



**ATHENS UNIVERSITY
OF ECONOMICS AND BUSINESS**
DEPARTMENT OF STATISTICS
POSTGRADUATE PROGRAM

**SMOOTHING SEGMENTATED IMAGES:
A REVIEW**

By

Stamatina S. Nikolaou

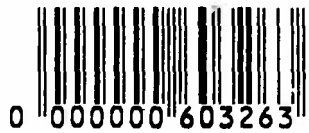
A THESIS

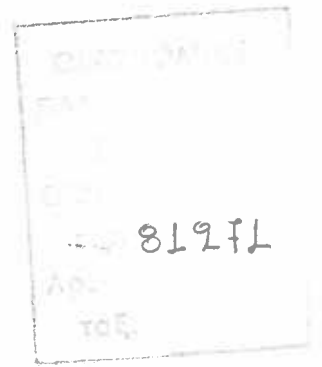
Submitted to the Department of Statistics
of the Athens University of Economics and Business
in partial fulfilment of the requirements for
the degree of Master of Science in Statistics

Athens, Greece
2007



ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΚΑΤΑΛΟΓΟΣ





**ATHENS UNIVERSITY
OF ECONOMICS AND BUSINESS**

DEPARTMENT OF STATISTICS

POSTGRADUATE PROGRAM

**SMOOTHING SEGMENTATED IMAGES:
A REVIEW**

By

Stamatina S. Nikolaou

A THESIS

Submitted to the Department of Statistics
of the Athens University of Economics and Business
in partial fulfilment of the requirements for
the degree of Master of Science in Statistics

Athens, Greece
March 2007







81291

**ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

**ΕΞΟΜΑΛΥΝΣΗ ΚΑΤΑΤΜΗΜΕΝΩΝ ΕΙΚΟΝΩΝ:
ΜΙΑ ΑΝΑΣΚΟΠΗΣΗ**

Σταματίνα Σταύρου Νικολάου

ΔΙΑΤΡΙΒΗ

Που υποβλήθηκε στο Τμήμα Στατιστικής
του Οικονομικού Πανεπιστημίου Αθηνών
ως μέρος των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος Ειδίκευσης στη Στατιστική

Αθήνα
Μάρτιος 2007



ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ ΚΑΙ ΕΚΜΟΧΕΥΣΗ

ΠΡΟΣΧΕΔΙΟ ΠΡΟΓΡΑΜΜΑΤΟΣ ΕΡΓΑΣΙΑΣ
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ





**ATHENS UNIVERSITY
OF ECONOMICS AND BUSINESS
DEPARTMENT OF STATISTICS**

A Thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

**SMOOTHING SEGMENTATED IMAGES:
A REVIEW**

Stamatina S. Nikolaou

Approved by the Graduate Committee

P. Tsiamyrtzis
Lecturer
Thesis Supervisor

I. Ntzoufras
Assistant Professor
Members of the Committee

D. Karlis
Assistant Professor

Athens, June 2007

**Epameinondas Panas, Professor
Director of the Graduate Program**



ACKNOWLEDGEMENTS

I would like to thank my supervisor, Lecturer Panagiotis Tsiamyrtzis, for his support during the development of this thesis. I would also like to thank Professor Ioannis Pavlidis of Computational Physiology Lab, Dept. of Computer Science, University of Houston for providing me with the data set used in this thesis.



VITA

I was born in Athens in 1979. I graduated from Lyceum in 1996 and studied Mathematics in National University of Athens during the years 1996 and 2002. In 2003 I started my postgraduate studies in the Athens University of Economics and Business at the Statistics department.





ABSTRACT

Stamatina S. Nikolaou

SMOOTHING SEGMENTATED IMAGES: A REVIEW

March 2007

The aim of this thesis is to smooth images that have come out from segmentation.

The original images have been recorded by a thermal video camera. After applying a segmentation method, the pixels have been grouped into two clusters, one denoting the background and the other denoting the foreground of the image. Segmentation adds noise to the input image, since some foreground pixels have lower temperature value than others and are mistakenly considered to be background. Thus, we need to improve the output image by removing the noise.

We use for this purpose mathematical morphology based on set theory, spatial statistical analysis and MCMC algorithms.

At the end of each section some applications are included and the results of each method are compared with each other.





ΠΕΡΙΛΗΨΗ

Σταματίνα Νικολάου

ΕΞΟΜΑΛΥΝΣΗ ΚΑΤΑΤΜΗΜΕΝΩΝ ΕΙΚΟΝΩΝ: ΜΙΑ ΑΝΑΣΚΟΠΗΣΗ

Μάρτιος 2007

Ο σκοπός αυτής της διατριβής είναι η βελτίωση εικόνων που έχουν προκύψει από κατάτμηση.

Οι αρχικές εικόνες έχουν καταγραφεί από θερμική κάμερα. Μετά από την εφαρμογή μιας μεθόδου κατάτμησης, τα εικονοστοιχεία (pixels) έχουν ομαδοποιηθεί σε δύο σύνολα, όπου το ένα αναφέρεται στο φόντο, ενώ το άλλο αναφέρεται στο προσκήνιο της εικόνας. Η κατάτμηση προσθέτει θόρυβο στην αρχική εικόνα εφόσον μερικά εικονοστοιχεία του προσκηνίου έχουν χαμηλότερη θερμοκρασία από άλλα και θεωρούνται λανθασμένα ότι ανήκουν στο φόντο. Συνεπώς, χρειάζεται να βελτιώσουμε το αποτέλεσμα απομακρύνοντας τον θόρυβο.

Για αυτό τον σκοπό χρησιμοποιούμε μαθηματική μορφολογία που βασίζεται στην θεωρία συνόλων, στατιστική ανάλυση του χώρου και αλγορίθμους MCMC.

Στο τέλος κάθε ενότητας έχουν περιληφθεί μερικές εφαρμογές καθώς και συγκρίσεις των αποτελεσμάτων.



ΠΕΡΙΛΗΨΗ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ



TABLE OF CONTENTS

Page

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: IMAGE SEGMENTATION

2.1 INTRODUCTION 3

2.2 THE K-MEANS ALGORITHM..... 3

2.2.1 DESCRIPTION OF THE ALGORITHM 3

2.2.2 INITIALIZATION 4

2.2.3 THE DISTANCE..... 4

2.2.4 TERMINATING CONDITION..... 5

2.2.5 WEAKNESSES OF THE ALGORITHM 6

2.2.6 EXAMPLE 6

2.3 OTSU’S THRESHOLD ALGORITHM 7

2.3.1 DESCRIPTION OF THE METHOD 7

2.3.2 EXAMPLE 8

2.4 COMPARING THE TWO ALGORITHMS 9

CHAPTER 3: BINARY MORPHOLOGY

3.1 INTRODUCTION 11

3.2 STRUCTURING ELEMENT 11

3.3 MORPHOLOGICAL OPERATORS 12

3.3.1 BINARY TRANSLATION..... 13

3.3.2 BINARY DILATION 14

3.3.2.1 EXAMPLES OF DILATION 17

3.3.3 BINARY EROSION 21

3.3.3.1 EXAMPLES OF EROSION..... 22

3.3.4 BINARY OPENING 24

3.3.4.1 EXAMPLES OF OPENING 24

3.3.5 BINARY CLOSING 26

3.3.5.1 EXAMPLES OF CLOSING..... 26

3.3.6 BOUNDARY CONDITIONS 30

3.3.6.1 IGNORE THE REMAINING PIXELS..... 30

3.3.6.2 ADD EXTRA PIXELS TO THE IMAGE..... 31



CHAPTER 4: MARKOV RANDOM FIELDS AND STOCHASTIC IMAGE MODELS

4.1 INTRODUCTION 33
4.2 SOME BAYESIAN STATISTICS 33
4.3 NEIGHBORHOOD SYSTEMS AND CLIQUES 34
4.4 MARKOV RANDOM FIELDS 36
4.4.1 THE ISING MODEL 37
4.4.2 AN APPLICATION OF THE ISING MODEL 39

CHAPTER 5: MARKOV CHAIN MONTE CARLO METHODS

5.1 INTRODUCTION 45
5.2 MARKOV CHAINS 46
5.3 MARKOV CHAIN MONTE CARLO METHODS 47
5.3.1 THE GIBBS SAMPLER 47
5.3.1.1 THE GIBBS SAMPLER FOR THE ISING MODEL 48
5.3.1.2 THE GIBBS SAMPLER FOR THE POSTERIOR 50
5.3.2 THE METROPOLIS-HASTINGS SAMPLER 54
5.3.2.1 THE METROPOLIS-HASTINGS SAMPLER FOR THE ISING MODEL 55
5.3.2.2 THE METROPOLIS-HASTINGS SAMPLER FOR THE POSTERIOR OF INTEREST 57
5.3.3 IMAGE ESTIMATION 60
5.3.3.1 THE MARGINAL POSTERIOR MODE ESTIMATOR 60
5.3.3.2 THE MAXIMUM A POSTERIORI ESTIMATOR 62

CHAPTER 6: FACE DETECTION

6.1 THE IMPORTANCE OF FACE DETECTION 67
6.2 FACE DETECTION USING K-MEANS 67
6.3 SOME MORE APPLICATIONS 70

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS 73
7.2 FUTURE WORK 73

APPENDIX 75

REFERENCES 99



LIST OF TABLES

Table	Page
Table 2.4.1: The outcome of the subtraction of (a) and (b) in Figure 2.4.1. This table shows that the K-means method and the Otsu’s threshold “agree” in all skin pixels and in 57554 out of 57556 background pixels	10
Table 4.4.2.1: The probabilities given by the Ising model, for several values of ising constant β and number of black neighbors k	40





LIST OF FIGURES

Figure	Page
Figure 2.2.6.1 : (a) The initial thermal image (the mean temperature is equal to 30.59 and the variance is equal to 3.49) and (b) the binary image we get after segmentation into two groups with the k-means algorithm. The initial points have been chosen at random.	6
Figure 2.3.2.2: (a) The input image and (b) the image after applying Otsu’s algorithm	9
Figure 2.4.1: (a) The output of the segmentation after using the k-means algorithm, (b) the output of the Otsu’s segmentation method.	10
Figure 3.2.1:(a) Diamond-shaped structuring element with radial $R=1$, where R is the distance between the origin and the vertices of the rhomb, (b) square structuring element with width $w=3$, (c) rectangular structuring element with 4 rows and 5 columns.	12
Figure 3.3.1.1: (a) The input image, (b) the structuring element with the gray pixel indicating the origin, (c) the translation of A by b	13
Figure 3.3.1.2: (a) The input image, (b) the structuring element with the gray pixel indicating the origin, (c) the translation of A by b . The result of the translation is the shift of the image one pixel to the right.	14
Figure 3.3.2.1: A is the input binary image, B is the diamond-shaped structuring element with radial $R=2$ and $A \oplus B$ is the dilated image.....	15
Figure 3.3.2.2: (a) The input image and (b) the output of dilation of (a) by diamond structuring element with $R=3$ (see Figure 3.3.2.4).....	15
Figure 3.3.2.3: The sketch is obtained by subtracting from image (b) the original image (a) in Figure 3.3.2.2.....	16
Figure 3.3.2.4:The diamond structuring element with $R=3$, i.e. the distance between the origin and the vertices is equal to 3.	16
Figure 3.3.2.1.1: The original binary image.....	17
Figure 3.3.2.1.3: (a) Radial equal to 1, (b) radial equal to 2, (c) radial equal to 4, (d) radial equal to 7. For radial values smaller than 4 the differences are unnoticed, as we can see in Figure 3.3.2.1.3.....	18
Figure 3.3.2.1.4: The octagonal structuring element with radial 3.	19



Figure 3.3.2.1.5: Dilation of the input image for radial values (a) R=3 and (b) R=9. 19

Figure 3.3.2.1.6: The rectangular 2x4 structuring element. 20

Figure 3.3.2.1.7: Dilation of the input image for different rectangular structuring elements. (a) M=4 and N=8 (b) M=6 and N=18 (c) M=6 and N=12 (d) M=8 and N=4..... 20

Figure 3.3.3.1: A is the input image. The gray pixel indicates the origin. B is the structuring element and $A \ominus B$ is the eroded image. 21

Figure 3.3.3.2: Edge detection obtained from erosion of image 3.3.2.1.1. The output of erosion is (a), while (b) is the difference of the two images. The structuring element we used is the diamond-shaped with radial equal to 3... 22

Figure 3.3.3.1.1: The radial of the structuring element is (a) R=1 and (b) R=3. 23

Figure 3.3.3.1.2: Erosion by octagonal structuring element with (a) R=3 and (b) R=6..... 23

Figure 3.3.4.1.1: Opening of the input image by the diamond structuring element for different values of radial (a) R=2, (b) R=3, (c) R=6, (d) R=7.... 25

Figure 3.3.4.1: Octagon with radial (a) R=6 and (b) R=9..... 25

Figure 3.3.5.1: A is the image we wish to perform the closing operation and B is the structuring element. Firstly, we dilate A by B to get $A \oplus B$ and then we erode $A \oplus B$ by B and get $A \bullet B$ 26

Figure 3.3.5.1.1: Diamond shaped structuring element for (a) R=12, (b) R=15, (c) R=20, (d) R=25, (e) R=35, (f) R=45. 27

Figure 3.3.5.1.2: Octagon for (a) R=9, (b) R=12, (c) R=15, (d) R=21, (e) R=24, (f) R=27..... 28

Figure 3.3.5.1.3: Rectangular with dimensions (a) M=10 and N=40, (b) M=10 and N=60, (c) M=10 and N=80, (d) M=20 and N=20, (e) M=20 and N=40, (f) M=20 and N=60 29

Figure 3.3.6.1: The structuring element B when applied to the boundary pixels of A exceeds the image (the structuring element is shown with bold lines).. 30

Figure 3.3.6.1.2 The result of erosion A by B of Figure 3.3.6.1 when the “out of the boundary” pixels ignored..... 31



Figure 3.3.6.2.1 Adding outer pixels for performing morphological operations in the image 31

Figure 3.3.6.2.2 The result of erosion A by B of Figure 3.3.6.1 when foreground pixels were added to the borders..... 32

Figure 4.3.1: (a) nearest-neighbor scheme, (b) 8-neighbor lattice scheme, (c) 12-neighbor lattice scheme and (d) 24-neighbor lattice scheme. 35

Figure 4.3.2: Some cliques for (a) the nearest-neighbor lattice scheme, (b) the 8-neighbor scheme, (c) the 12-neighbor scheme and (d) the 24-neighbor scheme..... 36

Figure 4.4.1.1: The plots of $P(x)$ versus k in the 8-neighbor scheme, for several values of β . (a) $\beta=0.01$, (b) $\beta=0.5$, (c) $\beta=0.7$, (d) $\beta=0.8$, (e) $\beta=1.0$, (f) $\beta=5.0$ 38

For $\beta=0.1$ $P(x)$ increases in an almost linear way. The larger values of β give probability $P(x)$ close to zero for less than 4 black neighbors and probability close to 1 for more than 4 black neighbors. 39

Figure 4.4.2.2: The initial binary image 40

Figure 4.4.2.3: The Ising model for number of iterations: (a) $M=5$, (b) $M=10$, (c) $M=15$ and (d) $M=20$ 41

Figure 4.4.2.4: Iterative applications of the Ising model. The algorithm terminates when at least a ratio R of the total number of pixels has changed value. The respective ratio numbers are: (a) $R=0.025$, (b) $R=0.075$, (c) $R=0.085$ and (d) $R=0.10$ 42

Figure 4.4.2.5: The Ising model for $R=0.85$ and different values of beta: (a) $\beta=0.3$, (b) $\beta=0.5$, (c) $\beta=0.7$ and (d) $\beta=0.9$ 43

Figure 4.4.2.6: The Ising model for $R=0.85$ and different values of beta: (a) $\beta=0.3$, (b) $\beta=0.5$, (c) $\beta=0.7$, (d) $\beta=0.9$ 44

Figure 5.3.1.1.1: Simulation from the Ising model using Gibbs sampler (20 iterations) for different values of beta constant. (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.8$, (e) $\beta=0.9$, (f) $\beta=1.0$ 49

Considering the equations (5.3.1.2.4) and (5.3.1.2.5), the Gibbs sampler for the noisy data (Hurn, Husby and Rue 2001) can be written:..... 51

Figure 5.3.1.2.1: The results of Gibbs sampler for the posterior distribution with beta values: (a) $\beta=0.4$, (b) $\beta=0.6$, (c) $\beta=0.8$, (d) $\beta=1.0$ 52



Figure 5.3.1.2.2: The Gibbs sampler for the posterior for different values of the probability p that we have observed the wrong image: (a) $p=0.1$, (b) $p=0.4$, (c) $p=0.5$, (d) $p=0.8$ 53

Figure 5.3.1.2.3: The initial image that was obtained by segmentation. The segmentation method that has been used is the K-means algorithm. The initial centroids were chosen at random. The values of the pixel temperatures in this image are the x_i 's in Gibbs sampler. 54

Figure 5.3.2.1.1: Simulation from the Ising model using the Metropolis sampler for different values of beta constant. (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.8$, (e) $\beta=0.9$, (f) $\beta=1.0$ 56

Figure 5.3.2.2.1: The results of the simulation from the noisy Ising model for flip noise. The beta parameter is respectively: (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.9$ 58

Figure 5.3.2.2.2: The results of the simulation from the noisy Ising model for flip noise. The different values of the probability p are respectively: (a) $p=0.1$, (b) $p=0.3$, (c) $p=0.5$, (d) $p=0.7$, (e) $p=0.8$, (f) $p=0.9$ 59

Figure 5.3.3.1.1: The MPM estimator for different number k of samples. (a) $k=10$ ($\hat{L}_{MPM} = 42645$), (b) $k=20$ ($\hat{L}_{MPM} = 46331$), (c) $k=30$ ($\hat{L}_{MPM} = 48169$), (d) $k=40$ ($\hat{L}_{MPM} = 49757$)..... 62

Figure 5.3.3.2.1: The MAP estimator for beta constant equal to 0.4 and 20 samples from the posterior. The probability p that we have obtained the wrong image is respectively (a) $p=0.1$, (b) $p=0.3$, (c) $p=0.5$, (d) $p=0.7$, (e) $p=0.8$, (f) $p=0.9$ 64

Figure 5.3.3.2.2: The MAP estimator for beta constant $\beta=0.4$ and probability of recording the wrong image $p=0.8$. The number k of samples is respectively (a) $k=5$, (b) $k=10$, (c) $k=15$, (d) $k=20$ 65

Figure 6.2.1: The blue column contains the head. The upper blue row contains the head, while the lower one contains the shoulders. 68

Figure 6.2.2: The frame has been divided into three vertical areas. The central area includes the face..... 68

Figure 6.2.3: The head lies within the blue columns..... 69

Figure 6.2.4: The rectangular region that contains the head. 69



Figure 6.2.5: The areas that result from the application of K-means to the rows of the image in Figure 6.2.4 70

Figure 6.2.6: The rectangular has detected the face 70

Figure 6.3.1: Face detection for different orientations of the head..... 71





CHAPTER 1

INTRODUCTION

In the present thesis we are dealing with image processing techniques. The data are frames of M rows and N columns ($M \times N$ pixels) that have been obtained from thermal video recordings of individuals. Each pixel is the temperature of the respective element in the frame. In thermal images the elements are usually presented with colors. The red color indicates high temperature, while the blue color indicates low temperature. The in-between temperatures are denoted by shades of red and blue.

We are interested in separating the image into two regions (hot corresponding to individuals and cold corresponding to background), using segmentation methods. Unfortunately, segmentation does not work properly for the in-between (not too hot, not too cold) temperatures. Thus, we try to reduce noise from the segmented image and improve it. Throughout this thesis we will focus on the warmer area, which corresponds to facial region (when the image contains an individual).

In chapter 2 we segment the image elements into two homogeneous, non-overlapping groups. One group consists of the elements that have higher temperature and as a result they contain mainly skin-related pixels, while the other group consists of the lower temperature pixels. The output is a binary image, since every element has one of two possible values. From now on the warm pixels will be referred as skin pixels, while the cold will consist the non-skin pixels (1 and 0 respectively in a binary notation). The binary image that comes out from the segmentation is not an exact representation of the real scene. This happens because some face areas such as the ears, the nose or the area surrounding the mouth have lower blood flow and as a result they have lower temperature. Thus, quite often the segmentation algorithm falsely



classifies these regions to be non-skin. The analogous happens with the skin regions that are covered by clothes, hair, etc.

In chapter 3 we try to improve the appearance of the image by eliminating misclassified pixels using set theory and in chapters 4 and 5 we cope with the same problem using Bayesian methods. Specifically, in chapter 4 we make use of the spatial dependence of the image temperatures and apply Markov Random Fields (MRF) theory to the data. In chapter 5 we simulate from the posterior distribution of the data and estimate the true image with Markov Chain Monte Carlo (MCMC) algorithms.

The goal of chapter 6 is to detect the position of the face in any frame. Some applications are presented for different positions of the head

Chapter 7 is a conclusion of the thesis, containing some future work.

Finally, in the appendix we include all the algorithms used, written in Matlab code.



CHAPTER 2

IMAGE SEGMENTATION

2.1 INTRODUCTION

Segmentation in image analysis is the partitioning of a digital image into multiple regions according to a specific criterion. In our problem we have a thermal image, which is a frame of M rows and N columns, where each pixel is associated with a temperature value. The aim is to separate the pixels into two groups. The pixels of the first group will indicate the foreground, while the second group pixels will indicate the background of the image. In other words, after the segmentation the continuous image will turn into a binary one, where the value 1 denotes foreground and the value 0 denotes background pixels. There are many algorithms that can be used to achieve image segmentation. Here we will present the K-means algorithm (MacQueen 1967) and the Otsu's threshold algorithm (Otsu 1979).

2.2 THE K-MEANS ALGORITHM

The K-means method (MacQueen 1967) uses an iterative algorithm to group data into k groups. The algorithm treats the observations as points having a location in space. Each group is defined by its member observations and its center or centroid. The centroid for each cluster is chosen so that the sum of distances from all objects in that cluster is minimized.

2.2.1 DESCRIPTION OF THE ALGORITHM

The first step is to decide the number k of clusters. One restriction of the method is that the number k is fixed a priori. Then, k centroids must be



defined, one for each cluster. In our case we have to define 2 initial centroids, since we need two groups. The next step is to associate each of the pixels in the data set to one of the centroids. The criterion for this association is that the pixel is being grouped with its nearest centroid. After all pixels have been associated to a centroid, k groups have been generated. At this point, new centroids must be decided. The new centroids are calculated using the values within the groups (in our study we will use the mean of the values in the group and this is consistent with the definition of the centroid). Again, each pixel is associated to its nearest centroid and new groups are generated. The procedure is being repeated until the centroids do not change any more or change less than a prespecified threshold. The aim of the algorithm is to minimize the sum of distances from each object to its cluster centroid, over all clusters.

