change} = 0$.

In the sampling part of the above algorithm, we set the candidate configuration x_i' to be the new current x with probability a_i and retain the current x with probability $1 - \sum_{i=1}^{n_b} a_i$, where

$$a_i = \frac{p_\tau(x_i')}{p_\tau(x) + \sum_{j=1}^{n_b} p_\tau(x_j')}.$$

Note that the unknown normalising constant for p_τ is not required for the calculation of a_i.

For direct simulated annealing, the temperature sequence τ_t is chosen according to the straight line schedule (see Stander and Silverman, 1994, for example) as we described in Section 1.3.4:

$$\tau_t = \frac{l - f}{T - 1}(t - 1) + f, \tag{2.3}$$

where t is the update number, T is the total number of iterations for simulated annealing, and f and l are the starting and ending temperatures respectively. The other temperature schedules described in Chapter 1 can also be applied to the cascade algorithm, but the results are almost the same. Hurn and Jennison (1995) consider similar schedules that sometimes offer an advantage when used with cascade. We now propose three temperature schedules derived from the linear schedule for the cascade based algorithm.

- Monotonically decreasing schedule

  The update number t is increased through all blocking stages of the cascade except the ICM parts. We define f and l only once at the beginning of the whole procedure. In this case, the total number T of iterations for simulated annealing is estimated as

$$T \approx \texttt{no.sweeps} \times N_0 \times \sum_{s=1}^{\texttt{no.stages}} 2^{s-1}$$

  where N_0 is the number of edge blocks in the initial configuration for the first stage. There are two possible temperature situations at the end of the procedure: the procedure stops before it reaches l, or the procedure reaches l before stopping. We do not mind the former situation, but for the later situation, we suggest that the procedure is kept running at the lowest temperature l until it stops. An example of the resulting temperature schedule for three blocking stages is shown in Figure 2.3(a).

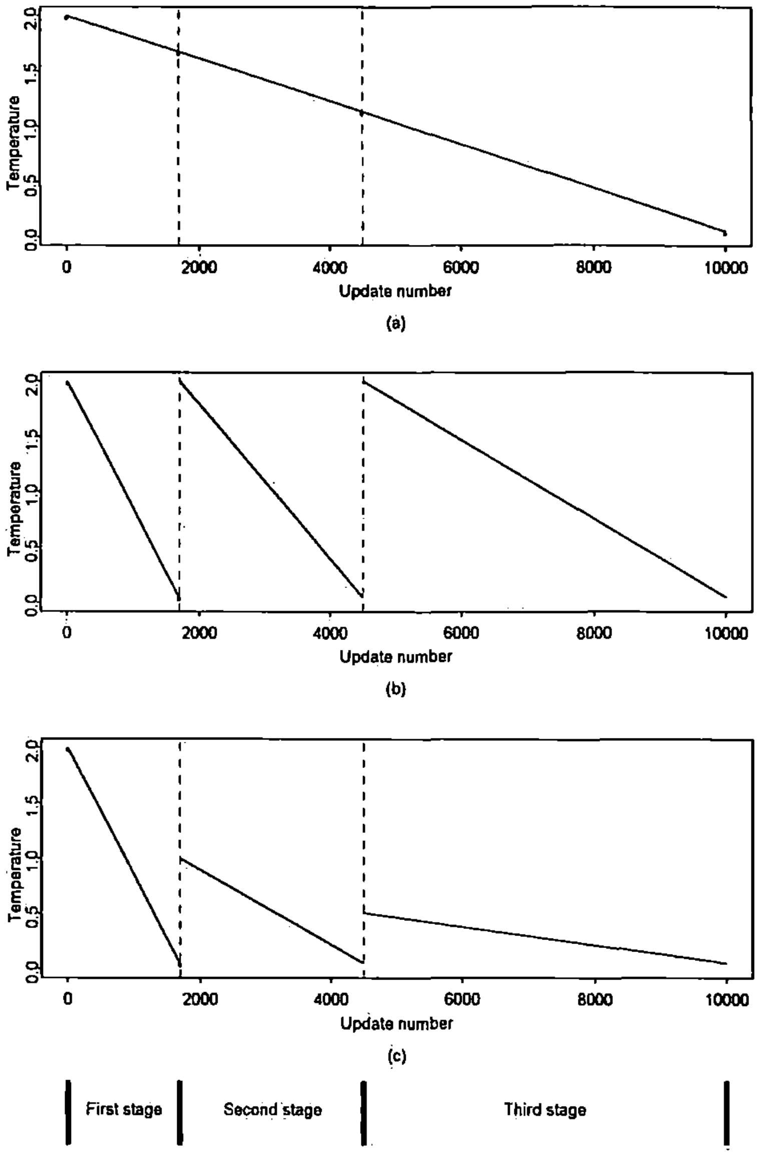


Figure 2.3 Cascade temperature schedules. (a) Monotonically decreasing linear schedule. (b) Independent linear schedule. (c) Cascade stage linear schedule.

- Independent schedule

  For each blocking stage of the cascade, the temperature starts from the same f and ends at the same l. An example of the resulting temperature schedule for three blocking stages is shown in Figure 2.3(b).

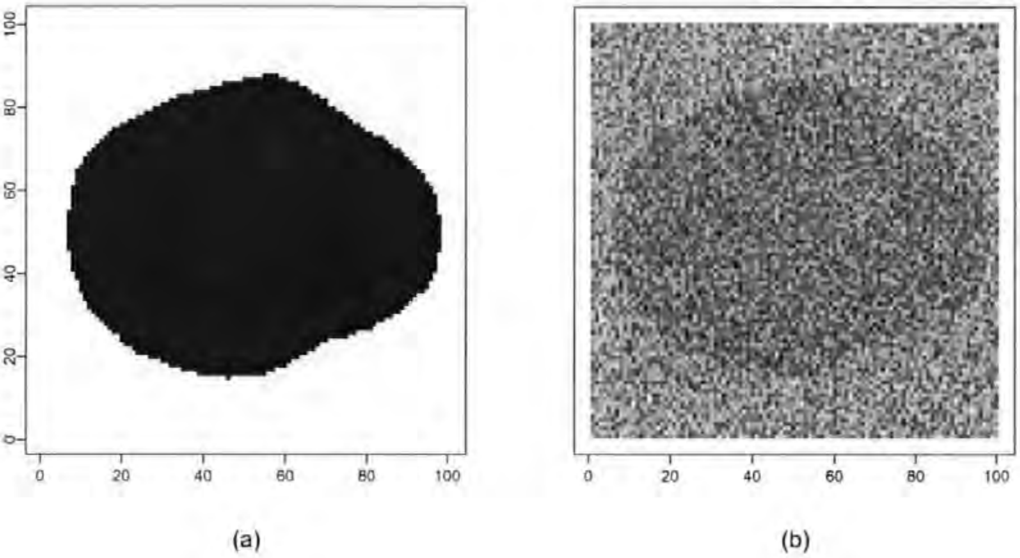
- Cascade stage schedule

  The third linear temperature schedule for the cascade blocking algorithm that we shall consider is similar to the second, but the first temperature f_s for s^{th} blocking stage of the cascade decreases as the stage s increases, i.e., $f_1 > f_2 > \cdots > f_{\text{no.stages}}$. For example, we can take $f_{s+1} = \frac{1}{2} f_s$. An example of the resulting temperature schedule for three blocking stages is shown in Figure 2.3(c).

In the next section we present the results of a simulation study designed to illustrate the above algorithm.

2.4 *Simulation Study*

This simulation study is based on the head shape shown in Figure 2.4(a), which is one of the head shapes considered by Wright, Stander and Nicolaides (1997). This original image comprises 100×100 pixels and is a binary image; black pixels (interior) correspond to the numerical value 1, whereas white pixels (exterior) correspond to 0. The simulated data, obtained by adding independent $N(0,1)$ noise to the original image, is presented in Figure 2.4(b).

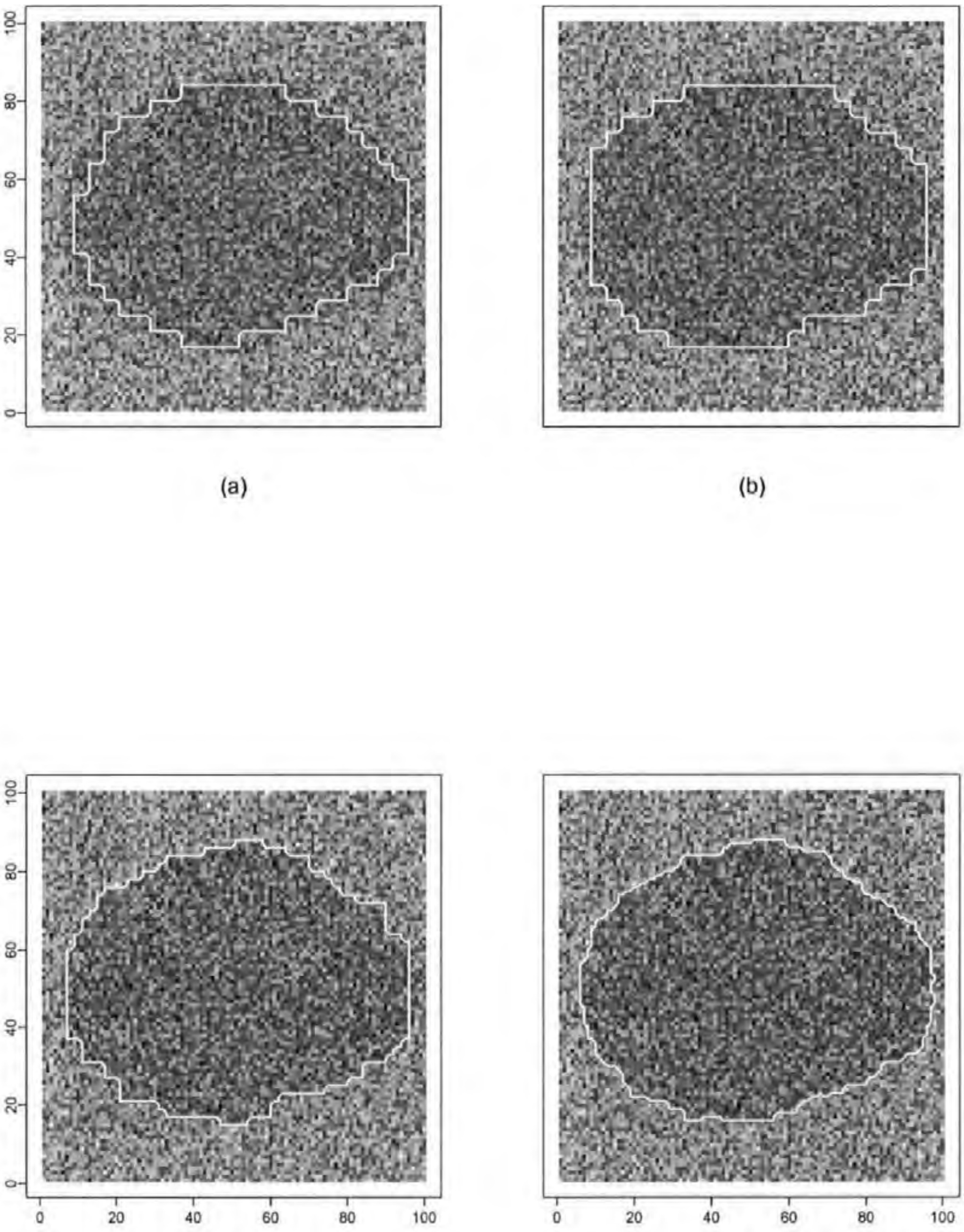


(a)

(b)

Figure 2.4 (a) The known head shape. This 100×100 binary image displays one of the heads considered by Wright *et al.* (1997). (b) The simulated data obtained by adding independent $N(0,1)$ noise to the image (a).

We apply the cascade algorithm described above with three blocking stages to the image presented in Figure 2.4(b) in order to estimate the edge of the shape. The number of sweeps of simulated annealing is taken to be ten at each blocking stage. In our experience, ten sweeps of simulated annealing at each cascade blocking stage are often sufficient to obtain good quality results. The weight α_1 and α_2 of the components that contribute to the energy function (2.2) are chosen to be 8 and 0.01 respectively. We set $f = 2$ and $l = 0.1$ and use the independent cascade linear temperature schedule.



(a) (b) (c) (d)

Figure 2.5 (a) The initial contour obtained by first thresholding the record averaged over the sixteen pixels of each block and then finding the convex hull. (b) The result of the first cascade stage on the 25×25 grid. (c) The result of the second cascade stage on the 50×50 grid using the edge found in (b) as starting point. (d) The result of the final cascade stage on the original 100×100 grid using the edge found in (c) as starting point.

The first stage of the cascade is performed on a 25×25 grid. Each block of this grid comprises 4×4 original pixels. Figure 2.5(a) shows the initial contour obtained by first thresholding the record averaged over the sixteen pixels of each block using the intermeans algorithm discussed by Glasbey and Horgan (1995), pp. 97–98, and then finding the convex hull of the resulting data using the `chull` function of S-Plus. The intermeans method worked well here, but is not always guaranteed to do so. Other thresholding methods are discussed in Glasbey and Horgan (1995). This contour obtained above serves as the initial contour of our cascade algorithm. The original 100×100 record z is used in the evaluation of the energy function U throughout the algorithm. Figure 2.5(b) is the result of simulated annealing followed by ICM for the first cascade stage on the 25×25 grid. The second cascade stage is performed on a 50×50 grid, each block of which comprises 2×2 original pixels. The initial contour for this cascade stage is the one that resulted from the first cascade stage and that is shown in Figure 2.5(b). Figure 2.5(c) presents the result of the second cascade stage on the 50×50 grid. The final cascade stage is performed on the original 100×100 grid with the initial contour obtained from the second cascade stage shown in Figure 2.5(c). Figure 2.5(d) is the result of simulated annealing followed by ICM for the final cascade stage on the original grid.

The final result produced by the above cascade algorithm may be considered a little too rough. Indeed, Figure 2.6(a), which presents the true shape (darker line) together with the contour produced by the cascade algorithm, confirms this. A smoother result could be obtained by re-running the algorithm with different weights. Alternatively, and more pragmatically, we smooth the edge shown in Figure 2.5(d) by using a median smoothing operator. The method is based on the shape S defined by edge. A pixel defined to be part of S has its definition changed if it has more than five first- and second-order neighbours defined to be outside S; a similar treatment is given to pixels defined to be outside S. Once these changes have been made, the smoothed edge is defined as the boundary of the modified S. The result of this smoothing procedure is presented in Figure 2.6(b) together with the true edge (darker line). The number of pixels that differ between the true shape and the estimated shape reduced from 160 before smoothing to 152.

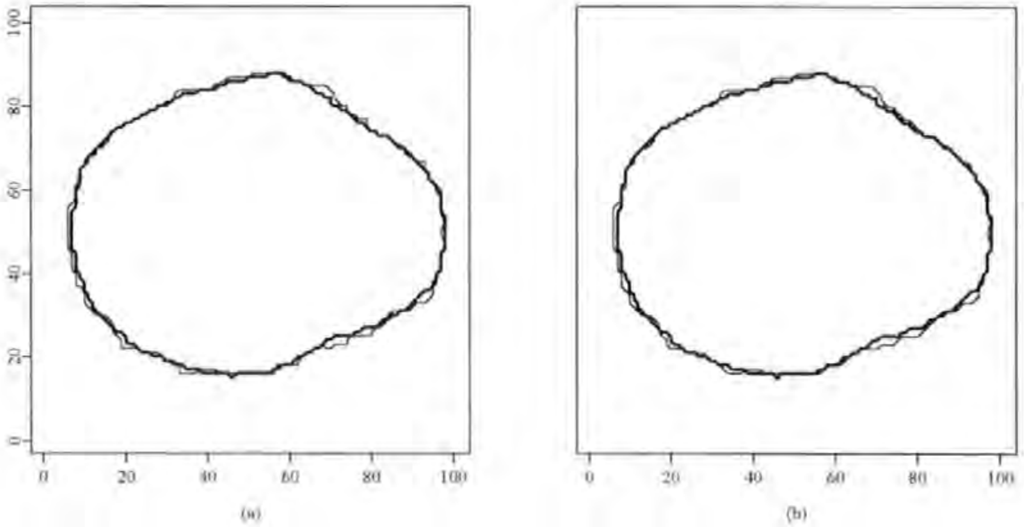


Figure 2.6 (a) The edge produced by the cascade procedure together with the true edge (darker line). (b) The result after applying a median smoothing operator to (a) together with the true edge (darker line).

We see that a good estimate has been obtained from the highly degraded image shown in Figure 2.4(b).

The whole procedure with smoothing took 4349 seconds DOS-time on a Pentium 75MHz PC. Figure 2.7 presents a plot of the value of the energy function U against update number as the algorithm proceeds. The unbroken vertical lines mark the stages of the cascade algorithm, whereas the broken vertical lines mark the beginning of each ICM phase. The plot clearly shows that simulated annealing allows increases in U, whereas ICM does not. As we would expect, there is an increase in energy due to smoothing.

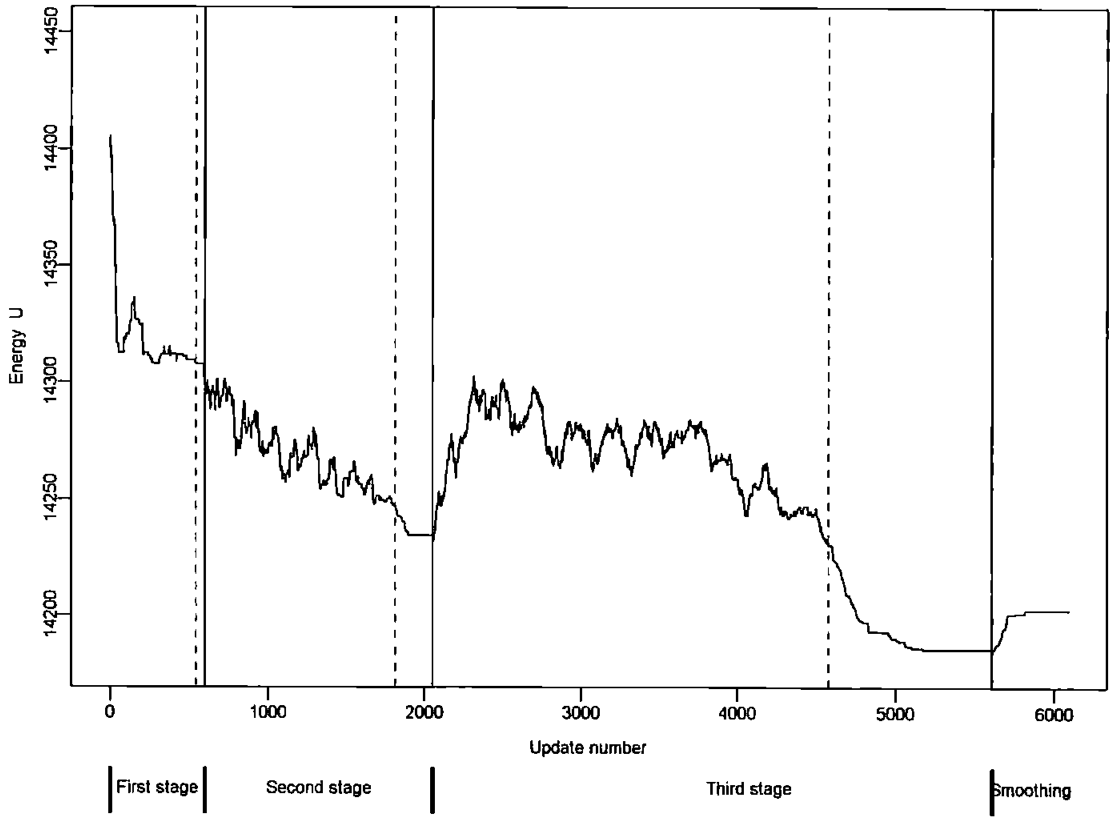
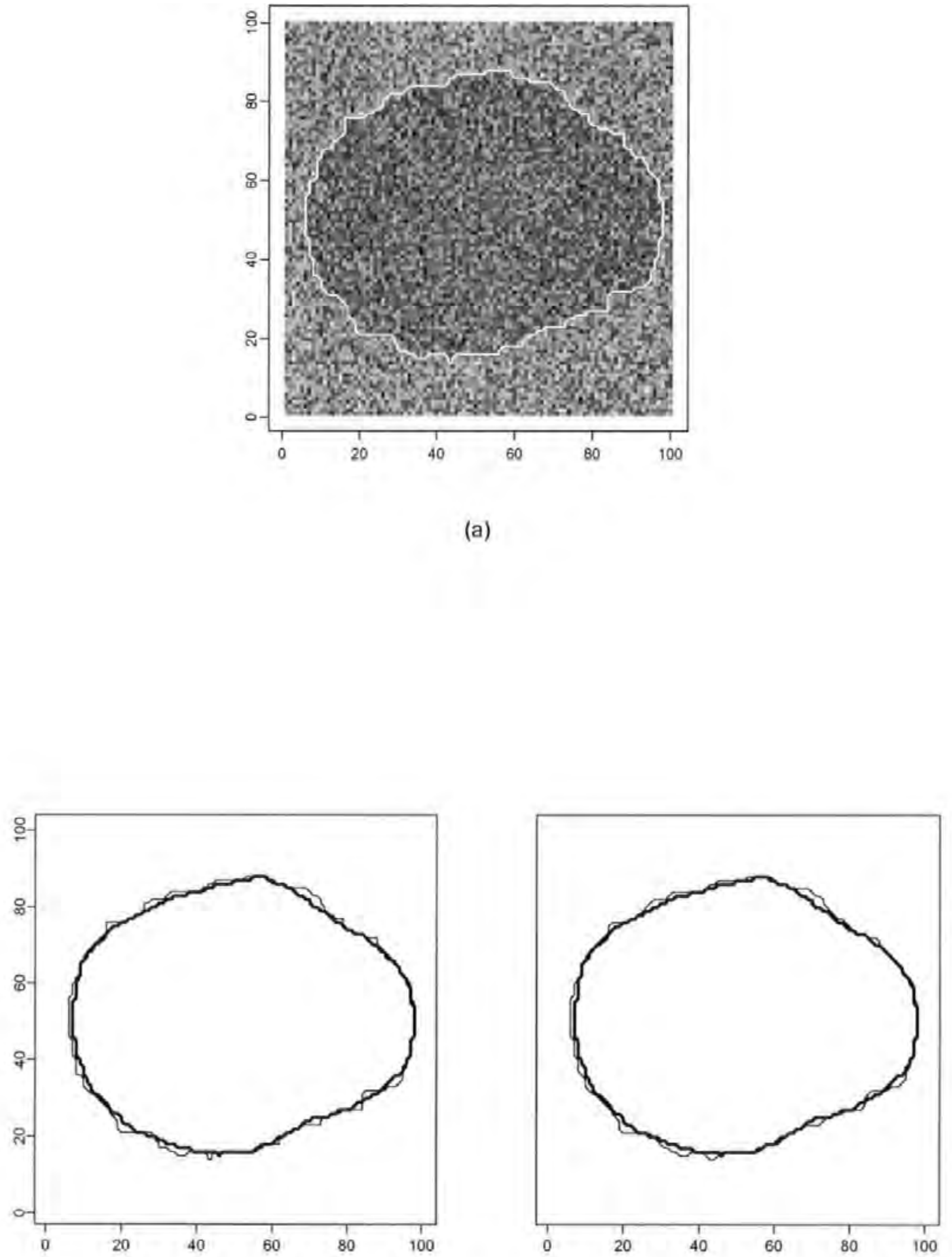


Figure 2.7 The value of the energy function U against update number for the three stages of the cascade algorithm followed by smoothing. The unbroken vertical lines mark the stages of the cascade algorithm, whereas the broken vertical lines mark the beginning of ICM.

In order to compare our cascade-type blocking algorithm with the direct simulated annealing approach, we applied the non-cascade simulated annealing algorithm to the noisy image shown in Figure 2.4(b). We used the contour shown in Figure 2.5(a) as the initial contour. Since this contour was obtained using blocking, it may be considered to be a generous starting point for the direct simulated annealing approach. We set α_1 and α_2 to be the same values as used above. The temperature schedule is the straight line schedule. The values of f and l are unchanged. The number of sweeps of simulated annealing was set to thirty, the total number of sweeps of simulated annealing employed in the cascade-type blocking algorithm. The ICM algorithm followed by median smoothing was again applied at the end of the algorithm.

Figure 2.8(a) presents the result obtained by using the non-cascade simulated annealing algorithm. Figure 2.8(b) presents the true contour (darker line) together with the contour produced by the direct procedure. The median smoothing operator is employed to the rough contour and the result of this smoothing operation is presented in Figure 2.8(c) together with the true edge (darker line). This whole procedure with smoothing took 6761 seconds DOS-time on a Pentium 75MHz PC, which is about 1.5 times as much computing time as the three stages cascade algorithm.

Figure 2.9 presents a plot of the value of the energy function U against update number as the algorithm proceeds. The broken vertical line marks the beginning of ICM, whereas the unbroken vertical line indicates the start of smoothing. The lower horizontal line (long and short dashes) marks the final energy produced by the above cascade algorithm before smoothing, while the higher horizontal line (long dashes) marks the corresponding energy after smoothing. The final energies of the two methods are quite different: 14202 for the three stages cascade algorithm with smoothing and 14244 for the non-cascade algorithm with smoothing, while their lowest energies are 14185 and 14223 respectively.



(a)

(b)

(c)

Figure 2.8 (a) Edge estimate obtained by using the non-cascade simulated annealing algorithm. (b) The edge produced by the direct procedure together with the true edge (darker line). (c) The result of applying a median smoothing operator to the edge produced by the direct procedure together with the true edge (darker line).

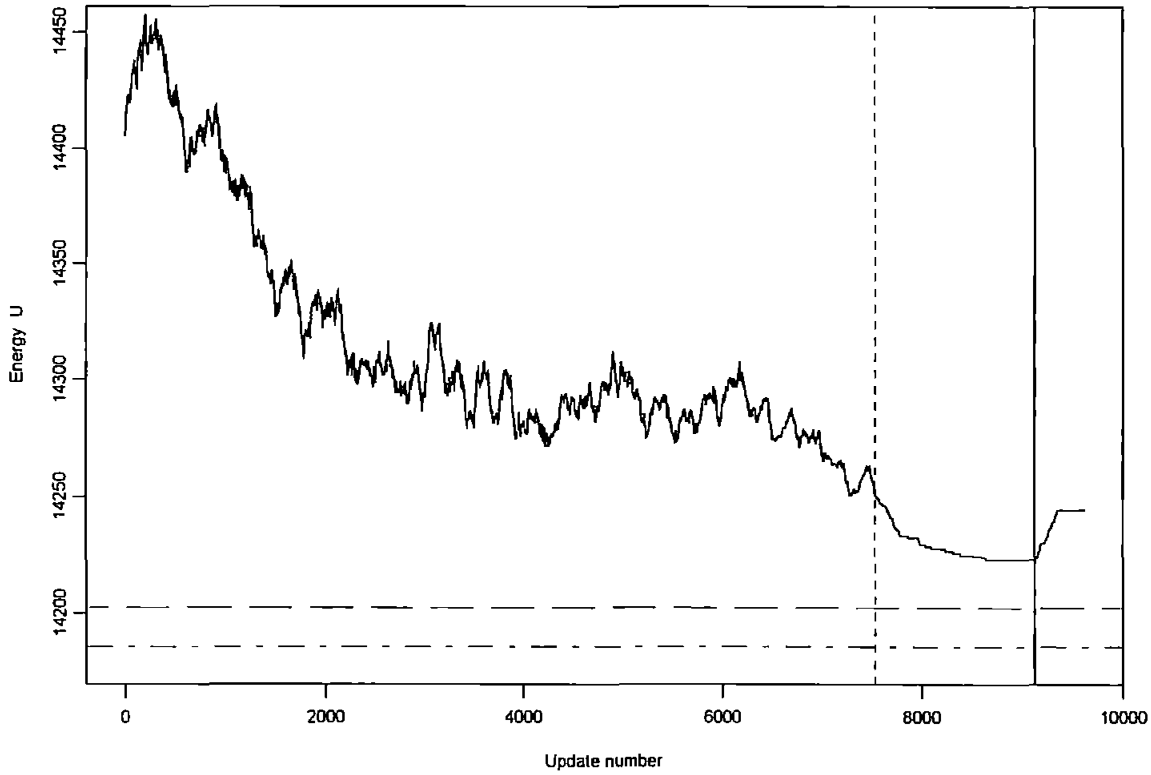
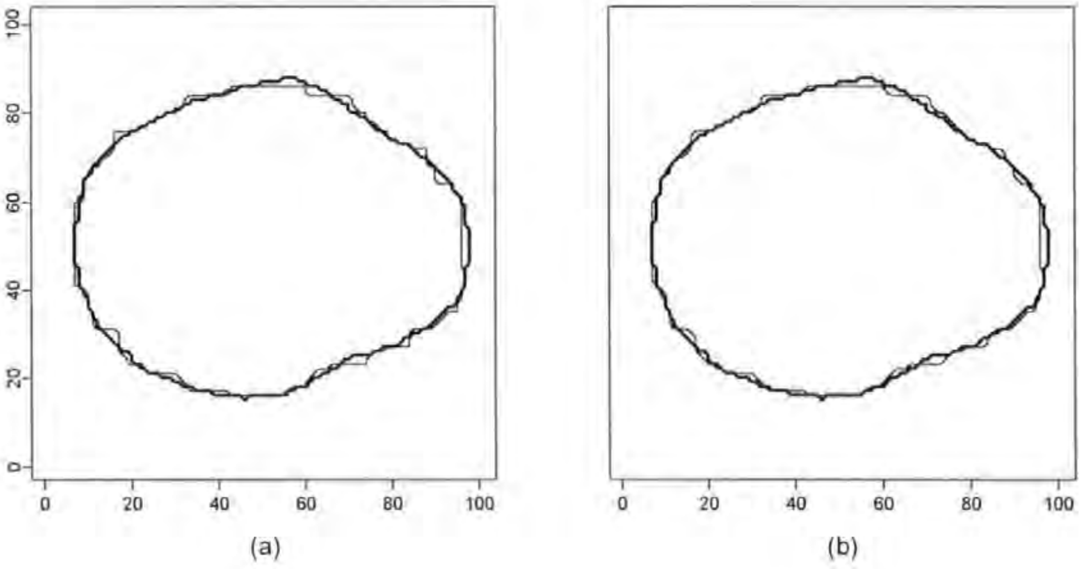


Figure 2.9 The value of the energy function U against update number for the direct simulated annealing algorithm followed by ICM and smoothing. The broken vertical line marks the beginning of ICM, whereas the unbroken vertical line indicates the start of smoothing. The lower horizontal line (long and short dashes) marks the final energy produced by the above cascade algorithm before smoothing, while the higher horizontal line (long dashes) marks the corresponding energy after smoothing.

Even with several thousand more updates the direct simulated annealing approach does not achieve a minimum as low as the cascade algorithm. Moreover, the direct simulated annealing approach takes about 1.5 times as much computing time as the cascade algorithm. The number of pixels that differ between the true shape and the estimated shape has risen from 152 to 175 (from 160 to 189 before smoothing). We remark that the resulting edge estimates are quite similar, with the one from the non-cascade algorithm being slightly less good than the one from the cascade algorithm; see Figure 2.6(b) and Figure 2.8(c).

In the above cascade simulation, we employed the independent linear temperature schedule for the three cascade stages. Another two cascade temperature schedules, the monotonically decreasing linear schedule and the cascade stage linear schedule, were introduced in Section 2.3. These three cascade schedules are illustrated in Figure 2.3. In order to compare the three cascade temperature schedules, we re-ran the cascade algorithm with the other two schedules on the same noisy image shown in Figure 2.4(b).



(a) (b)

Figure 2.10 Results of employing the monotonically decreasing linear temperature schedule in the cascade algorithm. (a) The edge produced by the cascade procedure together with the true edge (darker line). (b) The result of applying a median smoothing operator to the edge produced by the cascade procedure together with the true edge (darker line).

Figure 2.10(a) presents the result of employing the monotonically decreasing linear temperature schedule in the cascade algorithm (without median smoothing). Figure 2.10(b) shows the result of applying a median filter to this edge estimate. The number of pixels that differ between the true shape and the estimated shape is 160 (175 before smoothing). Figure 2.11 presents a plot of the value of the energy function U against update number. The final energy is 14194 and its lowest energy is 14179.