2.2.2 INITIALIZATION

There is not a standard way of deciding the initial centroids. The resulting groups might depend on the centroids selected. The most common solution is to arbitrarily assign groups to the pixels by assigning to the i^{th} pixel the i modulo k cluster. Another way is to choose k pixels at random with the constraint that the distance of the initial pixels is large enough. For example, we may require that the distance is much larger than k and smaller than the total number of pixels (Leeser, 1999).

2.2.3 THE DISTANCE

The most common distance used in the K-means algorithm is the Euclidean distance:

$$d(x, y) = \left(\sum_{i=1}^n (x_i - y_i)^2 \right)^{\frac{1}{2}} \quad (2.2.3.1)$$



where $x=(x_1, x_2, \dots, x_n)$ and $y=(y_1, y_2, \dots, y_n)$ are two observations. In other words, the Euclidean distance is a generalization of the Pythagorean theorem to more than two coordinates.

Another distance that can be used is the Hellinger distance, which is defined as:

$$d(x, y) = \sum_{i=1}^n (\sqrt{x_i} - \sqrt{y_i})^2 \tag{2.2.3.2}$$

The Canberra distance is:

$$d(x, y) = \sum_{i=1}^n \frac{|x_i - y_i|}{|x_i + y_i|} \tag{2.2.3.3}$$

The Canberra distance makes a summation of a series of ratios between corresponding coordinates of two points x and y. This measure takes into account the physical distance of the two points and their relation to the origin, as well. Therefore, it is biased around the origin.

In cases where the Hellinger and the Canberra distance is used, the observations must be shifted to positive values.

Finally, another popular distance is the Manhattan distance:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i| \tag{2.2.3.4}$$

From now on we will use the Euclidean distance, since it coincides with the direct distance.

2.2.4 TERMINATING CONDITION

Many terminating conditions are used for the K-means algorithm. Theoretically, the iterations stop when there is small change in the clusters. This means that if the result is not global optimal, at least it will be local optimal. However, sometimes the convergence time of the algorithm may be long. In such cases, the algorithm stops after maximum number of iterations that we have decided.



2.2.5 WEAKNESSES OF THE ALGORITHM

As mentioned above, the number of groups must be known a priori and cannot be decided during the execution of the algorithm. Another problem is that the result depends on the initial centroids. Different initial centroids may lead to different grouping. If the number of data is much larger than the number of clusters, as it happens in our data, this problem is eliminated. Moreover, the use of the mean value for calculating the centroids is not robust to outliers and a way to overcome this weakness is to use the median instead of the mean value. Finally, different choice of distance will lead to different results. In order to overcome this, we can standardize the values so that they don't depend on the distance measure.

2.2.6 EXAMPLE

The frame we use in the example has 254 rows and 318 columns. The values associated with the pixels are continuous and indicate the temperature of each pixel. The data are univariate, which means that $n=1$. We choose number of groups equal to 2 while the initialization of centroids is made at random and the distance measure is the Euclidean one. The results of the k-means image segmentation can be seen in Figure 2.2.6.1.

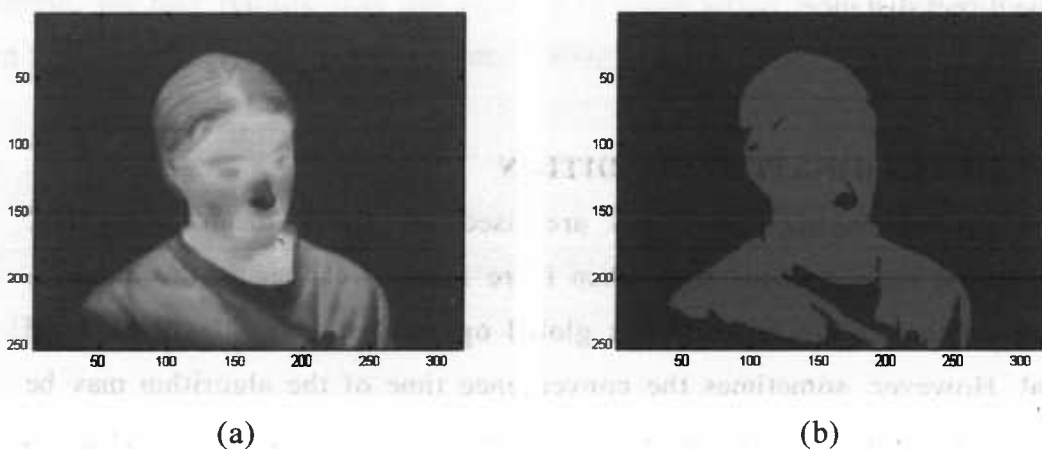


Figure 2.2.6.1 : (a) The initial thermal image (the mean temperature is equal to 30.59 and the variance is equal to 3.49) and (b) the binary image we get after segmentation into two groups with the k-means algorithm. The initial points have been chosen at random.



2.3 OTSU'S THRESHOLD ALGORITHM

The Otsu's algorithm (Otsu 1979) is based on the intensity histogram, which is a graph, showing the number of pixels in an image at each different intensity value found in that image. In the case of a thermal image, the horizontal axis contains the temperatures, while the vertical axis contains the number pixels that correspond to those temperatures. The Otsu's method relies on the assumption that thermal pictures have a bi-modal distribution, with the lower lobe for background and the upper lobe for foreground. We try to find a point between the two distributions such that the variance between those groups is maximized, while the variance within groups is minimized.

2.3.1 DESCRIPTION OF THE METHOD

Suppose that the input image has L levels of intensity, in other words it has L different pixel values and the number of pixels at the intensity level i is n_i . If the total number of pixels is N , then the probability of each pixel value occurring is

$$p_i = \frac{n_i}{N} \quad (2.3.1.1)$$

The mean value of the image is

$$\mu = \sum_{i=1}^L \frac{i \cdot p_i}{\omega} \quad (2.3.1.2)$$

where

$$\omega = \sum_{i=1}^L p_i \quad (2.3.1.3)$$

The total variance is

$$\delta_i^2 = \sum_{i=1}^L i^2 \cdot p_i - \mu^2 \quad (2.3.1.4)$$

If the threshold is at level j , then two classes have been formed. The first includes the levels 1 to j and has mean intensity value

$$\mu_0 = \sum_{i=1}^j \frac{i \cdot p_{(i)}}{\omega_0} \quad (2.3.1.5)$$

where

$$\omega_0 = \sum_{i=1}^j p_{(i)} \tag{2.3.1.6}$$

The second includes the levels j+1 to L and has mean intensity value

$$\mu_1 = \sum_{i=j+1}^L \frac{i \cdot p_{(i)}}{\omega_1} \tag{2.3.1.7}$$

where

$$\omega_1 = \sum_{i=j+1}^L p_{(i)} \tag{2.3.1.8}$$

The between class variance is

$$\delta_b^2 = \omega_0 (\mu_0 - \mu_t)^2 + \omega_1 (\mu_1 - \mu_t)^2 \tag{2.3.1.9}$$

If we consider that

$$\mu_t = \omega_0 \mu_0 + \omega_1 \mu_1 \tag{2.3.1.10}$$

then

$$\delta_b^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2 \tag{2.3.1.11}$$

The criterion to choose the threshold is to maximize the ratio:

$$\frac{\delta_b^2}{\delta_t^2} \tag{2.3.1.12}$$

2.3.2 EXAMPLE

We use the same continuous image we used for the k-means method. In figure 2.3.2.1 is shown the histogram of the temperatures. The threshold has been found at temperature 31.3880. All pixels that have temperatures bellow this threshold obtain value 0 while the others obtain value 1. In Figure 2.3.2.2 can be seen the segmentation of the image according to Otsu's threshold.



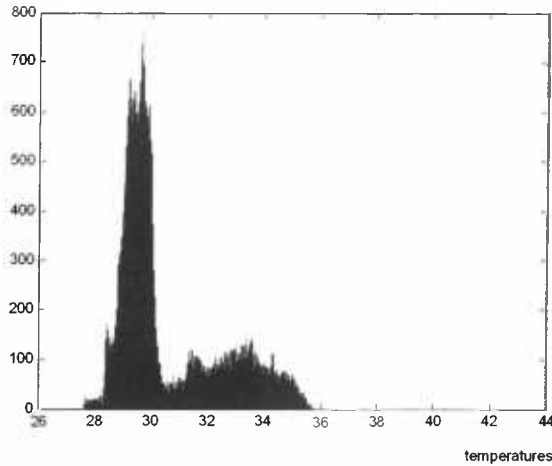


Figure 2.3.2.1: The histogram of temperatures. The group of the lower temperatures has mean temperature equal to 29.5 and variance 0.34, while the group of higher temperatures has mean 33.27 and variance 1.2



Figure 2.3.2.2: (a) The input image and (b) the image after applying Otsu's algorithm

2.4 COMPARING THE TWO ALGORITHMS

Obviously, the two methods gave similar results, which means that they are stable. This can be shown in Table 2.4.1, while Figure 2.4.1 shows the output pictures of both algorithms. Specifically for the same image only two pixels out of 80772 have different values in the output images, as shown in Figure 2.4.2. That is because we have chosen to separate the image into two regions skin and no skin. The Otsu's algorithm works well for higher values of the between clusters variance. If the between variance is small, the



segmentation with Otsu's threshold gives bad segmentation. In the example the two clusters are distinct and both methods worked fairly well.

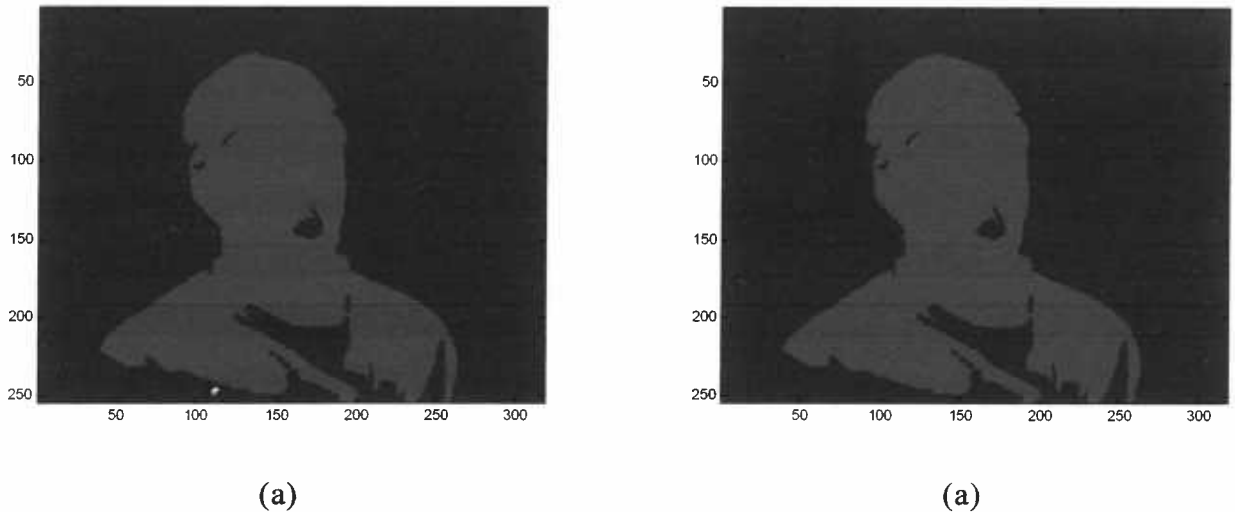


Figure 2.4.1: (a) The output of the segmentation after using the k-means algorithm, (b) the output of the Otsu's segmentation method.

		Otsu's method	
		Skin pixels	Background pixels
K-means	Skin pixels	23216	0
	Background pixels	2	57554

Table 2.4.1: The outcome of the subtraction of (a) and (b) in Figure 2.4.1. This table shows that the K-means method and the Otsu's threshold "agree" in all skin pixels and in 57554 out of 57556 background pixels



CHAPTER 3

BINARY MORPHOLOGY

3.1 INTRODUCTION

Mathematical morphology is a methodology for analyzing the shape of objects contained in an image, using set theory and shift operators. Mathematical morphology can provide boundaries of the objects, their skeletons or it can be used in order to fill ‘holes’ in images expand or shrink edges. The image consists of pixels, each having associated with it a value. In binary images each pixel can take one of two possible values, usually 1 or 0. In this way the image consists of two sets. The points contained in one set are called foreground and have value one, while those contained in the complement set are called background. In grayscale images the value of a pixel shows its brightness and it is an integer between 0 and 255. The 0 represents the black color, the 255 represents white and all other values are taken for shades of gray. The whole idea of morphology is implemented with the aim of the morphological operators, which transform the original image into another through interaction with an image of certain shape and size called structuring element.

3.2 STRUCTURING ELEMENT

The structuring element contains a pattern, which depends on the pixels relative to some origin. A flat structuring element consists of a 2-dimensional matrix of 1’s and 0’s and is smaller than the image being processed. The pixel that identifies the pixel of interest in the image is called the origin of the structuring element. Usually, the origin is the center pixel, but this is not always the case. The pixels that have the value 1 define the neighborhood.



The most useful structuring elements are the diamond, the rectangular, the square and the octagonal structuring elements. As regards the diamond-structuring element, a number defines the neighborhood, which is the radial of the diamond. Specifically, the diamond with radial R is a rhomb shaped neighborhood with a horizontal and a vertical diagonal. Each diagonal contains $2R+1$ foreground pixels, the central one and R pixels in each side. In other words, R is the distance between the center and the vertices of the rhomb. The octagonal structuring element with radial R is a flat octagon shaped neighborhood, where R specifies the distance from the origin to the sides of the octagon, as measured along the horizontal and vertical axes. R must be a nonnegative multiple of 3.

The geometrical features of the image that are similar to those of the structuring element are preserved, while the others are suppressed. Some examples of structuring elements are shown in Figure 3.2.1. The gray colored pixel shows the origin of the structuring elements.

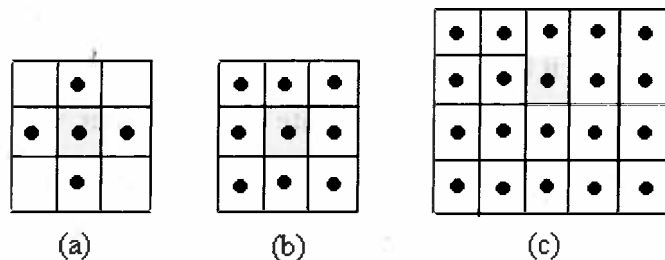


Figure 3.2.1:(a) Diamond-shaped structuring element with radial $R=1$, where R is the distance between the origin and the vertices of the rhomb, (b) square structuring element with width $w=3$, (c) rectangular structuring element with 4 rows and 5 columns.

3.3 MORPHOLOGICAL OPERATORS

Morphological operators take a binary image and a structuring element as input and combine them using a set operator (intersection, union, inclusion, complement). They process objects in the input image based on characteristics of its shape, which are encoded in the structuring element. The most

fundamental morphological operators are dilation and erosion, both based on translation, introduced below.

3.3.1 BINARY TRANSLATION

Given a binary image A and a structuring element b , the translation of A by b is defined as

$$A_b = A + b = \{a + b : a \in A\} \tag{3.3.1.1}$$

In other words, A_b is the set of elements that we get if we add all the elements of A with b . So, A will be shifted by the displacement b .

In Figure 3.3.1.1, translation of A by b will shift A , such that A is centered at pixel (3,3).

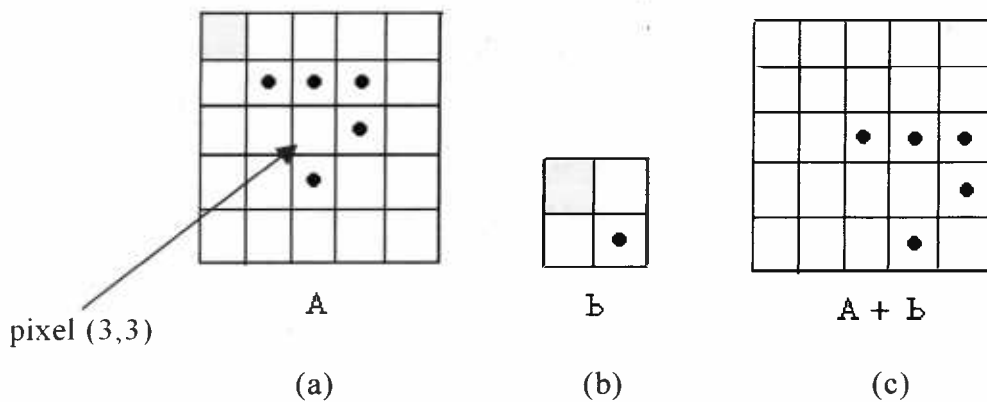


Figure 3.3.1.1: (a) The input image, (b) the structuring element with the gray pixel indicating the origin, (c) the translation of A by b



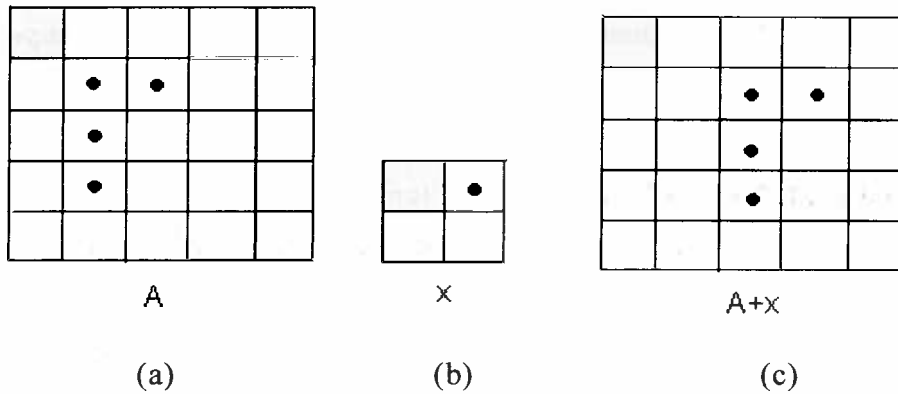


Figure 3.3.1.2: (a) The input image, (b) the structuring element with the gray pixel indicating the origin, (c) the translation of A by b. The result of the translation is the shift of the image one pixel to the right.

3.3.2 BINARY DILATION

Dilation of a binary image A by structure element B is defined as

$$A \oplus B = \{a + b \mid a \in A, b \in B\} \tag{3.3.2.1}$$

For example, if $A = \{1, 2, 5\}$ and $B = \{-1, 2\}$, then $A \oplus B = \{0, 1, 3, 4, 7\}$.

Dilation is also known as the Minkowski addition and is found if we apply the origin of the structuring element on every foreground pixel of the original image, which is called input pixel and take the copy of the structuring element.

Dilation is equivalent to a union of translates of the original image with respect to the structure element (Morse 2000).

$$A \oplus B = \bigcup_{b \in B} A_b \tag{3.3.2.2}$$

Dilation expands the image and as a result it eliminates small holes in it, while translation only shifts the image by structuring element. Moreover, it can be used for edge detection, by dilating an image and then subtracting away the original one. Figure 3.3.2.1 shows how dilation works with a simple example. In Figures 3.3.2.2 and 3.3.2.3 we can see what the outcome of dilation is and how we can get the sketch of a picture by subtracting the original from the dilated one.



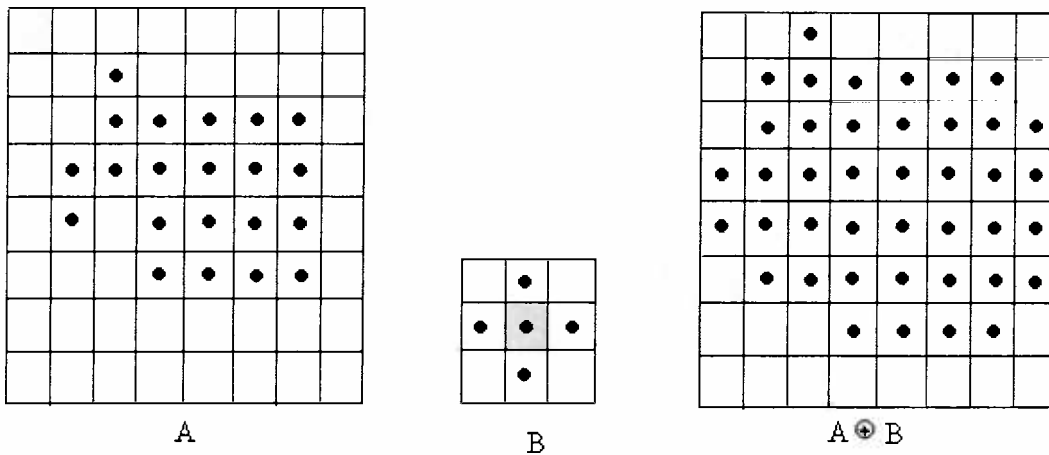


Figure 3.3.2.1: A is the input binary image, B is the diamond-shaped structuring element with radial $R=2$ and $A \oplus B$ is the dilated image.

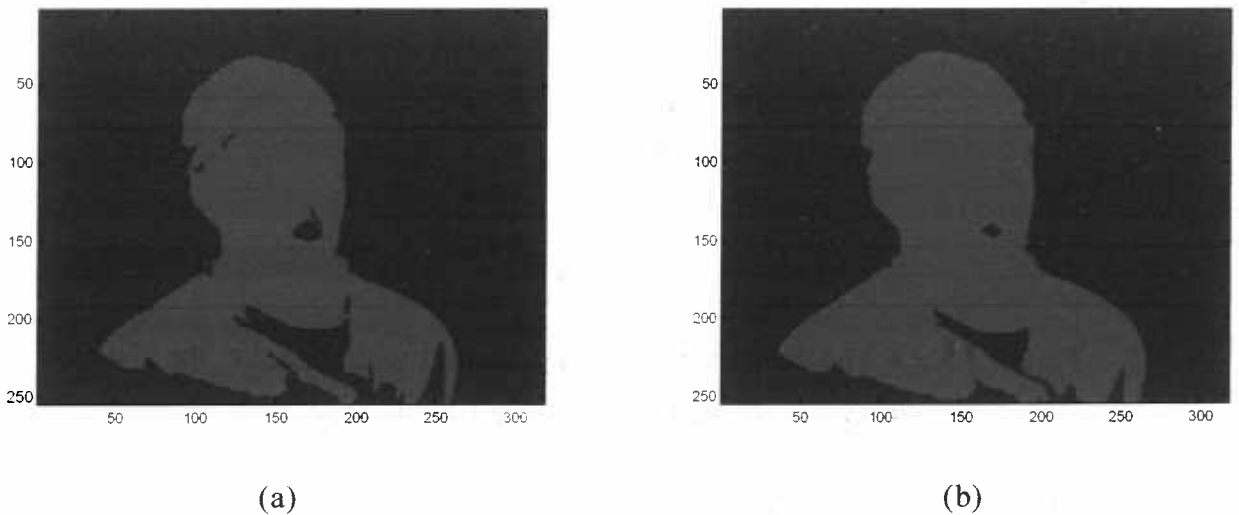


Figure 3.3.2.2: (a) The input image and (b) the output of dilation of (a) by diamond structuring element with $R=3$ (see Figure 3.3.2.4).