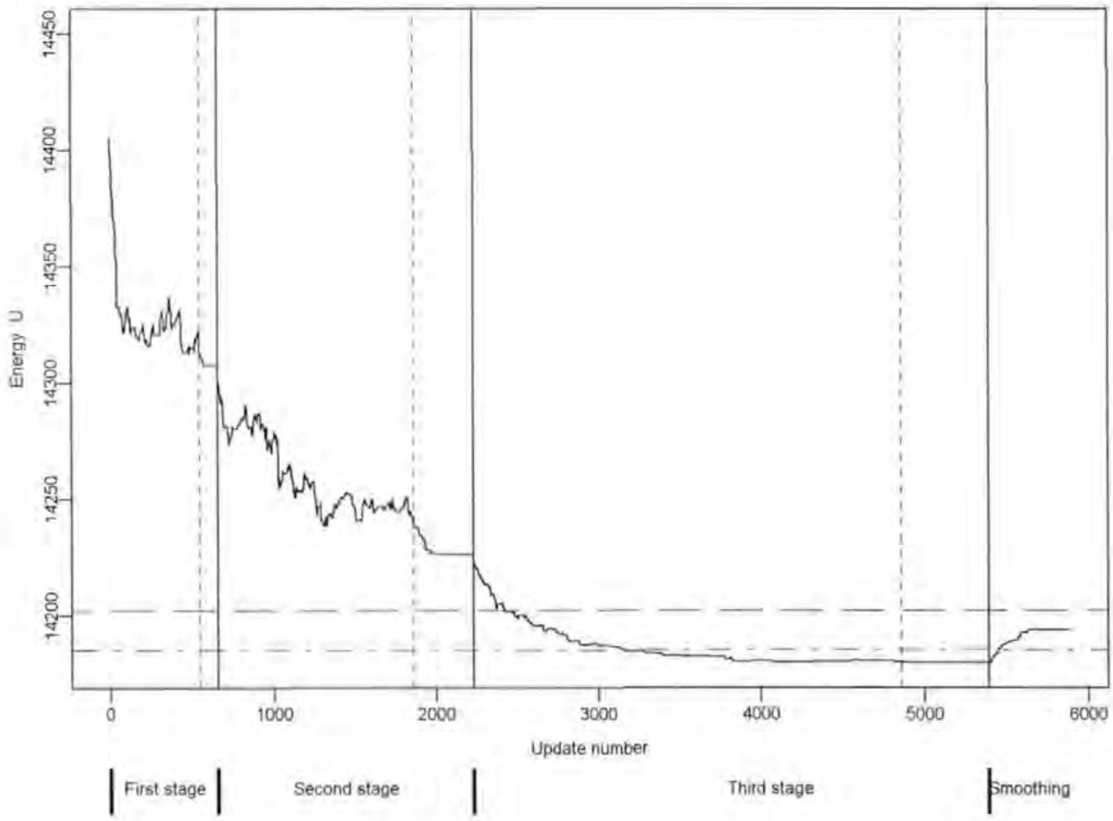


Figure 2.11 The value of the energy function U against update number for the three stages of the cascade algorithm with the monotonically decreasing linear temperature schedule followed by smoothing. The unbroken vertical lines mark the stages of the cascade algorithm, whereas the broken vertical lines mark the beginning of ICM. The lower horizontal line (long and short dashes) marks the final energy produced by the cascade algorithm with the independent linear temperature schedule before smoothing, while the higher horizontal line (long dashes) marks the corresponding energy after smoothing.

Figure 2.12(a) presents the result of employing the cascade stage linear temperature schedule in the cascade algorithm (without median smoothing). Figure 2.12(b) shows the result of applying a median filter to this edge estimate. The number of pixels that differ between the true shape and the estimated shape is 169 (179 before smoothing). Figure 2.13 presents a plot of the value of the energy function U against update number. The final energy is 14197 and its lowest energy is 14182.

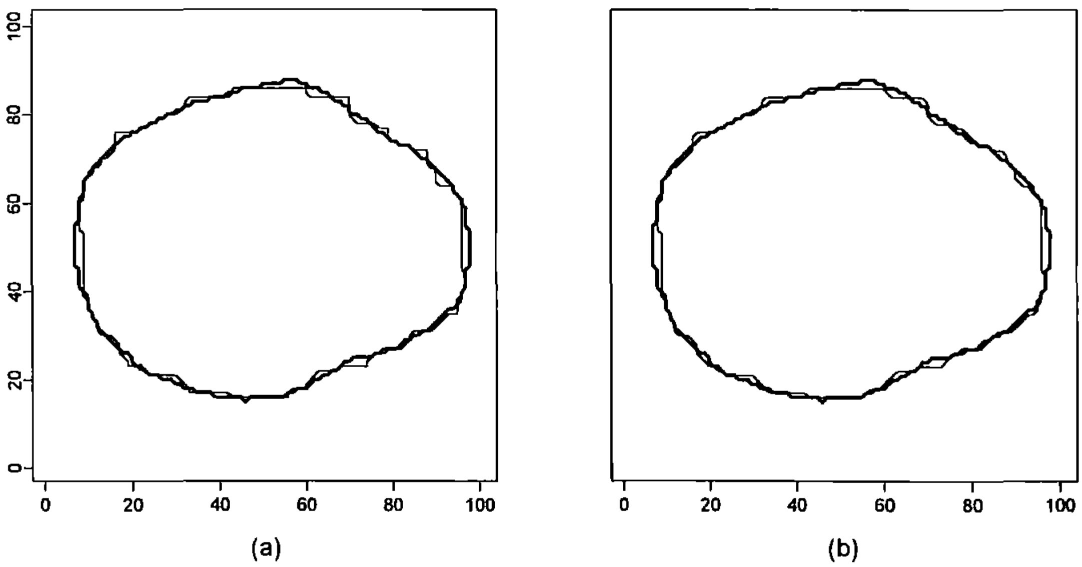


Figure 2.12 Results of employing the cascade stage linear temperature schedule in the cascade algorithm. (a) The edge produced by the cascade procedure together with the true edge (darker line). (b) The result of applying a median smoothing operator to the edge produced by the cascade procedure together with the true edge (darker line).

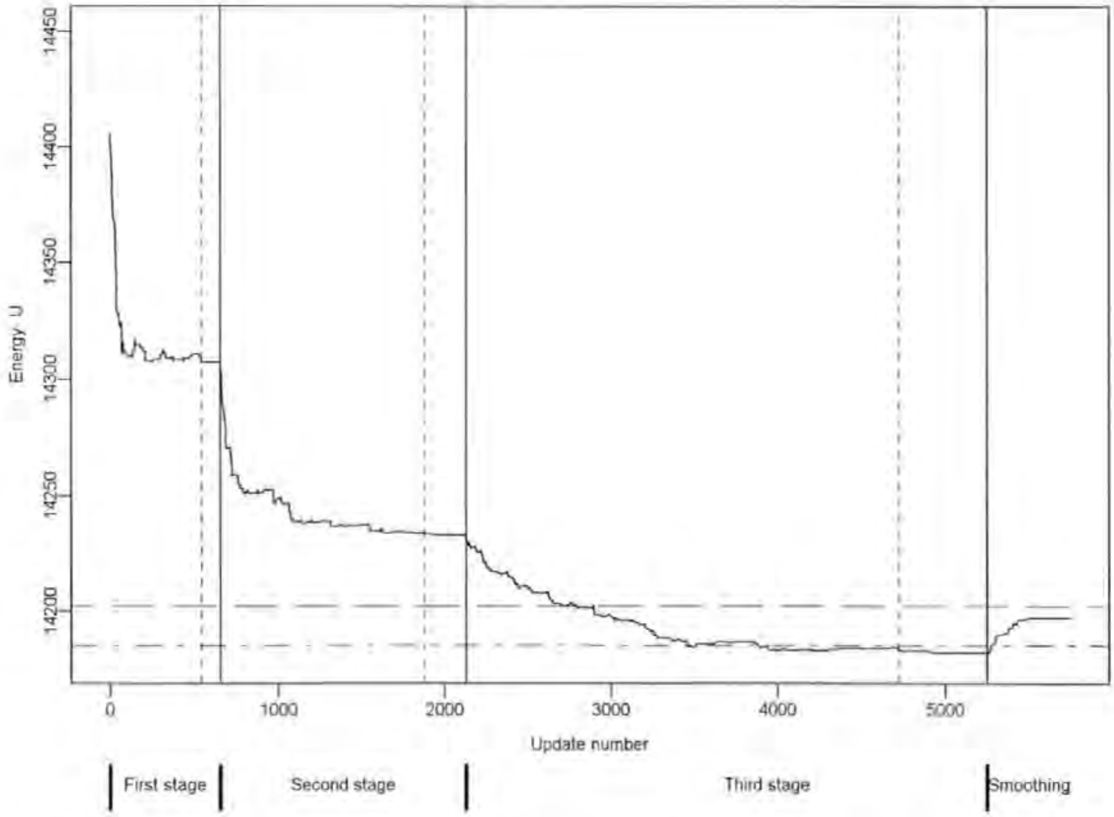


Figure 2.13 The value of the energy function U against update number for the three stages of the cascade algorithm with the cascade stage linear temperature schedule followed by smoothing. The unbroken vertical lines mark the stages of the cascade algorithm, whereas the broken vertical lines mark the beginning of ICM. The lower horizontal line (long and short dashes) marks the final energy produced by the cascade algorithm with the independent linear temperature schedule before smoothing, while the higher horizontal line (long dashes) marks the corresponding energy after smoothing.

For ease of comparison in Table 2.1 we present the results of applying the cascade algorithm with the three cascade temperature schedules and of applying the non-cascade algorithm. By the number of differing pixels we mean the number of pixels that differ between the true shape and the estimated shape. In Figure 2.14 we display these results by means of bar plots.

Table 2.1 Results of applying the cascade algorithm with the three cascade linear temperature schedules and of applying the non-cascade algorithm on the noisy image shown in Figure 2.4(b)

Algorithm	Cascade Temperature schedule	Before smoothing		After smoothing	
		Energy	Number of Differing pixels	Energy	Number of Differing pixels
Cascade	Independent	14185	160	14202	152
	Monotonically decreasing	14179	175	14194	160
	Cascade stage	14182	179	14197	169
Non-cascade		14223	189	14244	175

We can see that the final energy before smoothing and after smoothing obtained when using the monotonically decreasing and the cascade stage temperature schedules are lower than the corresponding energy obtained when using the independent temperature schedule. Even so, the independent temperature method gives us a little smoother result. There is not much difference between the final energies in the cascade algorithm in this example. One message from these results is that the lowest energy edge does not necessarily correspond to the best edge. We have found this result to be especially true for images that have been degraded by high levels of noise.

The result of applying the non-cascade algorithm is slightly less good than the results of applying the cascade algorithms in this example. The edge that results from the non-cascade algorithm has much higher energy than those that result from the cascade algorithms. We remark that the resulting edge estimates are quite similar; see Figures 2.6(b), 2.8(c), 2.10(b) and 2.12(b).

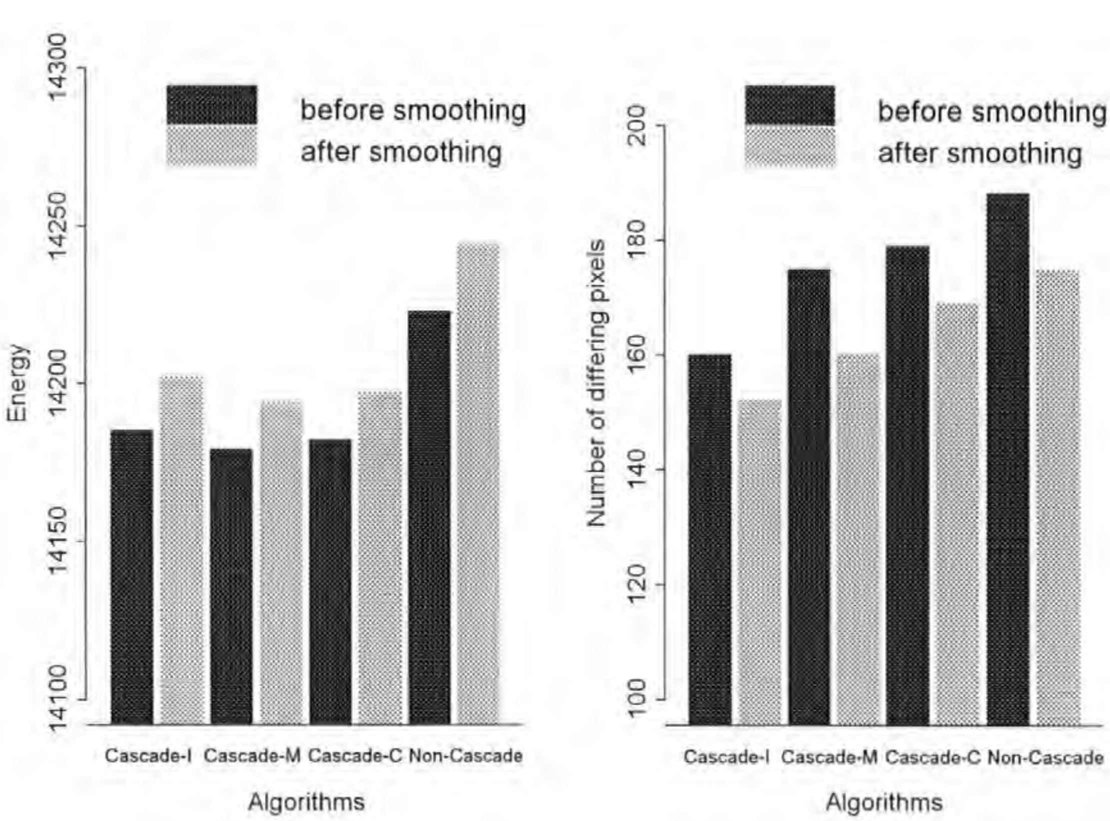


Figure 2.14 Bar plots of the energies and number of differing pixels before and after smoothing using four different algorithms. Cascade-I ≡ cascade with independent linear schedule, Cascade-M ≡ cascade with monotonically decreasing linear schedule, Cascade-C ≡ cascade with cascade stage linear schedule.

In order to make a wide comparison of the performance of the cascade-based algorithm with that of the direct simulated annealing algorithm, we conducted a simulation study based on the original image shown in Figure 2.4(a). Our data were obtained by adding independent Gaussian noise to the image. For each of three different noise variances $\kappa = 0.5, 1.0$ and 2.0, we applied both the cascade-based algorithm with independent linear temperature schedule and the direct simulated annealing algorithm to 100 noisy images. The mean and the standard deviation (sd) of the number of pixels that differ between the true shape and the estimated shape over realisations of the noise process are recorded in Table 2.2.

Table 2.2 Results of the simulation study to compare the cascade-based algorithm with the direct simulated annealing algorithm

	Cascade		Direct		
variance κ	mean (pixels)	sd (pixels)	mean (pixels)	sd (pixels)	p-value
0.5	101.9	15.2	100.7	17.0	0.41
1.0	182.4	24.4	204.3	30.3	0.00
2.0	275.1	43.0	341.7	71.3	0.00

Table 2.2 suggests that for the case $\kappa = 0.5$ there is little difference in the performance of the algorithms. To check this we tested $H_0: \mu_{\text{cascade}} = \mu_{\text{direct}}$ against $H_1: \mu_{\text{cascade}} \neq \mu_{\text{direct}}$ using a paired t-test, where μ_{cascade} and μ_{direct} are the mean number of pixels that differ between the true shape and the estimated shape for the cascade-based algorithm and the direct simulated annealing algorithm; we did not reject H_0. As κ increases the quality of the performance of the cascade-based algorithm decreases less rapidly than that of the direct simulated annealing; the p-value of a paired t-test of $H_0: \mu_{\text{cascade}} \geq \mu_{\text{direct}}$ against $H_0: \mu_{\text{cascade}} < \mu_{\text{direct}}$ is essentially zero for both $\kappa = 1.0$ and $\kappa = 2.0$. Hence the use of the cascade-based method leads to a significant increase in the performance of the algorithm when the noise level is high. Moreover, we found that the cascade-based algorithm is about 1.5 times faster than the direct simulated annealing approach.

In order to see whether there is a significant difference between the fixed and the random edge blocking visiting schedule described in Section 2.3, we conducted a simulation study based on the cascade algorithm and the original image shown in Figure 2.4(a). Sixteen noisy images were obtained by adding independent standard Gaussian noise to that image. We performed the simulated annealing procedure (with ICM and smoothing) on these images using both fixed and random visiting schedules. The number of pixels that differ between the true shape and the estimated shape over realisations of the noise process is listed in Table 2.3. The mean number of differing pixels is 191 for the fixed and 186 for the random visiting schedule. The p-value of a paired t-test of $H_0: \mu_{\text{fixed}} = \mu_{\text{random}}$ against $H_1: \mu_{\text{fixed}} \neq \mu_{\text{random}}$ is 0.4327, where μ_{fixed} and μ_{random} are the mean number of pixels that differ between the true shape and the estimated shape for fixed and random visiting schedules respectively. Hence there is no statistically significant difference between the fixed and random visiting schedules.

Table 2.3 The number of pixels that differ between the true shape and the estimated shape over realisations of the noise process in 16 noisy images for fixed and random edge block visiting schedule

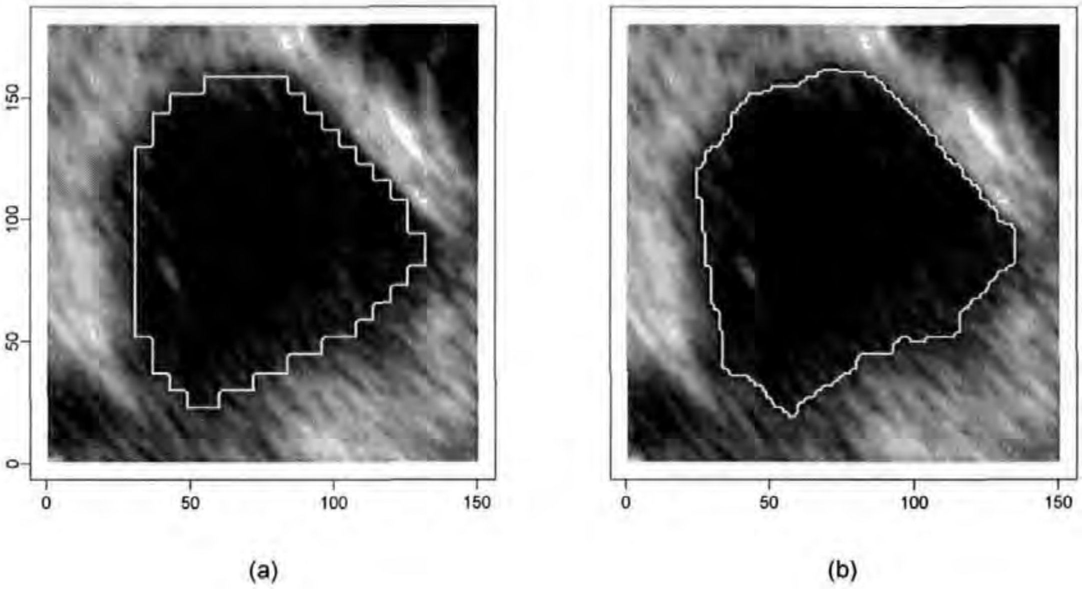
Sample	Fixed	Random	Sample	Fixed	Random
1	170	186	9	211	198
2	251	263	10	140	159
3	238	180	11	219	185
4	159	130	12	179	212
5	175	197	13	188	177
6	178	152	14	161	156
7	183	200	15	239	191
8	163	187	16	196	188
			Mean	191	186
			SD	29.6	32.4

2.5 *Application to real data*

In this section we apply the cascade based simulated annealing algorithm and the non-cascade simulated annealing algorithm to the real ultrasound image of a human ovarian cyst that is presented in Figure 1.8.

Figure 2.15(a) presents this real ultrasound image of a human ovarian cyst together with an initial contour. The cyst can be defined by its edge and pixels inside and outside the cyst have different records. We took $\alpha_1 = 8$ and $\alpha_2 = 0.01$, and we set $f = 2$ and $l = 0.1$. We considered a three stage cascade procedure with the monotonically decreasing linear temperature schedule followed by smoothing. The number of sweeps of simulated annealing was taken to be ten at each blocking stage. The initial contour shown in panel (a) is obtained by first thresholding the record averaged over the sixteen pixels of each block using the intermeans algorithm and then finding the convex hull of the resulting data using the `chull` function of S-Plus. The white line in panel (b) is the final estimate of the outline of the shape by the three stage cascade based algorithm.

In order to make a comparison with the non-cascade algorithm, we applied the direct simulated annealing algorithm to the ultrasound image using the same values of parameters and starting from the same initial contour. The number of sweeps was set to thirty. The result is presented in panel (b) of Figure 2.15 by means of the black line. We see that the results obtained from these two algorithms are almost the same. However, the non-cascade procedure took 5639 seconds DOS-time on a Pentium 75MHz PC and the final energy is 51325, whereas the cascade procedure took only 3721 seconds DOS-time and the final energy is 51268.



(a)

(b)

Figure 2.15 (a) A real ultrasound image of a human ovarian cyst together with the initial contour. (b) Estimate of the cyst shape produced by a three stage cascade procedure (white outline) together with the estimate produced by the direct simulated annealing algorithm (black outline).

The cascade based algorithm seems well able to define the edge of the underlying shape in the ultrasound image. In Section 3.8 of the next chapter, we present the result of applying another algorithm to this ultrasound image. That algorithm is a modified version of the kernel algorithm that we present in the initial sections of Chapter 3.

We believe that the algorithms presented in this chapter and in Chapter 3 would work well on the ultrasound image of the left ventricle presented in Storvik (1994); we have, however, been unable to obtain Storvik's data.

2.6 *Discussion*

We have suggested a cascade-based modification for increasing the speed of an algorithm proposed by Storvik (1994) suitable for edge detection when the shape is defined by its edge and pixels inside and outside the shape have different records. We have illustrated by means of a simulated example that the use of the cascade-based method leads to a considerable increase in the speed of the algorithm. Three different temperature schedules for the cascade based simulated annealing algorithm have been considered and the results (see Table 2.1) tell us that the monotonically decreasing cascade temperature schedule can lead to the edge with the lowest energy. However, in our simulation the independent cascade temperature schedule leads to an edge whose associated shape has the lowest number of differing pixels, although the final energy of the edge is a little higher than that obtained by using the monotonically decreasing schedule. We have also applied our algorithm to real data with success.

From Table 2.2 we may conclude that the cascade based method leads to a significant increase in the performance of the algorithm when the noise level is high.

This algorithm would not work well for the real ultrasound fetal head image given in Figure 1.2(a) because the grey-levels of pixels inside and outside the head shape seem to have very similar distributions. In fact, the algorithm described in this chapter seems to work well only for solid shapes such as the one presented in Figure 2.4(a). Even if we set $u_3(x) \equiv 0$, that is, we disregard the likelihood term, the algorithm does not work well since the smoothness energy $u_1(x)$ and the gradient operator $u_2(x)$ do not satisfactorily detect the edge.

Another problem that the image given in Figure 1.2(a) presents is the obliteration of the edge of the head in certain regions caused by high levels of degradation. The kernel algorithm that we present in the next chapter enables us to overcome this problem and to produce an acceptable estimate of the edge.

Chapter 3

Edge Detection Using Kernel Functions

3.1 *Definition of the kernel function*

In this chapter we consider images that consist of a single object whose shape is defined only by its edge; in other words, we consider images that comprise a closed curve degraded by noise. An example of such an image is shown in Figure 3.1 (This image was also shown in Figure 1.2(a)). In Figure 3.1 the head shape is defined by a thin outline of pixels with records that are different from those pixels that lie inside and outside the shape. The records at these interior and exterior pixels are, however, similar. Much degradation is present in Figure 3.1, and this has almost obliterated part of the outline. Accordingly, standard edge detection algorithms such as the Prewitt and Canny filters are not able to detect the head edge. In Chapter 1 we reported that the results from both the Prewitt filter and the Canny filter are poor; see Figure 1.2. In addition, such algorithms often yield many artefactual edges elsewhere in the image; see Qian, Titterington and Chapman (1996) for another example of this.

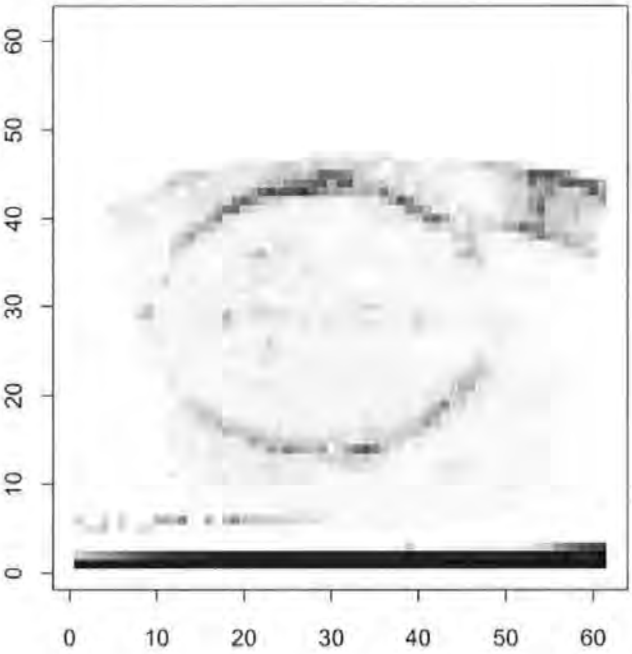


Figure 3.1 The ultrasound image of a cross-section of a fetal head.

The approach described in Chapter 2 does not consider objects of this type; see the discussion at the end of that chapter. A shape that is defined only by its edge presents special problems because the edge can be almost lost by high levels of degradation in certain regions, as can be seen in Figure 3.1. Accordingly, for a degraded closed curve image, such as an ultrasound fetal head shape, we need a special method to estimate the edge of the shape.

Recently, several authors have worked on similar problems. These include Bowtell and Patefield (1997) who develop techniques for fitting circular functional relationships, and Pursey and Taylor (1995) who present a method for edge detection called route tracing based on finding and following the perimeter of a shape. Pursey and Taylor (1995) apply their methods with considerable success to complicated digital images of fungal spores. Their images are not, however, subjected to high levels of degradation. Van Lieshout (1995) discusses the use of the generalised Hough Transform in object recognition; see also Illingworth and Kittler (1988) for a survey of the Hough transform. Rue and Husby (1997) use deformable templates and destructive deformation fields to identify partly destroyed human melanoma cancer cells with good results. Other authors have used methods based on mathematical morphology (see, for example, Glasbey and Horgan, 1995) for extracting measurements from ultrasound images. These include Thomas, Peters II and Jeanty (1991) who measure the femur length and Matsopoulos and Marshall (1994) who measure the fetal head.

In this chapter we present a new algorithm for the detection of connected shapes in noisy images such as the one shown in Figure 3.1. The algorithm is based on a specially designed kernel function that iteratively identifies the outline pixels of the head. Once the outline pixels have been found, the shape is defined to be these pixels together with the pixels inside the outline.

Our kernel function algorithm is a type of greedy algorithm as it chooses the most 'outline like' pixel at each step. It is somewhat reminiscent of the iterative algorithm to track roads in satellite images presented by Geman and Jedynak (1996). The images analysed by these authors are far more complicated than those that we consider, although the level of degradation is far less. Our task is conceptually much simpler than that addressed by Geman and Jedynak (1996) and our algorithm is, accordingly, much less complicated.

This chapter is organised as follows. In Section 3.2, we describe the kernel function, and in Section 3.3, we define the kernel algorithm used in edge detection. A simulation study that aims to illustrate features of the proposed algorithm is presented in Section 3.4. In Section 3.5, a modification to the kernel algorithm is discussed and we show that this modification improves the performance of the kernel algorithm. In Section 3.6 we apply the kernel algorithm to two real ultrasound images. Section 3.7 summarises our findings about the use of the kernel algorithm for detecting shapes defined only by their edge. In Section 3.8, we present a related kernel algorithm to detect the closed boundary of an object in a noisy image, where the pixels inside and outside the edge have grey levels with different distributions. We illustrate this kernel algorithm on a real ultrasound image by detecting the shape of the human ovarian cyst shown in Figure 1.8. Finally, in Section 3.9 we make some suggestions for further work.

3.2 *The kernel function*

Let us suppose that the recorded image z comprises $(2m+1)\times(2m+1)$ pixels: for the image shown in Figure 3.1, $m=30$. We think of this image as part of a plane defined in two dimensions and we define the co-ordinates from $-m$ to m on both axes, so that every integer co-ordinate in the interval $(-m, m)$, say (x, y), corresponds to one pixel. For example, let $m=30$, then the point $(12,\ 25)$ corresponds to the row $43=(30+1+12)$ and column $56=(30+1+25)$ of the image. In this way there is a one-to-one correspondence between points and pixels. We write $z(x,y)$ for the value taken by the image at pixel co-ordinate (x,y). These values are represented by grey-levels in Figure 3.1.

In the following we will define the kernel function centred at point (x_0,y_0). Let θ be the angle through which the kernel is allowed to rotate in order to find its best alignment with the edge of the shape. Then a new (x',y') co-ordinate system centred at the point (x_0,y_0) in the old (x,y) co-ordinate system and rotated through an angle θ is defined as

$$x'=(x-x_0)\cos\theta-(y-y_0)\sin\theta$$

$$y'=(x-x_0)\sin\theta+(y-y_0)\cos\theta.$$

We define the kernel function at pixel co-ordinate (x,y), centred at pixel co-ordinate (x_0,y_0) and rotated through an angle θ anticlockwise, to be

$$K\{(x,y);\ (x_0,y_0),\theta\}=\frac{C}{\sigma(x')}\exp\left\{-\frac{x'^2}{2\sigma_2^2}-\frac{y'^2}{2\sigma^2(x')}\right\}, \tag{3.1}$$

where C is an arbitrary constant, where

$$\sigma(x')=\sigma_0\exp\left(\frac{x'^2}{2\sigma_1^2}\right),$$

and where σ_0, σ_1 and $\sigma_2>0$. Up to a normalising constant, the kernel function is a product of a marginal density in the x' direction and a conditional density in the y' direction. Both densities have mean zero. The variance of the conditional density increases as $|x'|$ increases.

An example of the kernel function $K\{(x,y);\ (x_0,y_0),\theta\}$ is shown in Figure 3.2 by means of a contour plot, where the kernel is centred at pixel co-ordinate $(x_0,y_0)=(0,\ 0)$ with rotation $\theta=0$, and the values of the parameters are chosen as $\sigma_0=1$, $\sigma_1=5$ and $\sigma_2=5$. The kernel function shown in Figure 3.2 will be the one used to detect the fetal head shape from Figure 3.1. Figure 3.2 enables the size of the kernel to be compared with the image given in Figure 3.1 for which $-30\le x\le 30$ and $-30\le y\le 30$.

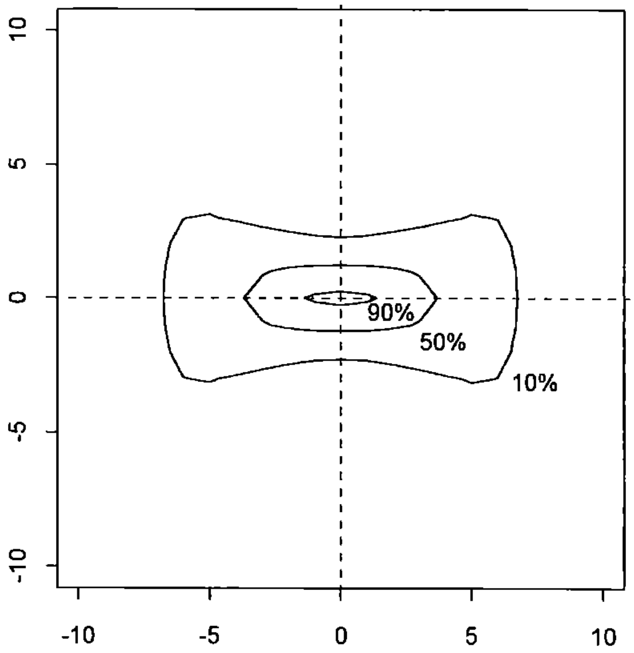


Figure 3.2 An example of a kernel function centred at pixel co-ordinate $(x_0,y_0)=(0,0)$ with rotation $\theta=0$. Contours are drawn at 10%, 50% and 90% of the maximum height. The values of the parameters are $\sigma_0=1$, $\sigma_1=5$ and $\sigma_2=5$. The scales on the axes refer to pixels.

The idea of this is that such a kernel function will be able to capture the curvature of the shape. Other kernel definitions may be appropriate for detecting different types of shapes.

The kernel function was designed intuitively to enable the algorithm to look forward towards new edge pixels, so bridging any apparent gaps. Other kernel definitions may be appropriate for detecting different types of shapes.