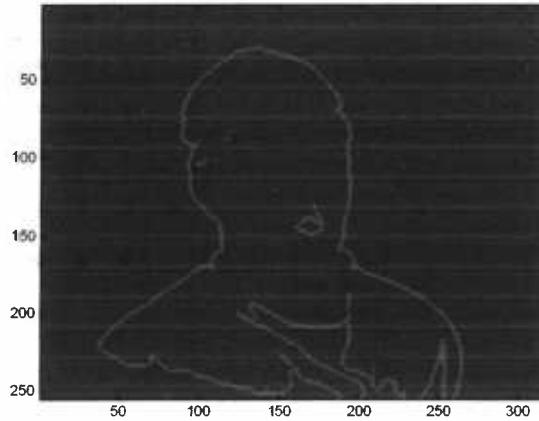


Figure 3.3.2.3: The sketch is obtained by subtracting from image (b) the original image (a) in Figure 3.3.2.2.

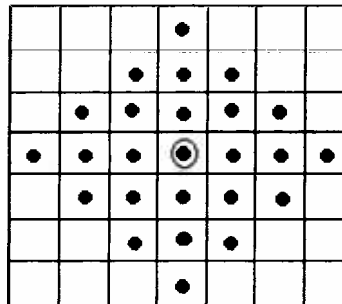


Figure 3.3.2.4: The diamond structuring element with R=3, i.e. the distance between the origin and the vertices is equal to 3.

Properties of binary dilation (Wayne, Lin, Wei-Cheng 2000).

1. The result of the translation of A dilated by structuring element B is the same as the translation of A dilated by structuring element B.

$$(A \oplus B)_x = A_x \oplus B$$

2. If more than two dilations are to be made, the order in which they are done does not matter.

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

3. If A is a subset of B, then dilation of A by C is a subset of the dilation of B by C.

$$A \subseteq B \Rightarrow A \oplus C \subseteq B \oplus C$$



3.3.2.1 EXAMPLES OF DILATION

The image we obtained from segmentation has some gaps in some face and chest regions. We use the dilation operator in order to eliminate the holes, trying not to expand the image more than it is necessary. Some applications with different shapes of structuring elements appear below. The input frame is shown in Figure 3.3.2.1.1.

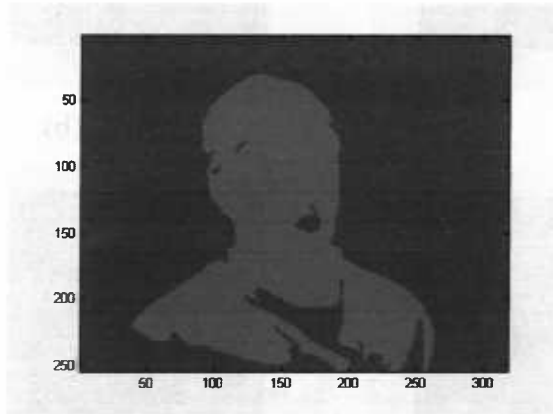


Figure 3.3.2.1.1: The original binary image.

THE DIAMOND-SHAPED STRUCTURING ELEMENT

We perform dilation using the diamond shaped structuring element with different values of radial. In Figure 3.3.2.1.2 is shown the one with radial equal to 2.

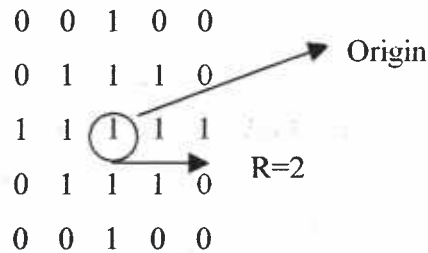


Figure 3.3.2.1.2: The diamond shaped structuring element with radial 2



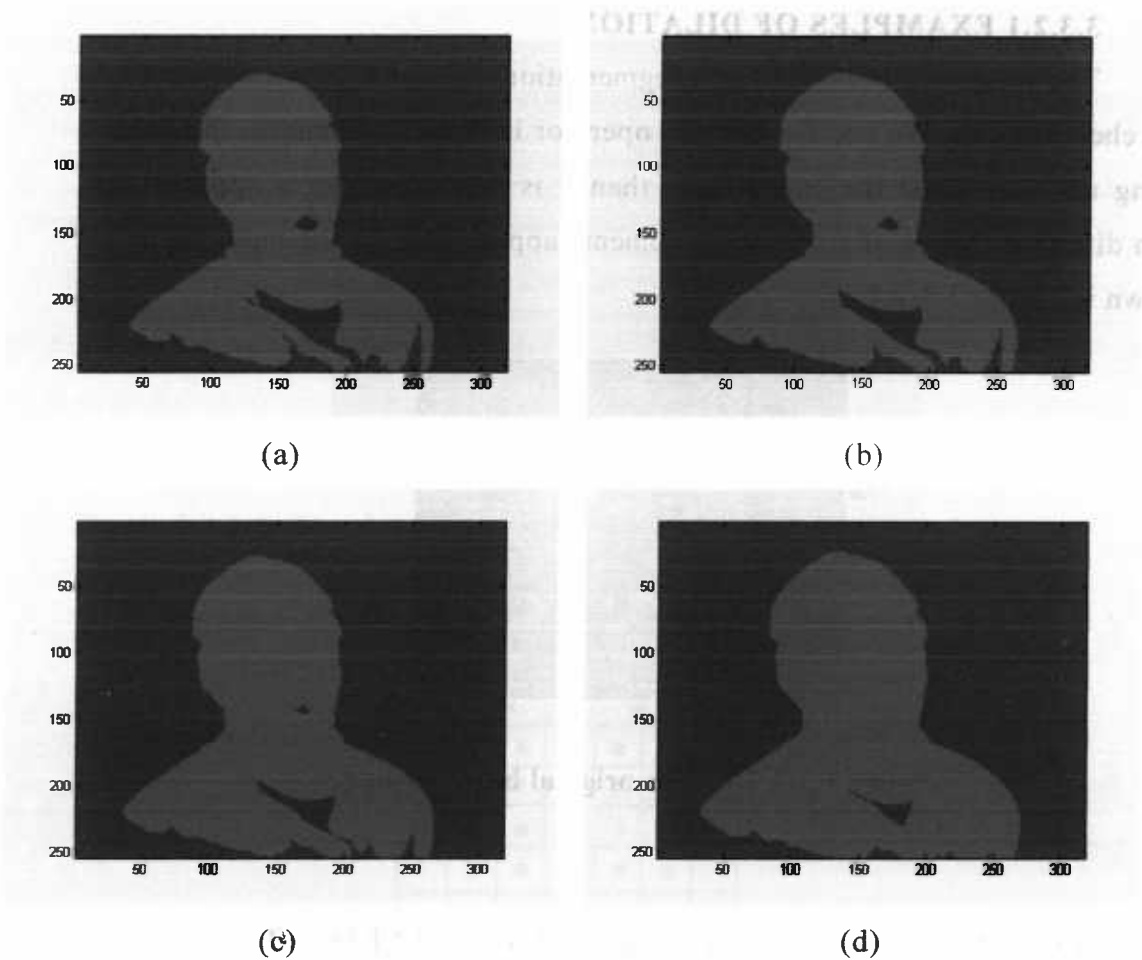


Figure 3.3.2.1.3: (a) Radial equal to 1, (b) radial equal to 2, (c) radial equal to 4, (d) radial equal to 7. For radial values smaller than 4 the differences are unnoticed, as we can see in Figure 3.3.2.1.3.

THE OCTAGONAL STRUCTURING ELEMENT

The octagonal structuring element is shown in Figure 3.3.2.1.4 for radial equal to 3. The radial R specifies the distance from the origin to the sides of the octagon as measured horizontally and vertically. R must be a nonnegative multiple of 3.



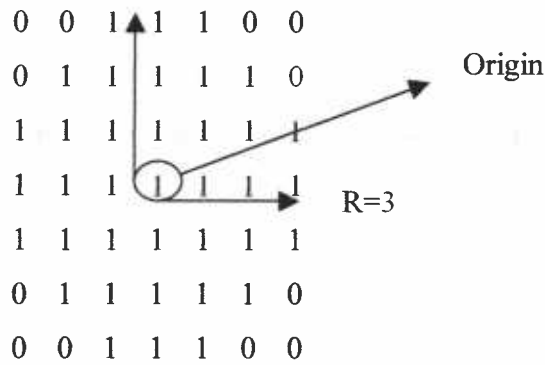


Figure 3.3.2.1.4: The octagonal structuring element with radial 3.

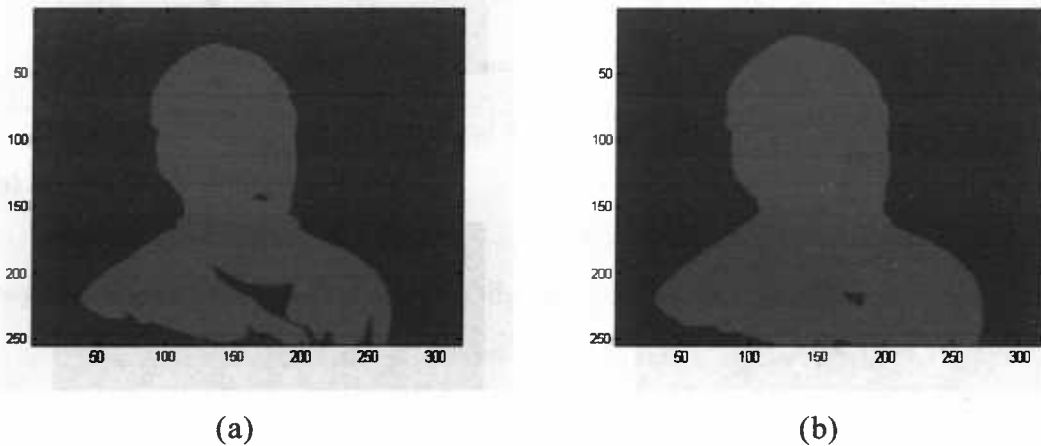


Figure 3.3.2.1.5: Dilation of the input image for radial values (a) $R=3$ and (b) $R=9$.

The results of the dilation by the octagonal structuring element are shown in Figure 3.3.2.1.5. Dilation by octagon structuring element with $R=3$ fills the holes in the face, except from the mouth. For the rest of the body it didn't work so well. For $R>9$ the image is expanded a lot.

THE RECTANGULAR STRUCTURING ELEMENT

In the rectangular structuring element the foreground pixels form a rectangle with M rows and N columns. In Figure 3.3.2.1.6 it is shown for $M=2$ and $N=4$.



```

1 1 1 1
1 1 1 1

```

Figure 3.3.2.1.6: The rectangular 2x4 structuring element.

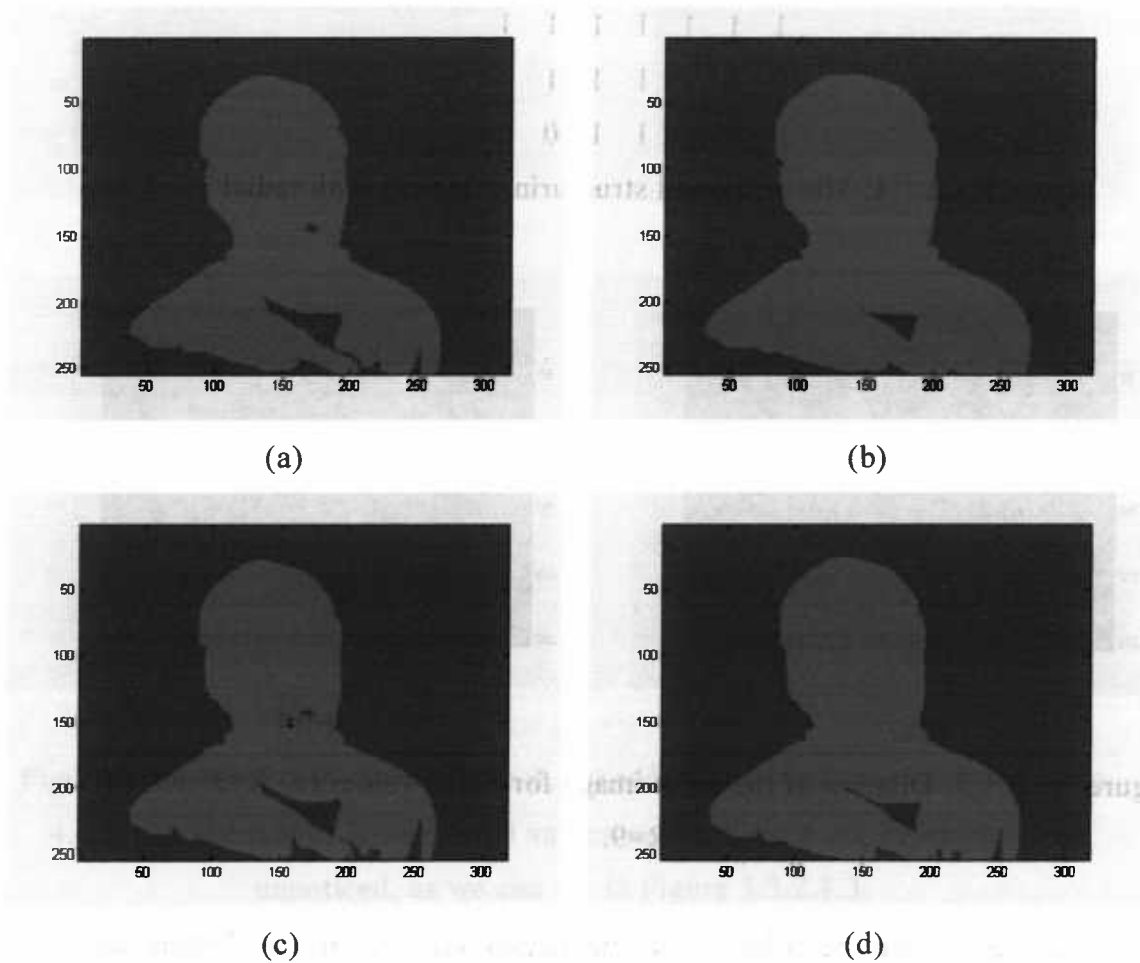


Figure 3.3.2.1.7: Dilation of the input image for different rectangular structuring elements. (a) $M=4$ and $N=8$ (b) $M=6$ and $N=18$ (c) $M=6$ and $N=12$ (d) $M=8$ and $N=4$

It is obvious from Figure 3.3.2.1.7 that the 4x8 rectangular structuring element used fills some small holes in the face. The 6x18 rectangle covers all gaps in the face and most of the rest image, but changes the shape of the image. The 12x6 fills the gaps in the face, but not in the rest of the image, while the 8x4 rectangle works better than the others, although it expands the right side of the head.



3.3.3 BINARY EROSION

Erosion of a binary image A by structure element B is defined as

$$A \ominus B = \{p \mid p + b \in A, \forall b \in B\} \tag{3.3.3.1}$$

Erosion is also called the Minkowski subtraction.

An interpretation of erosion is that we implement the origin of B at each foreground pixel in A. If every foreground pixel in the structuring element B coincides with a foreground pixel in the image, the input pixel remains as it is. Else, the input pixel is set to be background. This is equivalent to taking the intersection of the negative translations of A by B (Morse 2000).

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{3.3.3.2}$$

Erosion shrinks the foreground image and as a result it eliminates small peaks, as shown in Figure 3.3.3.1.

Dilation and erosion are two opposite concepts, since dilation of the foreground means erosion of the background.

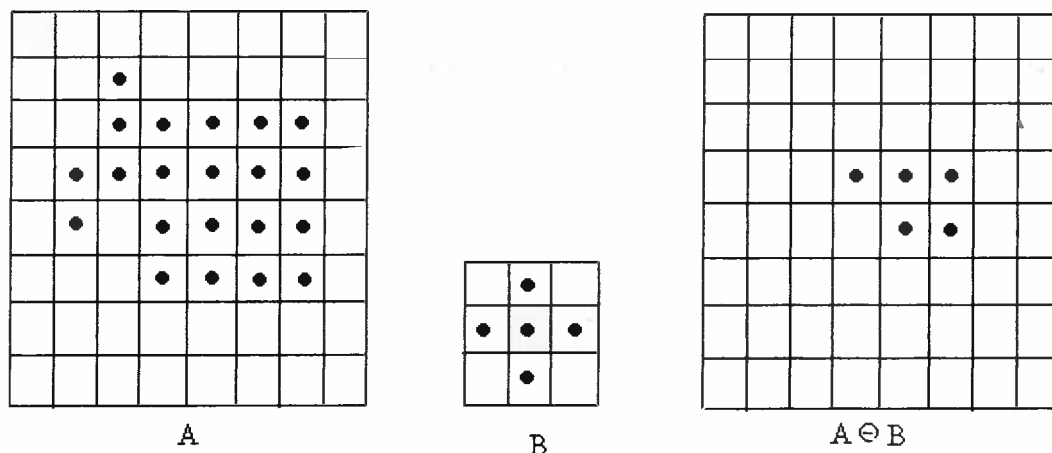


Figure 3.3.3.1: A is the input image. The gray pixel indicates the origin. B is the structuring element and $A \ominus B$ is the eroded image.

Erosion can also be used for edge detection of the image, by subtracting the eroded image from the input image. In Figure 3.3.3.2 (a) we have the eroded image by diamond structuring element with radial 3. In Figure 3.3.3.2 (b) the contour obtained by subtraction of (a) from the original image.



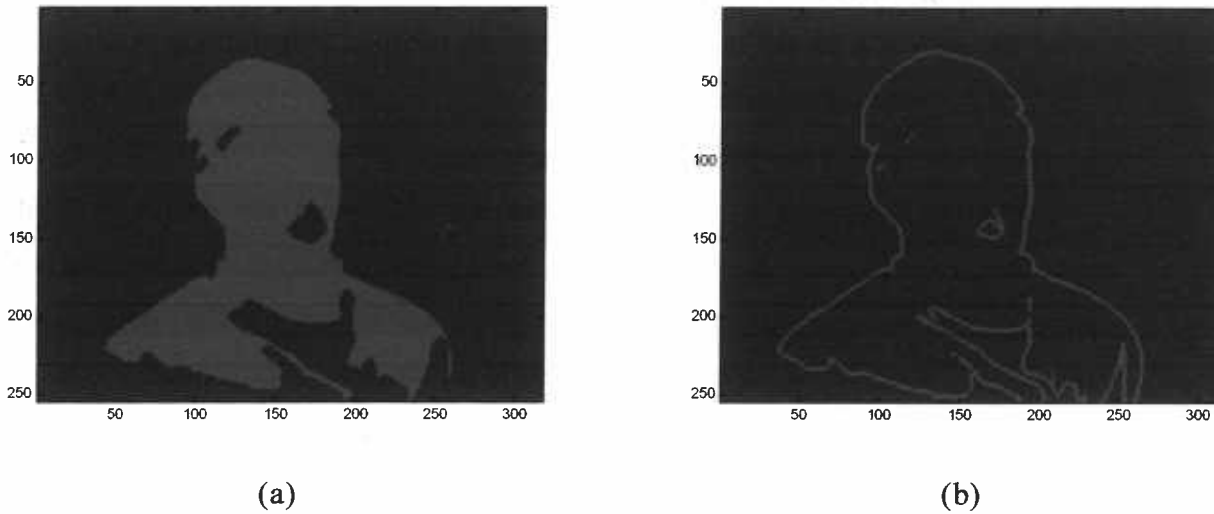


Figure 3.3.3.2: Edge detection obtained from erosion of image 3.3.2.1.1. The output of erosion is (a), while (b) is the difference of the two images. The structuring element we used is the diamond-shaped with radial equal to 3.

Properties of binary erosion (Wayne, Lin, Wei-Cheng 2000).

1. The result of the translation of A eroded by structuring element B is the same as the translation of A eroded by structuring element B.

$$(A \ominus B)_x = A_x \ominus B$$

2. If more than two dilations are to be made, the order in which they are done does not matter.

$$(A \odot B) \odot C = A \odot (B \odot C)$$

3. If A is a subset of B, then dilation of A by C is a subset of the dilation of B by C.

$$A \subseteq B \Rightarrow A \odot C \subseteq B \odot C$$

3.3.3.1 EXAMPLES OF EROSION

We use as the input image the one of Figure 3.3.2.1.1. Erosion should make the edges smoother and eliminate possible noise caused by



segmentation. Unfortunately, erosion did not give the expected results, since it suppressed a lot the foreground pixels and narrowed the image.

THE DIAMOND SHAPED STRUCTURING ELEMENT

The diamond structuring element with $R=1$ reduces the foreground pixels and as the radial value increases, the background pixels replace more skin pixels, as shown in Figure 3.3.3.1.1.

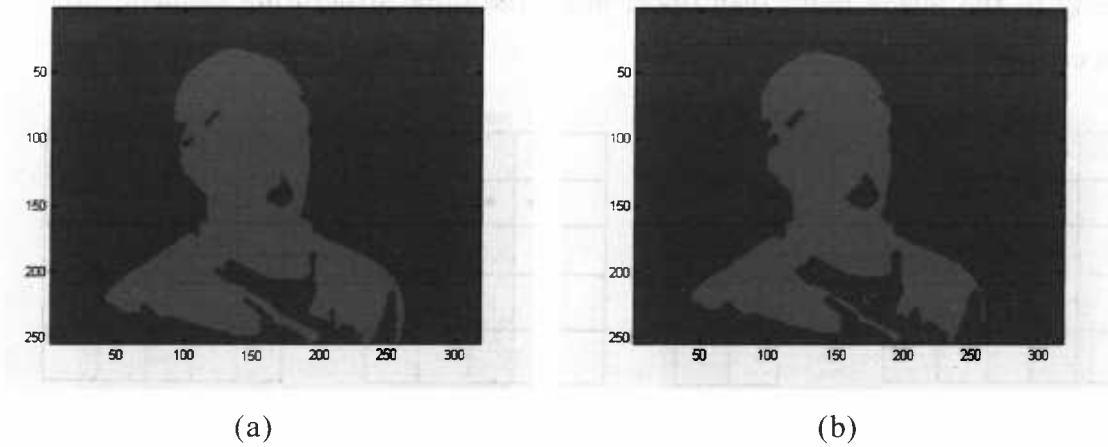


Figure 3.3.3.1.1: The radial of the structuring element is (a) $R=1$ and (b) $R=3$.

THE OCTAGONAL STRUCTURING ELEMENT

The octagon reduces the number of skin pixels causing larger gaps in the image. In (a) and (b) of Figure 3.3.3.1.2 we have used the octagonal structuring element with radial 3 and 6 respectively.

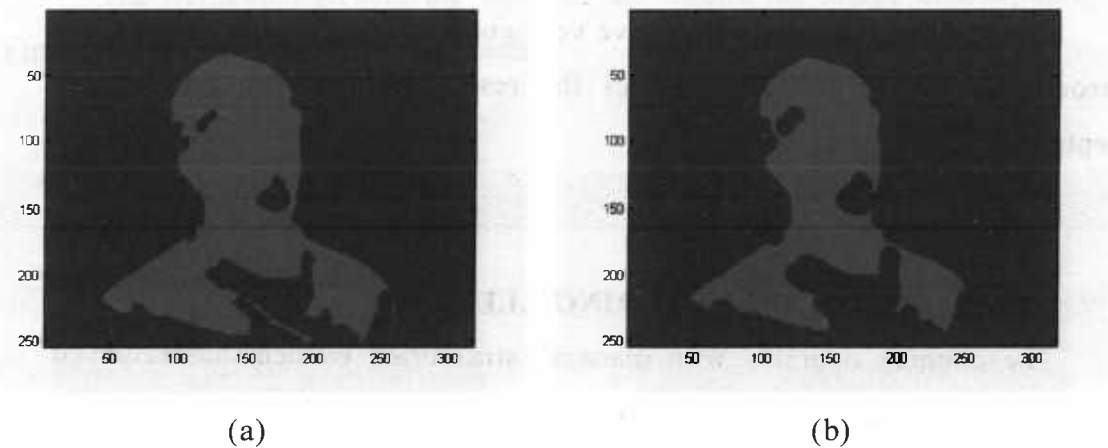


Figure 3.3.3.1.2: Erosion by octagonal structuring element with (a) $R=3$ and (b) $R=6$.



3.3.4 BINARY OPENING

Opening of a binary image A by structuring element B is defined as

$$A \circ B = (A \ominus B) \oplus B \tag{3.3.4.1}$$

Firstly, the original image is being eroded and the resulting image is being dilated. Binary opening smoothes contours, enlarges narrow gaps and eliminates thin protrusions. Any edge that is smaller than the structuring element is removed, but the rest of the image remains unchanged. If we apply opening to the image more than once, using the same structuring element, no extra changes will be made.

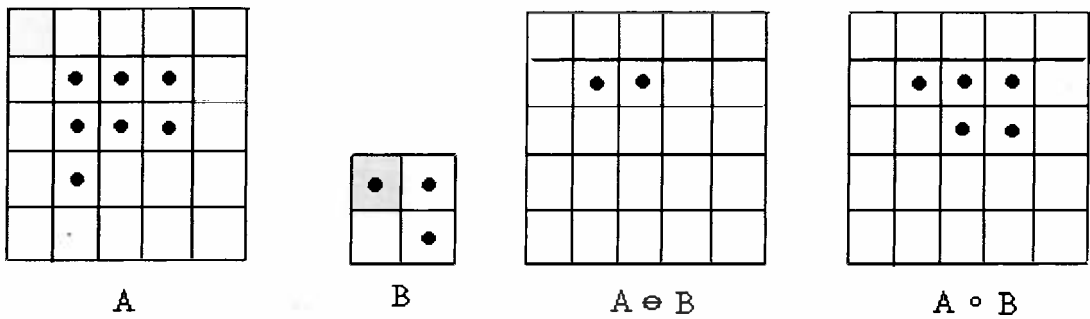


Figure 3.3.4.1: A is the input image and B the structuring element. Firstly, we perform erosion and obtain $A \ominus B$. Then we perform dilation to the eroded image and take $A \circ B$.

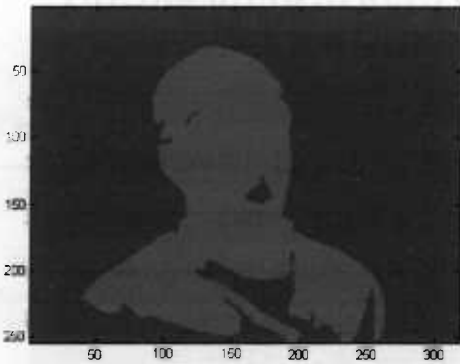
3.3.4.1 EXAMPLES OF OPENING

The opening operator didn't give very good results. That is because first it erodes the image and then dilates the result, but erosion didn't work acceptably.

THE DIAMOND STRUCTURING ELEMENT

The opening operator with diamond structuring element has removed the edges, but the image did not improve. The results for different radii are shown in Figure 3.3.4.1.1.

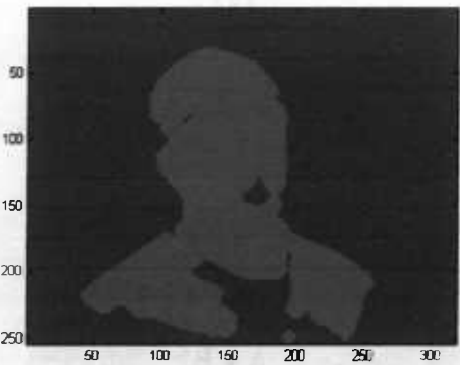




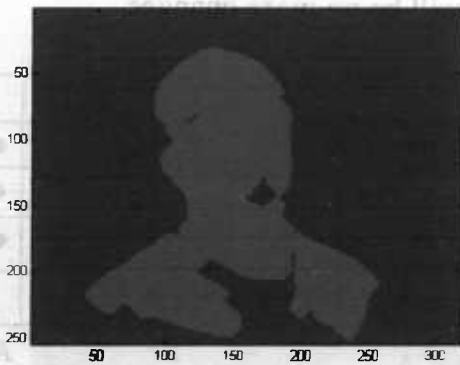
(a)



(b)



(c)



(d)

Figure 3.3.4.1.1: Opening of the input image by the diamond structuring element for different values of radial (a) $R=2$, (b) $R=3$, (c) $R=6$, (d) $R=7$.

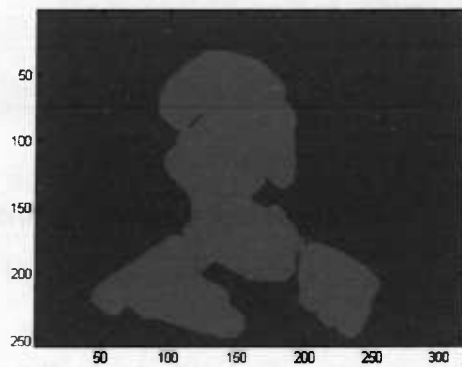
THE OCTAGONAL STRUCTURING ELEMENT

The octagonal structuring element has shrunk the image unacceptably.

Figure 3.3.4.1 shows two of the results.



(a)



(b)

Figure 3.3.4.1: Octagon with radial (a) $R=6$ and (b) $R=9$



3.3.5 BINARY CLOSING

Closing of a binary image A by structure element B is defined as

$$A \bullet B = (A \oplus B) \odot B \tag{3.3.5.1}$$

The image is being eroded and then dilated. The result of erosion is the filling of the holes in the image. The closing of a binary image includes all the points satisfying the condition that anytime the point is covered by a translation there must be some point in common between the translated structure and the original. Closing is used to fill narrow gaps, holes and small breaks. If we use closing more than once, with the same structuring element, there will be no more changes.

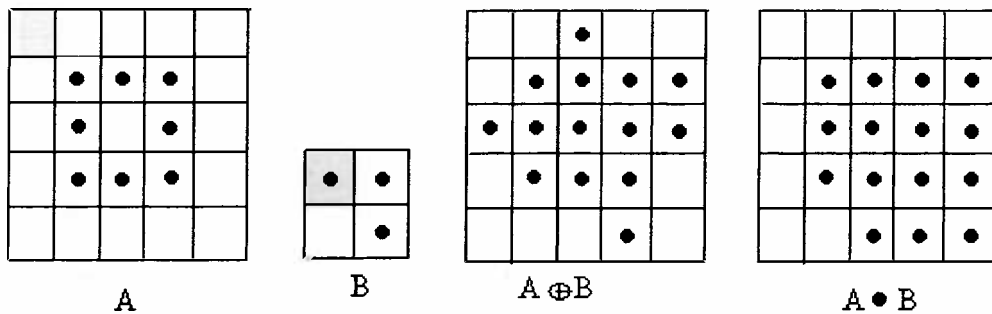


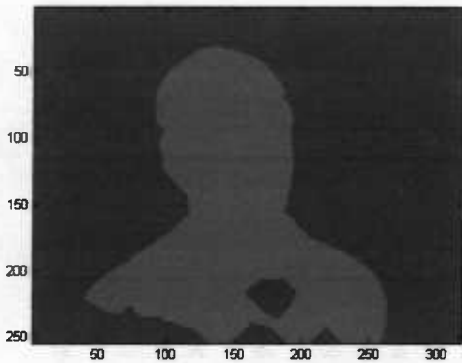
Figure 3.3.5.1: A is the image we wish to perform the closing operation and B is the structuring element. Firstly, we dilate A by B to get $A \oplus B$ and then we erode $A \oplus B$ by B and get $A \bullet B$.

3.3.5.1 EXAMPLES OF CLOSING

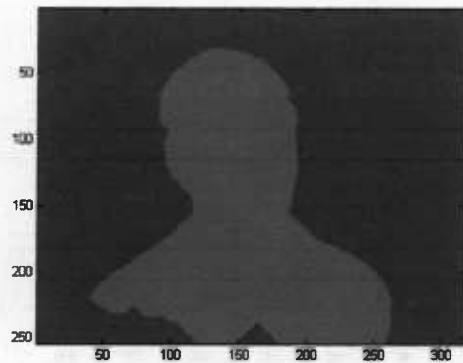
The closing operator is the most suitable for our data, since it fills most of the holes without changing the border of the image.



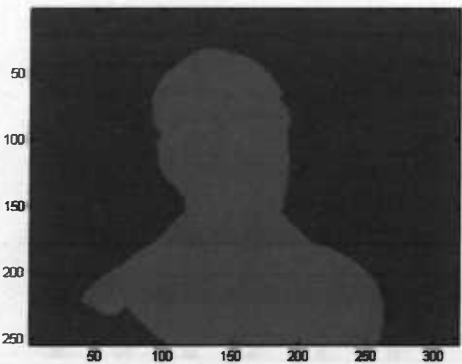
THE DIAMOND STRUCTURING ELEMENT



(a)



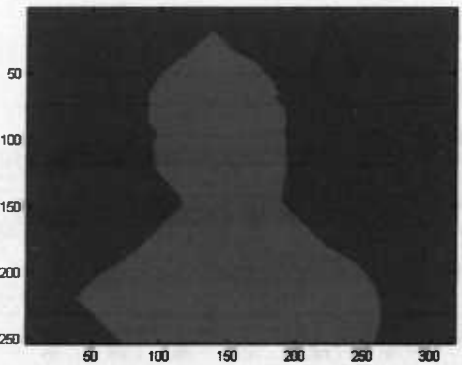
(b)



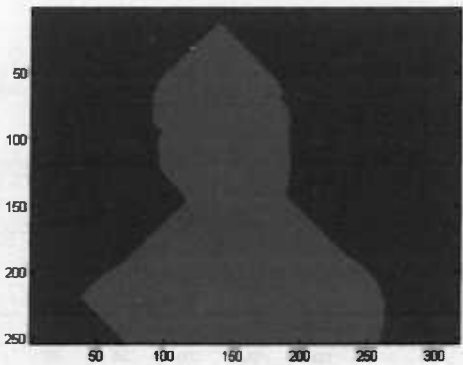
(c)



(d)



(e)



(f)

Figure 3.3.5.1.1: Diamond shaped structuring element for (a) $R=12$, (b) $R=15$, (c) $R=20$, (d) $R=25$, (e) $R=35$, (f) $R=45$.

The results are very good for $R=25$, while for $R>30$ the image is being disfeatured, as we can see in Figure 3.3.5.1.1.



THE OCTAGONAL STRUCTURING ELEMENT

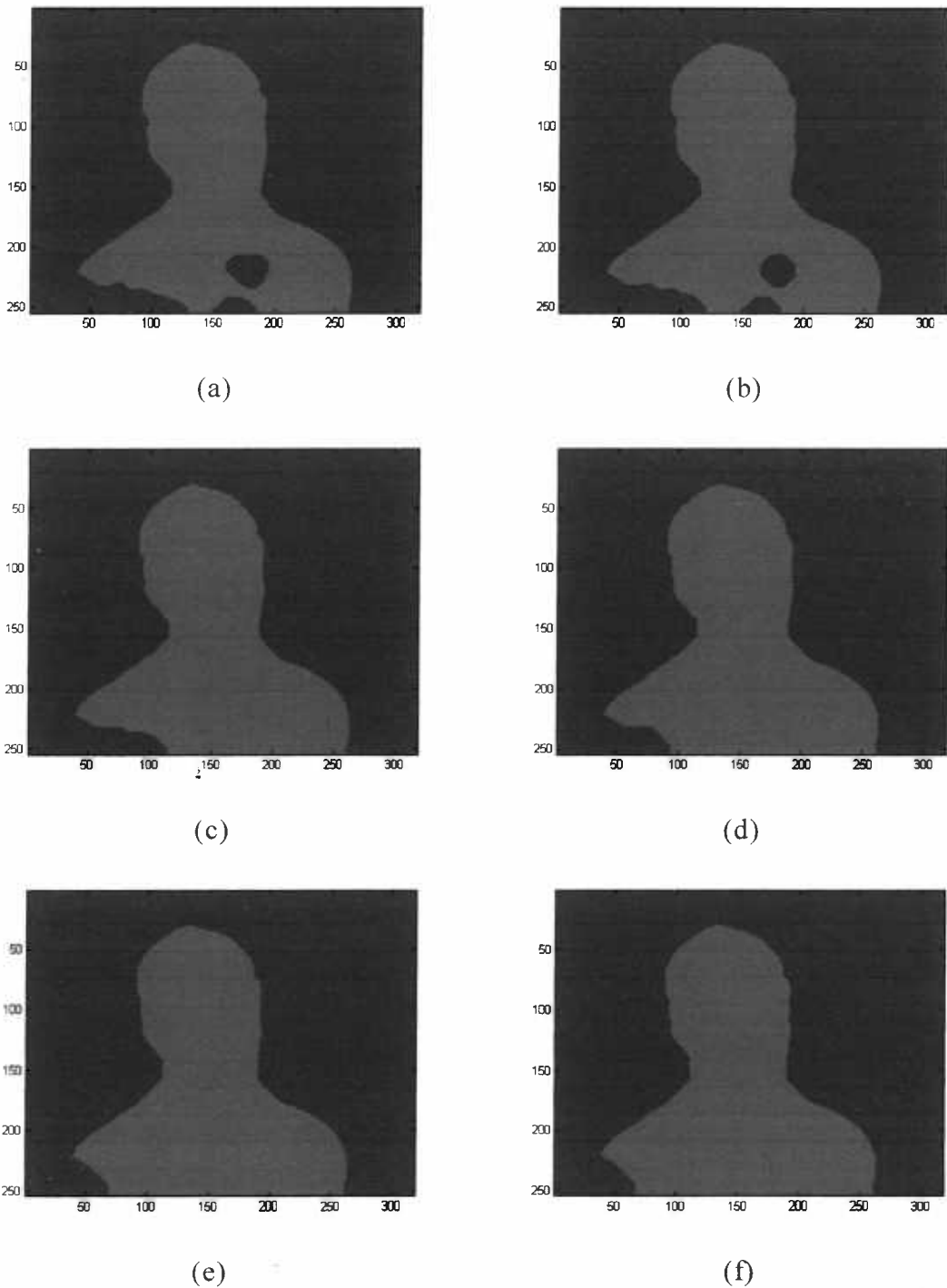


Figure 3.3.5.1.2: Octagon for (a) $R=9$, (b) $R=12$, (c) $R=15$, (d) $R=21$, (e) $R=24$, (f) $R=27$

For $R>15$ the operator eliminates the holes, without expanding the image. For $R>24$ the image spreads to the right. Those results are shown in Figure 3.3.5.1.2.



THE RECTANGULAR STRUCTURING ELEMENT

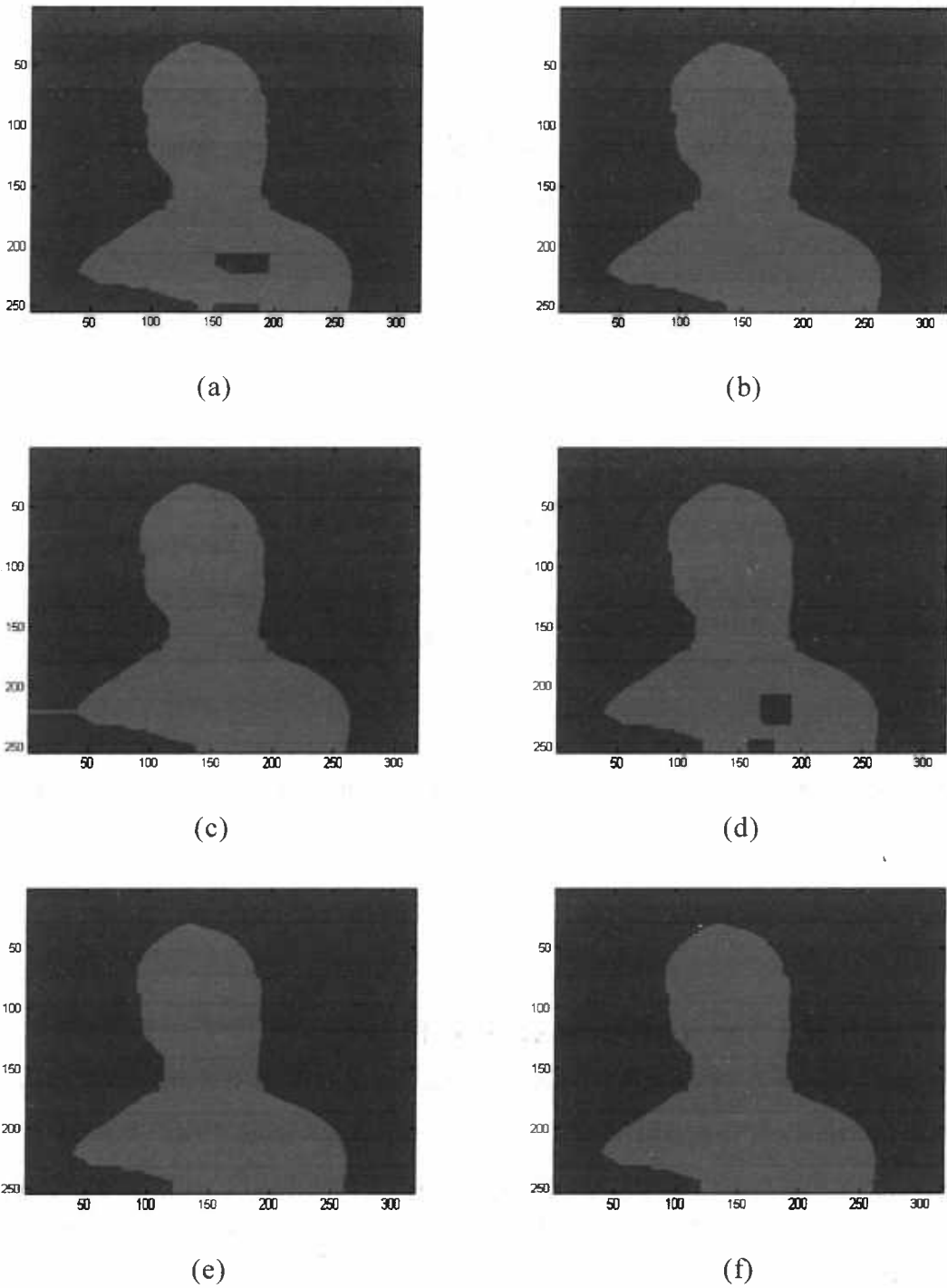


Figure 3.3.5.1.3: Rectangular with dimensions (a) $M=10$ and $N=40$, (b) $M=10$ and $N=60$, (c) $M=10$ and $N=80$, (d) $M=20$ and $N=20$, (e) $M=20$ and $N=40$, (f) $M=20$ and $N=60$



3.3.6 BOUNDARY CONDITIONS

In all the above cases we did not refer to what happens with the border pixels. In that case some pixels of the structuring element lie outside the image. In figure 3.3.6.1 we have the case where the boundary has foreground pixels and the structuring element exceeds the image. In such situations there are several methods that we can apply (Bloomberg 2002). We describe the two most common.

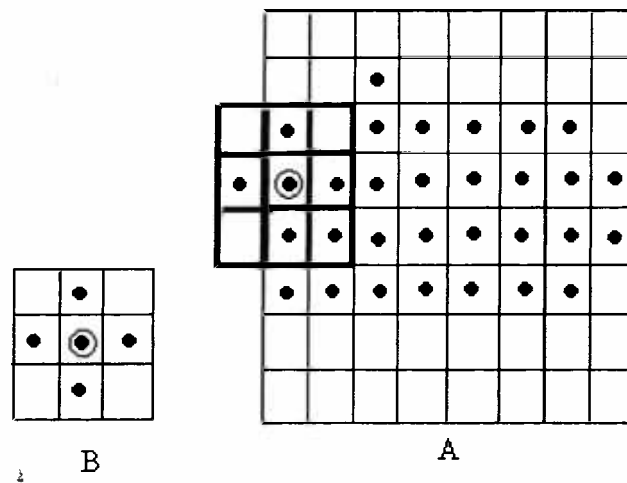
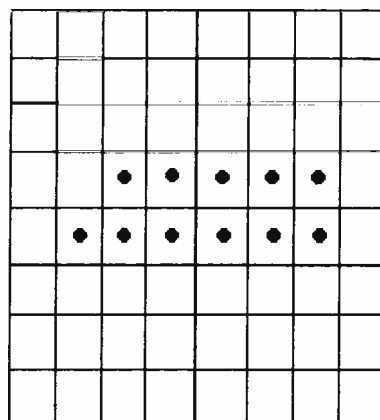


Figure 3.3.6.1: The structuring element B when applied to the boundary pixels of A exceeds the image (the structuring element is shown with bold lines).