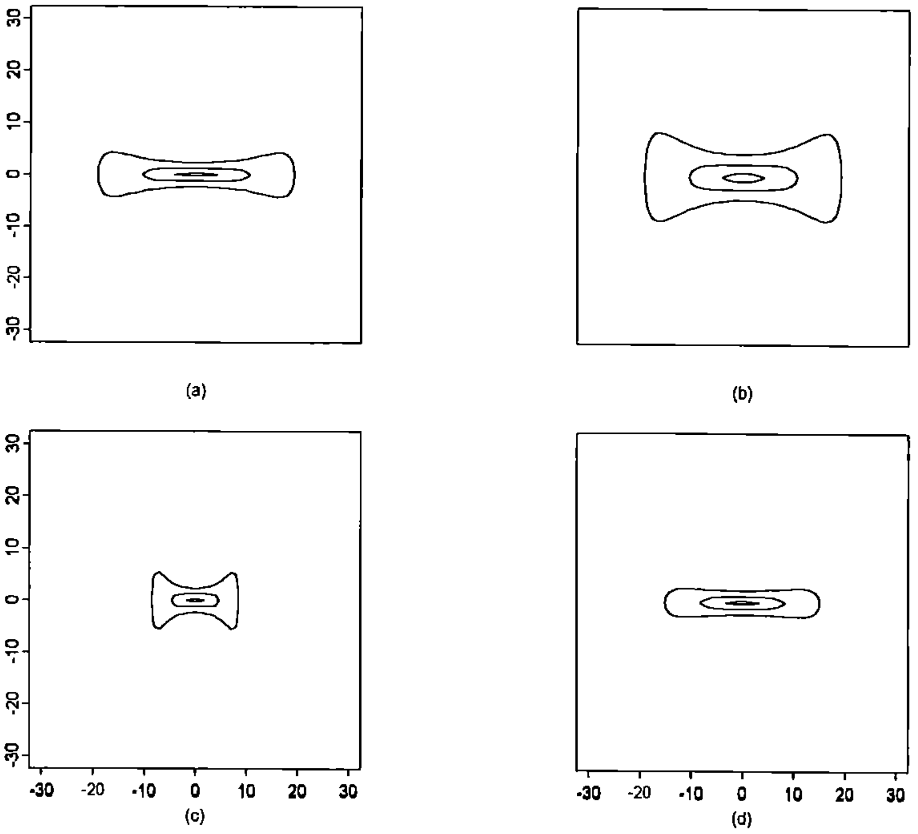


Figure 3.3 Four kernels functions with different parameters:

(a) $\sigma_0 = 1$ (low), $\sigma_1 = 10$ (high) and $\sigma_2 = 20$ (high) causing a narrow, long kernel;

(b) $\sigma_0 = 2$ (high), $\sigma_1 = 10$ (high) and $\sigma_2 = 20$ (high) causing a wide, long kernel;

(c) $\sigma_0 = 1$ (low), $\sigma_1 = 4$ (low) and $\sigma_2 = 20$ (high) causing a narrow, short kernel;

(d) $\sigma_0 = 1$ (low), $\sigma_1 = 10$ (high) and $\sigma_2 = 10$ (low) causing a less curved kernel than (a).

The scales on the axes refer to pixels.

In our algorithm the kernel function shown in Figure 3.2 is allowed to be centred at any pixel co-ordinate and rotated through an angle θ. In the formula for $K\{(x,y);\ (x_0,y_0),\theta\}$, there are three parameters σ_0, σ_1 and $\sigma_2 > 0$ that affect the shape of the kernel function: σ_0 controls mainly the width of K, while σ_1 controls mainly its length and σ_2 mainly its curvature. The effects of varying σ_0, σ_1 and σ_2 are illustrated in Figure 3.3 where we present some examples of kernel functions with different parameter values. The choice of the kernel function is discussed further in Section 3.5 and in Chapter 4.

3.3 *The kernel shape detection algorithm*

Our kernel shape detection algorithm is based on the *convolution* f of the image z with the kernel function $K\{(x,y);\ (x_0,y_0),\theta\}$ defined as

$$f\{(x_0,y_0),\theta\} = \sum_{\substack{\text{pixel}\\ \text{co-ordinates}\\ (x,y)}} K\{(x,y);\ (x_0,y_0),\theta\} z(x,y) \tag{3.2}$$

where $z(x,y)$ is the grey level at the pixel co-ordinate (x,y), and the sum is over all pixel co-ordinates in the region $\{(-m,m)\times(-m,m)\}$.

The maximum of f over rotation θ gives a measure of how 'edge like' the pixel corresponding to the co-ordinates (x_0,y_0) is. The idea of the edge detection algorithm is to use this measure to trace out the outline pixel by pixel (point by point). We do this in a clockwise direction.

The main steps of the edge detection algorithm are as follows:

1. The user selects a starting edge pixel (point) where the outline is obvious and chooses a suitable starting angle for the kernel function centred at this point so that it lines up with any visible edge.
2. Call this point the *current* point, mark it as an outline point and record the rotation angle as θ^*.
3. At each *candidate* point (some of the neighbours of the current point, as discussed below), find the value of f over the set $\{\theta^* - 2\delta, \theta^* - \delta, \theta^*, \theta^* + \delta, \theta^* + 2\delta\}$ of rotation angles, where the fixed rotation parameter $\delta > 0$ is chosen by the user.
4. Identify the candidate point and rotation angle that correspond to the maximum of the values of f found in step 3.
5. Go to step 2, until the starting point becomes a candidate point, so closing up the outline.
6. Stop.

For all the ultrasound head images with which we have experience step 1 has always been possible. For all these images the above algorithm stopped. When the algorithm stops, a closed outline of pixels results. The estimate of the shape is then defined to be all pixels on and inside the outline. Of course, the algorithm is not guaranteed to stop, but it would not be hard to modify it so that termination was ensured. This could be done, for example, by increasing the grey level at the starting pixel to such an extent as to force the kernel function to draw the edge back to that pixel. We do not discuss this further here, but in

Section 3.7 we describe the results of a simulation study designed to illustrate how much noise the kernel algorithm can tolerate before failing to stop.

The values of the kernel parameters σ_0, σ_1 and σ_2, and the rotation parameter δ of the kernel were chosen here by experience. In the next chapter, we will explore the effect of the kernel parameters by means of a simulation study. This study will help us to know well the effect of the parameters and so enable us to choose appropriate values for them.

We also need to specify the rule that chooses the candidate points over which the kernel function is to be optimised. In normal human fetuses, the head shape when viewed from above tends to be roughly ellipsoidal, whereas in fetuses affected by Neural Tube Defects the head is approximately lemon shaped. This is discussed more fully in Chapters 1 and 2. Accordingly, the normal and affected fetal head shapes are of quite different nature, the former having convexities and the latter having both convexities and concavities. However, in both cases a regular pattern for searching the candidate points can be defined when the centre of the image lies within the shape of interest and all parts of the edge are visible from the centre.

We now present the rule according to which the candidate points are chosen. Suppose that we look for the next point in a clockwise direction. If a first or second order neighbour of the current point lies on the clockwise side of a line drawn from the centre of the image (assumed to be inside the head shape) through the current point, we say this point is a candidate point. Hence, the next point will always be one of the neighbours shown in Figure 3.4, where the symbol • represents the current point and the symbols ∘ represent candidate points from which the next point is selected. We need only consider eight different sets of possible candidate points corresponding to the position of the current point.

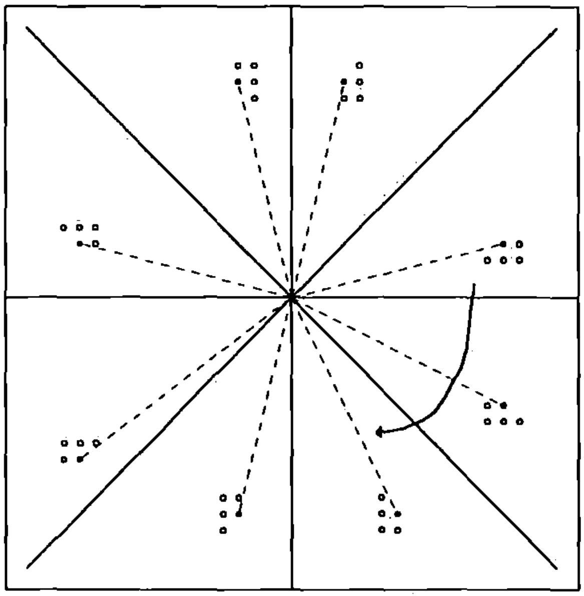


Figure 3.4 The choice of candidate points when searching takes place in a clockwise direction. The symbol • represents the current point, while the symbol ∘ represents the candidate points from which the next point is selected.

In our algorithm the edge is traced out iteratively by selecting the next edge pixel as the one from a set of candidate edge pixels that maximises the convolution f of the specially designed kernel function K with the image z. At each candidate edge pixel the kernel function is allowed to rotate in order to find its best alignment with the edge of the head. This is illustrated in Figure 3.5. Figure 3.5(a) shows the contours of kernel functions at every candidate pixel without rotation, while Figure 3.5(b) shows the contours at one candidate pixel with rotations $\pm 0.02\pi$.

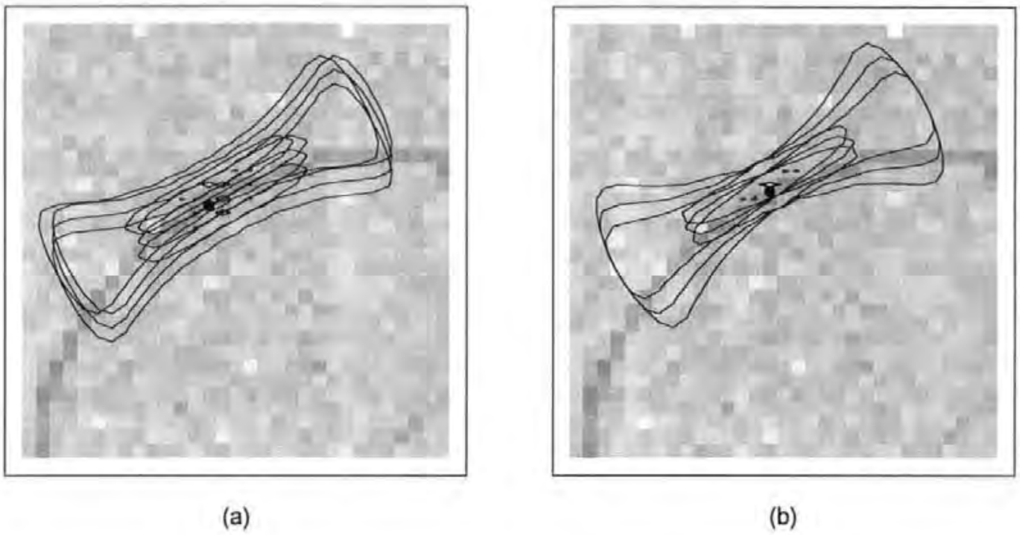

(a) (b)

Figure 3.5 The search procedure: (a) kernel contours at candidate points without rotation; (b) kernel contours at one candidate point with rotation. The symbol • represents the current point.

In the next section we present a simulation study that not only illustrates our algorithm, but also suggests the modifications to the kernel function and the refinements to the algorithm that will be discussed in Section 3.5.

3.4 *A simulation study*

The true binary image, which comprises 61×61 pixels and is an ellipse, is presented in Figure 3.6(a). The outline is intended to represent a cross-section of a fetal skull. The records of the image take the value 1 at the black edge pixels and the value 0 at the remaining white pixels. Figure 3.6(b) shows a noisy image obtained by adding independent $N(0, 1)$ noise to the original true image shown in Figure 3.6(a), and represents the simulated data to which our algorithm is applied.

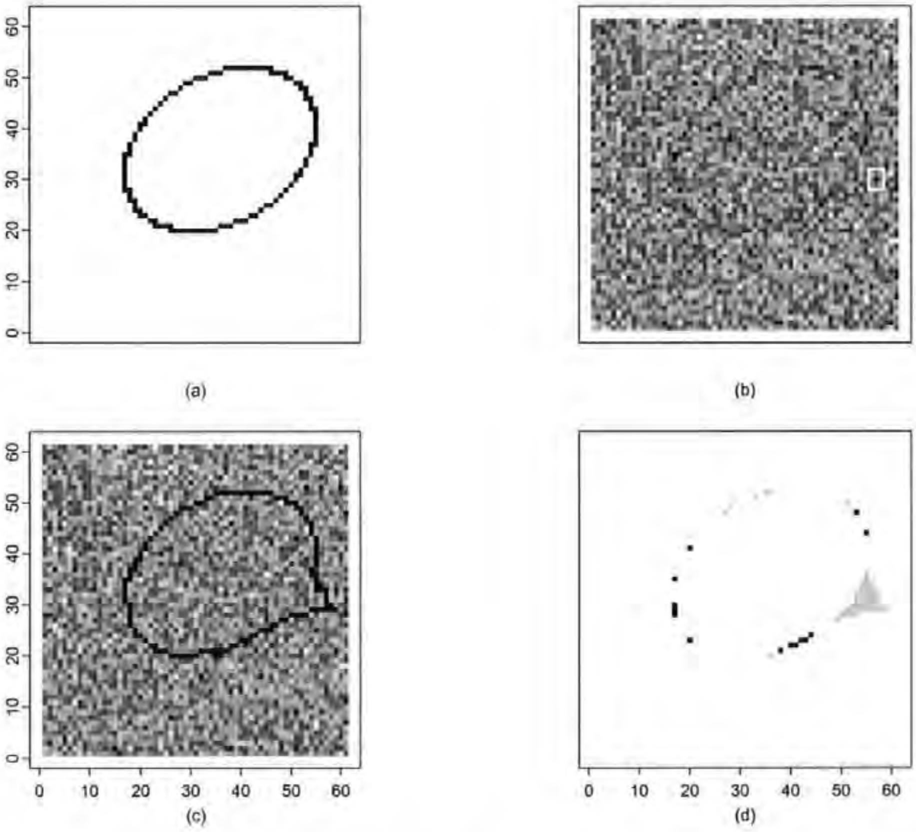


Figure 3.6 (a) The original 61×61 binary pixel image. (b) The data obtained by adding independent $N(0, 1)$ noise to the original image. (c) The estimate of the outline obtained by the proposed algorithm. (d) The difference between the original shape and the estimated shape. Black pixels are inside the original shape but outside the estimated shape, while grey pixels are outside the original shape but inside the estimated shape. The black pixel indicated by the white box in (b) has a large effect on the detected outline.

The estimate of the edge obtained by the proposed algorithm is shown in black in Figure 3.6(c). The kernel that was used had $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$ and is shown in Figure 3.3(c). The rotation parameter δ was set to 0.02π. Figure 3.6(d) illustrates the difference between the true shape and the estimated shape; black pixels are inside the original shape but outside the estimated shape, while grey pixels are outside the original shape but inside the estimated shape.

The proposed algorithm has performed well, except that in the estimated shape there is a protuberance on the lower right. This is caused by the large effect of the black pixel indicated by the white box in Figure 3.6(b). In Section 3.5 we present a modification to the kernel function that reduces the sensitivity of the algorithm to such outlying records. Table 3.1 gives a numerical summary of the errors made by the estimate. Note that edge pixels are thought of as part of the shape.

Table 3.1 Numerical summary of the error made by the estimate

	Estimate shape		
True shape	pixel present	pixel absent	Total
pixel present	965	14	979
pixel absent	47	2695	2742
Total	1012	2709	3721

3.5 *Modification of the kernel function and refinement of the algorithm*

The shape of the kernel function can of course be chosen to suit the application. In our experience with simulation studies such as those presented in Section 3.4, the 'peanut' or 'bowtie' shaped kernel function shown in Figure 3.2 generally performed well, but could be affected by isolated pixels with outlying records. Such behaviour was illustrated in Figure 3.6(c), where a substantial deviation of the outline was brought about by the black pixel indicated by the white box in Figure 3.6(b). In this case the kernel function causes the algorithm to look too far ahead and to be deviated by this outlying record.

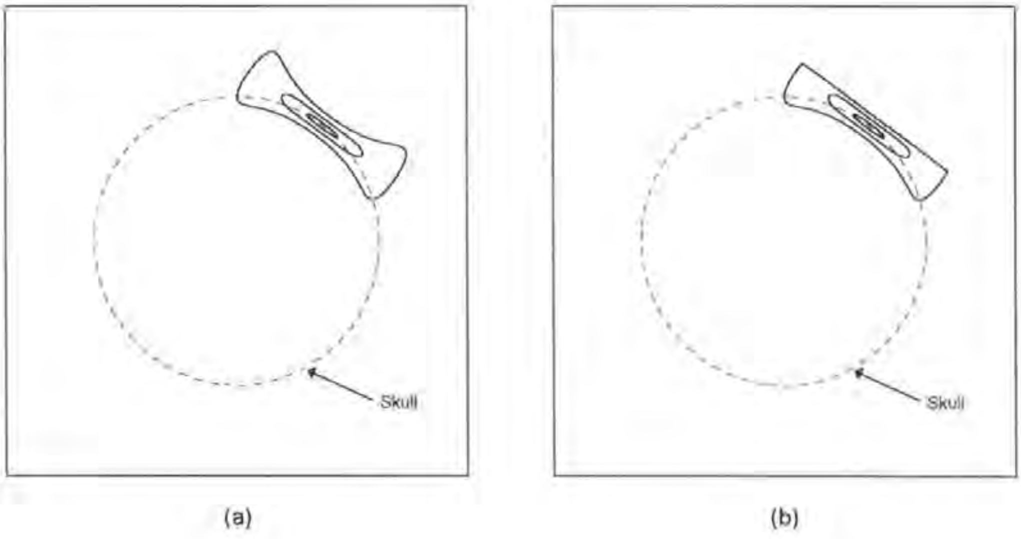
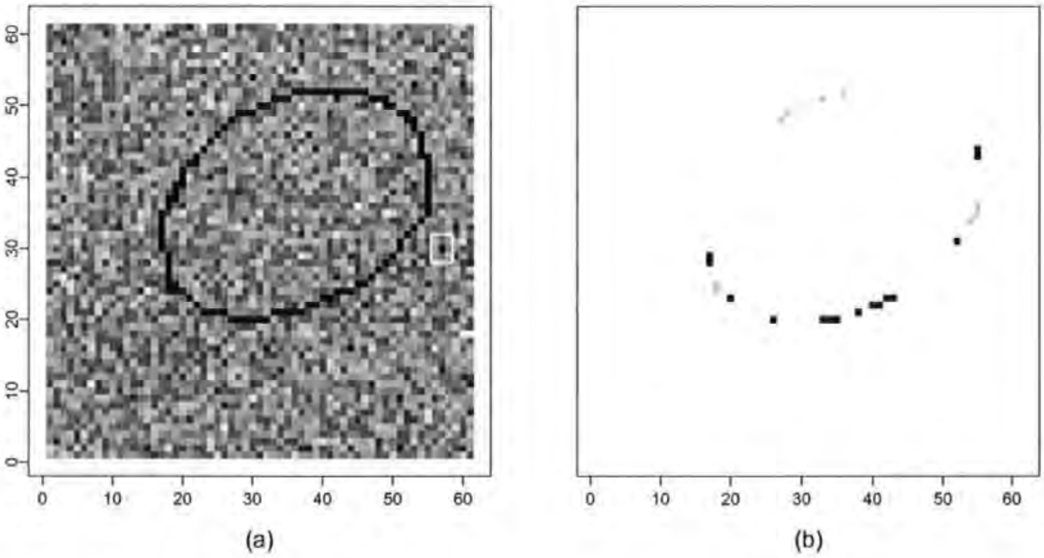


Figure 3.7 The choice of the kernel function in relation to the underlying shape. (a) The original 'peanut' shaped kernel function. (b) The modified 'banana' shaped kernel function. The banana kernel follows the curvature of the skull better than the peanut kernel.

A 'banana' shaped kernel function sometimes turns out to be more suitable for human fetal head shape detection as it follows more closely the curvature of the skull cross-section. This is illustrated in Figure 3.7.

The banana shaped kernel function can be obtained by cutting off the parts of the peanut shaped kernel that bulge away from the centre of the image.



(a)

(b)

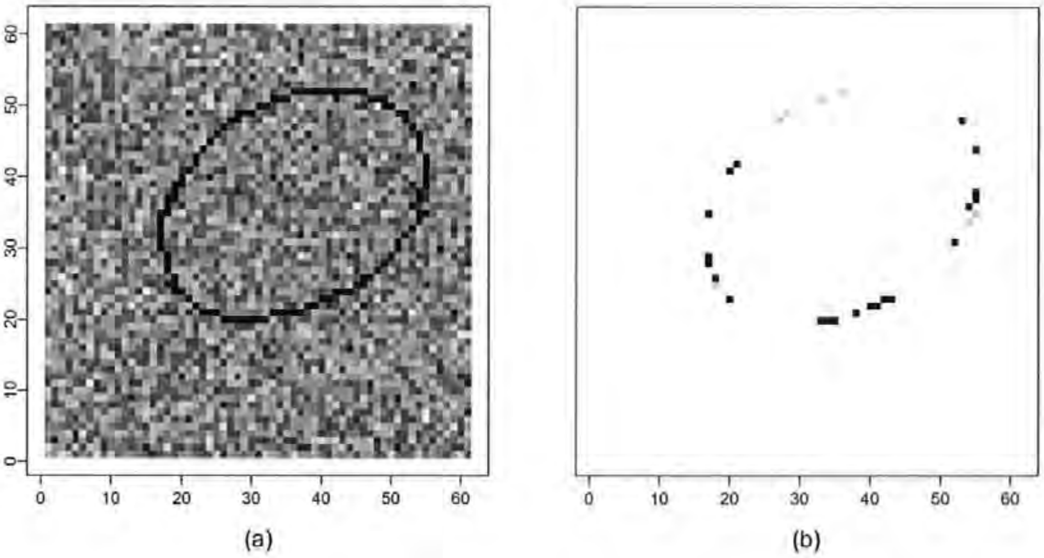
Figure 3.8 (a) The estimate of the outline obtained by applying the proposed algorithm with the banana shaped kernel function to the data in Figure 3.6(b). The white box indicates the black pixel that caused problems when the peanut shaped kernel function was employed. (b) The difference between the original shape and the estimated shape. The colour code of Figure 3.6(d) is employed.

We applied the banana shaped kernel function with the same parameter values as before to the image data in Figure 3.6(b). The estimated outline is shown in Figure 3.8(a). This estimate seems not to have been affected by the black pixel indicated by the white box. Figure 3.8(b) illustrates the difference between the true shape and the estimated shape. A comparison of Figure 3.8(b) with Figure 3.6(d) shows that the banana shaped kernel function has led to a substantial improvement in the performance of the algorithm. The numerical comparison between the true and estimated shapes is given in Table 3.2.

Table 3.2 Numerical summary of the error made by the estimate when the banana shaped kernel function is applied

	Estimate shape		
True shape	pixel present	pixel absent	Total
pixel present	964	15	979
pixel absent	9	2733	2742
Total	973	2748	3721

In the above experiments, the angles over which the function f is maximised at each candidate point are restricted to the discrete set of rotation angles $\left(\theta^* - 2\delta, \theta^* - \delta, \theta^*, \theta^* + \delta, \theta^* + 2\delta\right)$, where θ^* is the current angle of the kernel function, and the parameters σ_0, σ_1 and σ_2 are all fixed. We now consider allowing the rotation angle of the kernel to vary in a continuous interval $\left(\theta^* - \varepsilon,\ \theta^* + \varepsilon\right)$ for some fixed $\varepsilon > 0$; we now refer to θ as a dynamic parameter. We keep $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$, we let $\varepsilon = 0.1\pi$, and we use the banana shaped kernel function. We employ the `optimize` function of S-Plus to perform the search over θ. Figure 3.9 is the estimated shape, and the numerical comparison between the true shape and estimated shape is given in Table 3.3. Comparing Figure 3.9(b) with Figure 3.8(b) and Table 3.3 with Table 3.2, we see that optimising over θ does not lead to a better result in this simulation study.



(a)

(b)

Figure 3.9 (a) The estimate of the outline obtained by applying the proposed algorithm with the banana shaped kernel function and dynamic parameter $\theta \in \left(\theta^* - 0.1\pi,\ \theta^* + 0.1\pi\right)$ to the data in Figure 3.6(b). (b) The difference between the original shape and the estimated shape. The colour code of Figure 3.6(d) is employed.

<u>**Table 3.3**</u> Numerical summary of the error made by the estimate when the banana shaped kernel function and dynamic parameter θ are applied

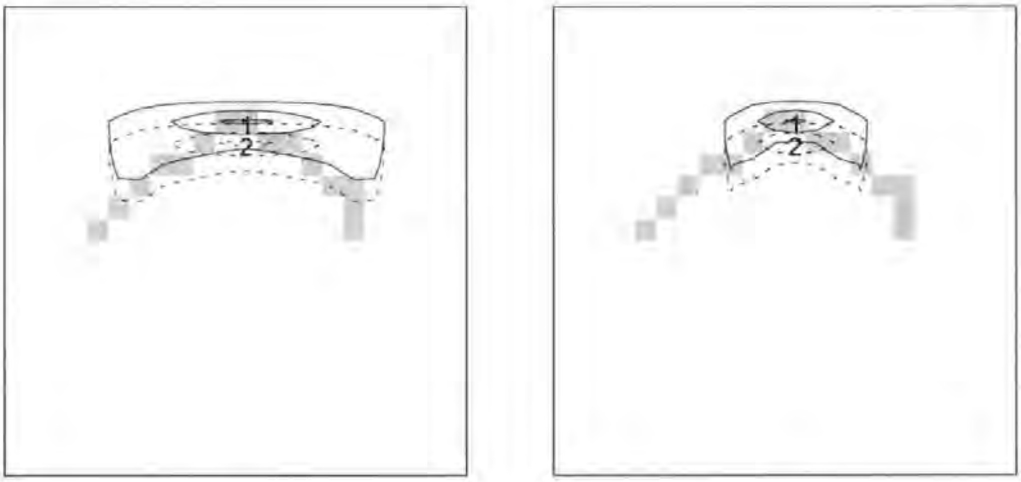
	Estimate shape		
True shape	pixel present	pixel absent	Total
pixel present	958	21	979
pixel absent	7	2735	2742
Total	965	2756	3721

Another possible refinement is to allow all the parameters θ, σ_0, σ_1 and σ_2 to vary continuously in suitable continuous intervals, and to optimise over these parameters at each pixel. Because the shape and hence volume of the kernel function will change as these parameters change, we need to standardise the convolution in order that legitimate comparisons may be made; we set

$$f_{\text{std}}(x_0,y_0)=\frac{f(x_0,y_0)}{\sum\limits_{\substack{\text{pixel}\\ \text{co-ordinates}\\ (x,y)}} K\{(x,y);\ (x_0,y_0),\theta\}},$$

where $f(x_0,y_0)$ is the value of f at (x_0,y_0), and the sum is over all pixels.

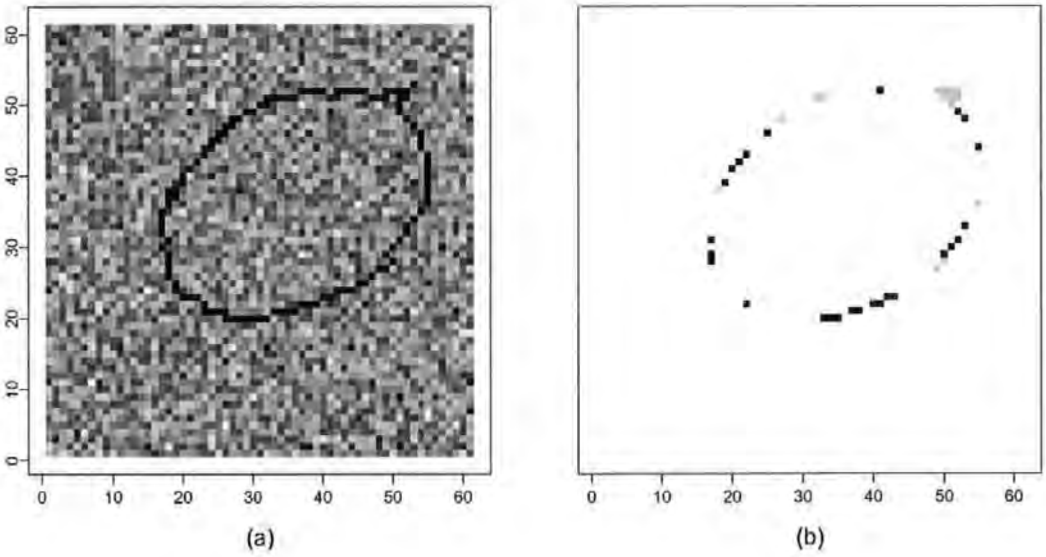
For our purposes, we assume that θ, σ_0, σ_1 and σ_2 are all dynamic parameters, with $\theta \in (\theta^* - 0.1\pi,\ \theta^* + 0.1\pi)$, $\sigma_0 \in (0.1,\ 1)$, $\sigma_1 \in (1,\ 10)$ and $\sigma_2 \in (1,\ 40)$. We use the S-Plus function `nlminb` to optimise the kernel function over θ, σ_0, σ_1 and σ_2. The result obtained by applying the proposed algorithm to the image shown in Figure 3.6(b) is very poor. We terminated the programme when the detected edge began to go far away from the true shape. The reason for this behaviour is that the length and the width of the kernel function may not be chosen suitably by the optimisation routine. For example, if the kernel is so short or narrow that it only covers a few pixels, the standardised convolution may be very high at a point whose immediate neighbours have quite high grey levels, even though it does not lie on the edge. In other words, the optimisation routine may prefer a small kernel function with large support that detects an uninteresting local effect to a big kernel function that detects the edge. There is a similar problem if the kernel function is too long or too wide. To illustrate this, Figure 3.10 shows part of an edge defined image with kernel contours added to pixel 1 and pixel 2, where pixel 1 is on the edge but pixel 2 is not. In Figure 3.10(a) kernel functions with parameters $\sigma_0 = 0.5$, $\sigma_1 = 3$ and $\sigma_2 = 20$ have been added and the standard convolution value f_{std} is 0.1023 at point 1 and 0.0232 at point 2, whereas in Figure 3.10(b) kernel functions with parameters $\sigma_0 = 0.4$, $\sigma_1 = 1.5$ and $\sigma_2 = 20$ are shown and the standard value f_{std} is 0.0909 at point 1 and 0.1134 at point 2. The incorrect pixel was identified in this case because we obtain the biggest value of f_{std} at point 2 in Figure 3.10(b).



(a)

(b)

Figure 3.10 Example of the effect of the parameters. (a) Contours with parameters $\sigma_0 = 0.5$, $\sigma_1 = 3$ and $\sigma_2 = 20$. f_{std} is 0.1023 at point 1 and 0.0232 at point 2. (b) Contours with parameters $\sigma_0 = 0.4$, $\sigma_1 = 1.5$ and $\sigma_2 = 20$. f_{std} is 0.0909 at point 1 and 0.1134 at point 2. The maximum value of f_{std} is therefore obtained at the incorrect point 2 in (b).



(a)

(b)

Figure 3.11 (a) The estimate of the outline obtained by applying the proposed algorithm with the banana shaped kernel function and dynamic parameters $\theta \in (\theta^* - 0.1\pi,\ \theta^* + 0.1\pi)$, $\sigma_1 \in (3,\ 10)$ and $\sigma_2 \in (5,\ 40)$, and fixed parameter $\sigma_0 = 1$ to the data in Figure 3.6(b). (b) The difference between the original shape and the estimated shape. The colour code of Figure 3.6(d) is employed.

If we fix the width of the kernel by letting $\sigma_0 = 1$, and let $\theta \in (\theta^* - 0.1\pi,\ \theta^* + 0.1\pi)$, $\sigma_1 \in (3,\ 10)$ and $\sigma_2 \in (5,\ 40)$ be dynamic, we obtain the result shown in Figure 3.11 after 19832 seconds DOS-time on a Pentium 75MHz PC, while the non-dynamic algorithm that leads to Figure 3.8 took 706 seconds only. The numerical comparison between the true and estimated shapes is given in Table 3.4. Although this time the program works and gives a very acceptable solution, Table 3.4 shows that the performance is still poor in comparison with Table 3.1, 3.2 and 3.3. Hence, in practice we need to ensure that the parameters and parameter intervals used do not lead to kernel functions that have too large or too small a support. Our experience suggests that we should choose the parameters σ_0, σ_1 and σ_2 so that at the centre the width of the kernel contour at 90% of the maximum height should cover about 5-7 pixels and the length should cover about 15-25 pixels.

Table 3.4 Numerical summary of the error made by the estimate when the banana shaped kernel function and dynamic parameters θ, σ_1 and σ_2 are applied

	Estimate shape		
True shape	pixel present	pixel absent	Total
pixel present	953	26	979
pixel absent	15	2727	2742
Total	968	2753	3721

Neither of these dynamic variations produced any noticeable improvement in results, despite a massive increase in computational burden. Maximising f over σ_0, σ_1 and σ_2 can lead to meaningless results if the range of these parameters is not restricted.

In Section 3.7 we will discuss further the effect of varying the parameters.