3.3.6.1 IGNORE THE REMAINING PIXELS

When there are foreground pixels in the boundary of the image, we use only the information from the pixels of the structuring element that lays to the image. Figure 3.3.6.1.2 shows the result of the erosion of A by structuring element B in Figure 3.3.6.1.1. The erosion set the boundary pixels to background.





$A \ominus B$

Figure 3.3.6.1.2 The result of erosion A by B of Figure 3.3.6.1 when the “out of the boundary” pixels ignored

3.3.6.2 ADD EXTRA PIXELS TO THE IMAGE

Another way of dealing with the above problem is to add background or foreground pixels to the boundary, when the structuring element is at the edge. The number of the extra pixels is such that the structuring element does not exceed the new image, but we do not apply the operator to the extra pixels. Figure 3.3.6.2.1 shows the image with the extra pixels, while Figure 3.3.6.2.2 shows the erosion of A by the structuring element B of Figure 3.3.6.1.

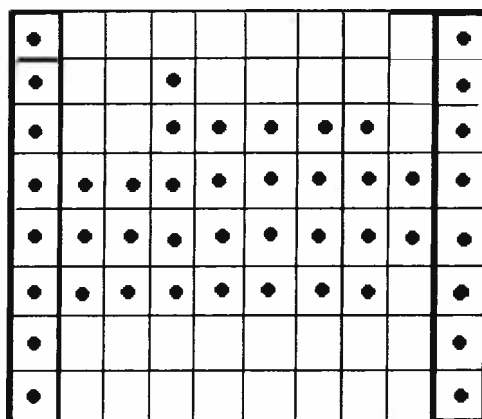
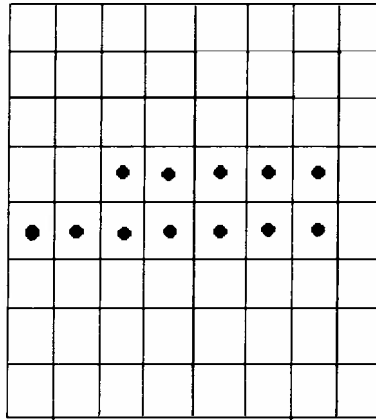


Figure 3.3.6.2.1 Adding outer pixels for performing morphological operations in the image





$$A \ominus B$$

Figure 3.3.6.2.2 The result of erosion A by B of Figure 3.3.6.1 when foreground pixels were added to the borders.



CHAPTER 4

MARKOV RANDOM FIELDS AND STOCHASTIC IMAGE MODELS

4.1 INTRODUCTION

When taking pictures from a thermal camera we do not record the true scene x , but instead we observe noisy data y . This happens because in some face areas the temperature is close enough to the environmental temperature and the segmentation process considers them to be background. The aim is to approach the real data using the spatial dependence between the pixels. Such problems can be confronted in a Bayesian framework, with the use of Markov Random Fields (MRF). In other words, we shall use the MRF processes, in order to find the maximum a-posteriori (MAP) estimate of the underlying image.

4.2 SOME BAYESIAN STATISTICS

Suppose that X is a random variable from a distribution with unknown parameter θ . We have taken a random sample x from the distribution $f(x|\theta) = P(X=x|\theta)$ and we need to draw statistical inference for the population parameter θ . The estimation of θ in Frequentist's analysis is based on the maximum likelihood principle, which is: "the values of θ that give greater probability to the observed data x are more possible than those that give x smaller probability". The fundamental difference between Bayesian and Frequentist statistics is that θ is a random variable too. This means that inference is based upon $f(\theta|x)$ and not upon $f(x|\theta)$. Within the Bayesian framework it is necessary to know the a-priori probability density $f(\theta)$,



which contains our prior beliefs about θ , before looking at the data. The distribution $f(\theta|x)$ is called a-posteriori probability density.

Bayesian statistics makes use of the Bayes theorem which says that if $f(x|\theta)$ is the likelihood of x given θ and $f(\theta)$ is the a-priori density, then:

$$f(x|\theta) = \frac{f(\theta|x) \cdot f(\theta)}{\int f(\theta|x) \cdot f(\theta) d\theta} \tag{4.2.1}$$

The Bayes theorem can be extended to more events. Suppose that B_1, B_2, \dots, B_n are events that constitute a partition of the sample space Ω such that $B_k \cap B_l = \emptyset$ for $k \neq l$ and $B_1 \cup B_2 \cup \dots \cup B_n = \Omega$. Then:

$$P(B_i | A) = \frac{P(A | B_i) \cdot P(B_i)}{\sum_{j=1}^n P(A | B_j) \cdot P(B_j)} \tag{4.2.2}$$

4.3 NEIGHBORHOOD SYSTEMS AND CLIQUES

An image is considered to be an $m \times n$ lattice of sites. When we refer to a site we mean a point or region in the Euclidean space and in order to define its position we use an ordered pair of variables (i,j) , where $1 \leq i \leq m$ and $1 \leq j \leq n$. The set of all sites is represented by S and usually we assume that all sites are ordered and $S = \{1, 2, \dots, N\}$, where $N = m \cdot n$. Each site is assigned to a random variable X_{ij} , called a label, which may be either continuous or discrete. The sites in our data are the pixels and the labels are the temperatures.

We shall make two assumptions concerning the types of spatial systems we are interested in (Besag 1974). Firstly, the position of each site will be known and their relative positions will be specific. Secondly, the observations will be available only at a single instant.

Suppose that $S = \{1, 2, \dots, N\}$ is a finite set of sites and $X = \{X_1, X_2, \dots, X_N\}$ is the respective set of labels. If the labels are independent, we can compute the joint distribution as $P(X) = \prod_{i \in S} P(X_i)$. In image analysis the labels are not independent and for each site i the conditional distribution



$P(X_i | X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_N)$ of X_i given all other site values is specified for which we need the joint distribution of all variables. For this reason we will use Markov random fields. Before introducing Markov random fields we need to define the neighboring relation between sites.

Definition 4.3.1: A site $j \neq i$ is said to be a neighbor of i and is denoted by $i \sim j$ if the functional form of $P(X_i | X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_N)$ depends on X_j .

We shall refer to the set of all sites neighboring to i with the symbol \hat{c}_i . Thus, for the site at location (i,j) if we have that $P(X_{i,j} | \text{all other site values})$ depends only on $X_{i-1,j}, X_{i+1,j}, X_{i,j-1}$ and $X_{i,j+1}$, then we have the nearest-neighbor lattice scheme (Figure 4.3.1 (a)). Similarly, we can define several neighbor systems, some of them shown in Figure 4.3.1.

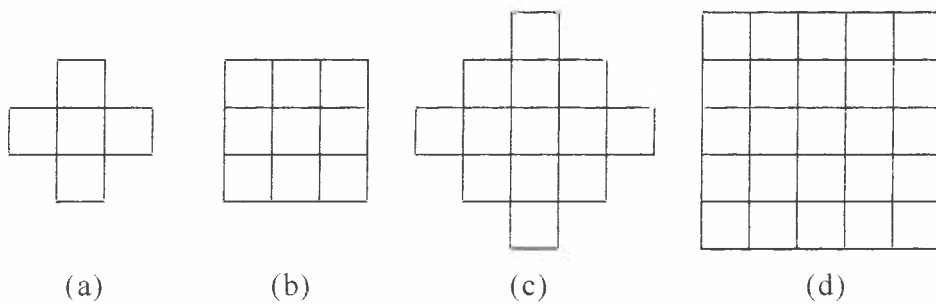


Figure 4.3.1: (a) nearest-neighbor scheme, (b) 8-neighbor lattice scheme, (c) 12-neighbor lattice scheme and (d) 24-neighbor lattice scheme.

Definition 4.3.2: A single site or a set of sites in which every site is a neighbor of every other site in the set is called a clique. C denotes the set of all cliques.

Some examples of cliques are shown in figure 4.3.2.



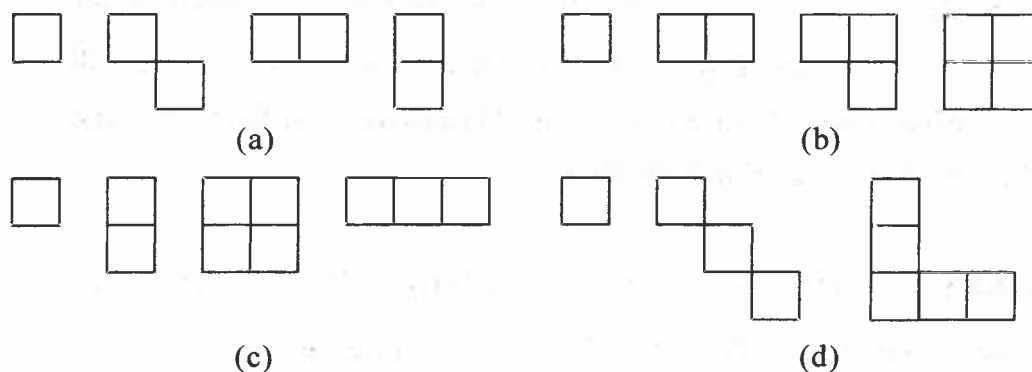


Figure 4.3.2: Some cliques for (a) the nearest-neighbor lattice scheme, (b) the 8-neighbor scheme, (c) the 12-neighbor scheme and (d) the 24-neighbor scheme.

4.4 MARKOV RANDOM FIELDS

Definition 4.4.1: Let $X = \{X_i : i \in S\}$ be a set of random variables. X is a Markov random field (MRF) if $P(X_i | \text{all other site values})$ depends only on the values of the neighbors. When for each site in location (i,j) $P(X_{i,j} | \text{all other site values})$ depends only on the values of the four, eight, twelve etc. neighbors we have respectively the first, second, third etc. order Markov Random Fields (MRF).

Definition 4.4.2: A MRF is said to be homogeneous if $P(X_i | \partial_i)$ is independent of the relative position of site i in S . In our case, we are concerned in homogeneous MRFs.

Definition 4.4.3: Let $X = \{X_1, X_2, \dots, X_n\}$ be a set of random variables from a distribution F . We say that the density $p(X)$ is a Gibbs distribution if it can be written in the form

$$P(X) = \frac{1}{Z} \cdot \exp \left\{ -\frac{1}{T} \sum_{c \in C} V_c(X) \right\}, \tag{4.4.1}$$

where Z is the normalizing constant, which is called partition function; the functions $V_c(\cdot)$ are called clique potentials and function $\sum_{c \in C} V_c(\cdot)$ is called the energy function.



The disadvantage that arises when we are using MRF is that it is difficult to compute the joint distribution and estimate parameters. The potential functions can help us to specify the joint probability of a MRF via the Hammersley-Clifford theorem (Bouman 1995):

Theorem 4.4.1: We assume that for every site there are only a finite number of labels. X is a MRF with $P(X = x) > 0 \forall x$ if and only if $P(X = x)$ has the form of the Gibbs distribution.

4.4.1 THE ISING MODEL

Suppose that for every site i in S two values are available

$$X_i = \begin{cases} 1 & \text{if } i \text{ is black} \\ 0 & \text{if } i \text{ is white} \end{cases} \tag{4.4.1.1}$$

and we assume the nearest-neighbor scheme, where a clique is a single site or double site set. We are interested to determine whether a site i is black or white conditional on the color of its four neighbors. One idea would be to count the number of black neighbors and take:

$$P(X_i=1 \mid k \text{ neighbors are black}) \propto \exp(\beta \cdot k) \tag{4.4.1.2}$$

where $\beta > 0$. If we consider the Hammersley-Clifford theorem and take:

$$V_c(x_c) = \begin{cases} -\beta & \text{if both sites in the clique } C \text{ have the same color} \\ 0 & \text{else} \end{cases} \tag{4.4.1.3}$$

we have the Ising model, where:

$$P(X) = \frac{\exp\left(\beta \sum_{i-j} I_{[x_i=x_j]}\right)}{Z(\beta)} = \frac{\exp(\beta \cdot k)}{\exp(\beta \cdot k) + \exp(\beta \cdot (N-k))} \tag{4.4.1.4}$$

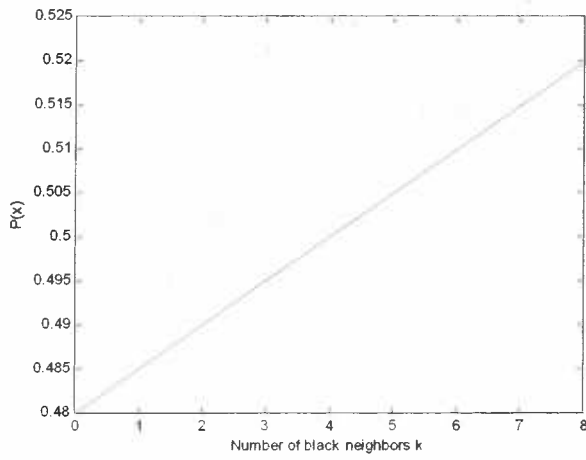
The normalizing constant

$$Z(\beta) = \sum_x \exp\left(\beta \sum_{i-j} I_{[x_i=x_j]}\right) \tag{4.4.1.5}$$

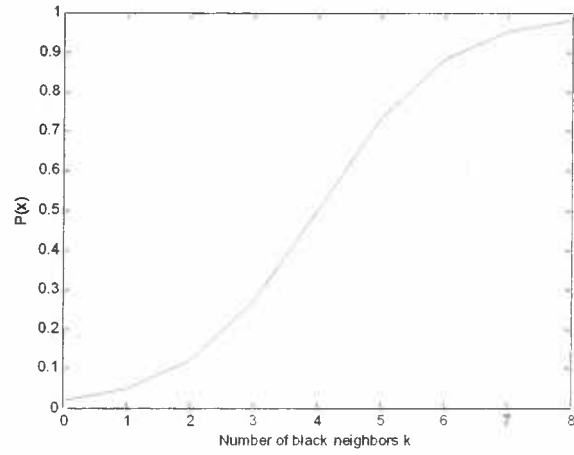
is a function of β .

In Figure 4.4.1.1 we have the plots of $P(x)$ versus the number of black neighbors k . We have assumed the 8-neighbors scheme.

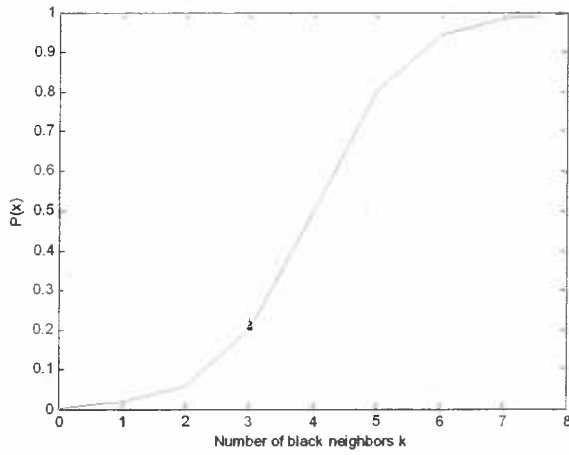




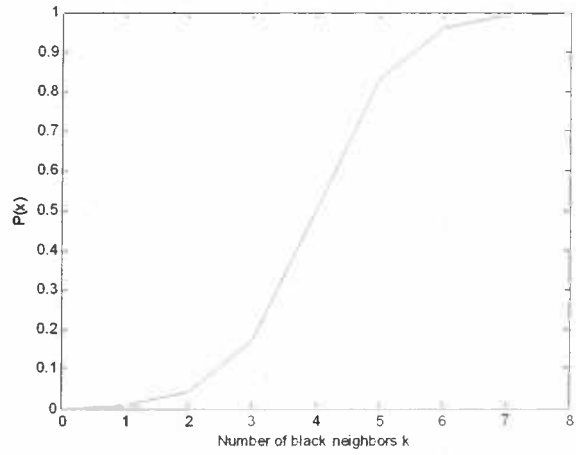
(a)



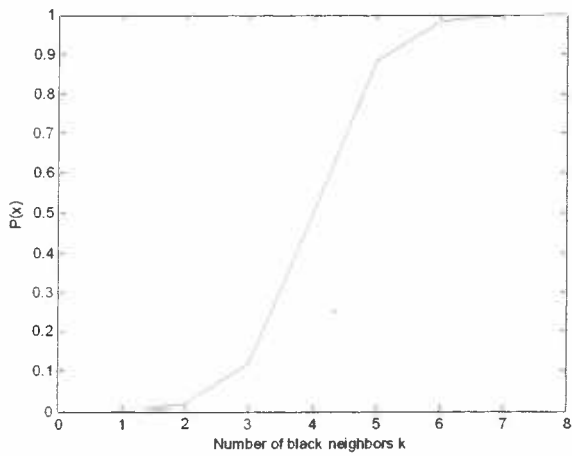
(b)



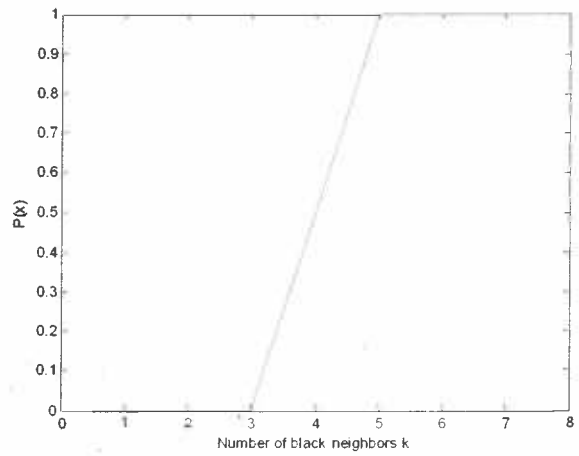
(c)



(d)



(e)



(f)

Figure 4.4.1.1: The plots of $P(x)$ versus k in the 8-neighbor scheme, for several values of β . (a) $\beta=0.01$, (b) $\beta=0.5$, (c) $\beta=0.7$, (d) $\beta=0.8$, (e) $\beta=1.0$, (f) $\beta=5.0$.



For $\beta=0.1$ $P(x)$ increases in an almost linear way. The larger values of β give probability $P(x)$ close to zero for less than 4 black neighbors and probability close to 1 for more than 4 black neighbors.

The value $\beta=\log(\sqrt{2}+1)\approx 0.881373$ is called critical value since for $\beta > \beta_{critical}$ the model changes behavior (Merilee Hurn, Oddvar Husby, Havard Rue 2001).

Consider an $m \times n$ rectangular lattice where values outside the boundary are given. Then, the effect of the boundary on the interior sites matters as $m \cdot n \rightarrow \infty$ for $\beta > \beta_{critical}$, while for $\beta < \beta_{critical}$ it does not. For more information, look into “Statistical Inference For Spatial Processes” by B.D.Ripley (1988).

4.4.2 AN APPLICATION OF THE ISING MODEL

We apply the Ising model in our data after turning the continuous image into binary, using the K-means method. For each site i we calculate the probability of being black given that it has k black neighbors. If this probability is greater than a threshold p then i is turned into black, else it gets white. The neighborhood consists of the four nearest neighbors. The probabilities for various values of β and k are in the table 4.4.2.1. For $k < 2$ the probability decreases as β increases. For $k > 2$ the probability increases as β increases, while for $k=2$ the probability is 0.5 independently of β . The results shown in Figure 4.4.2.3 are for different number of iterations M . The Ising constant that we have used is equal to 0.4 and the probability threshold is 0.35. If we chose probability threshold equal to 0.5 then in case that we have a pixel with 2 black neighbors it will be more possible to turn into white. This means that we need more than 2 black neighbors in order to turn a pixel into black and as a result we will have small improvement as regards the gap filling.



	k=1	k=2	k=3	k=4
$\beta=0.1$	0.4502	0.5000	0.5498	0.5987
$\beta=0.2$	0.4013	0.5000	0.5987	0.6900
$\beta=0.3$	0.3543	0.5000	0.6457	0.7685
$\beta=0.4$	0.3100	0.5000	0.6900	0.8320
$\beta=0.5$	0.2689	0.5000	0.7311	0.8808
$\beta=0.6$	0.2315	0.5000	0.7685	0.9168
$\beta=0.7$	0.1978	0.5000	0.8022	0.9427
$\beta=0.8$	0.1680	0.5000	0.8320	0.9608
$\beta=0.9$	0.1419	0.5000	0.8581	0.9734
$\beta=1.0$	0.1192	0.5000	0.8808	0.9820

Table 4.4.2.1: The probabilities given by the Ising model, for several values of Ising constant β and number of black neighbors k .

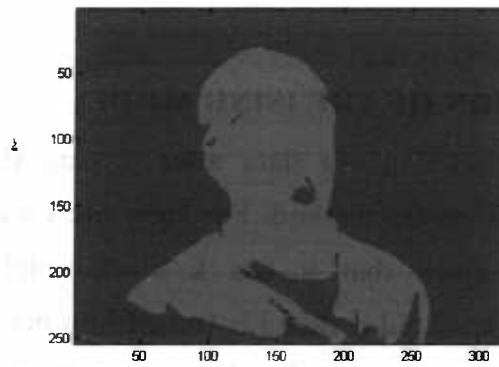


Figure 4.4.2.2: The initial binary image



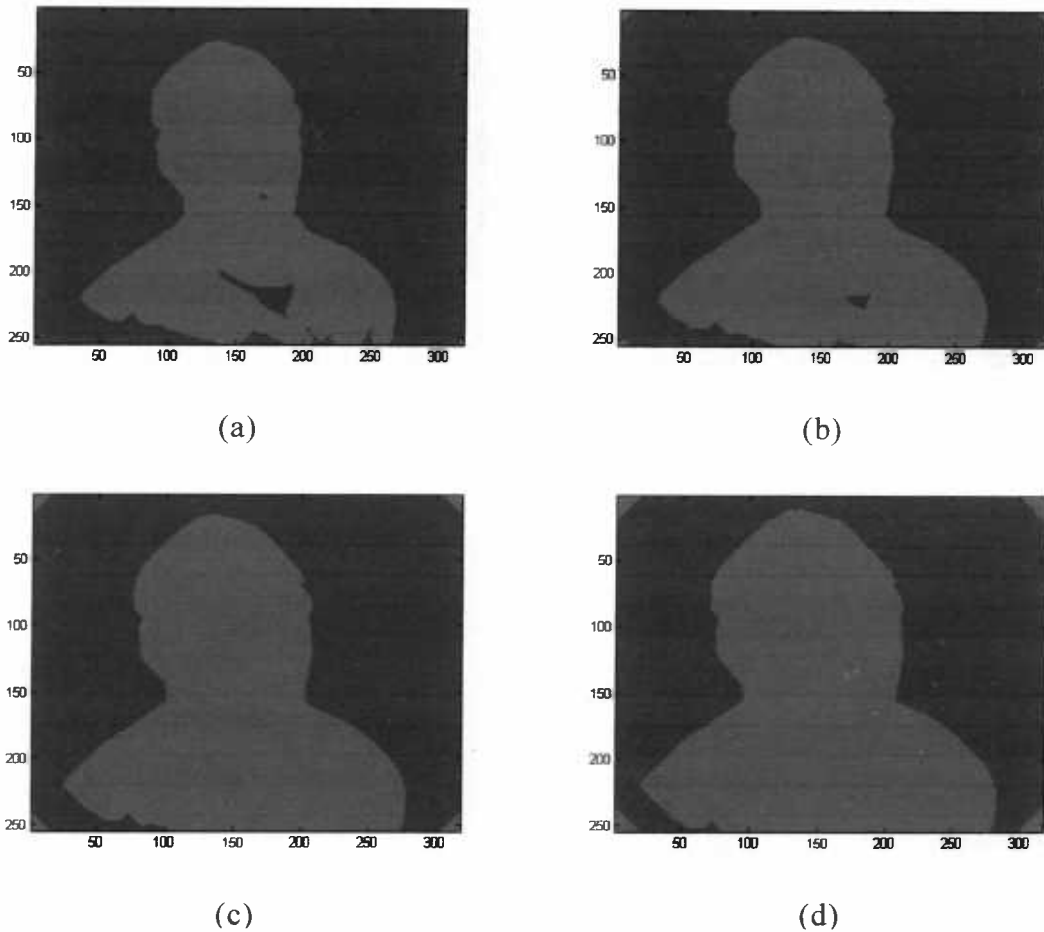


Figure 4.4.2.3: The Ising model for number of iterations: (a) $M=5$, (b) $M=10$, (c) $M=15$ and (d) $M=20$.

As the number of iterations increases the picture expands. This happens because we use the threshold 0.35 which means that we need at least two black neighbors to make a pixel black. The lineament pixels of the face have in general more than one black neighbor. As a result, at each iteration, there have been created more black pixels (for each iteration we are based on the previous iteration's result).

It is necessary to decide an appropriate number of iterations in order to avoid misshaping the image. This could be done using a terminating condition, for example, stopping when a predetermined ratio R of pixels has changed status. The results of this approach appear in Figure 4.4.2.4.



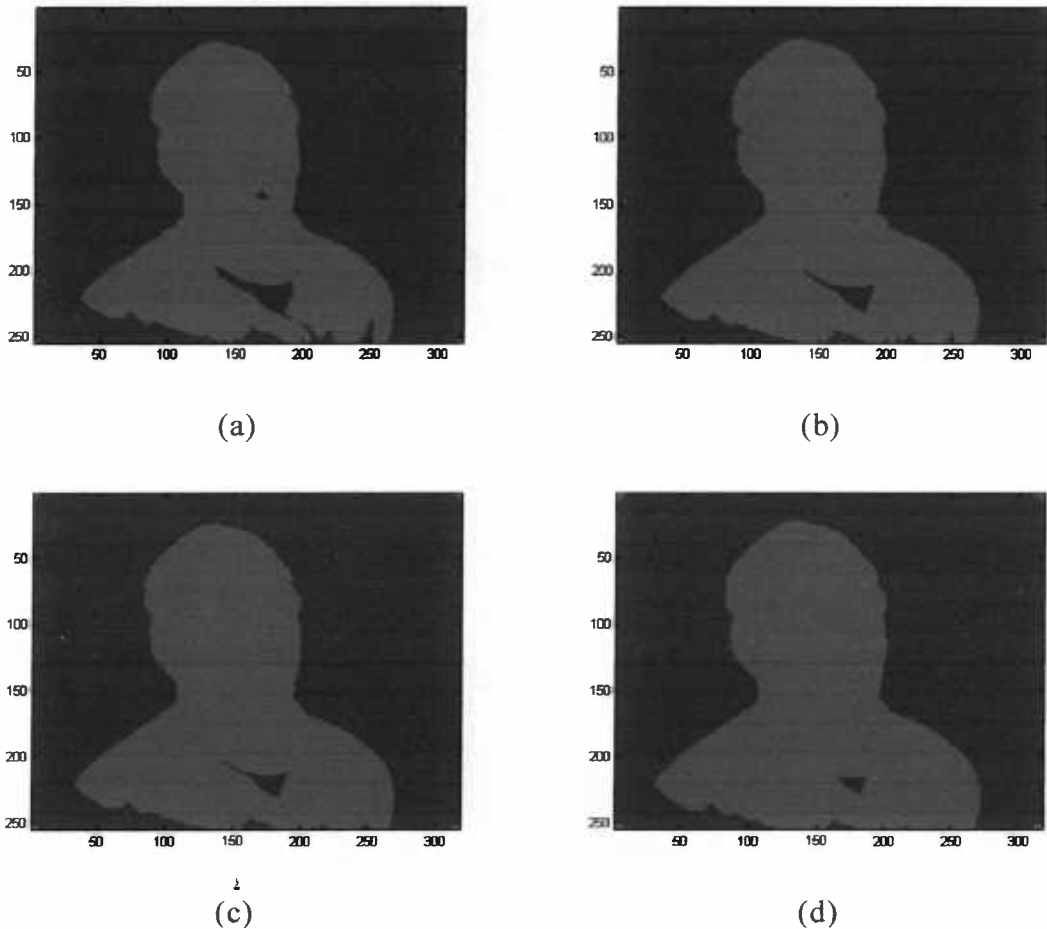


Figure 4.4.2.4: Iterative applications of the Ising model. The algorithm terminates when at least a ratio R of the total number of pixels has changed value. The respective ratio numbers are: (a) $R=0.025$, (b) $R=0.075$, (c) $R=0.085$ and (d) $R=0.10$.

The results for $R=0.075$ and $R=0.085$ are satisfying. For smaller values of R we have little change, while for greater values of R we have expansion of the image. Since the results for $R=0.85$ are quite good, we apply the same algorithm to the data, but this time we are trying different values of the Ising constant β .



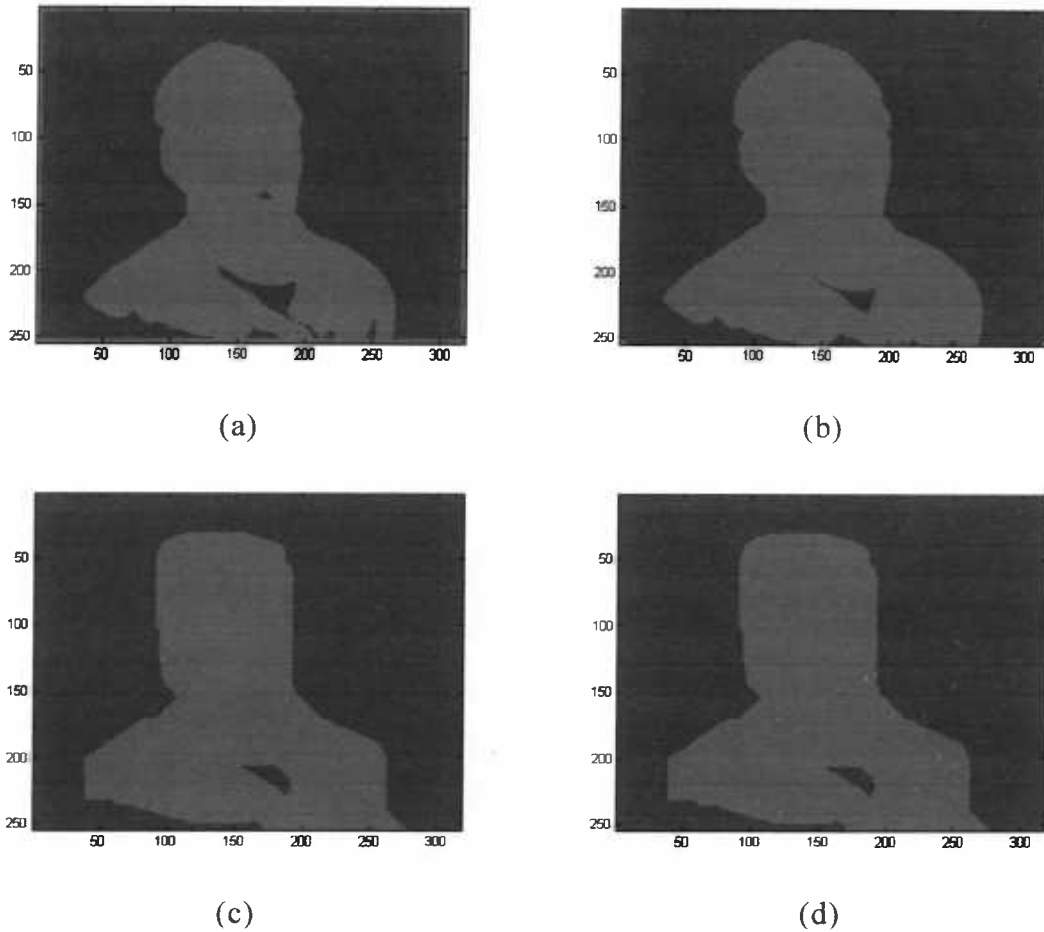


Figure 4.4.2.5: The Ising model for $R=0.85$ and different values of beta: (a) $\beta=0.3$, (b) $\beta=0.5$, (c) $\beta=0.7$ and (d) $\beta=0.9$.

In Figure 4.4.2.5 (a) it doesn't matter that the pixels in the boundary have turned into black. This happened because of the small value of β . If for example a pixel in the boundary has three white neighbors, then:

$$P(X_i=1 \mid 0 \text{ neighbors are black}) = \frac{\exp(0 \cdot 0.3)}{\exp(0 \cdot 0.3) + \exp(3 \cdot 0.3)} = \frac{1}{1 + \exp(0.9)} = 0.2891.$$

According to the algorithm we have used, a pixel becomes black if $P(X_i=1 \mid k \text{ neighbors are black})$ is greater than 0.35. For sensitivity purpose we carry out the same algorithm but this time we turn a pixel into black if $P(X_i=1 \mid k \text{ neighbors are black})$ is greater than 0.3. The results are shown in Figure 4.4.2.6.



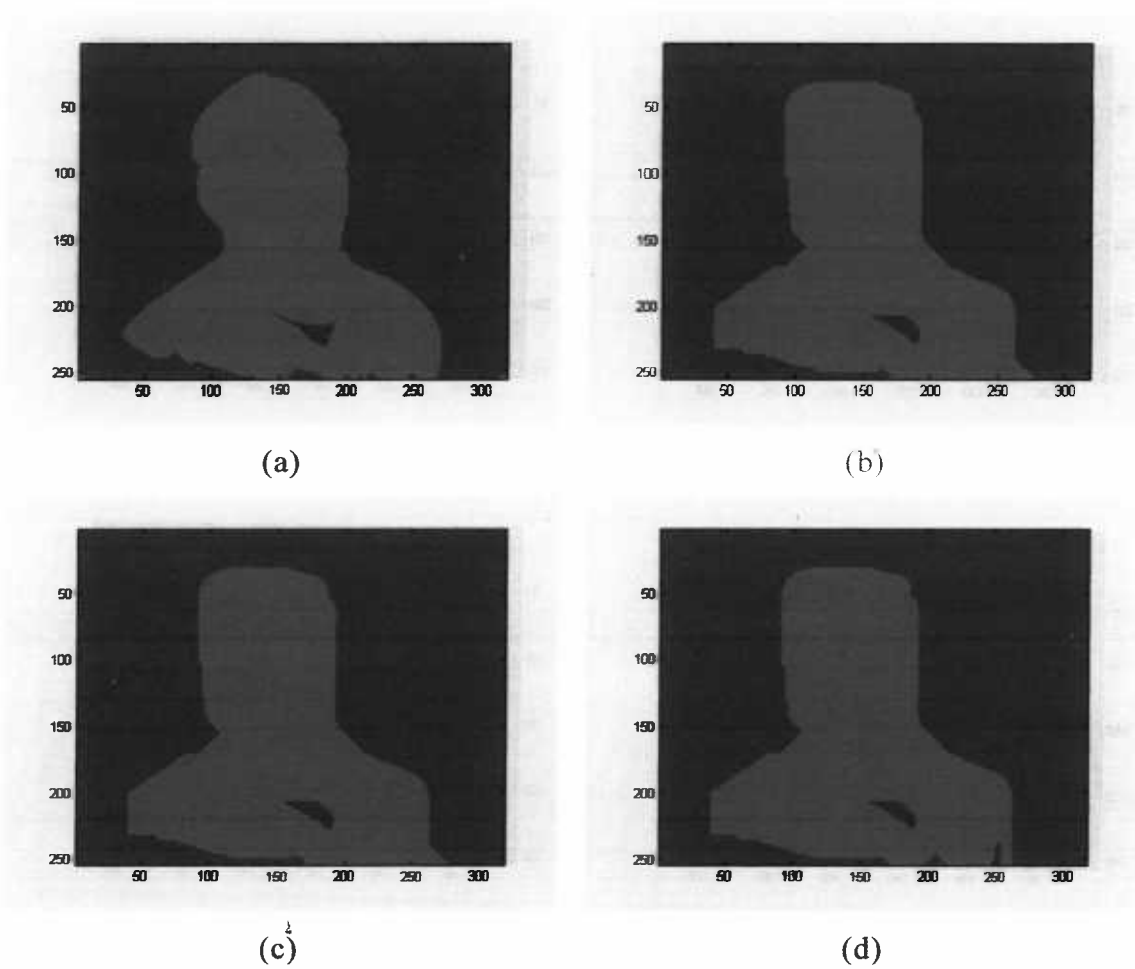


Figure 4.4.2.6: The Ising model for $R=0.85$ and different values of beta: (a) $\beta=0.3$, (b) $\beta=0.5$, (c) $\beta=0.7$, (d) $\beta=0.9$



CHAPTER 5

MARKOV CHAIN MONTE CARLO METHODS

5.1 INTRODUCTION

Suppose we are given a density function p on some state space \mathbf{X} , which may not be normalized, but it satisfies $0 < \int_{\mathbf{X}} p < \infty$. This density corresponds to a probability measure $p(\cdot)$ on \mathbf{X} , by

$$p(A) = \frac{\int_A p(x) dx}{\int_{\mathbf{X}} p(x) dx} \quad (A \subseteq \mathbf{X}). \quad (5.3.1)$$

If we want to estimate expectations of functions $f: \mathbf{X} \rightarrow \mathbb{R}$ with respect

to $p(\cdot)$, i.e. $p(f) = E[f(x)] = \frac{\int_{\mathbf{X}} f(x)p(x)dx}{\int_{\mathbf{X}} p(x)dx}$ and p has a complicated form or

\mathbf{X} is high-dimensional, then it is very difficult to compute the integrals.

The solution is to construct a Markov chain $\{X_n\}$ on \mathbf{X} , which has $p(\cdot)$ as a stationary distribution. Then for large values of n the distribution of X_n will be approximately stationary.

The construction of a Markov chain with the desired properties is not difficult. The difficulty is to reach the stationary distribution in a short time.

MCMC methods are very useful tools for sampling from unknown distributions. Before describing the MCMC algorithms we introduce some basic concepts of the Markov chains.



5.2 MARKOV CHAINS

Definition 5.2.1: A sequence of random variables $\{X_1, X_2, X_3, \dots\}$ is called a Markov chain if it has the Markov property, namely that

$$P(X_{n+1} = x | X_n = x_n, \dots, X_1 = x_1, X_0 = x_0) = P(X_{n+1} = x | X_n = x_n) \quad (5.2.1)$$

This means that the probability distribution of every state depends only on the previous state. The set S of all possible states is called state space and is a countable set.

Definition 5.2.2: Transition probability or transition kernel of a Markov chain is the probability

$$p_{ij} = P(X_{n+1} = s_j | X_n = s_i) \quad (5.2.2)$$

Transition probabilities declare the probability that we move from state x_i to state x_j at a single step. The probability of moving from state x_i to state x_j at $n+1$ steps is given by the **Chapman-Kolmogorov** equations:

$$\begin{aligned} p_i(n+1) &= P(X_{n+1} = s_i) \\ &= \sum_k P(X_{n+1} = s_i | X_n = s_k) \cdot P(X_n = s_k) \\ &= \sum_k p_{ik} \cdot p_k(n) \end{aligned} \quad (5.2.3)$$

where

$$p_k(n) = P(X_n = s_k) \quad (5.2.4)$$

Definition 5.2.3: A Markov chain is called stationary if

$$P(X_{n+1} = x | X_n = y) = P(X_n = x | X_{n-1} = y) \quad \text{for all } n \quad (5.2.5)$$

Definition 5.2.4: If a Markov chain is stationary with time-independent transition probabilities p_{ij} , then $\Pi = (\pi_1, \pi_2, \pi_3, \dots)$ is a stationary distribution if

$$\sum \pi_i = 1 \quad \text{and} \quad \pi_j = \sum_{i \in S} \pi_i \cdot p_{ij} \quad (5.2.6)$$

Definition 5.2.5: A Markov chain is called reversible with respect to $\pi(\cdot)$, if

$$\pi_i p_{i,j} = \pi_j p_{j,i} \quad (5.2.7)$$



Theorem 5.2.1: A Markov chain converges to a stationary distribution regardless of where it begins (Sahu 2002).

Theorem 5.2.2: If a Markov chain is reversible with respect to $\pi(\cdot)$, then $\pi(\cdot)$ is stationary for the chain. The proof is included in Roberts and Rosenthal (2004).

5.3 MARKOV CHAIN MONTE CARLO METHODS

Consider the noisy image y that has been derived by segmentation. We wish to improve it so that it gets closer to the real but unknown one x . It would be very useful if we could have the mathematical form of the posterior distribution $p(x|y)$. This is computationally difficult, since the posterior function requires the integration of high dimensional functions and for that reason we shall use Markov chain Monte Carlo (MCMC) methods. We introduce two MCMC algorithms, the Gibbs sampler (Geman, and Geman 1984) and the Metropolis-Hastings (Hastings 1970) sampler, which we use to sample from the posterior distribution of our data.

5.3.1 THE GIBBS SAMPLER

The steps of the algorithm are given bellow:

We start from any initial values $x_1^{(0)}, x_2^{(0)}, \dots, x_N^{(0)}$. At first iteration we generate the values

$$\begin{aligned} x_1^{(1)} &\sim P(x_1|x_2^{(0)}, x_3^{(0)}, \dots, x_N^{(0)}) \\ x_2^{(1)} &\sim P(x_2|x_1^{(1)}, x_3^{(0)}, \dots, x_N^{(0)}) \\ x_3^{(1)} &\sim P(x_3|x_1^{(1)}, x_2^{(1)}, x_4^{(0)}, \dots, x_N^{(0)}) \\ &\vdots \\ x_N^{(1)} &\sim P(x_n|x_1^{(1)}, x_2^{(1)}, \dots, x_{N-1}^{(1)}) \end{aligned}$$

The samples have autocorrelation, but as the procedure is being repeated, the autocorrelation is being reduced. The Gibbs sampler produces a



Markov chain $\{X_n\}$ which is reversible (Walsh 2004). This means that after k iterations we obtain $X_1^{(k)}, X_2^{(k)}, \dots, X_n^{(k)}$ and as $k \rightarrow \infty$

$$(X_1^{(k)}, X_2^{(k)}, \dots, X_n^{(k)}) \rightarrow (X_1, X_2, \dots, X_n) \sim P(X_1, X_2, \dots, X_n)$$

5.3.1.1 THE GIBBS SAMPLER FOR THE ISING MODEL

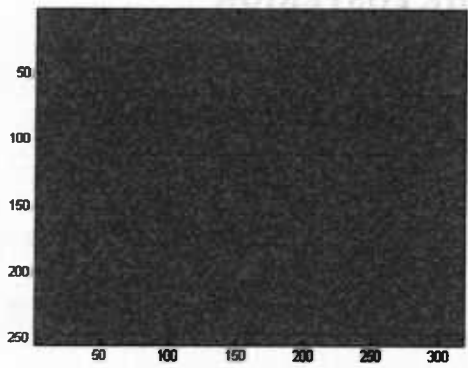
We assume the Ising model, which means that the value corresponding to each site depends only on the values of its four nearest neighbors and we will sample from $P(X_i|\partial_i)$, where ∂_i is the set of the neighbors of i . The algorithm (Hurn, Husby and Rue 2001) in that case is:

1. Initially give to each pixel i value 0 or 1 randomly.
2. For each site i :
 - a. Count the number of black neighbors denoted by b_i and the number of white neighbors denoted by w_i .
 - b. Calculate for each pixel the probability

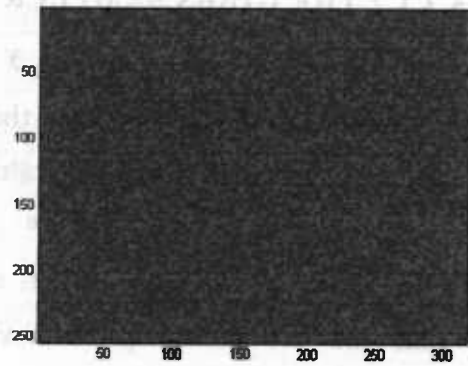
$$p_i = \frac{\exp(\beta \cdot b_i)}{\exp(\beta \cdot w_i) + \exp(\beta \cdot b_i)}$$
 - c. Generate a random value $U \sim \text{Uniform}(0,1)$
 - d. If $p_i > U$ then $x_i = 1$, else $x_i = 0$.
3. Repeat k times steps 1 and 2.

Before using the Gibbs sampler for the posterior, we give an example of simulation from the Ising model. In the next section we will add to the model the information we have observed.