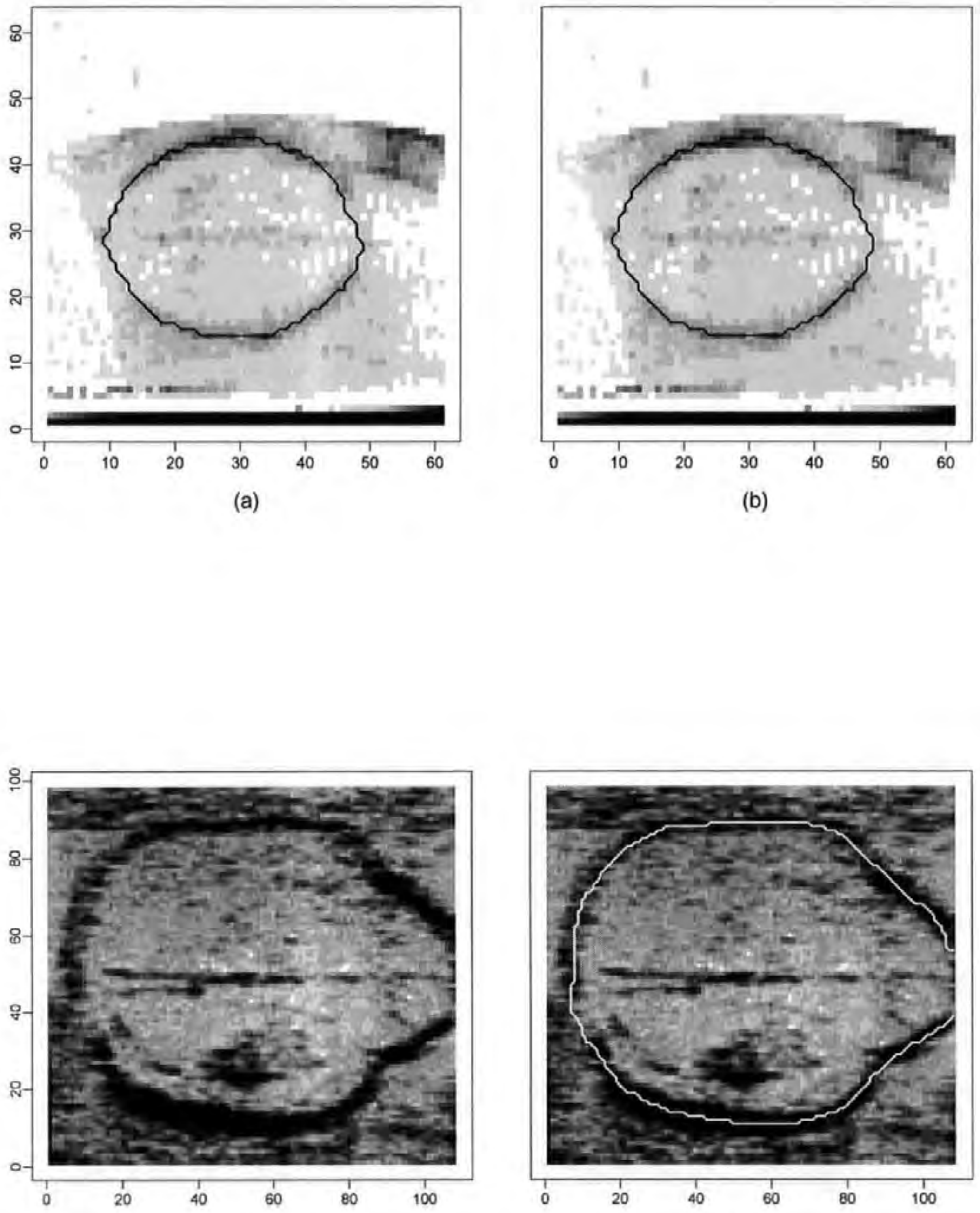
3.6 *Application to real data*

First we applied the proposed algorithm to the original ultrasound image of a cross-section of a fetal head shown in Figure 3.1. We used the peanut shaped kernel function shown in Figure 3.2, which has the parameter values $\sigma_0 = 1$, $\sigma_1 = 5$, $\sigma_2 = 5$, and we set

$\delta = 0.02\pi$. The resulting outline of the cross-section of the fetal head using the peanut kernel is shown in Figure 3.12(a). The procedure took 753 seconds DOS-time on a Pentium 75MHz PC. The algorithm seems to work well.

We repeated the above estimation procedure using the banana shaped kernel function and the same parameter values. The result is shown in Figure 3.12(b). We found that the estimated edge on the right of the shape differs a little in Figure 3.12(b) and in Figure 3.12(a). Of course, this difference may or may not be relevant from a medical point of view.

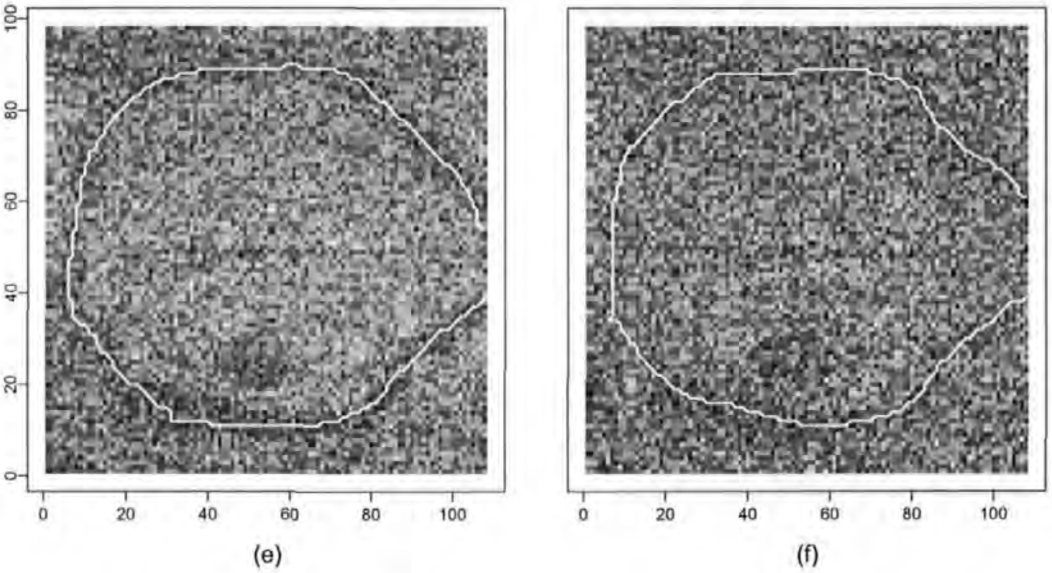
In this example, the peanut shaped kernel function works well. We can also use the peanut shaped kernel function for the detection of an affected head shape that shows the lemon sign. Figure 3.12(c) is the digitised version of the ultrasound image shown in Figure 1.1(b). This image comprises 108×98 pixels and there are 256 grey levels. The lemon sign is clear visible. Indeed it was known that the fetus was affected with open spina bifida. We applied the proposed algorithm to this image with the peanut shaped kernel function with parameters $\sigma_0 = 1.3$, $\sigma_1 = 6$ and $\sigma_2 = 10$, and with $\delta = 0.02\pi$. The starting point is taken to be the lower edge point at the right side of the image, where the image is not complete. The result is shown in Figure 3.12(d) where the white line represents the extracted outline of the head shape. The outline is cut off at its right side because the original image is not complete there. The lemon sign is obviously present in Figure 3.12(d).



(a) (b)

(c) (d)

Figure 3.12 Two real examples. (a) Ultrasound image of a cross-section of a fetal head with extracted edge using the peanut shaped kernel function. (b) Ultrasound image of a cross-section of a fetal head with extracted edge using the banana shaped kernel function. (c) Digitised version of a cross-section of a fetal head with open spina bifida showing the lemon sign. The original ultrasound image is shown in Figure 1.1(b). (d) The image (c) with extracted edge (white line) using the peanut shaped kernel function.



(e) (f)

Figure 3.12 (Continued) Noisy images together with the extracted edges (white lines) obtained by the kernel algorithm. The data were obtained by adding independent (e) $N\left(0, 100^2\right)$ and (f) $N\left(0, 150^2\right)$ noise to the already noisy ultrasound image shown in Figure 3.12(c).

A large difference between Figure 3.12(a) and Figure 3.12(c) is that the edge is quite well defined in Figure 3.12(c) whereas it is not in Figure 3.12(a). It would be an interesting experiment to add noise to the already noisy Figure 3.12(c) so as to make the edge much less clear, and then to run the algorithm. We obtained two images by adding independent $N\left(0, 100^2\right)$ and $N\left(0, 150^2\right)$ to the real ultrasound image respectively. Panels (e) and (f) of Figure 3.12 present the result. We see that the resulting images do not resemble an ultrasound image, but they do look like Figure 3.12(a) in the sense that there are parts of the edge that seem to the human eye almost completely obliterated. The proposed algorithm was applied to these images with the same kernel, same rotation angle and the same starting point as were used for Figure 3.12(d). The extracted outline of the head shape is expressed by the white lines. The lemon sign is obviously present in both results.

The kernel algorithm seems to work well on a large number of ultrasound images of cross-section of fetal heads. For these images the performance was reasonably robust to the choice of parameters.

3.7 *Discussion about the use of the kernel algorithm for detecting shapes defined only by their edges*

In the previous sections, we have described a kernel function based algorithm for detecting the edge of a closed curve from a noisy image such as an ultrasound image of a cross-section of a fetal head. In a simulation study we have seen that this algorithm gave a very good fit to a true ellipse corrupted by the addition of independent standard Gaussian noise; see Figures 3.6 and 3.8.

Our approach was implemented in several ways using peanut or banana shaped kernels with fixed or dynamic parameters. The results are shown in Figure 3.6(c) (peanut kernel with fixed parameters), Figure 3.8(a) (banana kernel with fixed parameters), Figure 3.9(a) (banana kernel with one dynamic parameter), and Figure 3.11(a) (banana kernel with three dynamic parameters). Comparing these results we find that the banana shaped kernel is more suitable for the ellipse. Because the kernel function can change its shape greatly in order to cover pixels with high records, the dynamic parameters and associated parameter intervals should be chosen with care. Because Figures 3.12(a) and (b) look very similar, we cannot say which kind of kernel is more suitable for fetal head shape detection from these two figures.

In order to assess the performance of the kernel function algorithm, we conducted a simulation study based on the original image shown in Figure 3.6(a). Our data were obtained by adding independent $N(0,\kappa)$ noise to that image. We judged the algorithm to have failed to stop if the resulting edge was not closed, or if the number of pixels in the resulting edge exceeded 1.5 times the number of pixels in the true edge so that the resulting edge was wandering around the image. For each of three different noise variances $\kappa = 0.25$, 0.5 and 1.0, we applied the peanut shaped kernel function algorithm to 100 noisy images. The starting pixel was selected at random from among all the known edge pixels. The mean and the standard deviation (sd) of the number of pixels that differ between the true shape and the estimated shape over realisations of the noise process for which the algorithm stopped are recorded in Table 3.5. Also presented in Table 3.5 is the number of times the algorithm failed to stop due to the resulting edge not being closed or due to the maximum number of edge pixels being exceeded.

Table 3.5 Results of the simulation study for the kernel function algorithm

Variance κ	Algorithm stops		Algorithm fails to stop	
	Measure of error		Occasions that resulting edge is not closed	Occasions that maximum number of pixels is exceeded
	mean (pixels)	sd (pixels)		
0.25	37	21.0	4	5
0.50	81	49.6	7	23
1.00	120	72.0	9	52

sd=standard deviation

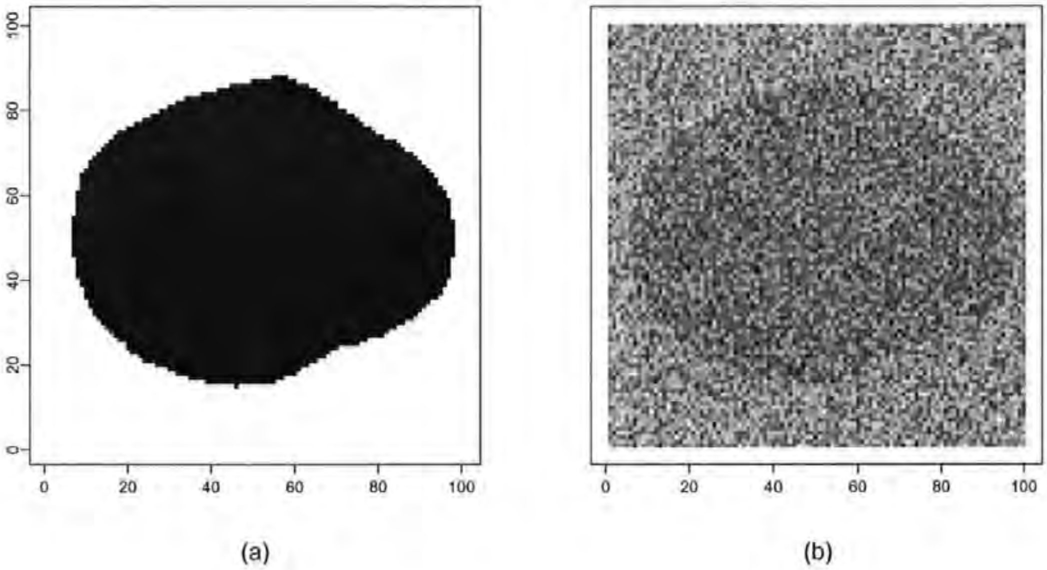
Table 3.5 shows that the performance of the algorithm is consistently good for $\kappa = 0.25$. As κ increases the quality of the performance decreases. Although the results seem disappointing for $\kappa = 1$, it should be remembered that the edge is just one pixel thick with the consequence that the edge detection problem is a very hard one.

3.8 *Shape detection using kernel algorithm*

We discussed the work of many authors on object recognition in Chapter 2; see in particular Section 2.1. The models used by these authors can be placed into a number of categories. The first category corresponds to traditional Bayesian type models as proposed in the imaging context by Geman and Geman (1984) and further discussed by Besag (1989). The second category corresponds to template models as used by Grenander, Chow and Keenan (1990). The template model is still Bayesian, but is based on the deformation of a special designed polygon. The third category corresponds to non-Bayesian models. The 'snakes' or active contour model, proposed by Kass, Witkin and Terzopoulos (1988), is a non-Bayesian model that uses the minimisation of a specially designed energy function to cause a deformable contour to change its shape with the aim of wrapping itself around the edge of the object. Kass, Witkin and Terzopoulos (1988) describe in detail an algorithm to perform this minimisation. We will discuss the snake model in Chapter 5. In Chapter 2 we mainly introduced Storvik (1994)'s model which we fitted using the cascade algorithm. Storvik (1994)'s approach is based on the minimisation of an energy function, and can sometimes be given a Bayesian interpretation.

In the previous sections of this chapter, a kernel method was introduced to detect in an image a single shape defined in terms of its edge with pixels inside and outside the edge having grey levels with the same or similar distributions; an example of such an image is given in Figure 3.1. In this section, the kernel algorithm is modified to detect in an image the closed boundary of an object, where the pixels inside and outside the edge have grey levels with different distributions. An example of such image data is given in

Figure 3.13(b). This is a simulated image obtained by adding independent $N(0, 1)$ noise to the true image shown in Figure 3.13(a). The images in Figure 3.13 were also shown in Figure 2.4. We will show that the kernel method can be adapted to identify the boundary of an object in a noisy image. After this, we present the result of applying the adapted algorithm to the real ultrasound image shown in Figure 1.8.



(a) (b)

Figure 3.13 (a) The known head shape. This 100×100 binary image displays one of the heads analysed by Wright *et al.* (1997). (b) The simulated data obtained by adding independent $N(0, 1)$ noise to the image (a).

3.8.1 *The idea*

Let us begin by considering a simple break detection problem in one dimension. Figure 3.14 presents a way of identifying the break point in a step function that uses a detection function D.

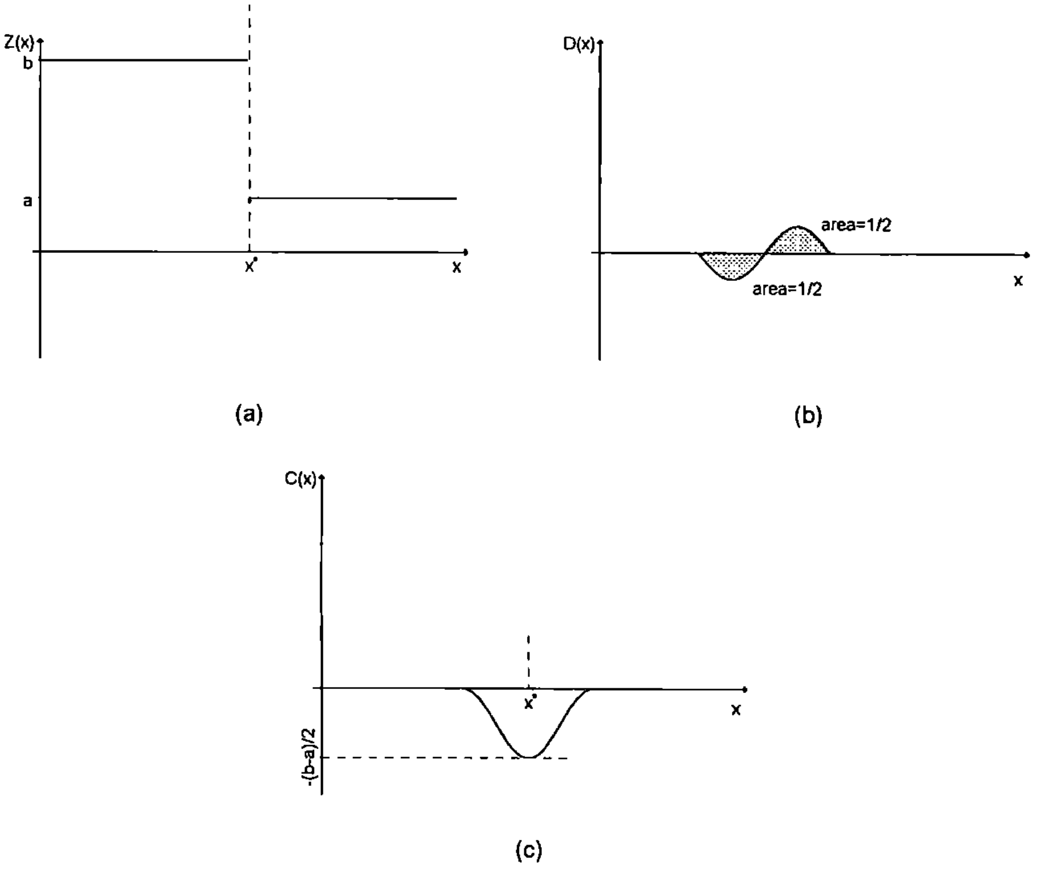


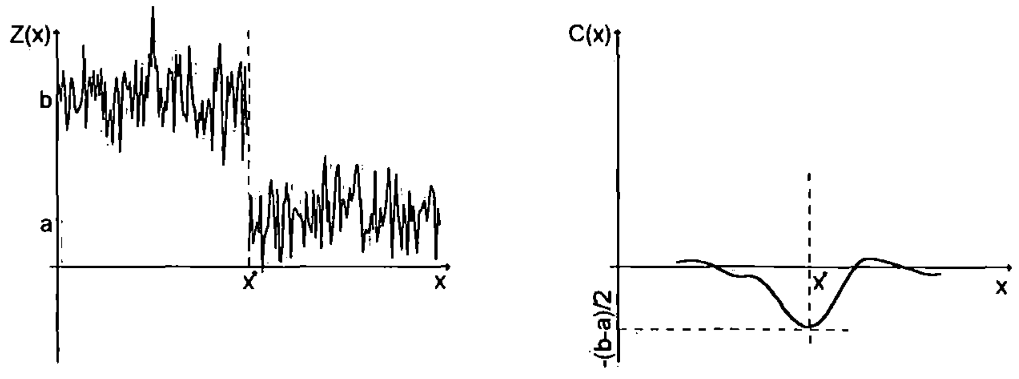
Figure 3.14 Illustration of the break point detection method in one dimension. (a) The step function $Z(x)$ has a break at an unknown point x^*. (b) The detection function $D(x)$ has two parts – a negative part and a positive part, each with the same area $1/2$. (c) The convolution $C(x)$ of the detection function $D(x)$ with the step function $Z(x)$. This indicates a break in $Z(x)$ at x^*.

Let $Z(x)$ be a step function with a break at an unknown point x^*:

$$Z(x) = \begin{cases} b & \text{if } x < x^*, \\ a & \text{if } x \geq x^*, \end{cases}$$

where $a < b$. The function $Z(x)$ is presented in Figure 3.14(a). A standard break detection algorithm would take a function $D(x)$ such as that shown in Figure 3.14(b), where the two shaded parts above and under the x – axis have the same area $1/2$ and would convolve $D(x)$ with the step function $Z(x)$ to obtain the convolution $C(x)$ presented in Figure 3.14(c). At points far away from the break point x^* the value of the convolution is zero. At points near to the break point, the value of the convolution becomes negative and at the break point x^* the minimum value $-(b-a)/2$ of the convolution is attained. This process is a sort of gradient operator. It identifies the break point as being the place where the maximum absolute value of the convolution occurs.

The same method could be used to estimate the change point for a much less smooth function, such as the function shown in Figure 3.15(a). This function was obtained by adding independent $N(0, 0.3^2)$ noise to the function $Z(x)$ shown in Figure 3.14(a); we set $a = 0.5$ and $b = 1.8$, and divide the x – axis into 200 points. Convolving the detection function $D(x)$ shown in Figure 3.14(b) with the noisy function $Z(x)$, we obtain the convolution $C(x)$ presented in Figure 3.15(b). Around the change point x^* the maximum absolute value of the convolution occurs. The estimated change point may differ from the exact break point because of noise.



(a)

(b)

Figure 3.15 Illustration of the change point detection method for a noisy function in one dimension. (a) The step function $Z(x)$ shown in Figure 3.14(a) has been corrupted by independent $N(0, 0.3^2)$ noise; we set $a = 0.5$ and $b = 1.8$, and we divided the x – axis into 200 points. (b) The convolution $C(x)$ of the detection function $D(x)$ with the noisy function $Z(x)$. This indicates a break in $Z(x)$ around x^*.

The form of the detection function used in this one dimensional case suggests to us how we should adapt the kernel function introduced in Section 3.2 to detect object boundaries in images such as Figure 3.13(b).

3.8.2 *Description of the kernel algorithm applied to shape detection*

Now we apply the above idea to the shape recognition problem in two dimensions. We shall assume that in the true image the whole shape is present; such a true image is shown in Figure 3.13(a). Such images exhibit a large change in grey levels at their edge. What we need to do is to find a detection function to estimate the changes in the true image from a noisy version of it.

Let us consider the kernel $K\{(x,y);(x_0,y_0),\theta\}$. Let us assume that the centre (x_0,y_0) of this kernel lies on the edge of an object, as shown in Figure 3.16. The line BB′ is the major axis of the kernel. We suppose that pixels inside the true object have higher grey level values than those outside the true object. If the angle of BB′ with respect to the horizontal is selected appropriately, and if the part of the kernel outside the object is multiplied by -1, the absolute value of the sum of the product of the resulting function with the image grey levels will reach its maximum around the edge point (x_0,y_0) along the line AA′ perpendicular to BB′ through the point (x_0,y_0). Accordingly, the kernel detection function $D\{(x,y);(x_0,y_0),\theta\}$ is defined as

$$D\{(x,y);(x_0,y_0),\theta\} = K\{(x,y);(x_0,y_0),\theta\} \times G\{(x,y);(x_0,y_0),\theta\}, \qquad (3.3)$$

where θ is the angle of rotation of the kernel, that is, the angle of rotation of BB′, and $G\{(x,y);(x_0,y_0),\theta\}$ is a marking function which takes the value -1 outside BB′ and $+1$ inside BB′.

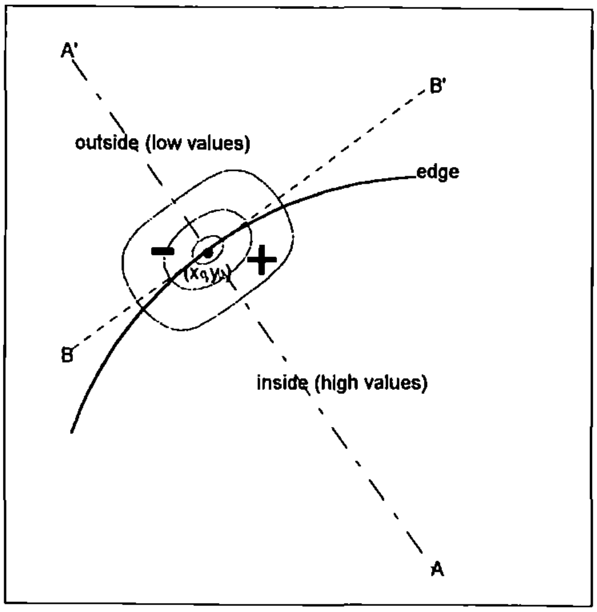


Figure 3.16 The detection function D. The kernel function K is illustrated by the contours, while the marking function G is shown by using + and −.

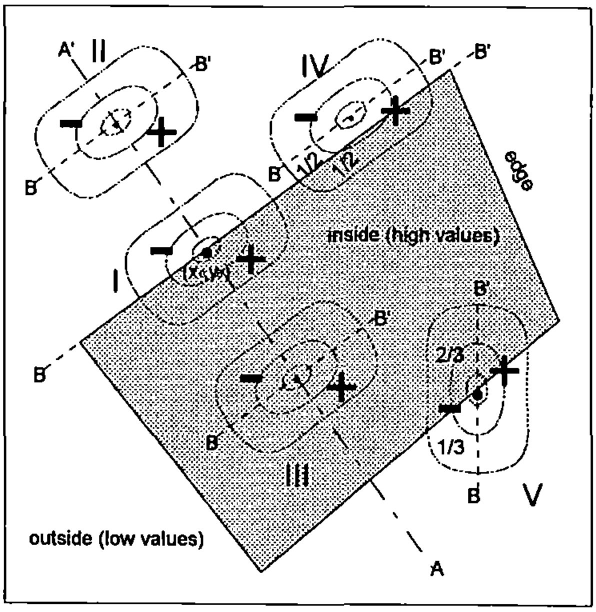


Figure 3.17 Five detectors. The highest absolute value is yielded by detector I.

Figure 3.17 shows how the detection function works. In Figure 3.17, detector I is located just on edge and the sum of its product with the image grey levels gives the value $high - low$; detector II is located completely outside the shape, and leads to the value $low - low = 0$; detector III is located completely inside the shape and leads to the value $high - high = 0$; detector IV overlaps the object much less than detector I and leads to the value $\left(\frac{1}{2} high + \frac{1}{2} low\right) - low = \frac{1}{2}(high - low)$; detector V has the wrong rotation angle and leads to the value $\left(\frac{1}{3} high + \frac{2}{3} low\right) - \left(\frac{2}{3} high + \frac{1}{3} low\right) = -\frac{1}{3}(high - low)$. Of all five detectors, detector I leads to the highest absolute value of the sum of its product with the image grey levels, as we would hope.

The shape detection now proceeds exactly as described in Section 3.2 and 3.3 except that the kernel function K is replaced by the detection function D.

We apply the modified kernel method to the object recognition problem shown in Figure 3.13. In Chapter 2 we applied Storvik (1994)'s algorithm to this problem. The noise level in image Figure 3.13(b) is $\kappa = 1$. The true edge (dark line) and its estimate by the modified kernel method are presented in Figure 3.18. The parameters of the kernel were chosen as $\sigma_0 = 4$, $\sigma_1 = 10$ and $\sigma_2 = 7$. The error for this estimation procedure (number of differing pixels between the true shape and the estimated shape) is 147, which is better than the error of 160 (152 after median smoothing) for the best estimate we obtained by searching over the two parameters of the energy function of Storvik's algorithm; this estimate was shown in Figure 2.6. The modified kernel method yielded an estimate with a smoother outline than the estimate produced by Storvik's algorithm at less computational expense.

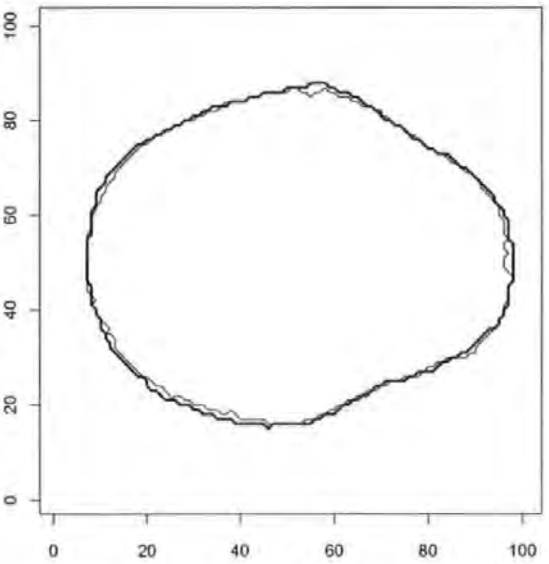
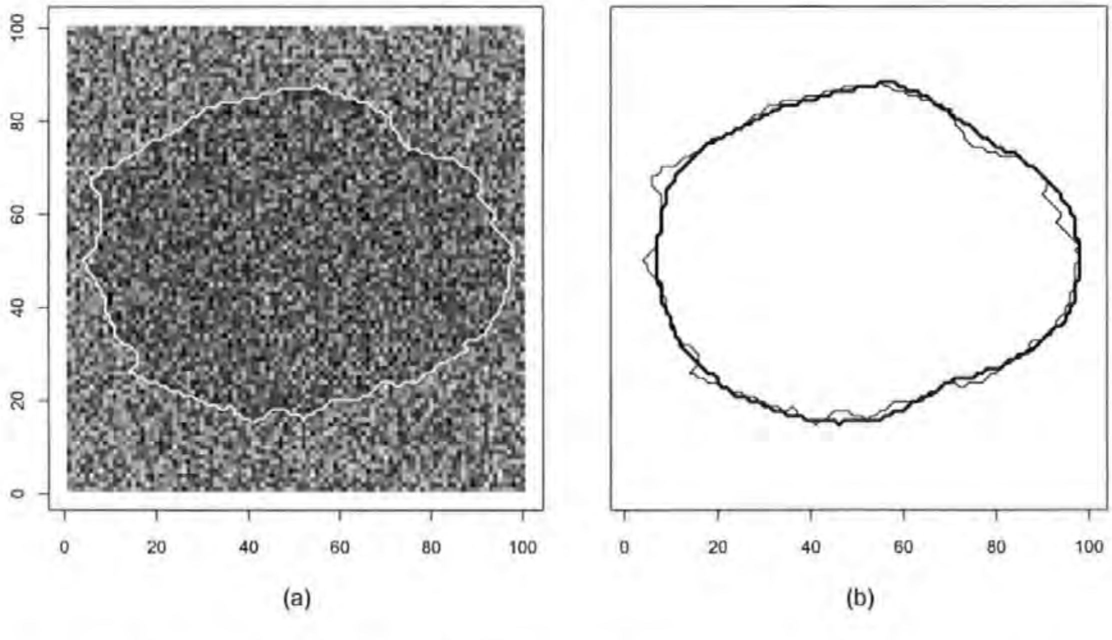


Figure 3.18 The edge produced by the modified kernel method together with the true edge (darker line).



(a)

(b)

Figure 3.19 (a) The noisy image together with the edge (white line) estimated by the modified kernel algorithm. The noisy image is obtained by adding independent $N(0, 3)$ noise to the true image shown in Figure 3.13(a). (b) The edge produced by the modified kernel procedure together with the true edge (darker line).

The detection function defined by (3.3) works well for images with relatively low levels of noise essentially because the large change in grey level at the edge of the image in the true image still remains in the noisy version. However, in a very noisy image, this change becomes much harder to identify and at some parts the edge may even be destroyed. For example, the image shown in Figure 3.19(a) is obtained from Figure 3.13(a) by adding independent $N(0, 3)$ noise. Some parts of the edge seem to have been completely destroyed by the high level of noise. The shape detected by the modified kernel method is presented in Figure 3.19(a) by the white line; we set $\sigma_0 = 3$, $\sigma_1 = 10$ and $\sigma_2 = 6$ to obtain the best estimate with an error of 236 pixels. From this figure we see that the estimated edge is attracted to regions where there is a large change in grey level, with the result that the deformable area cannot be restored properly. Figure 3.19(b) presents the true shape (darker line) together with the contour produced by the modified kernel algorithm; the result is rough.

3.8.3 *An attenuated detection function for very noisy images*

It may help if we reduce the contribution to the convolution from points that are close to the centre of the modified kernel. We can achieve this by adjusting the marking function $G\{(x,y);\,(x_0,y_0),\theta\}$. We shall say that the resulting detection function has been attenuated and we shall refer to the associated procedure as the attenuation technique. The idea behind the attenuated detection function is illustrated in one dimension in Figure 3.20; an analogous idea applies in the two dimensional case. Figure 3.20(a) is the kernel function $k(x)$ centred at $x = x_0$. The original marking function $g(x)$ is a step function from -1 to 1 and is presented in panel (b). The detection function obtained from the product $k(x)g(x)$ is shown in (c). At the centre of the detection function shown in panel (c), there is a big jump from negative to positive, and this makes the detection function very sensitive to changes in the data. A false break may be estimated by the detection function shown in (c) if high noise causes a large change at a non-change point.