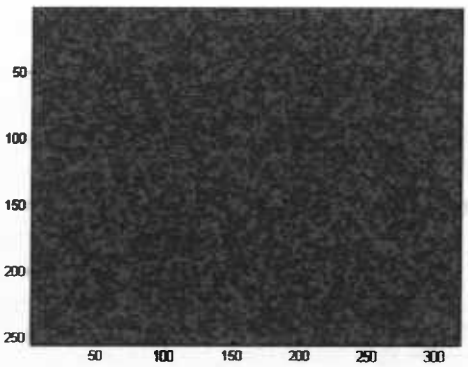




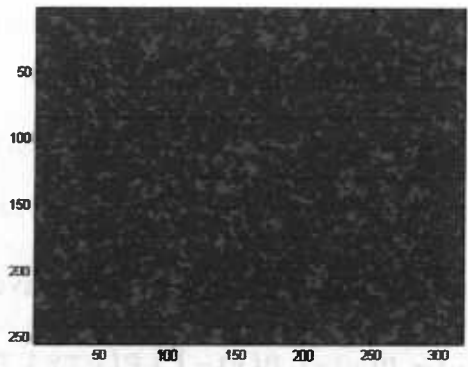
(a)



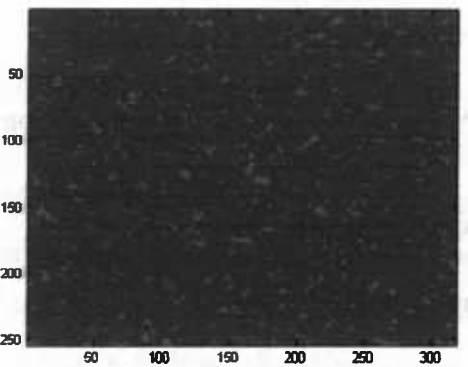
(b)



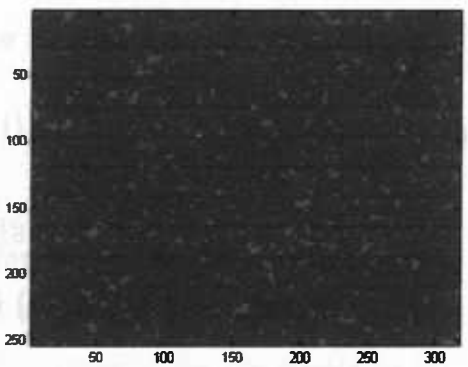
(c)



(d)



(e)



(f)

Figure 5.3.1.1.1: Simulation from the Ising model using Gibbs sampler (20 iterations) for different values of beta constant. (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.8$, (e) $\beta=0.9$, (f) $\beta=1.0$

5.3.1.2 THE GIBBS SAMPLER FOR THE POSTERIOR

Consider $\mathbf{x} = (x_1, x_2, \dots, x_N)$ and $\mathbf{y} = (y_1, y_2, \dots, y_N)$ to be the real and the recorded scene respectively (i.e. \mathbf{x} is the vector of the real pixel values, while \mathbf{y} is the vector of the observed pixel values). Then, we may write

$$\mathbf{y} = \mathbf{x} + \mathbf{e} \tag{5.3.1.2.1}$$

Where \mathbf{e} is the noise. We assume that \mathbf{e} is a flip noise or a binary noise. This means that the probability p of having recorded the wrong image is a Bernoulli probability. In other words p shows the rate of misclassification. By assuming that p is constant and that each pixel is recorded independently, we have

$$P(y_i | x_i) = \begin{cases} 1-p & \text{if } y_i = x_i \\ p & \text{else} \end{cases} \tag{5.3.1.2.2}$$

If we take $p = \frac{1}{2}$, it means that we have no information about \mathbf{x} in \mathbf{y} .

Regarding the posterior, we have

$$P(\mathbf{x} | \mathbf{y}) \propto P(\mathbf{y} | \mathbf{x}) \cdot P(\mathbf{x}) = \prod_{i=1}^N P(y_i | x_i) \cdot P(x_i), \tag{5.3.1.2.3}$$

where $P(x_i)$ is the a-priori density for every point x_i ,

If k ($1 \leq k \leq N$) is the number of sites where $y_i = x_i$, then

$$\begin{aligned} \prod_{i=1}^N P(y_i | x_i) &= (1-p)^k \times p^{N-k} = \exp\left(\log\left((1-p)^k \cdot p^{N-k}\right)\right) = \exp\left(k \cdot \log(1-p) + (N-k) \cdot \log(p)\right) \\ &= \exp\left(k \cdot \log\left(\frac{1-p}{p}\right) + N \cdot \log(p)\right) = \exp(N \cdot \log(p)) \cdot \exp\left(k \cdot \log\left(\frac{1-p}{p}\right)\right) \end{aligned}$$

Since $\exp(N \cdot \log(p))$ is constant, we have that

$$\prod_{i=1}^N P(y_i | x_i) \propto \exp\left(k \cdot \log\left(\frac{1-p}{p}\right)\right).$$

Moreover, if we assume Ising model for the true data, we have:

$$\prod_{i=1}^N P(x_i) = \exp\left(\beta \cdot \sum_{i-j} I_{[x_i=y_j]}\right).$$

In other words,

$$P(\mathbf{x} | \mathbf{y}) \propto \exp\left(k \cdot \log\left(\frac{1-p}{p}\right)\right) \cdot \exp\left(\beta \cdot \sum_{i-j} I_{[x_i=y_j]}\right)$$



$$P(x|y) \propto \exp\left(\sum_{i=1}^N I_{[x_i=y_i]} \cdot \log\left(\frac{1-p}{p}\right) + \beta \cdot \sum_{i \sim j} I_{[x_i=y_i]}\right)$$

$$P(x|y) \propto \exp\left(\sum_{i=1}^N h_i(x_i, y_i) + \beta \cdot \sum_{i \sim j} I_{[x_i=y_i]}\right) \quad (5.3.1.2.4)$$

Where

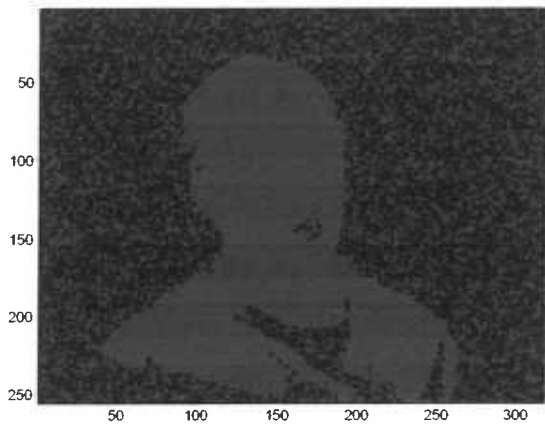
$$h_i(x_i, y_i) = I_{[x_i=y_i]} \cdot \log\left(\frac{1-p}{p}\right) \quad (5.3.1.2.5)$$

Considering the equations (5.3.1.2.4) and (5.3.1.2.5), the Gibbs sampler for the noisy data (Hurn, Husby and Rue 2001) can be written:

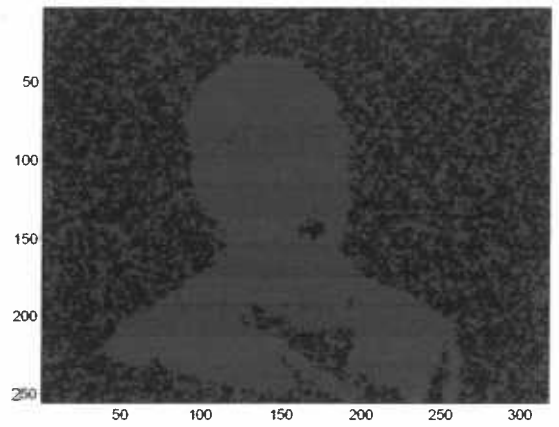
1. Initially give to each pixel i value 0 or 1 randomly.
2. For each site i :
 - a. Count the number of black neighbors denoted by b_i and the number of white neighbors denoted by w_i .
 - b. Calculate for each pixel the probability

$$p_i = \frac{\exp(\beta \cdot b_i + h_i(x_i, y_i))}{\exp(\beta \cdot w_i) + \exp(\beta \cdot b_i)}$$
 - c. Generate a random value $U \sim \text{Uniform}(0,1)$
 - d. If $p_i > U$ then $x_i = 1$, else $x_i = 0$.
3. Repeat for k iterations steps 1 and 2.

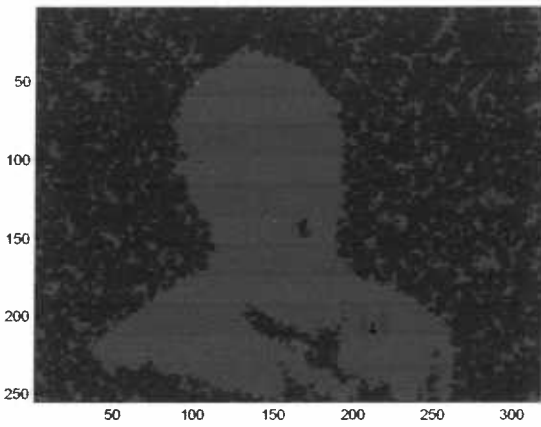
Consider the binary image we have used in all previous applications, which is shown in Figure 5.3.1.2.3. We will use the information that we have from this recorded image y in order to approach the true underlying image x . We assume that x follows the Ising model. The Gibbs sampler for the posterior for different values of Ising constant β and different probabilities p has given the results shown in Figures 5.3.1.2.1 and 5.3.1.2.2. We have performed the algorithm for 20 iterations.



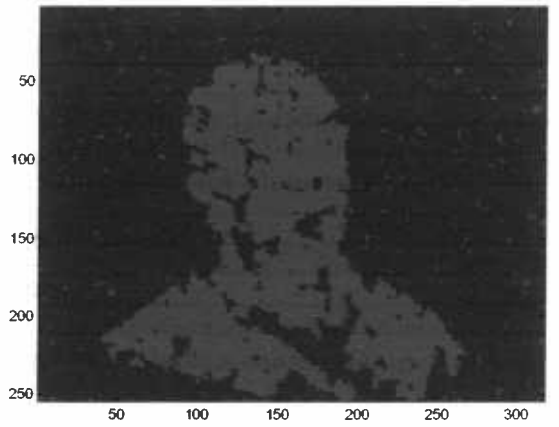
(a)



(b)



(c)

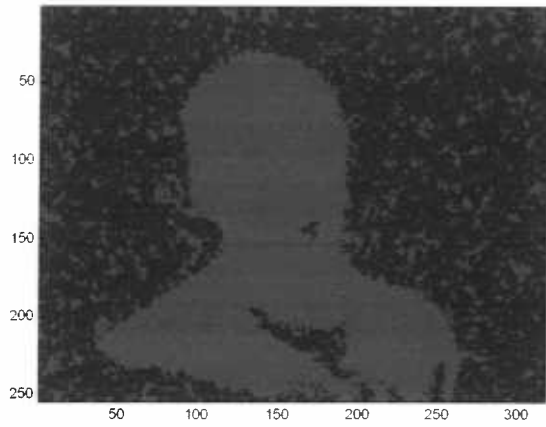


(d)

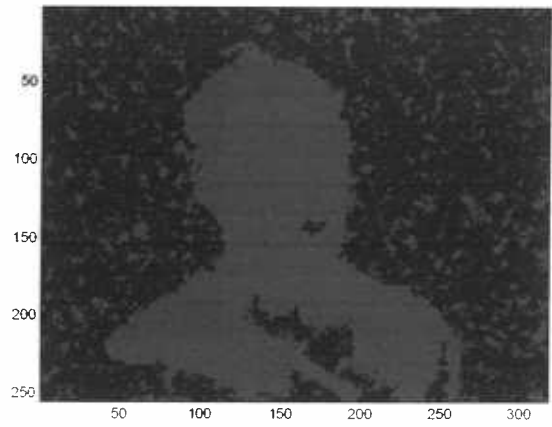
Figure 5.3.1.2.1: The results of Gibbs sampler for the posterior distribution with beta values: (a) $\beta=0.4$, (b) $\beta=0.6$, (c) $\beta=0.8$, (d) $\beta=1.0$.

The results show that a lot of noise has been added. This might mean that the underlying model for the true image \mathbf{x} is not the Ising model.

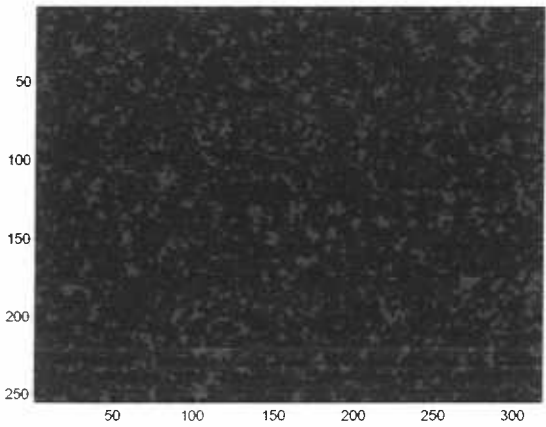




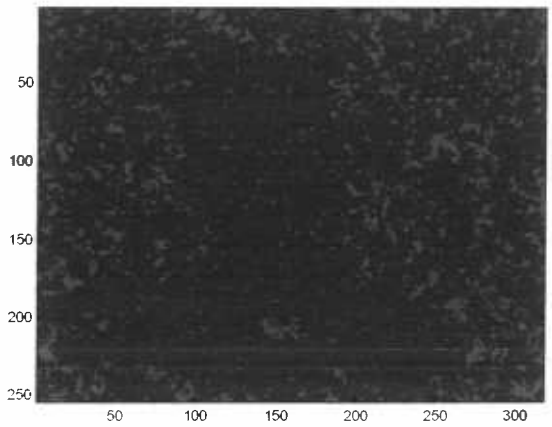
$p=0.1$



$p=0.4$



$p=0.5$



$p=0.8$

Figure 5.3.1.2.2: The Gibbs sampler for the posterior for different values of the probability p that we have observed the wrong image: (a) $p=0.1$, (b) $p=0.4$, (c) $p=0.5$, (d) $p=0.8$

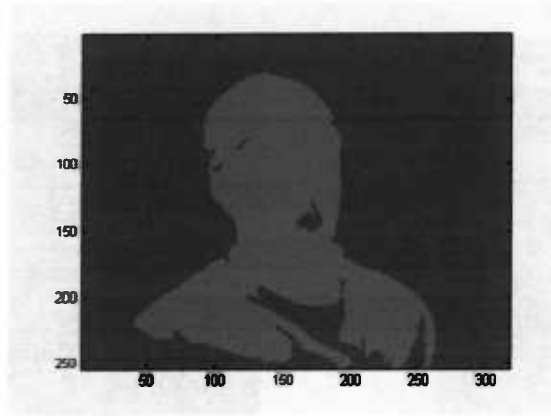


Figure 5.3.1.2.3: The initial image that was obtained by segmentation. The segmentation method that has been used is the K-means algorithm. The initial centroids were chosen at random. The values of the pixel temperatures in this image are the x_i' s in Gibbs sampler.

For $p=0.5$ we use no information from the data. Values of p those are lower than 0.5 show that we trust the observed image, while those that are greater than 0.5 show that we do not trust the data.

5.3.2 THE METROPOLIS-HASTINGS SAMPLER

The Metropolis-Hastings algorithm is a general case of the Gibbs sampler. The steps of the algorithm are described bellow:

1. Start with an initial value $x^{(0)}$
2. After having simulated t values, simulate a candidate value x^* from a transition kernel $q(x^* | x^{(t)})$.
3. Set $x^{(t+1)} = x^*$ with probability

$$a(x^* | x^{(t)}) = \min \left\{ \frac{q(x^{(t)} | x^*) \cdot f(x^*)}{q(x^* | x^{(t)}) \cdot f(x^{(t)})}, 1 \right\}$$

otherwise set $x^{(t+1)} = x^{(t)}$.

4. Repeat for k iterations steps 2 and 3.

We start from a depended sample and after some iteration the autocorrelation gets smaller. Theoretically, after infinite number of iteration, the sample will

be independent. In other words, the Metropolis-Hastings sampler produces a Markov chain $\{X_n\}$ which is reversible (Walsh 2004).

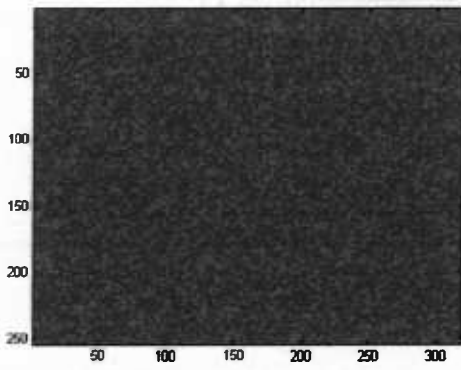
5.3.2.1 THE METROPOLIS-HASTINGS SAMPLER FOR THE ISING MODEL

At first, we try the sampler for the Ising model. The steps of the algorithm for simulating from the Ising model are (Hurn, Husby and Rue 2001):

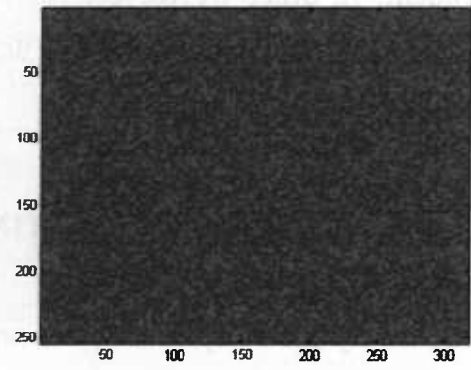
1. Give initial values x_i to each site 1 or 0 at random.
2. For each site i calculate $x_i' = 1 - x_i$.
 - a. Draw a random value $U \sim \text{Uniform}(0,1)$
 - b. Calculate $d = \exp(\beta \cdot b_i)$ and $d' = \exp(\beta \cdot w_i)$, where b_i is the number of the black neighbors of i and w_i is the number of its white neighbors.
 - c. Set $p = \min\{1, d'/d\}$.
 - d. If $U < p$ set $x_i = y_i$.
3. Repeat step 2 for k iterations.

The results of the Metropolis-Hastings algorithm for 25 iterations are shown in Figure 5.3.2.1.1 for several values of β .

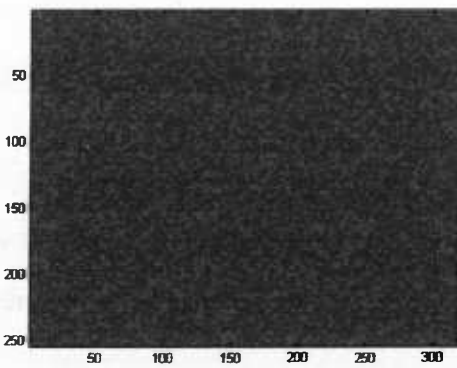




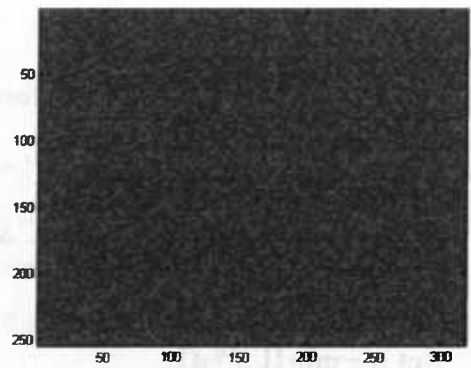
(a)



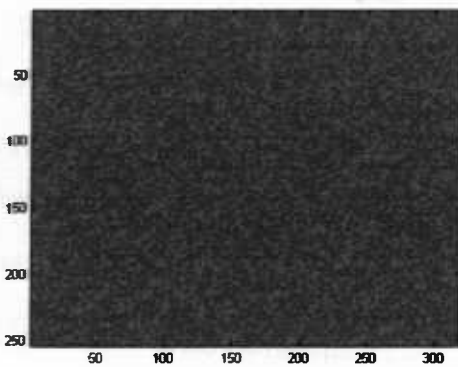
(b)



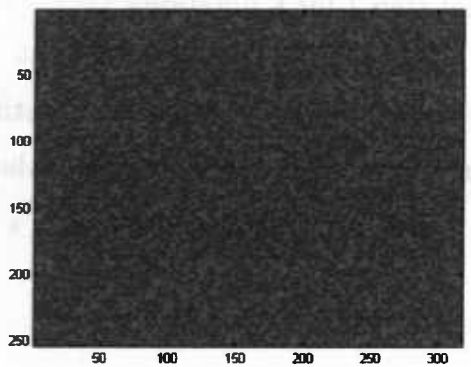
(c)



(d)



(e)



(f)

Figure 5.3.2.1.1: Simulation from the Ising model using the Metropolis sampler for different values of beta constant. (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.8$, (e) $\beta=0.9$, (f) $\beta=1.0$



5.3.2.2 THE METROPOLIS-HASTINGS SAMPLER FOR THE POSTERIOR OF INTEREST

We use the algorithm to simulate the posterior distribution (Hurn, Husby and Rue 2001). This means that we assume the Ising model for \mathbf{x} and try to find \mathbf{y} , where $\mathbf{y}=\mathbf{x}+\mathbf{e}$. The steps are modified as it follows:

1. Give initial values x_i to each site 1 or 0 at random.
2. For each site i :
 - a. Calculate $x'_i = 1 - x_i$.

- b. Draw a random value $U \sim \text{Uniform}(0,1)$

- c. Calculate

$$d = \exp(\beta \cdot b_i) + h_i(x_i, y_i)$$

and

$$d' = \exp(\beta \cdot w_i) + h_i(x'_i, y_i)$$

where the number of the black neighbors of i is b_i and w_i is the number of its white neighbors. The quantity $h_i(x_i, y_i)$ is the same we used in Gibbs sampler.

- d. Set $p = \min\{1, d'/d\}$.

- e. If $u < p$ set $x_i = y_i$.

3. Repeat step 2 for k iterations.

Figure 5.3.2.2.1 shows the results for different values of β and for 25 iterations of the algorithm.