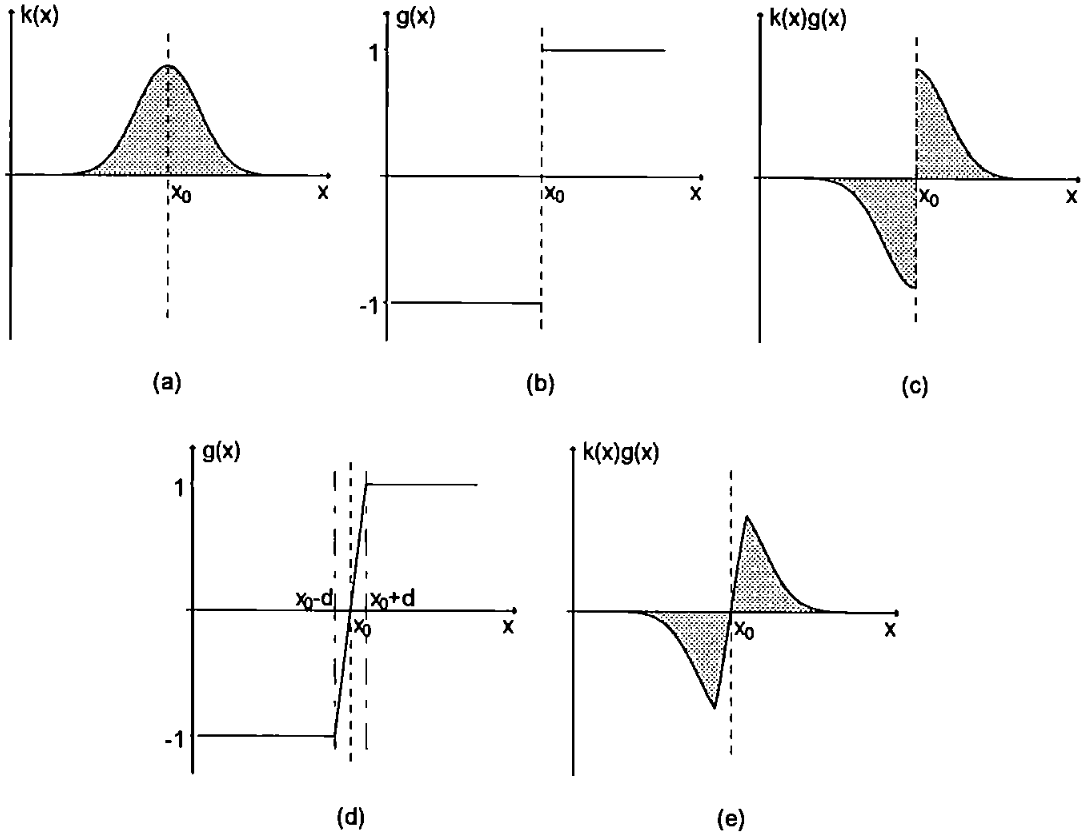


Figure 3.20 Illustration of the attenuated detection function in one dimension.

One way to overcome this problem is for the centre of detection function to vary continuously from negative to positive. In order to obtain such a detection function, we redefine the marking function. Instead of using the step function in Figure 3.20(c), we use a continuous function such as the one shown in Figure 3.20(d) in which the function $g(x)$ increases from -1 to 1 continuously in the interval $[x_0 - d, x_0 + d]$:

$$g(x) = \begin{cases} -1 & \text{if } x < x_0 - d, \\ \dfrac{x - x_0}{d} & \text{if } x_0 - d \le x < x_0 + d, \\ 1 & \text{if } x \ge x_0 + d. \end{cases}$$

The attenuated detection function is then obtained by $k(x)g(x)$ as shown in Figure 3.20(e). With such an attenuated detection function, values of Z immediately adjacent to x_0 contribute less to the value of the convolution at x_0, whereas values a little further away provide the main contribution. This means that the detection of false change points may be avoided.

We now work with the co-ordinates introduced in Section 3.2. Accordingly, $d = 1$ means that a band of width two pixels is put through the centre of the kernel detector along the major axis BB′ of the kernel. We now apply this technique with $d = 1$ to the image shown in Figure 3.19(a). We use the same values for the parameters as used for the estimate in Figure 3.19. The estimated shape is presented in Figure 3.21. A smooth and reasonable estimate is obtained. This estimate is much better than the one shown in Figure 3.19 obtained using the original detection function. The number of differing pixels for the current estimate is 178, while it was 236 for the estimate without using the attenuation technique. This example, along with other experience that we have, tells us that the attenuation technique plays an essential role in shape detection using the kernel algorithm when the image is corrupted by high levels of noise.

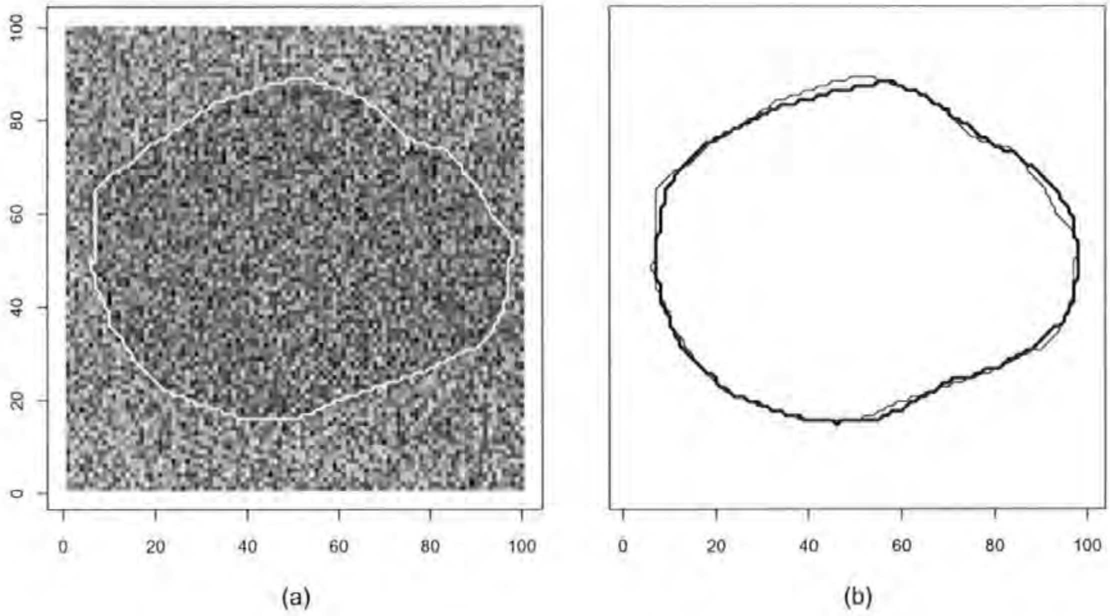
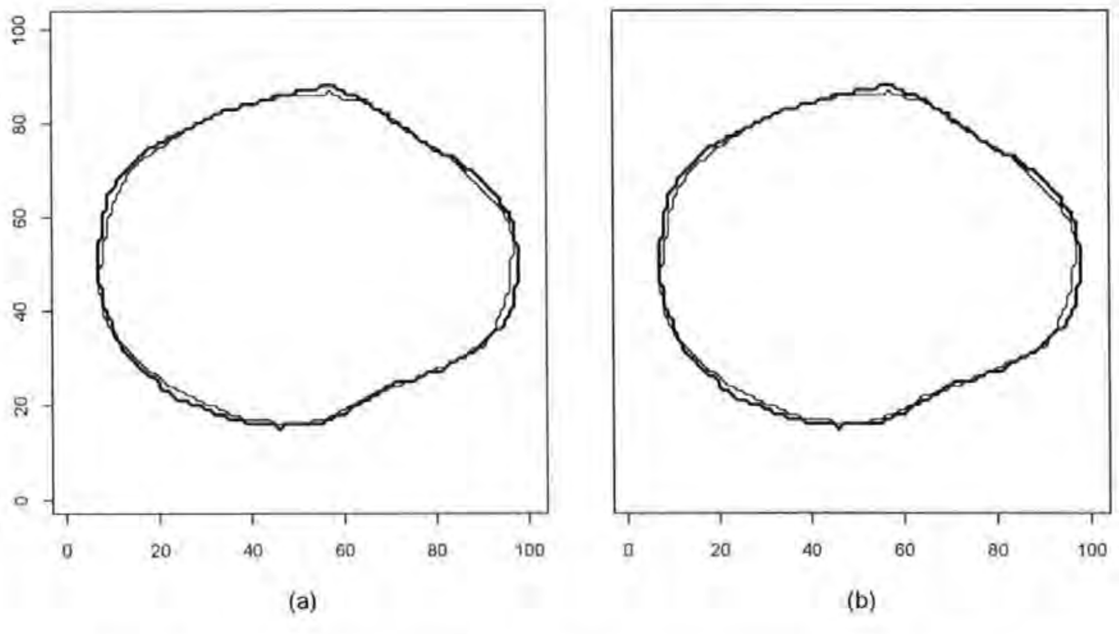


Figure 3.21 (a) The noisy image presented in Figure 3.19 together with the estimated edge (white line) detected using the attenuated detection function. (b) The estimated edge together with the true edge (darker line).

This attenuation technique can also be applied to images with low noise levels but the results are generally not better than those obtained without using the attenuated detection function. For example, we now apply the attenuation technique with $d = 1$ and $d = 2$ to the noisy image shown in Figure 3.13(b) that is corrupted by independent $N(0,1)$ noise. By searching over the kernel parameters, we found that the parameters $\sigma_0 = 3$, $\sigma_1 = 10$ and $\sigma_2 = 6$ give the best results for these values of d; these are presented in Figure 3.22. The numbers of differing pixels in these estimates are 183 and 182 for $d = 1$ and $d = 2$, respectively, while it is 147 in the best estimate without using the attenuation technique shown in Figure 3.18. These errors tell us that the attenuation technique does not work well when the image is corrupted by independent $N(0, 1)$ noise. However, if we compare the estimates in Figure 3.22 with the estimate in Figure 3.18, we can see that the attenuation technique leads to a more smooth estimate. The same phenomenon occurred in the estimate presented in Figure 3.21 compared with the estimate presented in Figure 3.20. Accordingly, the attenuation technique can lead to smoother estimates.



(a)

(b)

Figure 3.22 (a) The edge produced using the attenuated detection function with $d = 1$, together with the true edge (darker line). (b) The edge produced using the attenuated detection function with $d = 2$, together with the true edge (darker line).

Table 3.6 presents the errors for our estimation procedure for different noise levels κ and different values of d. The true image is the head shape shown in Figure 3.13(a). There is just one simulation for every noise level and value of d. This table indicates that the noise level κ determines whether or not the attenuation technique should be employed: when $\kappa \le 1.5$ the attenuation technique is not needed for shape detection, but may be useful for smoothing purposes; when $\kappa > 1.5$, the attenuation technique can improve the accuracy of the estimate. Some interesting points arise from this table. An almost stable estimate can be obtained when $d \ge 10$ in the 100×100 grid. When the attenuation technique does lead to an improved estimate, better results are obtained using higher values of d. In Section 3.8.4 we present a more thorough investigation of the performance of the algorithm for different κ and d. This investigation led to similar conclusions.

Table 3.6 Errors (number of differing pixels between the estimated shape and the true shape) for various values of κ and d

d	Noise level κ			
	1.0	1.5	2.0	3.0
0	147	161	210	236
1	183	244	158	178
2	182	243	148	179
3	200	238	152	180
4	197	240	157	180
10	197	240	147	179
15	198	240	147	179
25	198	239	147	179
50	198	239	147	179

3.8.4 *A simulation study*

In order to assess the performance of the modified kernel algorithm, a simulation study was conducted based on the original image shown in Figure 3.23. The ellipse shown in Figure 3.23 is the same shape as the ellipse shown in Figure 3.6(a) where it is presented by means of a one pixel thick edge.

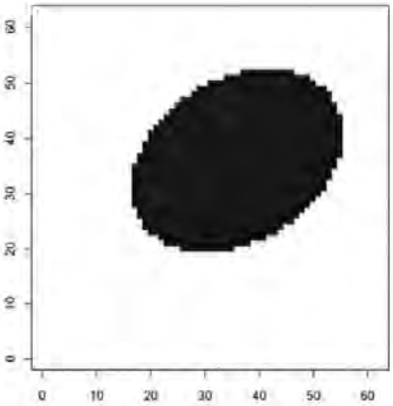


Figure 3.23 A binary image of an ellipse used for our simulation study.

Our data were obtained by adding independent $N(0,\kappa)$ noise to the original image. We used the stopping rule described in Section 3.7, that is, the algorithm was stopped as having failed if the resulting edge was not closed, or if the number of pixels in the resulting edge exceeded 1.5 times the number of pixels in the true edge so that the resulting edge was wandering around the image. For each level of four different noise variances κ, namely $\kappa = 0.5$, 1.0, 2.0 and 3.0, and for each of six different values of d, namely $d = 0, 1, 2, 5, 10$ and 20, we applied the adjusted kernel algorithm to 10 noisy images. For each of the four different noise levels, the kernel parameters were chosen by experience. The starting pixel was randomly selected from the true edge points and it was fixed in one simulation in order to investigate the effect of d.

Table 3.7 records the results of the simulation study. The numbers of pixels that differ between the true shape and the estimated shape for each simulation are listed if the algorithm stopped successfully; the symbol — is used if the algorithm failed. Also presented in Table 3.7 is the mean and standard deviation (sd) of the errors for each value of d. The mean provides an estimate of the error rate for each pair of κ and d.

When $\kappa = 0.5$, no failures occur and the mean and standard deviation for each value of d is calculated using all ten simulations. Table 3.7 shows that the performance of the algorithm is consistently good for $\kappa = 0.5$ and $d = 0$.

In Figure 3.24(a) error is plotted against d for each of the ten simulations with $\kappa = 0.5$. Each line represents one simulation. It is obvious that the lowest error always occurs when $d = 0$, and the error becomes stable as d increases.

Table 3.7 Results of the simulation study for the adjust kernel algorithm

Variance	*d*	Simulations											
		1	2	3	4	5	6	7	8	9	10	Mean*	sd
0.5	0	23	23	26	25	25	14	17	27	22	25	22.7	4.14
	1	38	37	33	26	36	36	29	46	42	43	36.6	6.19
	2	45	36	30	29	41	38	32	46	35	39	37.1	5.86
	5	47	41	29	43	44	35	32	48	29	41	38.9	7.14
	10	46	41	33	44	44	35	36	58	35	42	41.4	7.40
	20	46	41	34	44	45	35	36	58	35	42	41.6	7.32
1.0	0	—	58	97	75	99	—	47	77	49	64	70.8	19.96
	1	76	68	99	90	66	132	71	73	69	75	76.4	11.78
	2	81	74	86	81	65	128	63	66	71	71	72.1	8.01
	5	86	75	82	78	55	130	58	66	73	75	70.3	9.65
	10	86	75	94	80	57	146	57	69	72	75	72.4	12.05
	20	86	76	94	80	58	146	57	69	72	75	72.6	11.95
2.0	0	—	—	171	—	—	—	161	116	—	—	—	—
	1	86	140	83	88	112	137	142	107	186	94	109.9	24.27
	2	84	90	63	88	113	146	140	113	—	94	103.4	27.05
	5	84	90	70	88	109	147	144	114	—	95	104.6	26.60
	10	84	90	70	88	109	147	145	109	—	95	104.1	26.62
	20	84	90	70	88	109	147	145	109	—	95	104.1	26.62
3.0	0	—	—	—	—	—	—	—	—	—	—	—	—
	1	—	138	129	71	—	182	132	—	—	116	—	—
	2	—	145	137	74	—	166	133	—	—	110	—	—
	5	—	141	128	74	—	146	132	—	—	138	—	—
	10	—	142	125	74	—	—	132	—	—	136	—	—
	20	—	142	125	74	—	—	132	—	—	136	—	—

Notes:

1. For each simulation number we use the same realisation of the noise process.
2. Result — means the algorithm failed to stop due to the resulting edge not being closed or due to the maximum number of pixels being exceeded.
3. * For $\kappa = 0.5$, the mean and standard deviation are calculated by using the results from all ten simulations. For $\kappa = 1.0$, the mean and standard deviation are calculated by using the results from all simulations except the first and sixth. For $\kappa = 2.0$, the mean and standard deviation are calculated by using the results from all simulations except the ninth. For $\kappa = 2.0$ with $d = 0$ and for $\kappa = 3.0$, no mean and standard deviation are calculated because of too many failures.

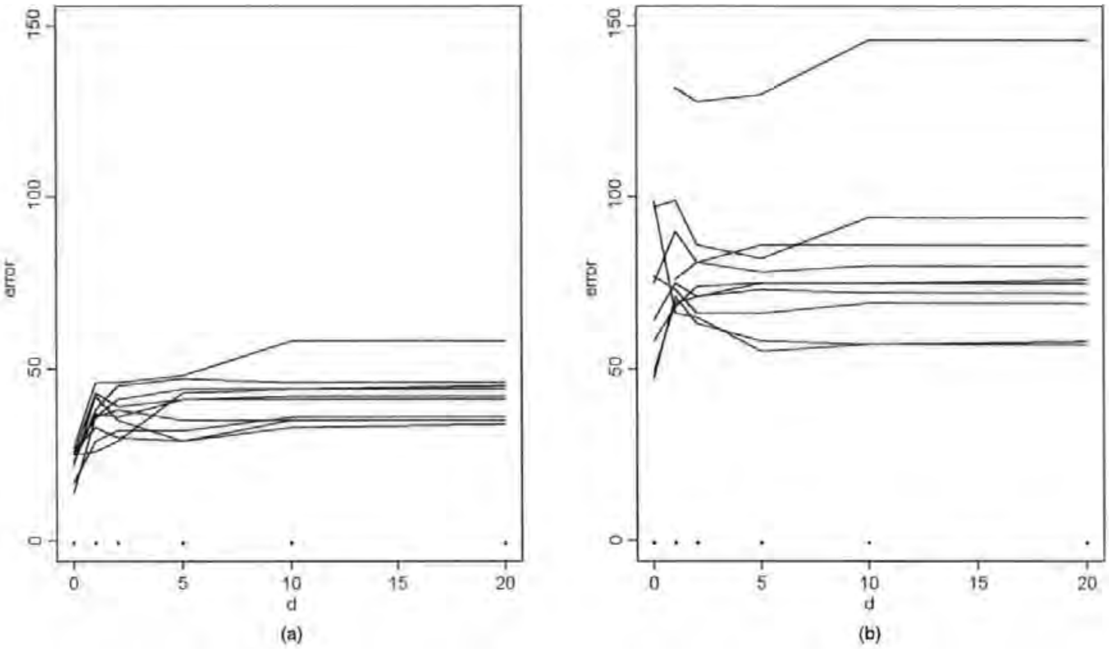


Figure 3.24 Error plot for ten simulations against $d = 0$, 1, 2, 5, 10 and 20. The dots indicates values of d. (a) $\kappa = 0.5$, (b) $\kappa = 1.0$

When $\kappa = 1.0$, failure occurs for the first and sixth simulation when $d = 0$. Among the other simulations, the best estimate was usually obtained when $d = 0$. The means and standard deviations for $\kappa = 1.0$ and for each value of d are calculated from all simulations except the first and sixth. The standard deviation of the results for $d = 0$ is higher than that of the results for the other values of d.

In Figure 3.24(b) error is plotted against d for each of the ten simulations with $\kappa = 1.0$. Each line indicates one simulation. Note that the first point is not plotted for the first and sixth simulations because the algorithm failed. From these plots we see that the lowest error sometimes occurs when $d = 0$ and sometimes for other values of d. We note that the results tend to stabilise as d increases.

When $\kappa = 2.0$ and $d = 0$, failure occurs for all but three simulations. When $d > 0$ good estimates are obtained except for the ninth simulation. For each value of $d > 0$, we calculated the mean and standard deviation across all simulations except the ninth. Clearly the use of the attenuated detection function is important here. As d increases the results seem to be quite stable.

When $\kappa = 3.0$ and $d = 0$, failure occurs for all simulations. For the second, third, fourth, seventh and tenth simulation, estimates are obtained when $d > 0$. Although most of these results are not good, it does appear that the attenuation technique allows estimates to be obtained sometimes for images corrupted by high levels of noise. Because there are many failures we have not calculated means and standard deviations for this level of noise. Obvious conclusions can, however, be drawn from the numbers themselves, that is, the use of the attenuation technique is important here, and as d increases the results seem to be quite stable. Although there are many failures for $\kappa = 3.0$, it should be remembered that the true shape is a binary image so that the resulting shape detection problem is very hard with this level of noise.

Table 3.7 suggests the same interesting result as Table 3.6: it seems that an almost stable estimate can be obtained for $d \geq 10$ in the 61×61 grid, and better results can be obtained with $d > 2$ if the attenuation technique can improve the estimate.

Accordingly, the conclusions from this simulation study are:

- As the variance κ increases the quality of performance decreases;
- For less noisy images, such as those obtained when $\kappa = 0.5$, the attenuation technique is not needed;
- For more noisy image, such as those obtained when $\kappa \geq 1$, the attenuation technique can improve the estimate;
- When applying the attenuation technique with the modified kernel algorithm to identify a shape in a very noisy image, a high value can be assigned to d and the quality of performance seems robust to the choice of d.

3.8.5 *Simulation study comparing the performance of the modified kernel algorithm with that of the cascade based simulated annealing algorithm*

In this section we present a small simulation study to compare the performance of the modified kernel algorithm with that of the cascade based simulated annealing algorithm that we described in Chapter 2. Our simulation study is still based on the binary image of an ellipse shown in Figure 3.23 and the data were obtained by adding independent $N(0,\kappa)$ noise, with $\kappa = 0.5, 1.0, 2.0$ and 3.0, to the original binary image. Following the results from the simulation study in Section 3.8.4, we chose $d = 0$ when $\kappa = 0.5$ and $d = 5$ when $\kappa = 1.0$, $\kappa = 2.0$ and $\kappa = 3.0$. For each value of κ, ten noisy images were considered. We applied both the modified kernel algorithm and the cascade based simulated annealing algorithm (CSA) to each image. For the modified kernel algorithm six failures occurred in the simulations when $\kappa = 3.0$.

Table 3.8 Results of the simulation study for comparing the performance of the modified kernel algorithm with that of the cascade based simulated annealing annealing algorithm

	κ							
	0.5		1.0		2.0		3.0	
	Kernel	CSA	Kernel	CSA	Kernel	CSA	Kernel	CSA
Simulation	25	34	62	66	72	104	—	131
	29	25	55	58	57	62	—	158
	25	34	64	61	71	78	105	110
	25	25	71	56	55	100	128	70
	28	25	67	59	124	51	—	97
	45	30	60	57	166	138	—	141
	44	37	61	69	101	127	161	128
	37	42	72	76	59	61	—	107
	28	38	70	63	77	110	—	91
	30	33	66	69	91	122	117	85
Mean	31.6	32.3	64.8	63.4	87.3	95.3	127.8	111.8
sd	7.7	6.0	5.4	6.5	35.1	30.6	24.1	27.4
p-value	0.79		0.54		0.25		—	

sd=standard deviation

Table 3.8 presents the results of the simulation study. The numbers of pixels that differ between the true shape and the estimated shape for each simulation are listed. The results of a paired t-test were that there is no significant difference between the two algorithms when $\kappa = 0.5, 1.0$, and 2.0. These results confirm that the modified kernel algorithm works well when $\kappa \leq 2$, but often fails when $\kappa > 2$. The procedures for both algorithms took almost the same computer time, but the kernel algorithm is considerably less complicated.

3.8.6 *Application to real data*

We now apply the algorithm described above to the ultrasound image of a human ovarian cyst shown in Figure 1.8 to which we applied the direct and the cascade based simulated annealing algorithms successfully and we obtained the estimated edge of the underlying shape in Chapter 2; see Figure 2.15.

We chose the parameters $\sigma_0 = 3$, $\sigma_1 = 5$, $\sigma_2 = 3$ and $d = 1$ that gave the best result as shown in Figure 3.25 by the white line. The result obtained by using the cascade based simulated annealing algorithm as shown in Figure 2.15(b) is presented in this figure by the black line for comparison. We see that the two estimates are quite similar. We mention here that the procedure took 3684 seconds DOS-time which is almost the same as that of the cascade based algorithm that we applied to the same image in Section 2.5 of Chapter 2, whereas the direct simulated annealing approach took about 1.5 times as much computing time.

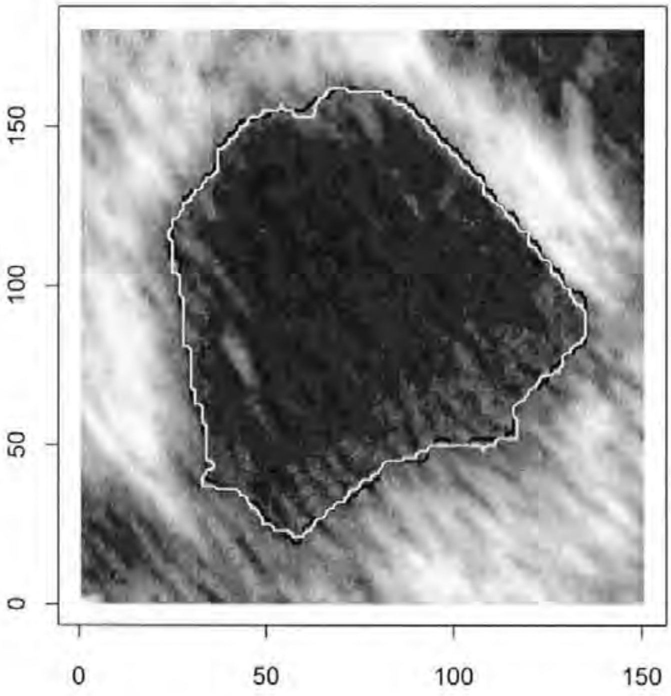


Figure 3.25 A real example. The original ultrasound image of a human ovarian cyst is shown in Figure 1.8. The white line is the estimate of the underlying shape, while the black line is the outline obtained in Chapter 2 by using the cascade based simulated annealing algorithm.

3.9 *Further works*

The kernel algorithm that we have discussed in this chapter is designed to detect edges of a shape that has an interior point from which every point of the edge is directly visible. For an arbitrary shape, such as the picture of a duck body presented in Pievatolo and Green (1998) and shown here in Figure 3.26, the above algorithm will not work properly. We are grateful to Dr A. Pievatolo and Professor P. J. Green for having sent us this binary image.

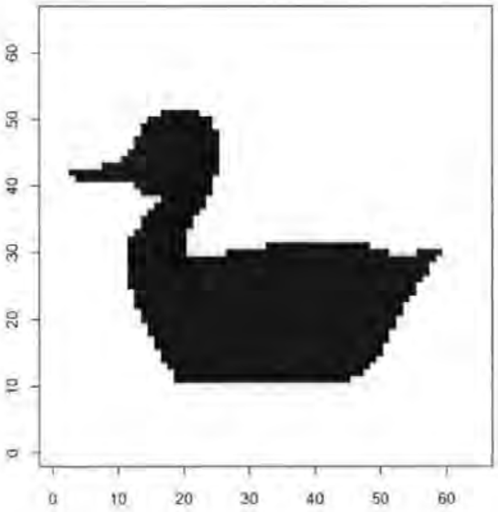


Figure 3.26 64×64 test image (Pievatolo and Green, 1998).

Pievatolo and Green (1998) add independent normal noise to the duck image and then estimate the outline by fitting a polygon with any number of sides. In order to do this they derive a new probabilistic model for the generation of a polygon in a compact subset of $\Re^2$

and use this model as a prior distribution in a Bayesian approach. Simulations from the prior and posterior distributions are carried out through the reversible jump Markov chain Monte Carlo algorithm proposed by Green (1995). The authors demonstrate that the results obtained from this approach are very much better than those obtained by pixel-based methods.

As a first step to making the kernel algorithm work on the duck image we extend the original set of candidate points described in Section 3.3 and presented in Figure 3.4 to include any one of the first and second order neighbours of the current edge point. To prevent the edge from retracing its steps, we exclude points that have been chosen in the last few steps from the set of candidate points. This modification is easy to implement but a new problem arises.

Around the beak and neck of the duck the direction of the outline changes suddenly often by angles around $\frac{\pi}{2}$. Accordingly, we applied the dynamic angle version of the kernel algorithm with $\theta \in \left(\theta^{*} - \varepsilon, \theta^{*} + \varepsilon\right)$, where θ^{*} is the current angle of the kernel function. We let $\varepsilon = \frac{\pi}{2}$ so as to allow the kernel function to change angle in a very wide range. Simulation results show that these modifications can cause the kernel to wander off in completely the wrong direction so leading to a meaningless shape.

What we now suggest is modifying the kernel function itself. The original kernel function (3.1) has a straight line as axis of symmetry; see Figure 3.3 and 3.17. We can change its shape by bending the axis of symmetry through an angle ϕ at its centre so enabling the kernel function to match better the edge of the shape. Figure 3.27 shows four examples of the bent kernel functions. In Figure 3.27 we indicate the position of the inside and outside of the shape near the point at which the kernel bends. The algorithm would then proceed by optimising over ϕ as well as the other parameters. We hope to develop this method in the near future.

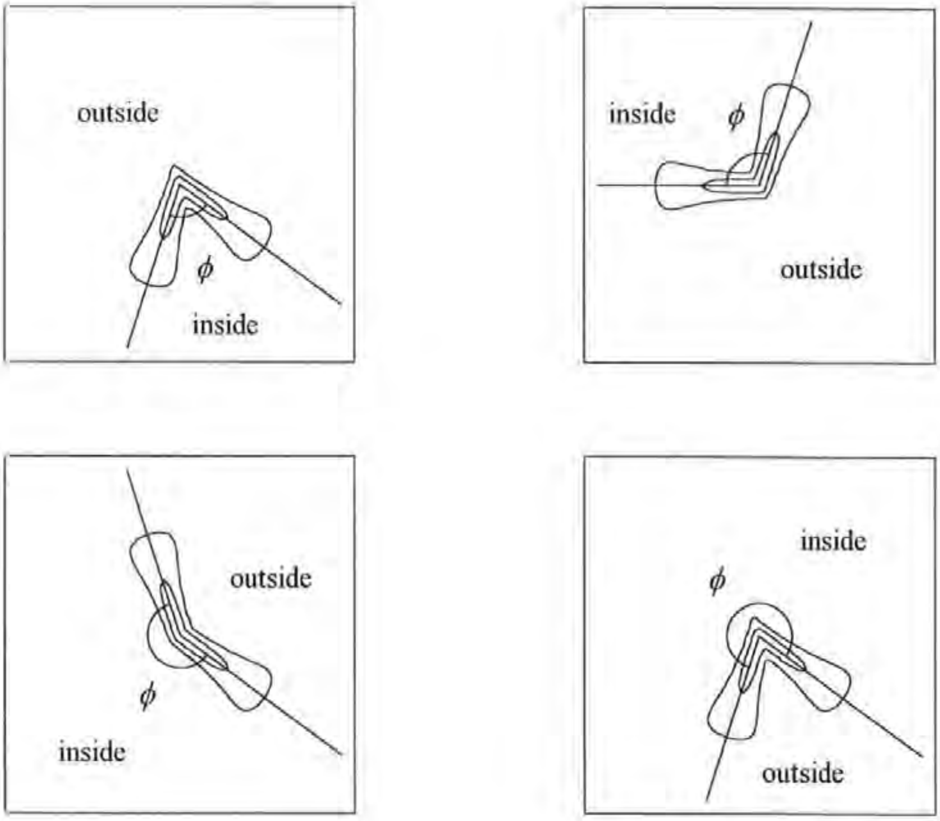


Figure 3.27 Examples of the bent kernel function.

Chapter 4

Study of the effect of the kernel parameters

4.1 *Introduction*

The kernel function $K\{(x,y);\ (x_0,y_0),\theta\}$ introduced in Chapter 3 is defined in terms of three parameters, σ_0, σ_1 and σ_2: σ_0 controls mainly the width of the kernel, while σ_1 controls mainly its length and σ_2 mainly its curvature. This is illustrated in Figure 3.3, for example. Therefore, edge detection using the kernel function method depends upon the choice of the values of these parameters.

In Section 3.5, we have discussed several variations of the algorithm described in Section 3.3. For example in steps 3 and 4 instead of maximising the convolution f over θ in the discrete set of rotation angles $\left(\theta^* - 2\delta, \theta^* - \delta, \theta^*, \theta^* + \delta, \theta^* + 2\delta\right)$, where θ^* is the current angle of the kernel, we maximised it over θ in the continuous interval

$(\theta^* - \varepsilon, \theta^* + \varepsilon)$ for some fixed $\varepsilon > 0$. We also allowed the shape of a standardised kernel to vary at each point by maximising f over the parameters σ_0, σ_1 and σ_2 as well as θ. Neither of these variations produced any noticeable improvement in results.

In this chapter we aim to illustrate the effect of the kernel parameters. For simplicity, we take our true image to be a circle in a 61×61 grid and add Gaussian noise to it. Figure 4.1 is a binary image comprising 61×61 pixels showing the outline of a circle centred at $(31, 31)$ with radius 16. The records of the image take the value one at the black outline pixels and the value zero at the remaining white pixels.

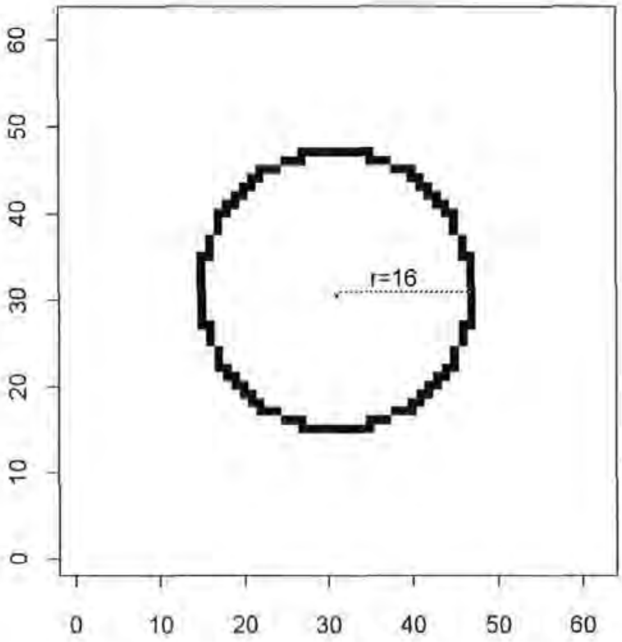


Figure 4.1 The original binary pixel image, a circle centred at (31,31) with radius 16.

In the following parts of this chapter, we think of this image as occupying part of the plane with co-ordinates from -30 to 30 on both axes, so that every integer co-ordinate corresponds to one pixel; see Section 3.2 for details. In these kernel co-ordinates, the radius of the circle in the image presented in Figure 4.1 remains at 16.

In Section 4.2, we apply the kernel algorithm to detect the circles such as the one shown in Figure 4.1 from their noisy images. We discuss how the error of the estimate obtained depends on the parameters σ_0, σ_1 and σ_2. Then in Section 4.3 we present a probability study about edge pixel detection.

4.2 *The effect of the kernel parameters*

We now apply the kernel algorithm to detect the circle from noisy images. We will use the number of pixels that differ between the true shape and the estimated shape as our measure of the error of the estimate. For example, the number of differing pixels in the estimate in Figure 3.6 is the total number of black and grey pixels in Figure 3.6(d).

Before we discuss the effect of the kernel parameters, we begin by studying the effect of different noise levels and circular radii.

We added independent Gaussian noise with variance $\kappa = 0.01$, 0.09, 0.25, 0.36 and 0.50 to the image shown in Figure 4.1. For every noise level, we simulated 20 realisations of the noise process. For each realisation we performed the estimation by applying the peanut kernel function with parameters $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$, and with the discrete set of rotation angles. The initial point was always chosen randomly from the true edge pixels.

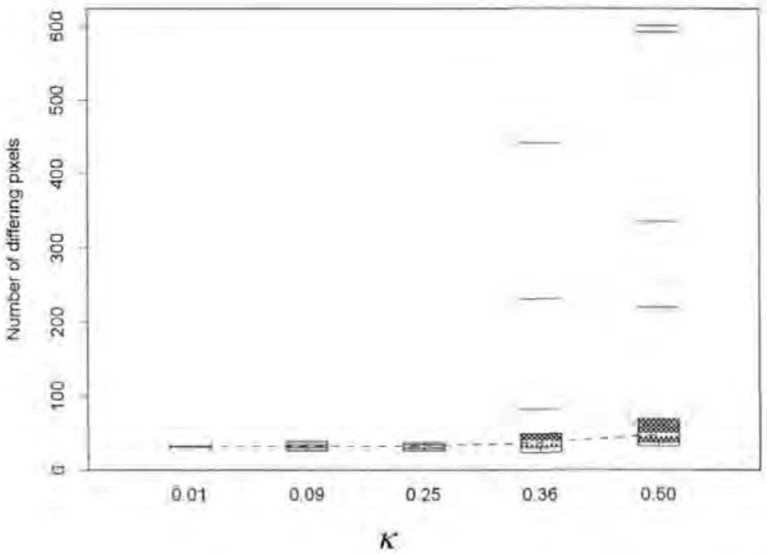


Figure 4.2 Boxplots of the number of differing pixels for the different noise levels added to the true image of a circle with radius 16. The dashed line links the medians of the five data sets, each of which is obtained from 20 realisations of the noise process. The kernel parameters were chosen as $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$.

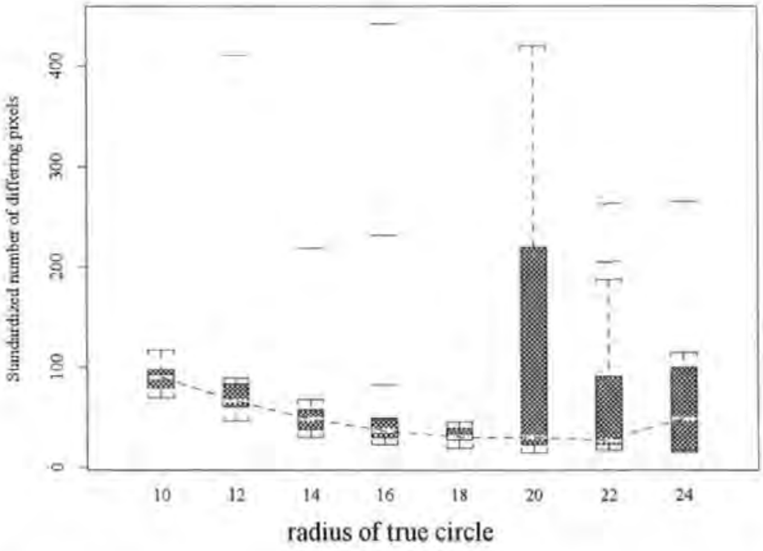


Figure 4.3 Boxplots of the standardised number of differing pixels for the different radii of the true circle. The dashed line links the medians of the eight data sets, each of which is obtained from 20 realisations of the noise process. The kernel parameters were chosen as $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$, and the noisy images are obtained by adding independent $N(0, 0.36)$ noise.

Boxplots of the number of differing pixels for each noise level κ are presented in Figure 4.2. The dashed line in the figure links the medians of the five data sets each of which is obtained from 20 realisations of the noise process. It is not difficult to understand that the lower the noise level, the lower and the more stable the number of differing pixels of the estimate.

In order to understand the relationship between the curvature of the edge curve and the parameters, we applied the peanut kernel function with $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$ to images based on circles with radii 10, 12, 14, 16, 18, 20, 22 and 24. We added independent $N(0, 0.36)$ noise to the original images, and twenty realisations of the noise process were used for each radius. Because the circles have different areas, we standardised the number of differing pixels according to the circle with radius 16:

$$\text{standardised number of differing pixels} = \text{number of differing pixels} \times \left(\frac{16}{\text{radius}}\right)^2.$$

Figure 4.3 presents boxplots of the number of differing pixels for different radii. The dashed line links the medians of the standardised numbers of differing pixels for each radius. It seems that the parameters we used in these simulations are not suitable for too small or too large radii in the 61×61 pixel image. For smaller circles, such as those with radius 10 or 12, the medians of the standardised number of differing pixels are much higher than those with radius 16, 18, 20 or 22. This indicates that the kernel is too big for smaller circles. For bigger circles with radius 24, the median of the standardised number of differing pixels is higher than those with radius 16, 18, 20 or 22. For circles with radii 20 and 22, the standardised numbers of differing pixels show higher variation than for other radii although the medians are lower. This indicates that the kernel is too small for bigger circles. For the parameters $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$, the minimum error occurs when the radius of the true circle is 16 or 18.

Now we study the effect of the parameters of the kernel function on the estimate of the circle with radius 16. The peanut kernel function was used and independent $N(0, 0.5)$ noise was added.

Figure 4.4 presents boxplots of the number of differing pixels for the following values of the parameter σ_0: 0.3, 0.7, 1.0, 1.5, 2.0, 2.5, 3.0 and 5.0. The dashed line links the medians of the eight data sets, each of which is obtained by 20 simulations as before. The other two kernel parameters were chosen as $\sigma_1 = 4$ and $\sigma_2 = 20$. The minimum error in Figure 4.4 seems to occur for values of σ_0 around 1.0 to 1.5. We have already explained that the parameter σ_0 controls mainly the width of the kernel. If a kernel is very narrow relative to the true shape, it may not be able to take into account sufficient edge information to detect true edge pixels. On the other hand, if a kernel is too wide relative to the true shape, it may take into account too much information from non-edge pixels. Hence, if the parameter σ_0 takes a very low value such as 0.3, then the kernel is too narrow with the result that it is easily attracted by pixels that are not on the edge but that have a high grey-level. If the parameter σ_0 takes a very high value, say 5.0, then the kernel covers lots of pixels that are not useful for detecting local edge pixels but whose values may substantially affect the result.

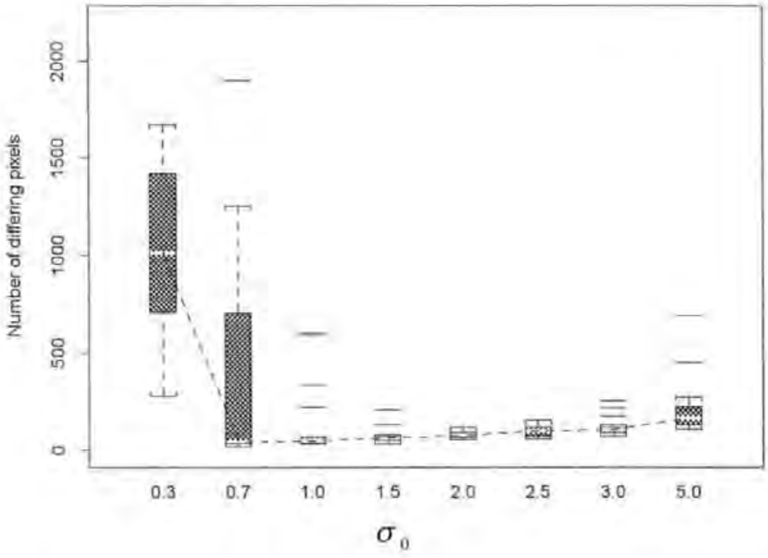


Figure 4.4 Boxplots for different values of the parameter σ_0. The dashed line links the medians of the eight data sets, each of which is obtained from 20 realisations of the noise process. The other two kernel parameters were chosen as $\sigma_1 = 4$ and $\sigma_2 = 20$, and the noisy images are obtained by adding independent $N(0, 0.5)$ noise.

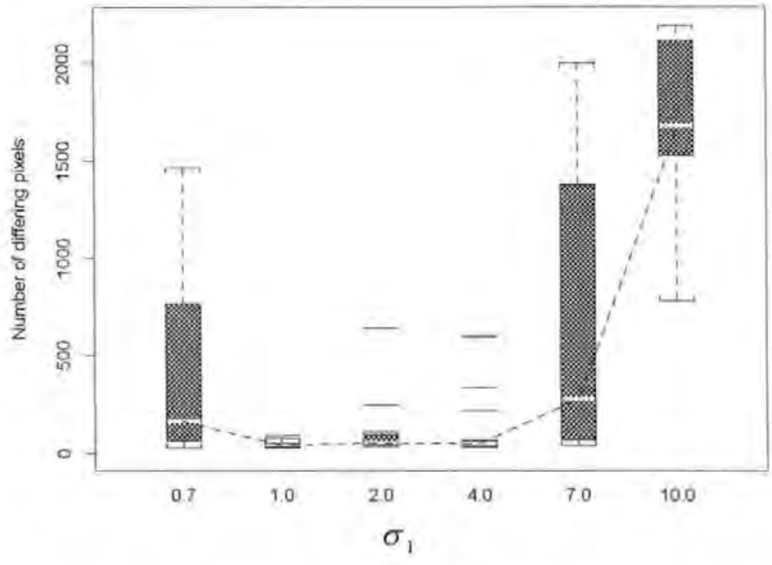


Figure 4.5 Boxplots for different values of the parameter σ_1. The dashed line links the medians of the six data sets, each of which is obtained from 20 realisations of the noise process. The other two kernel parameters were chosen as $\sigma_0 = 1$ and $\sigma_2 = 20$, and the noisy images are obtained by adding independent $N(0, 0.5)$ noise.

Figure 4.5 presents boxplots of the number of differing pixels for the following values of the parameter σ_1: 0.7, 1.0, 2.0, 4.0, 7.0 and 10. The dashed line links the medians of the six data sets, each of which is obtained by 20 simulations as before. The other two kernel parameters were chosen as $\sigma_0 = 1$ and $\sigma_2 = 20$. We see that poor performance results from very low and very high values of the parameter σ_1. We have already explained that the parameter σ_1 controls mainly the length of the kernel. So reasons similar to those presented in the discussion of Figure 4.4 apply here. If a kernel is very short relative to the true shape, it covers only a very short part of edge with the result that it may not be able to take into account sufficient edge information to detect true edge pixels. On the other hand, if a kernel is very long relative to the true shape, there may be no edge pixels covered by the two ends of kernel with the result that the kernel takes into account too much information that comes from non-edge pixels. Hence, if the parameter σ_1 takes a very low value such as 0.7, then the kernel is too short with the result that it is easily attracted by pixels that are not on the edge but that have a high grey-level. If the parameter σ_1 takes a very high value, say 10, then the kernel is very long and so covers lots of pixels that are not useful for detecting local edge pixels but whose values may substantially affect the result.

Figure 4.6 presents boxplots of the number of differing pixels for the following values of the parameter σ_2: 5, 10, 15, 20, 25 and 30. The dashed line links the medians of the six data sets each of which is obtained by 20 simulations as before. The other two kernel parameters were chosen as $\sigma_0 = 1$ and $\sigma_1 = 4$.

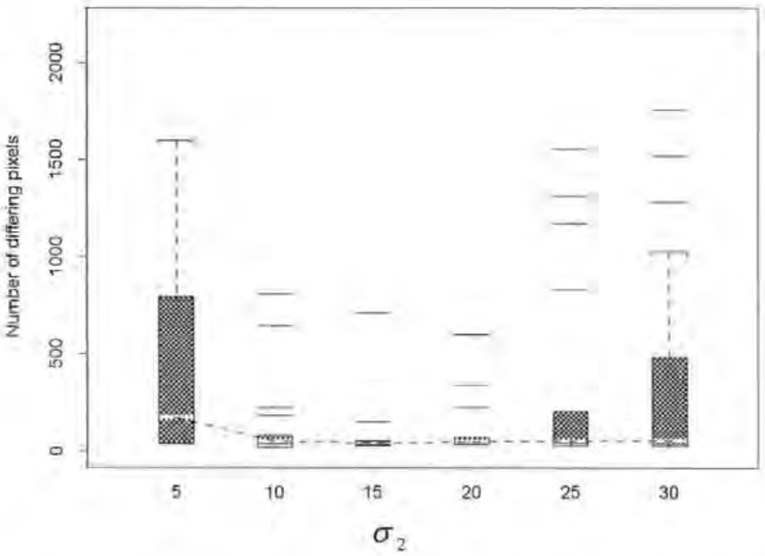


Figure 4.6 Boxplots for different values of the parameter σ_2. The dashed line links the medians of the six data sets, each of which is obtained from 20 realisations of the noise process. The other two kernel parameters were chosen as $\sigma_0 = 1$ and $\sigma_1 = 4$, and the noisy images are obtained by adding independent $N(0, 0.5)$ noise.

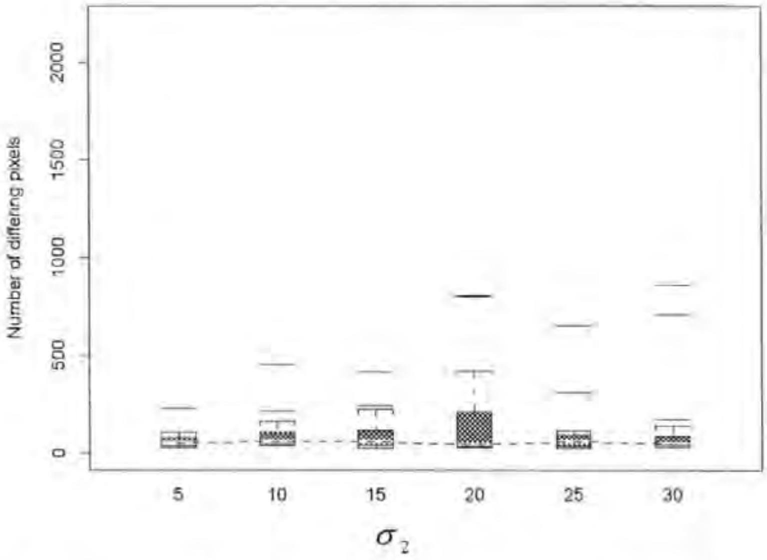


Figure 4.7 Boxplots for different values of the parameter σ_2. The dashed line links the medians of the six data sets, each of which is obtained from 20 realisations of the noise process. The other two kernel parameters were chosen as $\sigma_0 = 1$ and $\sigma_1 = 1$, and the noisy images are obtained by adding independent $N(0, 0.5)$ noise.

Figure 4.6 illustrates the dependence of the algorithm on the parameter σ_2; we see that values of σ_2 between 10 and 20 give us the minimum number of differing pixels. If the value of σ_2 is too small or too big, the result is poor. We know that the parameter σ_2 controls mainly the curvature of the kernel. A small value of σ_2 means that the kernel is too straight for it to follow the curvature of the circle with the result that less useful information is used in the calculation of the convolution. A big value of σ_2 means that the kernel is too curved for it to follow the curvature of the circle again with the result that less useful information is used.

We see from Figure 4.6 that the variances of the number of differing pixels are high for all the values of σ_2 we choose. This suggests that the values selected for the parameter σ_0 and σ_1 may be inappropriate. From Figure 4.5 we see that the value $\sigma_1 = 1$ gives us consistently good estimates when $\sigma_0 = 1$ and $\sigma_2 = 20$. So now we fix $\sigma_0 = 1$ and $\sigma_1 = 1$, and repeat the simulation study that led to Figure 4.6. Figure 4.7 presents boxplots of the result. The medians for the different values of σ_2 are almost the same in Figure 4.6 ($\sigma_1 = 4$) and Figure 4.7 ($\sigma_1 = 1$). However, the results when $\sigma_1 = 1$ are much less variable than those where $\sigma_1 = 4$ for all selected values of the parameter σ_2. This suggests that setting $\sigma_1 = 4$ leads to a kernel that is too long for the circle with radius 16 in the 61×61 array, while setting $\sigma_1 = 1$ leads to a very good choice of kernel. In fact, this agrees with Figure 4.5 in the sense that less variation is associated with $\sigma_1 = 1$ than with $\sigma_1 = 4$.

From Figures 4.4 to 4.7 we see that the parameters σ_0 and σ_1, that control mainly the width and the length of the kernel respectively, are the key parameters in the kernel algorithm. A good choice of the parameter σ_2 that controls the curvature of the kernel can reduce the error and its variability when the other two parameters σ_0 and σ_1 are chosen inappropriately; see Figure 4.6 for this. If the other two parameters σ_0 and σ_1 are chosen appropriately, there is not a large difference in the numbers of differing pixels for a very wide range of values of σ_2; see Figure 4.7. Figure 4.6 and Figure 4.7 tells us that the performance of the kernel algorithm is relatively robust to the choice of the parameter σ_2.

4.3 *A probability study*

In the previous section we discussed the effect of the kernel parameters. There, we measured the errors by the number of pixels that differ between the true shape and the estimated shape. Here we present a further study about the parameters based on a different measure of the quality of the estimated shape.

In this section we will study the probabilities that successive true edge pixels are found correctly, given that the initial pixel is a true edge pixel. We will estimate these probabilities by means of a simulation study in which we randomise over both realisations of the noise process and initial edge pixels.

The true shape is the one presented in Figure 4.1. We apply the kernel algorithm described in Chapter 3 to a noisy image derived from Figure 4.1 and focus attention on the detection of the first 20 edge pixels. One hundred simulations were performed for each fixed set of kernel parameters in order to estimate the probability of correctly detecting the i^{th} estimated edge pixel. For example, if the i^{th} edge pixel is correctly detected n times, then the probability that it is correctly detected is estimated as $n/100$. In order to help us understand what kind of kernel is suitable for different noise levels, we considered four different values of κ : 0.1, 0.5, 1 and 2.

Each simulation consists of generating a new noisy image from the true circle image, and estimating its edges by means of the kernel algorithm starting from an edge pixel randomly selected from among the true edge pixels. The initial kernel rotation angle is chosen to fit the circle according to the position of the starting point. Since our circle has been discretized according to a 61×61 grid, some parts of it look very different from others. This can be seen clearly in Figure 4.1 where the horizontal and vertical segments have a different form from the remainder of the discretized circle. In order to reduce any possible bias caused by this discretization effect, the initial edge pixel for the kernel algorithm is randomly selected from among the true edge pixels.

Figure 4.8 presents estimates of the probability of correct detection for the first 20 edge pixels for different values of κ and σ_0; the other parameters are fixed as $\sigma_1 = 4$ and $\sigma_2 = 20$. The values of σ_0 considered are 1.0, 1.5, 2.0, 2.5, 3.0 and 5.0. For each plot, one hundred different realisations of the noise process are employed.

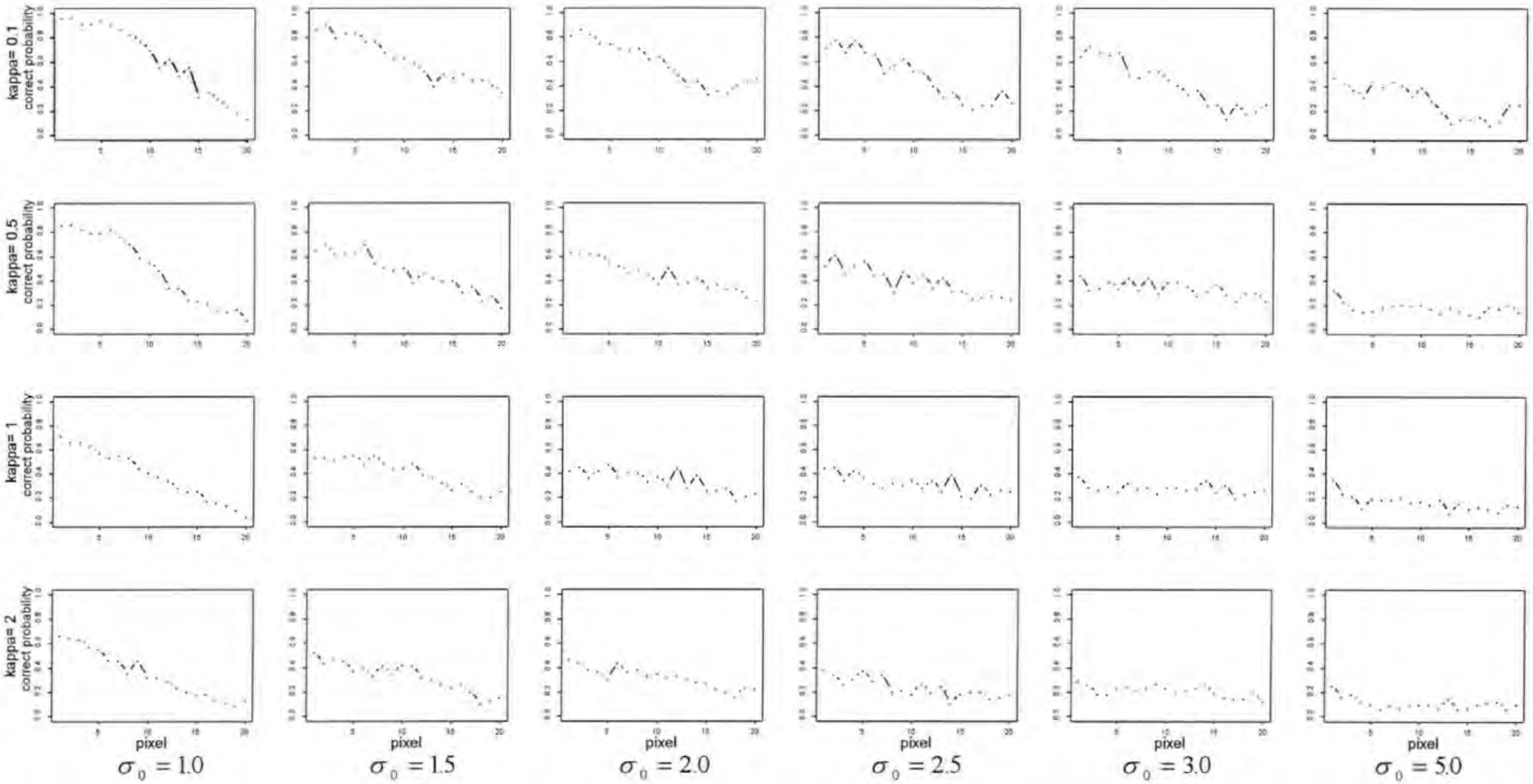
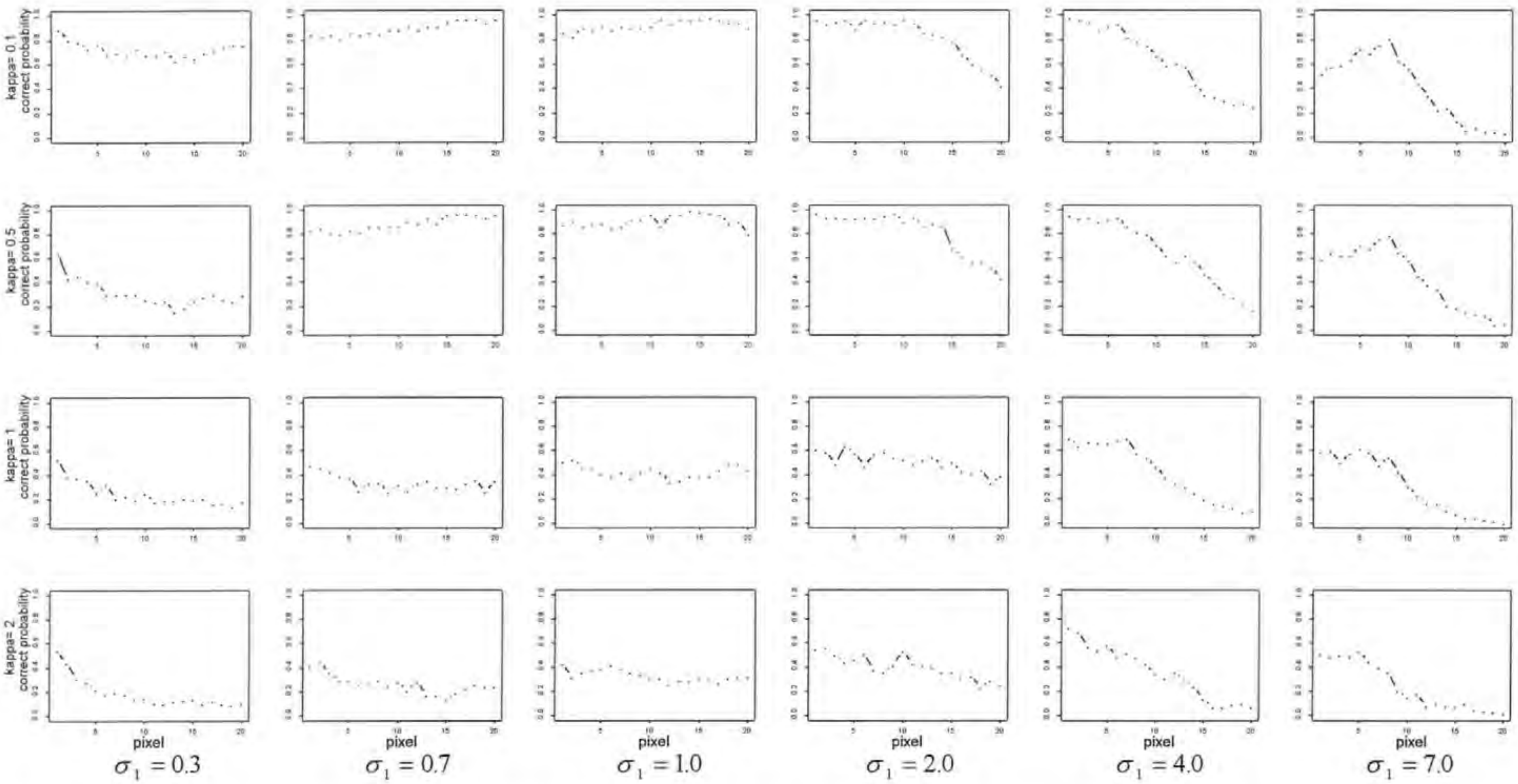


Figure 4.8 Estimates of the probability of correct detection for the first twenty edge pixels for different values of κ and σ_0; the other two kernel parameters are fixed as $\sigma_1 = 4$ and $\sigma_2 = 20$. For each plot, 100 different realisations of the noise process are employed with a randomly chosen starting point on the true circle edge.

Figure 4.8 indicates that choosing σ_0 from 1.5 to 2.0 with $\sigma_1 = 4$ and $\sigma_2 = 20$ gives higher estimated probabilities of correct detection for the first 20 edge pixels for $\kappa = 0.1$ to $\kappa = 2$. When $\sigma_0 = 1$, the estimated probabilities of correct detection decrease rapidly to a very low level. This tells us that the kernel is too narrow to cover enough useful edge information. Because the initial kernel rotation angle is chosen in an optimal way, the estimated probabilities for the first few detected pixels are high even though the kernel is too narrow. When $\sigma_0 = 5$, we see that the estimated probabilities of correct detection remain stable, although at a very low level. The reason for this is that the kernel is so wide that useless information is covered in the calculation of the convolution. Because the kernel is so wide, the rotation of the kernel almost loses its function in the kernel algorithm, and the probability remains low from the start. The conclusions presented in Section 4.1 from Figure 4.4 are confirmed by this probability study.

We now repeat the above study, but this time we fix $\sigma_0 = 1$ and $\sigma_2 = 20$ and take different values for σ_1, namely 0.3, 0.7, 1.0, 2.0, 4.0 and 7.0. The results are presented in Figure 4.9. It can be seen that when $\kappa = 0.1$ and $\kappa = 0.5$, higher and stable probabilities are obtained by choosing σ_1 between 0.7 and 1, and when $\kappa = 1$ and $\kappa = 2$ by choosing σ_1 between 1 and 2. When σ_1 is chosen very high, the probability decreases rapidly to a very low level, while when σ_1 is chosen very low the probability remains stable but at a low level.



$\sigma_1 = 0.3$ $\sigma_1 = 0.7$ $\sigma_1 = 1.0$ $\sigma_1 = 2.0$ $\sigma_1 = 4.0$ $\sigma_1 = 7.0$

Figure 4.9 Estimates of the probability of correct detection for the first twenty edge pixels for different values of κ and σ_1; the other two kernel parameters are fixed as $\sigma_0 = 1$ and $\sigma_2 = 20$. For each plot, 100 different realisations of the noise process are employed with a randomly chosen starting point on the true circle edge.

Since the kernel parameter σ_1 controls mainly the length of the kernel, Figure 4.9 tells us that when the kernel is too long (σ_1 too high), much useless pixel information is involved with the result that the probabilities decrease rapidly. The initial probabilities are higher because the initial kernel rotation angle is chosen in an optimal way so that the kernel lies along the true edge for the detection of the initial pixels. When the kernel is too short (σ_1 too low), only a little of the local information is used with the result that the probabilities remain low from the beginning. Another useful message from Figure 4.9 is that the higher the level of noise added to the true circle shape, the higher the value of σ_1 (the longer the kernel) that is needed. However, there is a limit to the length of the kernel that should be used.

Parameter σ_2 controls mainly the curvature of the kernel; see Section 3.1. We saw in Section 4.1 that σ_2 can be used to adjust the error when the other two parameters σ_0 and σ_1 are chosen inappropriately. We now repeat the above study, but this time we fix $\sigma_0 = 1$ and $\sigma_1 = 1$ and take different values for σ_2, namely 5, 10, 15, 20, 25 and 30. The values $\sigma_0 = 1$ and $\sigma_1 = 1$ have previously led to good results from a wide range of σ_2. The results are presented in Figure 4.10. All the probability plots are stable and the different values of parameter σ_2 do not significantly affect the estimated probabilities of the correct detection. Figure 4.10 confirms the conclusions obtained from Figure 4.7 in last section.

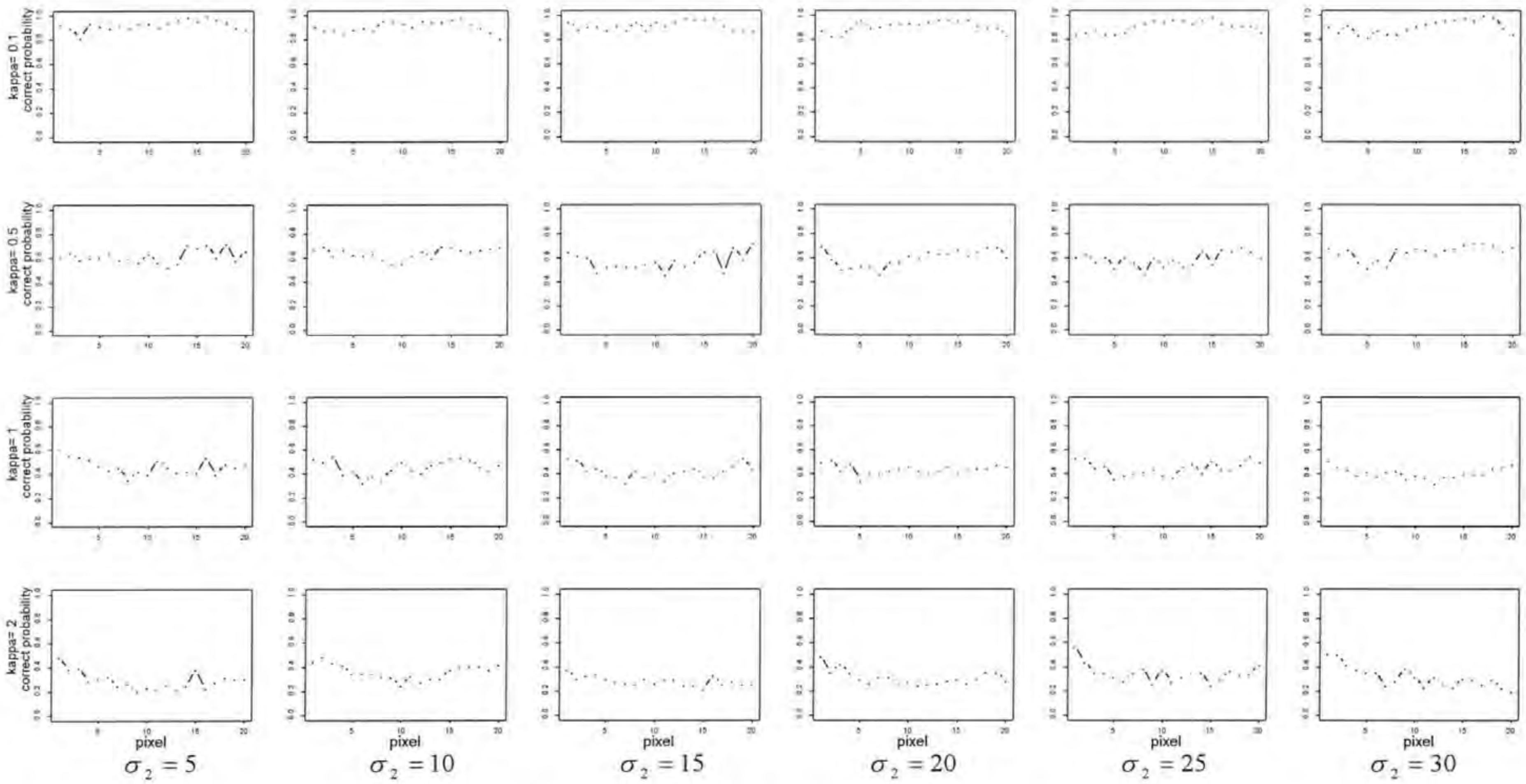


Figure 4.10 Estimates of the probability of correct detection for the first twenty edge pixels for different values of κ and σ_2; the other two kernel parameters are fixed as $\sigma_0 = 1$ and $\sigma_1 = 1$. For each plot, 100 different realisations of the noise process are employed with a randomly chosen starting point on the true circle edge.

4.4 *Conclusion*

From our error study and probability study, we conclude that in the kernel algorithm the kernel parameters σ_0 and σ_1 play the most important role, whereas the main function of the kernel parameter σ_2 is to reduce the error and its variability when σ_0 and σ_1 are chosen inappropriately. When σ_0 and σ_1 are selected appropriately, the performance of the algorithm is relatively robust to the choice of the kernel parameter σ_2.

Chapter 5

Active Contour Models: The Snake Model

The energy minimising active contour model, better known as the snake model, proposed by Kass, Witkin and Terzopoulos (1988), is a contour model for detecting edges in images. Since Kass *et al.* (1988), the snake model has been referenced in the literature on edge detection many times.

In this chapter, we introduce and discuss the snake model. In Section 5.1 we present a description of the snake energy function and the numerical algorithm used to minimise it. In Section 5.2 the snake model is applied to the ultrasound head image shown in Figure 1.2(a) and Figure 3.1. We also discuss the behaviour of the snake algorithm and then modify the energy function. In Section 5.3 we introduce two algorithms: the dynamic programming algorithm proposed by Amini, Weymouth and Jain (1990) and the balloon model proposed by Cohen (1991) and Cohen and Cohen (1993). These two approaches aim to improve the original snake methodology. In Section 5.4 we present a simulated annealing plus ICM algorithm to minimise the snake energy. In Section 5.5 we discuss the results of a simulation study designed to compare the kernel algorithm introduced in Chapter 3 with the snake algorithm and the simulated annealing plus ICM snake algorithm. Finally, a short summary is given in Section 5.6.

5.1 *The description of the active contour model of Kass, Witkin and Terzopoulos* (1988)

Kass *et al.* (1988) simplify the edge detection problem by reducing it to energy minimisation. The total energy U_S of an active contour S with parametric representation $v(s) = \left(x(s), y(s)\right)$, $s \in [0, 1]$, is defined as the sum of an internal energy and an external energy:

$$U_S = \int_0^1 U\{v(s)\}ds = \int_0^1 \left[U_{\text{int}}\{v(s)\} + U_{\text{ext}}\{v(s)\}\right]ds. \tag{5.1}$$

The internal energy

$$U_{\text{int}}\{v(s)\} = \frac{1}{2}\left\{\alpha(s)|v'(s)|^2 + \beta(s)|v''(s)|^2\right\} \tag{5.2}$$

represents the force that constrains the curve to be smooth, where $v'(s)$ and $v''(s)$ are the first and second derivatives of $v(s)$, and $\alpha(s)$ and $\beta(s)$ are weights.

The external energy comprises two parts

$$U_{\text{ext}}\{v(s)\} = U_{\text{con}}\{v(s)\} + U_{\text{image}}\{v(s)\}, \tag{5.3}$$

where

$$U_{\text{con}}\{v(s)\} = c|P(s) - Q| \tag{5.4}$$

is an external constraint energy that represents the energy of a spring connected between a point $P(s)$ on the snake contour $v(s)$ and some fixed point Q inside or outside the contour, and c is a constant. The constraint energy pulls the contour towards Q.

The image energy

$$U_{\text{image}}\{v(s)\} = w_1 U_{\text{line}}\{v(s)\} + w_2 U_{\text{edge}}\{v(s)\} + w_3 U_{\text{term}}\{v(s)\} \tag{5.5}$$

represents the forces derived from the image which constrain the curve to take the shape of certain features present in the image. The image energy U_{image} is expressed as a weighted combination of three energy functionals that attract the snake to edge features: the line energy $U_{\text{line}} = I(x,y)$ causes the snake to be attracted either by higher or lower grey levels in the image I depending on the sign of the weight w_1; when $w_2 > 0$, the edge energy $U_{\text{edge}} = -|\nabla I(x,y)|^2$ causes the snake to be attracted to pixels at which the image gradient is large; the termination energy U_{term} is the curvature of the level contours in a smoothed version of the image, and thus when $w_3 \geq 0$ causes the contour to be attracted towards line terminations.

The snake approach uses a technique called energy minimisation that causes it to change shape in order to minimise U_S. Kass *et al.* (1988) present an energy minimisation procedure, and the mathematics behind it in detail in the appendix to their paper.

From now on we shall assume that $\alpha(s) \equiv \alpha$ and $\beta(s) \equiv \beta$. From (5.1) and (5.2) we have

$$
\begin{aligned}
U_S &= \int_0^1 U_{\text{int}}\{v(s)\} + U_{\text{ext}}\{v(s)\}\,ds \\
&= \int_0^1 \left[\frac{1}{2}\left\{\alpha|v'(s)|^2 + \beta|v''(s)|^2\right\} + U_{\text{ext}}\{v(s)\}\right]ds \\
&= \int_0^1 \left[\frac{\alpha}{2}\left(x'^2 + y'^2\right) + \frac{\beta}{2}\left(x''^2 + y''^2\right) + U_{\text{ext}}(x,y)\right]ds
\end{aligned}
\tag{5.6}
$$

where

$$
x = x(s),\ y = y(s), x' = \frac{dx(s)}{ds},\ y' = \frac{dy(s)}{ds},\ x'' = \frac{d^2x(s)}{ds^2},\ y'' = \frac{d^2y(s)}{ds^2}.
$$

We now want to minimise the energy function U_S. Let

$$
f(x,y,x',y',x'',y'') = \frac{\alpha}{2}\left(x'^2 + y'^2\right) + \frac{\beta}{2}\left(x''^2 + y''^2\right) + U_{\text{ext}}(x,y), \tag{5.7}
$$

and consider

$$
U_S = \int_0^1 f(x,y,x',y',x'',y'')ds. \tag{5.8}
$$

Let $\bar{x}$ and $\bar{y}$ be the optimising functions and consider a small perturbation

$$
x(s) = \bar{x}(s) + \lambda\eta(s) \tag{5.9}
$$

$$
y(s) = \bar{y}(s) + \delta\rho(s) \tag{5.10}
$$

where $\lambda, \delta \in \Re$. We constrain $\eta(s)$ and $\rho(s)$ so that

$$
\eta(0) = \eta(1) = \eta'(0) = \eta'(1) = 0, \tag{5.11}
$$

$$
\rho(0) = \rho(1) = \rho'(0) = \rho'(1) = 0. \tag{5.12}
$$

Now think of U as a function of λ and δ:

$$
U(\lambda,\delta) = \int_0^1 f(\bar{x} + \lambda\eta, \bar{y} + \delta\rho, \bar{x}' + \lambda\eta', \bar{y}' + \delta\rho', \bar{x}'' + \lambda\eta'', \bar{y}'' + \delta\rho'')ds \tag{5.13}
$$

We want the minimum to occur when $\lambda = \delta = 0$, that is, we require

$$\left(\frac{\partial U}{\partial \lambda}\right)_{\lambda=\delta=0}=\int_0^1\left[\eta\left(\frac{\partial f}{\partial x}\right)_{\lambda=\delta=0}+\eta'\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0}+\eta''\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0}\right]ds=0 \qquad (5.14)$$

and

$$\left(\frac{\partial U}{\partial \delta}\right)_{\lambda=\delta=0}=\int_0^1\left[\rho\left(\frac{\partial f}{\partial y}\right)_{\lambda=\delta=0}+\rho'\left(\frac{\partial f}{\partial y'}\right)_{\lambda=\delta=0}+\rho''\left(\frac{\partial f}{\partial y''}\right)_{\lambda=\delta=0}\right]ds=0$$

Note that the partial derivatives in the above two functions are calculated at the optimising functions $\bar{x}$ and $\bar{y}$.

By condition (5.11) and by integration by parts, we have

$$\int_0^1 \eta'\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0} ds=-\int_0^1 \eta\frac{d}{ds}\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0} ds \qquad (5.15)$$

and

$$\int_0^1 \eta''\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0} ds=\int_0^1 \eta\frac{d^2}{ds^2}\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0} ds \qquad (5.16)$$

Hence, from (5.14),

$$\left(\frac{\partial U}{\partial \lambda}\right)_{\lambda=\delta=0}=\int_0^1\eta\left[\left(\frac{\partial f}{\partial x}\right)_{\lambda=\delta=0}-\frac{d}{ds}\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0}+\frac{d^2}{ds^2}\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0}\right]ds=0 \qquad (5.17)$$

This holds for all $\eta(s)$ such that $\eta(0)=\eta(1)=\eta'(0)=\eta'(1)=0$.

We now prove that

$$J(s)=\left(\frac{\partial f}{\partial x}\right)_{\lambda=\delta=0}-\frac{d}{ds}\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0}+\frac{d^2}{ds^2}\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0}=0$$

Suppose that there is a point ξ such that $J(\xi)>0$. By continuity, $J(s)>0$ must hold for a range of values $[\xi_0,\xi_1]$ say including ξ. Now take the function η to be such that

$$\eta(s) > 0 \quad \text{when } s \in [\xi_0, \xi_1],$$

$$\eta(s) = 0 \quad \text{when } s \notin [\xi_0, \xi_1].$$

This means that

$$\int_0^1 \eta(s) J(s) ds > 0 .$$

This is a contradiction since we know from (5.17) that $\int_0^1 \eta(s) J(s) ds = 0$ holds for any function $\eta(s)$ that satisfies (5.11). Hence there is no point ξ such that $J(\xi) > 0$. A similar argument gives that there is no point ξ such that $J(\xi) < 0$. Hence, we conclude that $J(s) = 0$, that is

$$\left(\frac{\partial f}{\partial x}\right)_{\lambda=\delta=0} - \frac{d}{ds}\left(\frac{\partial f}{\partial x'}\right)_{\lambda=\delta=0} + \frac{d^2}{ds^2}\left(\frac{\partial f}{\partial x''}\right)_{\lambda=\delta=0} = 0 . \tag{5.18}$$

A similar argument with $\left(\frac{\partial U}{\partial \delta}\right)_{\lambda=\delta=0} = 0$ gives

$$\left(\frac{\partial f}{\partial y}\right)_{\lambda=\delta=0} - \frac{d}{ds}\left(\frac{\partial f}{\partial y'}\right)_{\lambda=\delta=0} + \frac{d^2}{ds^2}\left(\frac{\partial f}{\partial y''}\right)_{\lambda=\delta=0} = 0 . \tag{5.19}$$

Accordingly, from (5.7), (5.18) and (5.19), we have

$$\begin{cases} \dfrac{\partial U_{\text{ext}}\{x(s), y(s)\}}{\partial x} - \alpha \dfrac{d^2 x(s)}{ds^2} + \beta \dfrac{d^4 x(s)}{ds^4} = 0 & (5.20) \\[2ex] \dfrac{\partial U_{\text{ext}}\{x(s), y(s)\}}{\partial y} - \alpha \dfrac{d^2 y(s)}{ds^2} + \beta \dfrac{d^4 y(s)}{ds^4} = 0 & (5.21) \end{cases}$$

Equations (5.20) and (5.21) are independent Euler-Lagrange equations; see Synge and Griffith (1959) for a definition of the Euler-Lagrange equation.

Since in practice the contour $v(s)=(x(s),y(s))$ is defined at a finite number of points $v_i=(x_i,y_i), i=1,\ldots,n$, equations (5.20) and (5.21) have to hold at all these points. Now

$$\frac{\partial U_{\text{ext}}(x_i,y_i)}{\partial x}\approx\frac{U_{\text{ext}}(x_i+h,y_i)-U_{\text{ext}}(x_i-h,y_i)}{2h} \tag{5.22}$$

and

$$\frac{\partial U_{\text{ext}}(x_i,y_i)}{\partial y}\approx\frac{U_{\text{ext}}(x_i,y_i+h)-U_{\text{ext}}(x_i,y_i-h)}{2h} \tag{5.23}$$

where $h\to 0$. Let $F_x(i)$ equal the right side of (5.22) and $F_y(i)$ equal the right side of (5.23).

For our application, we consider a closed curve defined by the points $v_i=(x_i,y_i), i=0,\ldots,n$, by linking v_0 and v_n: $v_0=v_n$. For easy of notation let $v_{-1}=v_{n-1}$, $v_{n+1}=v_1$ and $v_{n+2}=v_2$. At the point $(x_{i-1/2},y_{i-1/2})$, that is the midpoint between points (x_{i-1},y_{i-1}) and (x_i,y_i), we may approximate

$$\left.\frac{dx}{ds}\right|_{x_{i-1/2}}\approx\frac{\text{change in } x}{\text{change in } s}=\frac{x_i-x_{i-1}}{1/n}=n(x_i-x_{i-1}).$$

Hence

$$\begin{aligned}\left.\frac{d^2x}{ds^2}\right|_{x_i}&\approx\frac{\text{change in }\dfrac{dx}{ds}}{\text{change in } s}=\frac{\left.\dfrac{dx}{ds}\right|_{x_{i+1/2}}-\left.\dfrac{dx}{ds}\right|_{x_{i-1/2}}}{1/n}\\&\approx n\{n(x_{i+1}-x_i)-n(x_i-x_{i-1})\}\\&=n^2(x_{i+1}-2x_i+x_{i-1}).\end{aligned}$$

Similarly

$$\left.\frac{d^3x}{ds^3}\right|_{x_{i-1/2}} \approx \frac{\text{change in } \frac{d^2x}{ds^2}}{\text{change in } s} = \frac{\left.\frac{d^2x}{ds^2}\right|_{x_i} - \left.\frac{d^2x}{ds^2}\right|_{x_{i-1}}}{1/n}$$

$$\approx n\left\{n^2\left(x_{i+1} - 2x_i + x_{i-1}\right) - n^2\left(x_i - 2x_{i-1} + x_{i-2}\right)\right\}$$

$$= n^3\left(x_{i+1} - 3x_i + 3x_{i-1} - x_{i-2}\right),$$

and finally

$$\left.\frac{d^4x}{ds^4}\right|_{x_i} \approx \frac{\text{change in } \frac{d^3x}{ds^3}}{\text{change in } s} = \frac{\left.\frac{d^3x}{ds^3}\right|_{x_{i+1/2}} - \left.\frac{d^3x}{ds^3}\right|_{x_{i-1/2}}}{1/n}$$

$$\approx n\left\{n^3\left(x_{i+2} - 3x_{i+1} + 3x_i - x_{i-1}\right) - n^3\left(x_{i+1} - 3x_i + 3x_{i-1} - x_{i-2}\right)\right\}$$

$$= n^4\left(x_{i+2} - 4x_{i+1} + 6x_i - 4x_{i-1} + x_{i-2}\right).$$

From (5.22) and the above calculations we obtain

$$F_x(i) - \alpha n^2\left(x_{i+1} - 2x_i + x_{i-1}\right) + \beta n^4\left(x_{i+2} - 4x_{i+1} + 6x_i - 4x_{i-1} + x_{i-2}\right) = 0, \quad i = 1, \ldots, n \tag{5.24}$$

as the discretized version of equation (5.20), and, similarly,

$$F_y(i) - \alpha n^2\left(y_{i+1} - 2y_i + y_{i-1}\right) + \beta n^4\left(y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}\right) = 0, \quad i = 1, \ldots, n \tag{5.25}$$

as the discretized version of equation (5.21). Equations (5.24) and (5.25) can be written in matrix form as

$$\begin{cases} \mathbf{A}x + \mathbf{F}_x(x, y) = \mathbf{0} & (5.26) \\ \mathbf{A}y + \mathbf{F}_y(x, y) = \mathbf{0} & (5.27) \end{cases}$$

where $\mathbf{A}$ is a pentadiagonal banded matrix, x and y are position vectors:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

and $\mathbf{F}_x$ and $\mathbf{F}_y$ denote the vectors of forces at these points $v_i, i = 1, \ldots, n$:

$$\mathbf{F}_x(x,y)=\begin{pmatrix}F_x(1)\\F_x(2)\\\vdots\\F_x(n)\end{pmatrix},\qquad \mathbf{F}_y(x,y)=\begin{pmatrix}F_y(1)\\F_y(2)\\\vdots\\F_y(n)\end{pmatrix}.$$

Equations (5.26) and (5.27) can be solved by an iterative algorithm — Euler's method; see Burden and Faires (1989), pp. 225–232. Let $\left(x^0, y^0\right)$ be an initial contour, then Euler's method proceeds iteratively by letting

$$x^t=(\mathbf{A}+\gamma\mathbf{I})^{-1}\left\{\gamma x^{t-1}-\mathbf{F}_x\left(x^{t-1}, y^{t-1}\right)\right\} \tag{5.28}$$

$$y^t=(\mathbf{A}+\gamma\mathbf{I})^{-1}\left\{\gamma y^{t-1}-\mathbf{F}_y\left(x^{t-1}, y^{t-1}\right)\right\} \tag{5.29}$$

for t = 1, 2, ..., where γ is a constant. As $t\rightarrow\infty$, the contours $\left(x^t, y^t\right)$ tend to a contour that corresponds to a local minimum of U_S. According to our experience, if the image comprises $m_1\times m_2$ pixels the value of γ can be chosen in quite a wide range between $\sqrt{m_1\times m_2}$ and $m_1\times m_2$. We remark that Burden and Faires (1989) work with $h=\frac{1}{\gamma}$ which they refer to as the step size.

5.2 *The behaviour of the snake model*

The snake algorithm that we have just discussed is somewhat different from traditional statistical image analysis methods, such as those presented by Geman and Geman (1984), Besag (1986), and others. The snake model originates from mathematical physics where the forces, energy and method for solving the Euler-Lagrange equation were introduced.

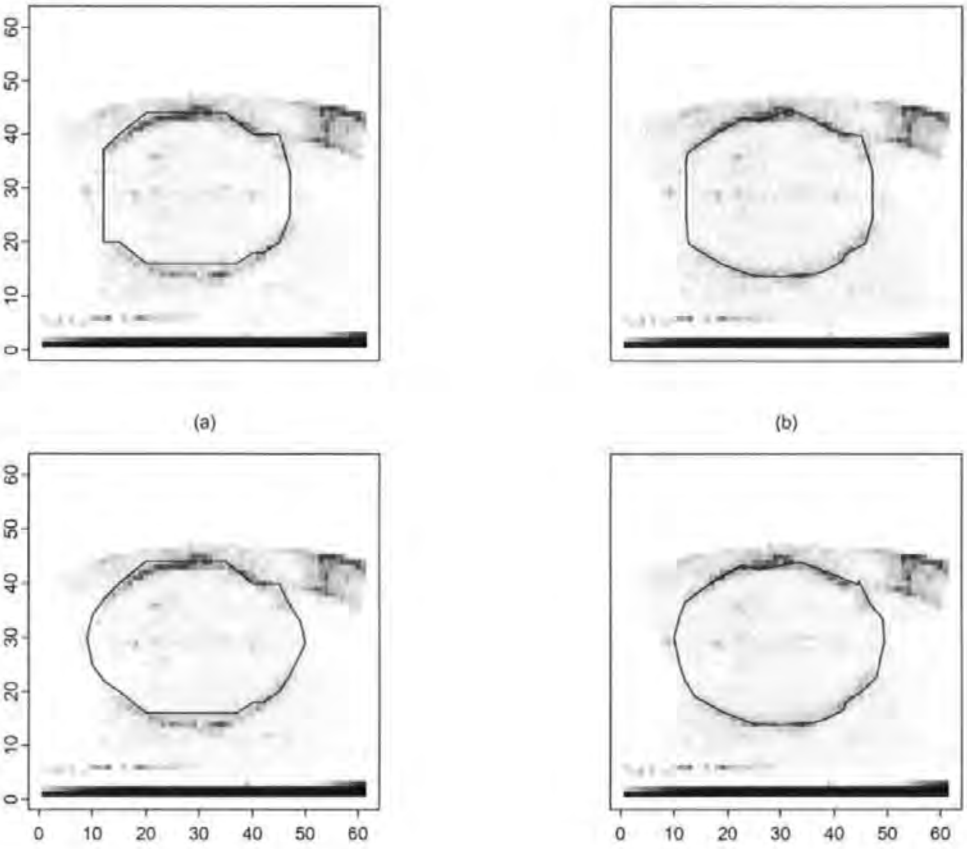
Since the energy function is not concave, the snake algorithm finds a local minimum of the energy function. Finding the global minimum of the energy does not necessarily have a meaning (see Cohen and Cohen, 1993). For example, if (x_m, y_m) is a point of the plane where U_{ext} has a global minimum, the constant curve $v(s) = (x_m, y_m)$ is a global minimum for the energy with periodic boundary conditions.

Kass *et al.* (1988) did not present details of the convergence properties of their snake algorithm. Instead, they stopped iterating when the difference between two successive contours becomes sufficiently small; that is, given a fixed $\varepsilon > 0$, if

$$\sum_{i=1}^{n} \sqrt{\left(x_i{}^{t} - x_i{}^{t-1}\right)^2 + \left(y_i{}^{t} - y_i{}^{t-1}\right)^2} < \varepsilon \tag{5.30}$$

then stop iterating and define the final contour (x, y) to be (x^t, y^t).

For the ultrasound fetal head image shown in Figures 3.1 and 1.2(a), we detected the head shape well using the kernel algorithm described in Chapter 3. We felt that the result obtained was especially good since part of the head outline had almost been completely obliterated. Figure 5.1 illustrates the results that we obtained by applying the snake algorithm of Kass *et al.* (1988) to this ultrasound image. Note that in this chapter the grey levels of this ultrasound image are standardized. The weights $\alpha(s)$ and $\beta(s)$ of the derivatives in (5.2) were taken to be constant. In the external energy (5.3), we used only the image energy term, and in the image energy term (5.5) we did not need the termination term U_{term} because our problem is to find a closed shape without line terminations.



(a) (b) (c) (d)

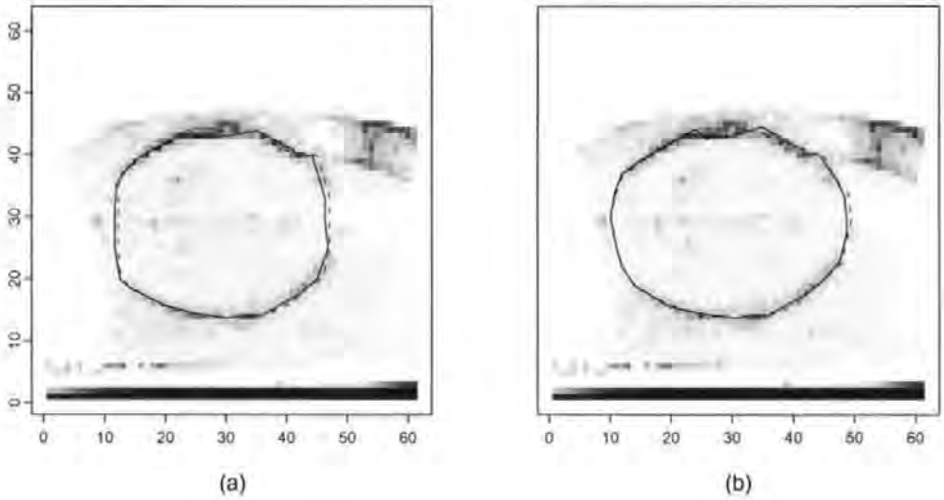
Figure 5.1 The results obtained by applying the snake algorithm to the ultrasound image of a fetal head. (a) An initial contour. This provides a poor fit to the head shape at its left and right sides. (b) The final contour obtained from the initial contour (a). (c) Another initial contour. This is much closer to the shape of a head. (d) The final contour obtained from the initial contour (c).

Figure 5.1(a) presents an initial contour for the snake algorithm. This contour was placed by hand on the points where the outline is obvious. We supposed that we did not have knowledge about the true shape so that we just gave straight lines for the left and right sides where the outline has almost been obliterated. We experimented with various parameters in the snake model defined by (5.1) to (5.5), and found that the best result was obtained by using $\alpha = 1 \times 10^{-6}$, $\beta = 0$, $w_1 = -1$, $w_2 = 2$ and $\gamma = 100$. This result is shown in Figure 5.1(b). The algorithm was terminated when the stopping condition (5.30) was met with $\varepsilon = 0.01$. We see that the snake algorithm performs well but the left and right sides of the head shape are not recovered. The reason for this is that the initial contour is not close enough to the true edge of the head shape with the result that the contour is not attracted by it.

Because we really do have knowledge about the head shape, we can pick an initial contour that fits better at the left and right sides of the image in the hope that this contour will lead to a better fit in these regions. The contour shown in Figure 5.1(c) is the initial contour that is thought to be closer to the edge of the head shape than the previous initial contour. The result of applying the snake algorithm using the same parameter values as before is shown in panel (d) of Figure 5.1. We see that the contour almost remains at its starting position at the left and right sides due to the lack of edge information in these two parts. Overall the result seems very good.

The original energy term in (5.5) is defined as $U_{\text{edge}} = -\left|\nabla I(x, y)\right|^2$ and is designed so that the snake is attracted to pixels at which the image gradient is large. However, in noisy images there may be many pixels at which the gradient is large but which do not correspond to real edges. In order to overcome this problem, we may use a smoothed version of the image in the definition of U_{edge}. A similar idea was applied by Canny (1986) in the Canny filter, which is really the Prewitt filter applied to a smoothed version of the image; see Glasbey and Horgan (1995). Let I_s be the smoothed version of the image I. We now define the edge energy to be $U_{\text{edge}} = -\left|\nabla I_s(x, y)\right|^2$.

We now re-implement the snake algorithm using this definition of the edge energy in (5.5) with I_S equal to the median smoothed version of the fetal head image. The median smoothing operator replaces the grey level at each pixel with the median of the grey levels at pixels in a $(2m+1)\times(2m+1)$ centred on the original pixel, where m is a positive integer. Glasbey and Horgan (1995) presents details of the median smoothing operator, and other smoothing operators. The choice of m depends on the size of image; taking $m=1$ is appropriate for our requirement of reducing the effect of noise on the gradient, whilst not losing too much edge information. Figure 5.2(a) and (b) present the results obtained starting from the initial contour shown in Figure 5.1(a) and (c), respectively. For the purpose of comparison, the results shown in Figure 5.1(b) and (d) are also presented in Figure 5.2 by the dashed line.



(a)

(b)

Figure 5.2 Results obtained by calculating the edge energy using the smoothed version of the head image. (a) Starting from the initial contour shown in Figure 5.1(a). (b) Starting from the initial contour shown in Figure 5.1(c). The dashed line corresponds to the contours shown in panels (b) and (d) of Figure 5.1.

The results obtained using the two different definitions of U_{edge} are very similar. It does seem, however, that a slightly smoother contour is obtained when U_{edge} is based on the median smoothed image.

From Figure 5.2 we may draw the same conclusion as before, that the result obtained from a good initial contour is better than that obtained from a less good initial contour. From these and other experiments we believe that the choice of the initial contour is very important in the snake algorithm as proposed by Kass *et al.* (1988). In order to obtain a good final contour, the initial contour should be as close as possible to the shape, or to what we believe that the shape should be.

In Kass *et al.* (1988) there is no discussion about the choice of the snake parameters. In our experience, a good choice of parameters is essential for a good final contour. High values of β cause the contour to shrink rapidly to a point. High values of α cause the contour to become a very elliptic shape no matter how the initial contour is defined. Generally, if there is a lack of edge information, the shape will shrink on itself.

5.3 *Discussion about snake models*

Since Kass *et al.* (1988), many different implementations of snake models or similar active contour models have been proposed in an attempt to solve some of the shortcomings of the original snake model or to provide better models. In this section, we introduce two approaches: the dynamic programming active contour model and the balloon model.

5.3.1 *Dynamic programming active contour model*

Suppose there are n points on the contour. If each of these points is allowed to take only M possible positions in the plane corresponding to the centre of pixels, then the number of possible contours is finite. In this situation, we could find the contour that corresponds to the global minimum of the energy function by exhaustive enumeration. However, this is impossible in practice because of the huge associated computation cost. A practical method for finding the minimising contour based on dynamic programming is given by Amini, Weymouth and Jain (1990). These authors also discuss how dynamic programming additionally allows hard constraints to be enforced on the behaviour of the solution.

The following is a brief description of the dynamic programming approach presented by Amini *et al.* (1990).

First, the derivatives in (5.2) are discretized to obtain

$$U_{\text{int}}(v_i) = \frac{1}{2}\left(\alpha n^2 |v_i - v_{i-1}|^2 + \beta n^4 |v_{i+1} - 2v_i + v_{i-1}|^2\right), \qquad i = 1,\dots,n.$$

This enables the integral in (5.1) to be discretized as

$$U_S \equiv U_S(v_1, v_2, \ldots, v_n) = \sum_{i=1}^{n} \{U_{\text{int}}(v_i) + U_{\text{ext}}(v_i)\}$$

$$= \sum_{i=1}^{n} \left\{ \frac{1}{2}\left(\alpha n^2 |v_i - v_{i-1}|^2 + \beta n^4 |v_{i+1} - 2v_i + v_{i-1}|^2 \right) + U_{\text{ext}}(v_i) \right\}.$$

If we let

$$U_i(v_{i-1}, v_i, v_{i+1}) = \frac{1}{2}\left(\alpha n^2 |v_i - v_{i-1}|^2 + \beta n^4 |v_{i+1} - 2v_i + v_{i-1}|^2 \right) + U_{\text{ext}}(v_i),$$

then the problem becomes minimising a function of the form

$$U_S(v_1, v_2, \ldots, v_n) = \sum_{i=1}^{n} U_i(v_{i-1}, v_i, v_{i+1}). \qquad (5.31)$$

Unfortunately, dynamic programming cannot be applied directly to this minimisation problem because of the restrictions imposed by having a closed curve, namely $v_0 = v_n$ and $v_{n+1} = v_1$. To overcome this Amini *et al.* (1990) slightly modify the above minimisation to that of minimising a function of the form

$$U(v_1, v_2, \ldots, v_n) = \sum_{i=2}^{n-1} U_i(v_{i-1}, v_i, v_{i+1}). \qquad (5.32)$$

This is an unrestricted minimisation over $(v_1, v_2, \ldots, v_n)$ and so standard dynamic programming can be applied. A closed curve is obtained by joining v_1 to v_n.

Therefore what we want is to find $(v_1', v_2', \ldots, v_n')$ such that

$$(v_1', v_2', \ldots, v_n') = \arg\min_{v_1, v_2, \ldots, v_n} U(v_1, v_2, \ldots, v_n).$$

Throughout we exclude self-intersecting contours.

To solve this minimisation problem, a sequence of functions of two variables is generated:

$$s_1(v_2, v_3) = \min_{v_1} U_2(v_1, v_2, v_3) \tag{5.33}$$

$$s_2(v_3, v_4) = \min_{v_2} \left[s_1(v_2, v_3) + U_3(v_2, v_3, v_4) \right] \tag{5.34}$$

....

$$s_i(v_{i+1}, v_{i+2}) = \min_{v_i} \left[s_{i-1}(v_i, v_{i+1}) + U_{i+1}(v_i, v_{i+1}, v_{i+2}) \right] \tag{5.35}$$

....

$$s_{n-2}(v_{n-1}, v_n) = \min_{v_{n-2}} \left[s_{n-3}(v_{n-2}, v_{n-1}) + U_{n-1}(v_{n-2}, v_{n-1}, v_n) \right], \tag{5.36}$$

where each point v_i can take on M possible values. At stage i, for each of the $M \times M$ possible pairs of (v_{i+1}, v_{i+2}), find the value of the point v_i that minimises

$$s_{i-1}(v_i, v_{i+1}) + U_{i+1}(v_i, v_{i+1}, v_{i+2})$$

and record $s_i(v_{i+1}, v_{i+2})$. The minimum value of the energy function can now be found to be

$$\min_{v_1, v_2, \ldots, v_n} U(v_1, v_2, \ldots, v_n) = \min_{v_{n-1}, v_n} s_{n-2}(v_{n-1}, v_n).$$

The minimising point $(v_1', v_2', \ldots, v_n') = \arg\min_{v_1, v_2, \ldots, v_n} U(v_1, v_2, \ldots, v_n)$ can be found by working backwards. First find (v_{n-1}', v_n') such that

$$(v_{n-1}', v_n') = \arg\min_{v_{n-1}, v_n} s_{n-2}(v_{n-1}, v_n).$$

Then from $s_{n-2}(v_{n-1}', v_n')$ find v_{n-2}' that makes (5.36) hold, and so on. Finally, from $s_1(v_2', v_3')$ find v_1' that makes (5.33) hold.

The above procedure leads to a global minimum of U. However, in practice M is very large with the result that the above minimisation procedure is not computationally feasible.

Accordingly, Amini *et al.* (1990) restrict their minimisations so that v_i, for example, takes positions in the set comprising the current position of v_i and its eight first and second order neighbouring pixels. Clearly, this restriction will lead to a local minimum of U since the complete space is not explored. Because of this the whole procedure outlined above is repeated a number of times until the value of U no longer falls. Each repetition is referred to as an iteration by Amini *et al.* (1990). Convergence is guaranteed since the total energy of the contour is reduced (or remains the same) at each iteration.

Although the above dynamic programming approach ensures the convergence of the energy minimisation process, it requires large storage requirements. For example, for the case when m possible choices are allowed for each point v_i, then the time complexity for each iteration of the algorithm is $O(nm^3)$, and the storage requirement is $O(nm^2)$ memory elements.

5.3.2 *Balloon model*

Cohen (1991) and Cohen and Cohen (1993) present an improved active contour model that they call the balloon model. In the snake model, $F(v) = -(\mathbf{F}_x(x,y), \mathbf{F}_y(x,y))^{\mathrm{T}}$ from the Euler-Lagrange equations (5.26) and (5.27) can be thought of as an external force applied to the curve. Let $\nabla P(v) = (\mathbf{F}_x(x,y), \mathbf{F}_y(x,y))^{\mathrm{T}}$. In order to find "good" edge points and to ensure a connected contour, Cohen (1991) introduced an external constraint force that inflates or deflates the snake. The new force defined by Cohen (1991) is

$$F(v) = k_1 \mathbf{n}(s) - k \frac{\nabla P}{\|\nabla P\|}$$

where $\mathbf{n}(s)$ is the balloon force, that is the normal unitary vector to the curve at point $v(s)$, and k_1 is the amplitude of this added force. The parameters k_1 and k are chosen to be of the same order, which is smaller than the size of a pixel, and k is slightly larger than k_1 so an edge point can stop the inflation force. If the sign of k_1 or the orientation of the curve is changed, the first term of F will deflate the contour instead of inflating it, or vice versa.

We believe that the dynamic programming algorithm and the balloon model can improve the original snake model. It would be of interest to compare the behaviour of the dynamic programming algorithm or the balloon model with that of the original snake model in our case, although we have not done this.

In the next section, we will use simulated annealing and ICM to minimise the energy function employed by the snake model. We will compare the results obtained with those presented in Section 5.2.

5.4 *Simulated annealing and ICM applied to the snake model*

We now consider the use of simulated annealing and ICM for minimising the energy function employed by the snake model. If we use simulated annealing and ICM to minimise the energy, we do not need to solve the Euler-Lagrange equations (5.26) and (5.27). Simulated annealing allows jumps out of local minima of the energy function to be made whereas under ICM (applied after simulated annealing) the energy only decreases; see Chapter 1 and Chapter 2 for more details about simulated annealing and ICM methods.

In our application of simulated annealing, the logarithmic temperature schedule

$$\tau_t = \frac{lf\{\log(T+1)-\log 2\}}{\{l\log(T+1)-f\log 2\}+(f-l)\log(t+1)}$$

was employed with temperature f for the first update, temperature l for the last update, and the finite total number of iterations T. See Section 1.3.4 for more details of temperature schedules in simulated annealing as discussed by Stander and Silverman (1994).

At each iteration of simulated annealing, one point chosen in turn from the contour points is allowed to change its position to a number of candidate points or remain in its current position. The candidate points for a selected point are those first and second order neighbours to which a move may be made without destroying the nature of the shape. For example, if a move to a neighbouring point results in the contour intersecting itself, the neighbouring point is not counted as a candidate point.

Suppose there are n_b candidate points for the selected point b. Let S be the current contour and let the candidate contours obtained by moving b to one of the associated candidate points be $S_1, S_2, \ldots, S_{n_b}$. Simulated annealing samples one of the contours $S_1, S_2, \ldots, S_{n_b}, S$ with probabilities proportional to $p_{\tau_t}(S_1), p_{\tau_t}(S_2), \ldots, p_{\tau_t}(S_{n_b}), p_{\tau_t}(S)$, where $p_{\tau_t}(S)$ is the distribution of contour S with temperature τ_t at iteration t:

$$p_{\tau_t}(S) \propto \exp\left(-\frac{U_S}{\tau_t}\right).$$

The sampled contour is then set to be the current contour S. Note that p_{τ_t} is defined up to a constant of proportionality known as a normalising constant, which, because of the size and complexity of the set of all contours, is unknown in practice. Fortunately, we do not need to know the normalising constant for sampling because it cancels out in the probabilities proportional to $p_{\tau_t}(S_1), p_{\tau_t}(S_2), \ldots, p_{\tau_t}(S_{n_b}), p_{\tau_t}(S)$.

A sweep comprises visiting all the contour points in turn. If the number of sweeps that have been performed exceeds a given number, we stop the simulated annealing algorithm, and apply ICM.

ICM is an algorithm that does not increase the energy. Instead of sampling one contour from $S_1, S_2, \ldots, S_{n_b}, S$, the ICM algorithm chooses the new contour to be the one that has the lowest energy. We now briefly describe the details of our implementation of ICM. Suppose there are n_b candidate points for the selected point b and S is the current contour. Let the candidate contours obtained by moving b to one of its candidate points be $S_1, S_2, \ldots, S_{n_b}$. Calculate the energies U_{S_1}, U_{S_2}, ..., $U_{S_{n_b}}$, U_S and choose the contour with the lowest energy as the new current S. If there is no change between two successive sweeps, ICM has converged. Because ICM never increases the energy, it is guaranteed to converge to a local minimum of the energy in a finite number of sweeps.

We now use simulated annealing and ICM to minimise the energy function we used in Section 5.2. The edge energy function is calculated using the median smoothed version of the image. For simulated annealing we set $T = 900$, $f = 2$ and $l = 0.1$.

In Figure 5.3 we present the results obtained by applying the simulated annealing plus ICM algorithm to the head image using the initial contours shown in Figure 5.1 (a) and (c), respectively. In Figure 5.3 the dashed line represents the result obtained from the snake algorithm as shown in Figure 5.2. The final contour in panel (a) is much better than the one shown in Figure 5.2(a), especially on the left and right sides, whereas the final contour in panel (b) is almost same as the one shown in Figure 5.2(b) although a little rougher.

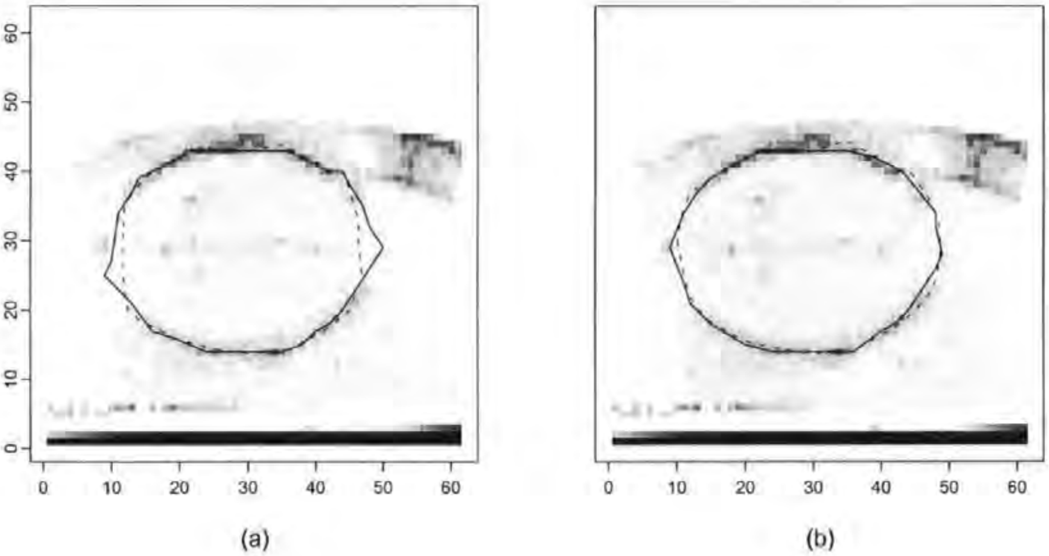


Figure 5.3 Results from the simulated annealing plus ICM algorithm. (a) Starting from the initial contour shown in Figure 5.1(a). (b) Starting from the initial contour shown in Figure 5.1(c). The outlines presented by the dashed line are the results obtained from the snake algorithm as shown in Figure 5.2.

We also performed the simulated annealing plus ICM algorithm with $U_{\text{edge}} = -|\nabla I(x,y)|^2$, where I is the original unsmoothed image. The results appear slightly rougher than those obtained by with $U_{\text{edge}} = -|\nabla I_S(x,y)|^2$.

It is of interest to compare the energy of the final contour obtained by the algorithm suggested by Kass *et al.* (1988) with the energy of the final contour obtained by simulated annealing plus ICM. Table 5.1 gives the initial and final energies based on the results that have led to Figures 5.2 and 5.3 for both algorithms and both initial contours.

Table 5.1 Final energies for the algorithm due to Kass *et al.* (1988) and the simulated annealing plus ICM algorithm based on the results that led to Figures 5.2 and 5.3

Algorithm	Initial contour (energy) Figure 5.1(a) (-0.6840)	 Figure 5.1(c) (-2.2482)
Kass *et al.* (1988)	-38.0290	-36.9765
Simulated annealing + ICM	-97.8373	-98.1155

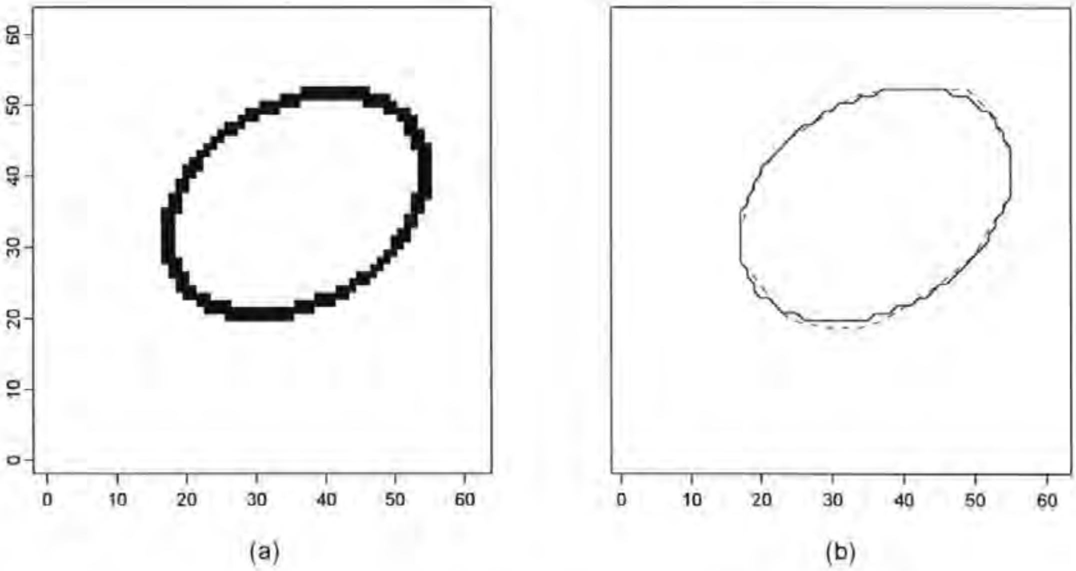
From Figure 5.3(a) and Table 5.1, we see that the algorithm due to Kass *et al.* (1988) applied to the initial contour presented in Figure 5.1(a) led to a worse result with higher final energy than simulated annealing plus ICM. From Figure 5.3(b) and Table 5.1, we see that the algorithm due to Kass *et al.* (1988) applied to the initial contour presented in Figure 5.1(c) led to a smoother result but with higher final energy than simulated annealing plus ICM. We have already said that the initial contour presented in Figure 5.1(a) fits badly at its left and right sides, while the initial contour presented in Figure 5.1(c) provides a better fit.

For both initial contours the simulated annealing plus ICM algorithm produces a final contour with energy lower than that of the final contour produced by the algorithm due to Kass *et al.* (1988). The simulated annealing plus ICM algorithm is able to make larger changes to the initial contour shown in Figure 5.1(a) than the algorithm due to Kass *et al.* (1988), leading to a possibly better final contour.

5.5 *Simulation study: comparison between the kernel algorithm, the snake algorithm and the simulated annealing plus ICM algorithm*

The most important difference between the snake model and the kernel method defined in Chapter 3 is that a closed curve always results from the former algorithm, but cannot be guaranteed when applying the latter algorithm if the shape is defined by a thin edge only. In order to compare the algorithms in a simulation study, we consider a shape which is defined by an edge that is two pixels thick. The original image is shown in Figure 5.4(a). It is almost the same as the image shown in Figure 3.6(a) except that the edge is a little thicker.

For our simulation study we generated noisy images by adding independent Gaussian noise $N(0,\kappa)$ to the true image shown in Figure 5.4(a). For the kernel algorithm, the starting point was selected randomly from the outer true edge and we set $\sigma_0 = 1$, $\sigma_1 = 4$ and $\sigma_2 = 20$ according to experience and the study into the effect of the kernel parameters presented in Chapter 4. For the snake algorithm and the simulated annealing plus ICM algorithm, the initial contour was chosen to be very close to the outline of the true shape in order to reduce the effect of high noise level and to avoid meaningless shapes resulting. This initial contour is shown by dashed line in Figure 5.4(b). The parameters for the snake algorithm were set to be $\alpha = 10^{-6}$, $\beta = 10^{-5}$, $w_1 = -1$, $w_2 = 2$, $w_3 = 0$ and $\gamma = 100$ by experience. The parameters for the simulated annealing plus ICM algorithm were $\alpha = 0.1$, $\beta = 0.1$, $w_1 = -1$, $w_2 = 2$, $w_3 = 0$, $f = 2$, $l = 0.1$, $T = 600$, and a logarithmic temperature schedule was employed. For both the original snake and the simulated annealing plus ICM algorithms, α and β were selected in order to optimise the results. The reason for the difference in parameters is connected to the poor performance of the original snake algorithm as far as minimising the energy U_S is concerned. For both the snake and the simulated annealing plus ICM algorithms, the edge energy function is calculated using the median smoothed version of the image.



(a) (b)

Figure 5.4 (a) A shape defined by an edge that is two pixels thick. (b) The initial contour (dashed line) and the outline (solid line) of the true shape shown in (a).

For the purpose of comparison, all three algorithms were applied to each noisy image. We simulated 10 noisy images for four noise levels $\kappa = 0.25, 0.5, 1.0 \text{ and } 2.0$. The mean and standard deviation of the number of pixels that differ between the true shape and the estimated shape are presented in Table 5.2 for the three algorithms and for the four different noise levels. When the kernel algorithm was employed, the algorithm failed to produce a closed curve on one occasion when $\kappa = 1$, and on two occasions when $\kappa = 2$.

Table 5.2 Mean (standard deviation) of the number of pixels that differ between the true shape and the estimated shape obtained by the kernel, snake and simulated annealing (SA) plus ICM algorithms

Variance κ	Size of sample	Algorithm		
		Kernel	Snake	SA+ICM
0.25	10	25.2 (3.61)	25.3 (8.00)	30.1 (6.10)
0.50	10	33.6 (4.38)	36.9 (10.68)	35.6 (7.04)
1.00	9	37.0 (8.03)	42.9 (7.30)	37.2 (3.35)
2.00	8	36.6 (9.49)	64.0 (6.48)	35.6 (6.55)

From Table 5.2, it seems that there is no real difference among the three algorithms when κ is small, but for higher values of κ the results of the snake algorithm seem worse than those of the other two algorithms. In order to test if there is a significant difference between any two algorithms, a paired *t*-test was employed. The means of the difference between the algorithms are presented in Table 5.3, and the symbol * is used to indicate that there is a statistically significant difference ($p < 0.05$) between the two algorithms considered. When $\kappa = 0.25$, the performance of the simulated annealing plus ICM algorithm is significantly worse than that of the kernel algorithm ($p = 0.0069$, one-tailed test). When $\kappa = 1.0$, the performance of the snake algorithm is significantly worse than that of the simulated annealing plus ICM algorithm ($p = 0.0293$, one-tailed test). When $\kappa = 2.0$, the performance of the snake algorithm is significantly worse than that of the kernel algorithm and the simulated annealing plus ICM algorithm ($p < 0.00005$, one-tailed test).

Table 5.3 Mean (standard deviation) of difference. The symbol * is used to indicate that there is a statistically significant difference ($p < 0.05$) between the two algorithms considered

Variance κ	Size of sample	Between algorithms		
		Snake–Kernel	(SA+ICM)–Kernel	(SA+ICM)–Snake
0.25	10	0.1 (7.64)	4.9 (5.09)*	4.8 (9.92)
0.50	10	3.3 (10.36)	2.0 (7.67)	–1.3 (8.81)
1.00	9	5.9 (11.42)	0.2 (8.64)	–5.7 (7.71)*
2.00	8	27.4 (7.73)*	–1.0 (8.05)	–28.4 (3.85)*

In the above implementation, the initial contour is very close to the outline of the true shape. If the initial contour is far away from the true shape, a worse result is obtained from the snake algorithm. This is particularly the case for high value of κ.

We finish by remarking that even when the edge of the true shape is two pixels thick, the kernel algorithm failed to produce a closed curve on three occasions. However, for these cases the value of κ was relatively high, $\kappa = 1$ and $\kappa = 2$.

5.6 *Conclusions*

Our general conclusions from the above analyses are

- A smoother contour usually results when U_{edge} in the snake model is calculated using the smoothed version of image.
- Lower energy does not necessarily mean a better estimate.
- In order to obtain an accurate estimate using the algorithm due to Kass *et al.* (1988), the initial contour must be very well chosen.
- If we lack edge information but have a good initial contour, then the algorithm due to Kass *et al.* (1988) may give a final contour that is better than that obtained by the simulated annealing plus ICM algorithm. On the other hand, if we lack edge information and we are unable to supply a good initial contour, the simulated annealing plus ICM algorithm will leads to a result that generally has lower energy than that obtained by the algorithm due to Kass *et al.* (1988).
- The kernel algorithm introduced in Chapter 3 usually performs well provided it produces a closed curve. However, for higher noise levels and thinner edges, there is a risk that the algorithm will fail to produce a closed curve.

We feel that it would also be interesting to compare the behaviour of the dynamic programming algorithm with that of the simulated annealing plus ICM algorithm. We would also like to work with the balloon models. We hope to perform this research in the near future.

Conclusions and Suggestions for Further Work

In this thesis we have introduced, developed and discussed statistical image analysis techniques which can be applied to data from medical ultrasound images. We have worked with two types of image: in the first a degraded outline of the shape is visible, whilst in the second the data are a corrupted version of the shape itself. In Chapter 2 we considered the algorithm proposed by Storvik (1994) for images of the second type and concluded by means of simulation studies that the algorithm can be speeded up and its performance improved by using a cascade based simulated annealing approach. The kernel function algorithm developed and discussed in Chapter 3 and Chapter 4 allows the outline of shape in images of the first type to be recovered when parts of it are almost completely obliterated by noise. Other algorithms often do not allow such outlines to be identified and tend to be more complicated. In Chapter 5 we showed that the 'snake' methodology applied to images of the first type is very sensitive to the initial estimate of the shape to be detected if the original fitting algorithm is used, but that this sensitivity is less marked if simulated annealing followed by ICM is employed.

There are several aspects of the work in this thesis which could be further developed. We now know that it is possible to run the kernel algorithm and the modified kernel algorithm much more quickly by using the same values of the kernel function or the detection function at all candidate points each time we attempt to identify a new edge pixel. We also hope to re-code all the programs in C++. This would provide us with a better idea about the possibilities for real time detection, an issue which is of interest to clinicians.

Other suggestions for further work include developing the kernel algorithm by using the bent kernel function; see Section 3.9 for more details. In fact the kernel algorithm does not work well on highly degraded images of the first type with a thin edge, and the modified kernel algorithm performs badly on the duck image of Pievatolo and Green (1998) presented in Figure 3.26. Further simulation studies aimed at investigating the interaction of the kernel parameters σ_0, σ_1 and σ_2, by fixing one parameter and allowing the other two parameters to change together, should be performed. We also plan to adapt the snake methodology so that it can be applied successfully to images of the second type. So far we have detected shapes in these images by means of our modified version of Storvik's algorithm and the modified kernel algorithm.

References

Amini, A. A., Weymouth, T. E. and Jain, R. C. (1990) Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-12**, 855–867.

Anderson, E. M. and Spain, B. (1977) *The Child with Spina Bifida.* Methuen, London.

Besag, J. E. (1986) On the statistical analysis of dirty pictures (with discussion). *Journal of the Royal Statistical Society* B, **48**, 259–302.

Besag, J. E. (1989) Digital image processing towards Bayesian image analysis. *Journal of Applied Statistics*, **16**, 395–407.

Besag, J. E. and Green, P. J. (1993) Spatial statistics and Bayesian computation. *Journal of the Royal Statistical Society* B, **55**, 25–38.

Besag, J. E., Green, P. J., Higdon, D. and Mengersen, K. L. (1995) Bayesian computation and stochastic systems. *Statistical Science*, **10**, 3–66.

Bowtell, P. and Patefield, M. (1997) The circular functional relationship. Research Report, Department of Applied Statistics, The University of Reading.

Brock, D. J. H. and Sutcliffe, R. G. (1972) Alpha-fetoprotein in the antenatal diagnosis of anencephaly and spina bifida. *Lancet*, **2**, 197–199.

Brooks, S. P. (1998) Markov chain Monte Carlo method and its application. *The Statistician*, **47**, 69–100.

Burden, R. L. and Faires, J. D. (1989) *Numerical Analysis* (Fourth edition). PWS-KENT, Boston.

Canny, J. (1986) A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-8**, 679–698.

Chalana, V., Linker, D. T., Haynor, D. R. and Kim, Y. M. (1996) A multiple active contour model for cardiac boundary detection on echocardiographic sequences. *IEEE Transactions on Medical Imaging*, **MI-15**, 290–298.

Chalana, V., Winter III, T. C., Cyr, D. R., Haynor, D. R. and Kim, Y. M. (1996) Automatic fetal head measurements from sonographic images. *Academic Radiology*, **3**, 628–635.

Cohen, L. D. (1991) Note on active contour models and balloons. *Computer Vision, Graphics and Image Processing: Image Understanding*, **53**, 211–218.

Cohen, L. D. and Cohen, I. (1993) Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-15**, 1131–1147.

Cowles, M. K. and Carlin, B. P. (1996) Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, **91**, 883–904.

Friedland, N. and Adam, D. (1989) Automatic ventricular cavity boundary detection from sequential ultrasound images using simulated annealing. *IEEE Transactions on Medical Imaging*, **MI-8**, 344–353.

Friedland, N. S. and Rosenfeld, A. (1992) Compact object recognition using energy-function-based optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-14**, 770–777.

Frigessi, A., Martinelli, F. and Stander, J. (1997) Computational complexity of Markov chain Monte Carlo methods for finite Markov random fields. *Biometrika*, **84**, 1–18.

Gelman, A. and Rubin, D. B. (1992) Inference from iterative simulation using multiple sequences. *Statistical Science*, **7**, 457–472.

Geman, D. and Jedynak, B. (1996) An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-18**, 1–14.

Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**, 721–741.

Geman, S. and McClure, D. E. (1987) Statistical methods for tomographic image reconstruction. In: *Bulletin of the International Statistical Institute*, **52**(Book 4), 5–21.

Geyer, C. J. (1992) Practical Markov chain Monte Carlo. *Statistical Science*, **7**, 473–511.

Gibbs, A. (1998) Bounding convergence time of the Gibbs sampler in Bayesian image restoration. Research report, Department of Statistics, University of Toronto, Canada.

Gidas, B. (1989) A renormalisation group approach to image processing problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-11**, 164–180.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1996) *Markov Chain Monte Carlo in Practice*. Chapman & Hall, London.

Glasbey, C. A. and Horgan, G. W. (1995) *Image Analysis for the Biological Sciences*. Wiley, Chichester.

Green, P. J. (1994) Contribution to the discussion of 'representations of knowledge in complex systems' (By Grenander, U. and Miller, M. I.). *Journal of the Royal Statistical Society* B, **56**, 589–590.

Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732.

Grenander, U., Chow, Y. and Keenan, D. M. (1990) *Hands: A Pattern Theoretic Study of Biological Shapes*. Spring-Verlag, New York.

Grenander, U. and Miller, M. I. (1994) Representations of knowledge in complex systems (with discussion). *Journal of the Royal Statistical Society* B, **56**, 549–603.

Hajek, B. (1988) Cooling schedules for optimal annealing. *Mathematics of Operation Research*, **13**, 311–329.

Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109.

Heikkinen, J. and Högmander, H. (1994) Fully Bayesian approach to image restoration with an application in biogeography. *Applied Statistics*, **43**, 569–582.

Helterbrand, J. D., Cressie, N. and Davidson, J. L. (1994) A statistical approach to identifying closed object boundaries in images. *Advances in Applied Probability*, **26**, 831–854.

Higdon, D. M., Bowsher, J. E., Johnson, V. E., Turkington, T. G., Gilland, D. R. and Jaszczak, R. J. (1997) Fully Bayesian estimation of Gibbs hyperparameters for emission computed tomography data. *IEEE Transactions on Medical Imaging*, **MI-16**, 516–526.

Hurn, M. A. and Jennison, C. (1995) A study of simulated annealing and a revised cascade algorithm for image-reconstruction. *Statistics and Computing*, **5**, 175–190.

Hurn, M. A. and Rue, H. (1997) High-level image priors in confocal microscopy applications. In: *Proceedings in the Art and Science of Bayesian Image Analysis* (eds Mardia, K. V., Gill, C. A. and Aykroyd, R. G.), pp. 36–43. Leeds University Press.

Illingworth, J. and Kittler, J. (1988) A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, **44**, 87–116.

Jubb, M. and Jennison, C. (1991) Aggregation and refinement in binary image restoration. In: *Spatial Statistics and Imaging* (ed. Possolo, A.), pp. 150–162. Institute of Mathematical Sciences Lecture Notes, Hayward, CA.

Karaman, M., Kutay, M. A. and Bozdagi, G. (1995) An adaptive speckle suppression filter for medical ultrasonic imaging. *IEEE Transactions on Medical Imaging*, **MI-14**, 283–292.

Kass, M., Witkin, A. and Terzopoulos, D. (1988) Snakes: active contour models. *International Journal of Computer Vision*, **1**, 321–331.

Kirkpatrick, S., Gellat, C. and Vecchi, M. (1983) Optimization by simulated annealing. *Science*, **22**, 671–680.

Laarhoven, P. J. M. and Aarts, E. H. L. (1987) *Simulated Annealing: Theory and Applications*. D. Reidel, Dordrecht.

van Lieshout, M. N. M. (1995) *Stochastic Geometry Models in Image Analysis and Spatial Statistics*. CWI Tract, Amsterdam.

Luan, J., Stander, J. and Wright, D. (1998) On shape detection in noisy images with particular reference to ultrasonography. Accepted for publication in *Statistics and Computing*.

Mardia, K. V. and Qian, W. (1995) Bayesian method for compact object recognition from noisy images. In: *Complex Stochastic System and Engineering* (ed. Titterington, D. M.), pp. 155–166. Clarendon Press, Oxford.

Matsopoulos, G. K. and Marshall, S. (1994) Use of morphological image processing techniques for the measurement of a fetal head from ultrasound images. *Pattern Recognition*, **27**, 1317–1324.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953) Equations of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**, 1087–1092.

Nicolaides, K. H., Campbell, S., Gabbe, S. G. and Guidetti, R. (1986) Ultrasound screening for spina bifida: cranial and cerebellar signs. *Lancet*, **2**, 72–74.

Pievatolo, A. and Green, P. J. (1998) Boundary detection through dynamic polygons. *Journal of the Royal Statistical Society* B, **60**, 609–626.

Pursey, G. J. and Taylor, P. C. (1995) Route tracing: a new method for edge detection. Technical Report 95/5, Department of Applied Statistics, The University of Reading.

Qian, W., Titterington, D. M. and Chapman, J. N. (1996) An image analysis problem in electron microscopy. *Journal of the American Statistical Association*, **91**, 944–952.

Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society* B, **59**, 731–792.

Roberts, C. J., Hibbard, B. M., Roberts, E. E., Evans, K. T., Laurence, K. M. and Robertson, I. B. (1983) Diagnostic effectiveness of ultrasound detection of neural tube defect–the South Wales experience of 2509 scans (1977-1982) in high-risk mothers. *Lancet*, **2**, 1068–1069.

Robert, C. P. (1995) Convergence control methods for Markov chain Monte Carlo algorithms. *Statistical Science*, **10**, 231–253.

Rohling, R. N., Gee, A. H. and Berman, L. (1996) Three-dimensional spatial compounding of ultrasound images. *Medical Image Analysis*, **1**, 177–193.

Rue, H. and Husby, O. (1997) Identification of partly destroyed objects using deformable templates. Research Report, Department of Mathematical Sciences, Norwegian University of Science and Technology, Norway.

Silverman, B. W., Jennison, C., Stander, J. and Brown, T. C. (1990) The specification of edge penalties for regular and irregular pixel images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-12**, 1017–1024.

Smith, A. F. M. and Roberts, G. O. (1993) Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society* B, **55**, 3–23.

Stander, J. (1992) Some topics in statistical image analysis. PhD thesis, University of Bath, UK.

Stander, J. and Silverman, B. W. (1994) Temperature schedules for simulated annealing. *Statistics and Computing*, **4**, 21–32.

Storvik, G. (1994) A Bayesian approach to dynamic contour through stochastic sampling and simulated annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-16**, 976–986.

Synge, J. L. and Griffith, B. A. (1959) *Principles of Mechanics* (Third edition), McGraw-Hill Book Company, Tokyo.

Teles de Figueiredo, M. and Leitão, J. M. N. (1992) Bayesian estimation of ventrical contours in angiographic image. *IEEE Transactions on Medical Imaging*, **MI-11**, 416–429.

Thomas, J. G., Peters II, R. A. and Jeanty, P. (1991) Automatic segmentation of ultrasound images using morphological operators. *IEEE Transactions on Medical Imaging*, **MI-10**, 180–186.

Thompson, A. M., Brown, J. C., Kay, J. W. and Titterington, D. M. (1991) A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-13**, 326–339.

UK Collaborative Study on Alpha-fetoprotein in Relation to Neural Tube Defects. (1977) Maternal serum-alpha-fetoprotein measurement in antenatal screening for anencephaly and spina bifida in early pregnancy. *Lancet*, **1**, 1323–1332.

Weir, I. S. (1997) Fully Bayesian reconstructions from single-photon emission computed tomography data. *Journal of the American Statistical Association*, **92**, 49–60.

Winkler, G. (1995) *Image Analysis, Random Fields and Dynamic Monte Carlo Methods: a Mathematical Introduction*, Springer-Verlag, Berlin.

Wright, D., Stander, J. and Nicolaides, K. H. (1997) Non-parametric density estimation and discrimination from images of shapes. *Applied Statistics*, **46**, 365–380.

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior consent.

To my wife Wenting and our son Lang