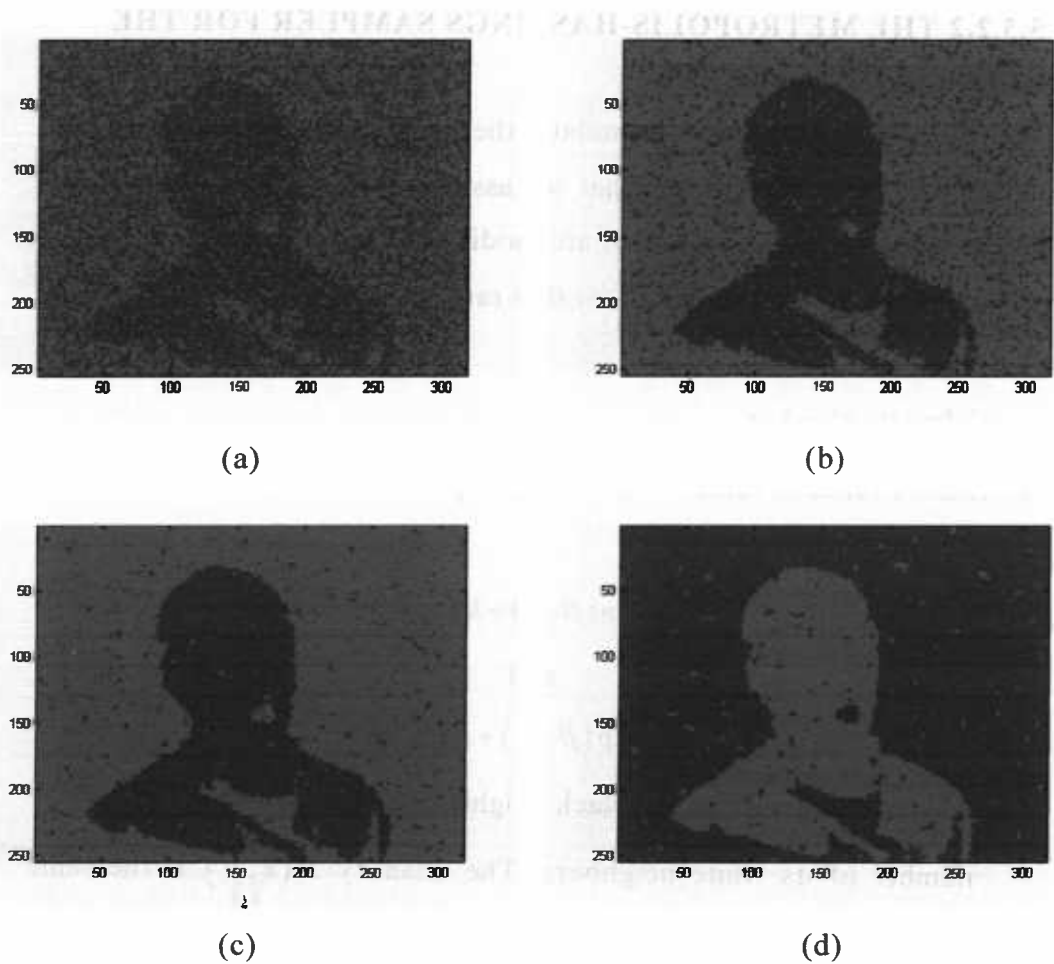
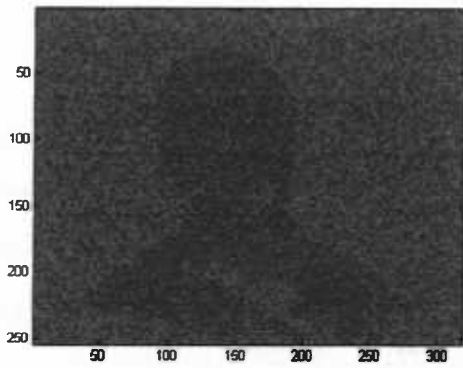


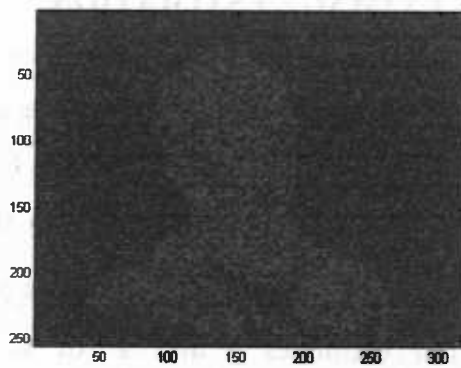
Figure 5.3.2.2.1: The results of the simulation from the noisy Ising model for flip noise. The beta parameter is respectively: (a) $\beta=0.2$, (b) $\beta=0.4$, (c) $\beta=0.6$, (d) $\beta=0.9$

In Figure 5.3.2.2.2 we have some outcomes of the algorithm for $\beta=0.4$, 25 iterations and different values of the probability that $y \neq x$.

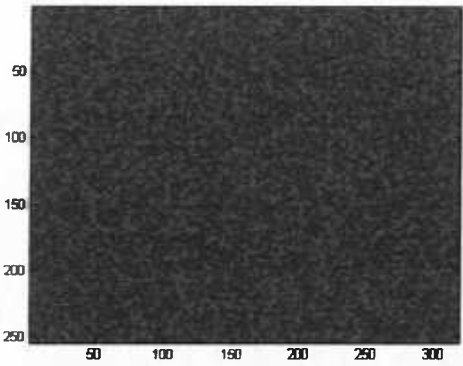




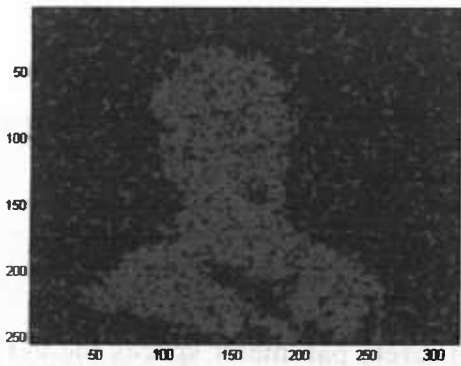
(a)



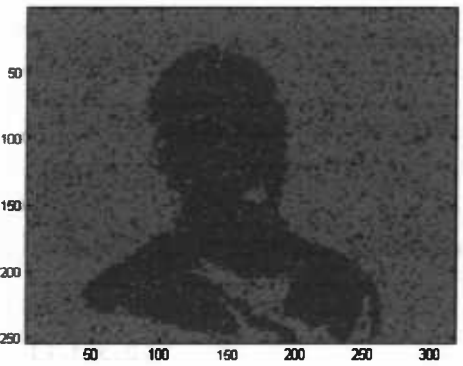
(b)



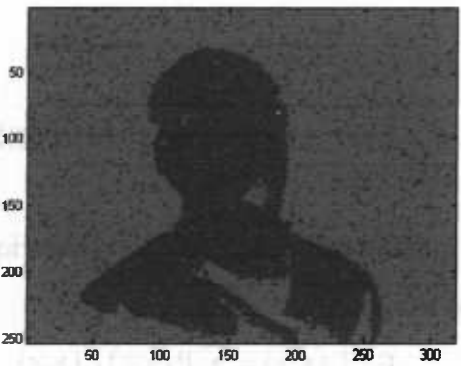
(c)



(d)



(e)



(f)

Figure 5.3.2.2.2: The results of the simulation from the noisy Ising model for flip noise. The different values of the probability p are respectively: (a) $p=0.1$, (b) $p=0.3$, (c) $p=0.5$, (d) $p=0.7$, (e) $p=0.8$, (f) $p=0.9$



5.3.3 IMAGE ESTIMATION

Consider the case where we have an estimate y of the true image x , and we need a measure of how wrong we are if we use y instead of x . In other words we need to know how close y is to the true data.

Definition 5.3.3.1: A function $L(x, y)$ that provides a measure of the distance between the real data x and an estimate y is called a loss function.

For estimates z and z' of x , we say that z is better than z' if $L(x, z) \leq L(x, z')$. For example, if we have point estimation, we can use the **squared error loss**:

$$L(x, y) = (y - x)^2 \tag{5.3.3.1}$$

or the **absolute error loss**:

$$L(x, y) = |y - x| \tag{5.3.3.2}$$

or for discrete parameter spaces the **0-1 loss**:

$$L(x, y) = \begin{cases} 0 & \text{if } y = x \\ 1 & \text{if } y \neq x \end{cases} \tag{5.3.3.3}$$

5.3.3.1 THE MARGINAL POSTERIOR MODE ESTIMATOR

Definition 5.3.1.1: The posterior expected distance between x and z is defined as:

$$E_{x|y} L(x, z) = \sum_x P(x|y) L(x, z) \tag{5.3.3.1}$$

Definition 5.3.1.2: The optimal Bayes estimate is the configuration that minimizes the posterior expected distance.

It can be proven that the optimal Bayes estimator is given by:

$$\hat{x} = \arg \min_z E_{x|y} L(x, z) \tag{5.3.3.1.2}$$

In order to obtain an estimate of the true image x we need to introduce a loss function. A possible choice could be



$$L_{\text{MPM}}(\mathbf{e}) = \sum_{i=1}^N e_i \tag{5.3.3.1.3}$$

where $\mathbf{e}=(e_1, e_2, \dots, e_N)$ is the error vector and $e_i = I_{[x_i \neq y_i]}$.

$L_{\text{MPM}}(\mathbf{e})$ leads to the **marginal posterior mode estimator** and counts the number of errors. For site i the MPM estimator is

$$\hat{x}_{\text{MPM},i} = \begin{cases} 1 & \text{if } \text{Prob}(x_i = 1 | y) \geq 0.5 \\ 0 & \text{if } \text{Prob}(x_i = 1 | y) < 0.5 \end{cases} \tag{5.3.3.1.5}$$

An algorithm to obtain an estimate of x_{MPM} is (Hurn, Husby and Rue 2001):

1. Produce k samples from the posterior
2. For each site i count the number b of times that $x_i = 1$.
3. If $b > k/2$ then $\hat{x}_{\text{MPM},i} = 1$, else $\hat{x}_{\text{MPM},i} = 0$.

Figure 5.3.1.1 shows the results of the algorithm for different number of samples. We have assumed the Ising model with constant $\beta=0.4$. The samples have derived from the Gibbs sampler, while the probability of having recorded the wrong image is equal to 0.6.



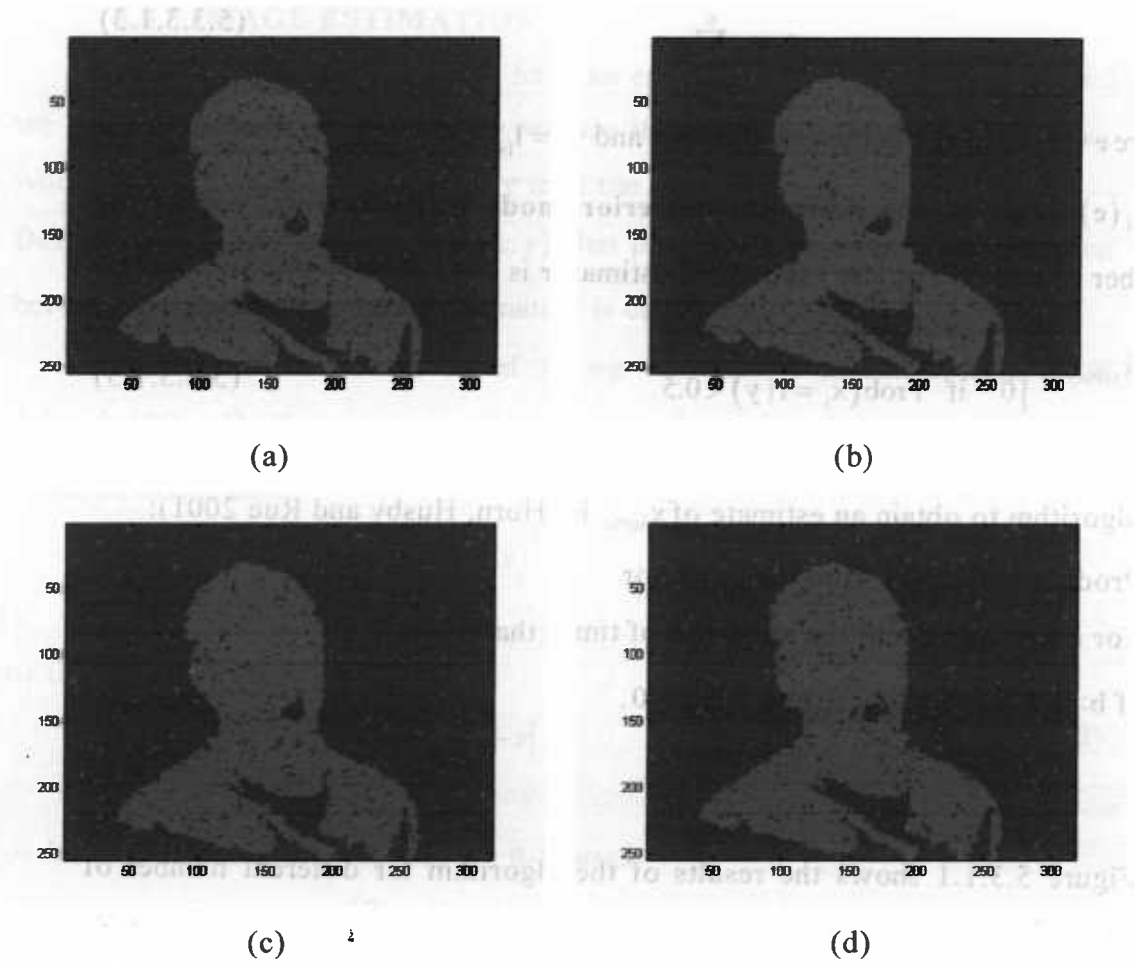


Figure 5.3.3.1.1: The MPM estimator for different number k of samples. (a) $k=10$ ($\hat{L}_{MPM} = 42645$), (b) $k=20$ ($\hat{L}_{MPM} = 46331$), (c) $k=30$ ($\hat{L}_{MPM} = 48169$), (d) $k=40$ ($\hat{L}_{MPM} = 49757$).

For number of samples equal to 10 or 20 the results are better than for 30 or 40 samples.

5.3.3.2 THE MAXIMUM A POSTERIORI ESTIMATOR

Another choice of loss function may be

$$L_{MAP}(\mathbf{e}) = 1 - (1 - \epsilon_1)(1 - \epsilon_2) \cdots (1 - \epsilon_n) \quad (5.3.3.2.1)$$

$L_{MAP}(\mathbf{e})$ is an extreme case of loss function since it takes value 1 if there is at least one error. This means that the number of errors does not matter, but any

image different from x is as wrong as any other image that differs from x . The MAP estimator is (Hurn, Husby and Rue 2001):

$$x_{MAP} = \arg \min_y P(y|x) \tag{5.3.3.2.2}$$

One estimator of x_{MAP} could be the mode of the posterior $P(x|y)$, but a more efficient way would be to sample from $P_T(x|y) \propto P(x|y)^{1/T}$ for $0 < T \ll 1$. The mode of $P_T(x|y)$ is the same as the mode of $P(x|y)$ for all $T > 0$ and as $T \rightarrow 0$ most of the probability mass will be on this mode (Besag 1986). The problem of decreasing T is that as T takes smaller values the algorithm may be trapped to a local mode of the posterior. To overcome this situation we should lower

T at each iteration t not faster than $T(t) = \frac{c}{\log(t+1)}$, where c is a constant

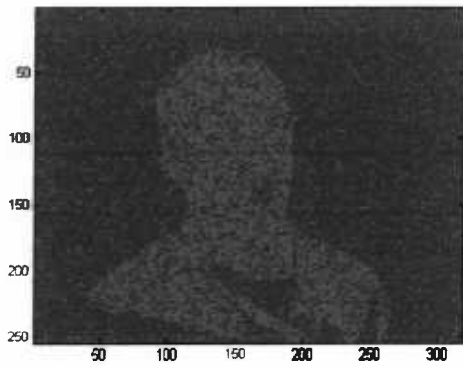
depending on the posterior (Hurn, Husby and Rue 2001). One choice is to take $T(t) = T_0 \cdot n^{-t}$, for $n < 1$. The algorithm that leads to the MAP estimator is (simulated annealing):

1. Give randomly value 1 or 0 to each site i .
2. For every random choice of a site i
 - a. Calculate $x'_i = 1 - x_i$.
 - b. Calculate

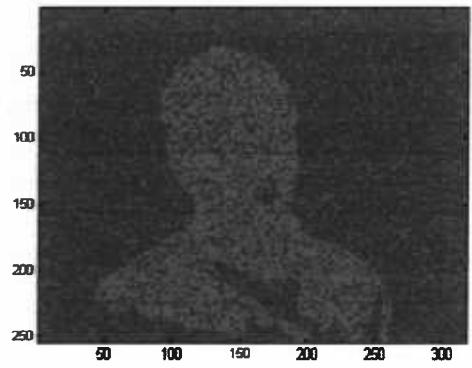
$$d' = \exp\left(\beta \sum_{j-i} I_{[x'_i=x'_j]} + h_i(x'_i, y_i)\right)$$
 and

$$d = \exp\left(\beta \sum_{j-i} I_{[x_i=x_j]} + h_i(x_i, y_i)\right)$$
 - c. Set $p = \left(\min\left(\frac{d'}{d}, 1\right)\right)^{1/T}$.
 - d. Draw a random value $U \sim \text{Uniform}(0,1)$.
 - e. If $U < p$ then $x_i = x'_i$.
3. Repeat steps 1 and 2 until all sites are updated once.
4. Set $T = T_0 \cdot n^{-t}$
5. Repeat 1 to 4 for k iterations.

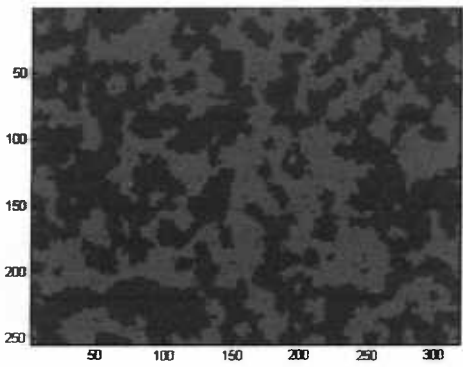




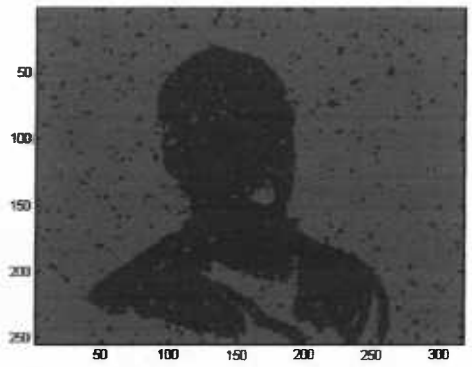
(a)



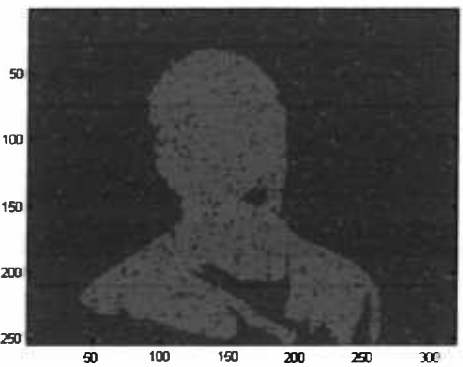
(b)



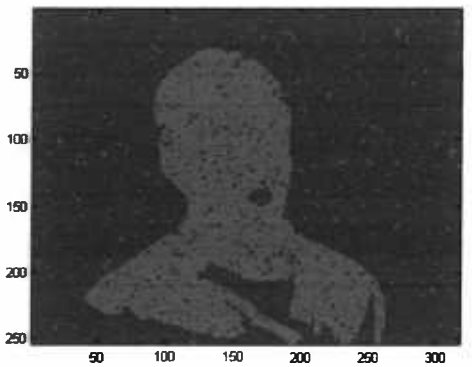
(c)



(d)



(e)



(f)

Figure 5.3.3.2.1: The MAP estimator for beta constant equal to 0.4 and 20 samples from the posterior. The probability p that we have obtained the wrong image is respectively (a) $p=0.1$, (b) $p=0.3$, (c) $p=0.5$, (d) $p=0.7$, (e) $p=0.8$, (f) $p=0.9$

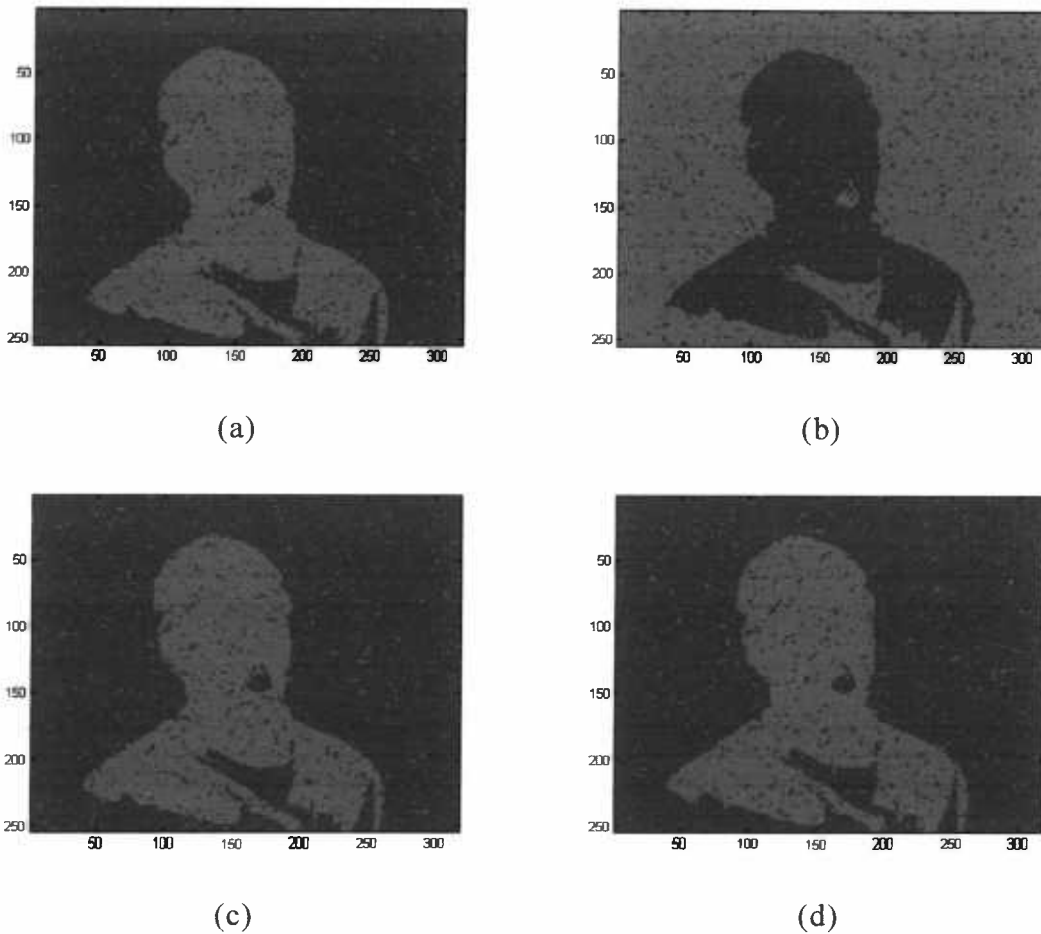


Figure 5.3.3.2.2: The MAP estimator for beta constant $\beta=0.4$ and probability of recording the wrong image $p=0.8$. The number k of samples is respectively (a) $k=5$, (b) $k=10$, (c) $k=15$, (d) $k=20$.

Small differences are noticed between the images. The results are similar to the results of the MPM estimator. In both cases (MPM and MAP) the algorithm added noise to the input image. A possible reason for that is that we have mistakenly assumed the Ising model.





CHAPTER 6

FACE DETECTION

6.1 THE IMPORTANCE OF FACE DETECTION

A usual problem in image analysis is to detect some features in the frame. The images we consider in this project are still from a thermal video recording. Such videos have a variety of applications. They may be used for detecting alcohol or drugs in the blood, since these substances cause vasodilatation, increasing the temperature in certain regions of the face. Moreover, a thermal video can be very useful in the lie detection procedure. Actually, when a person lies, it is likely to experience a certain level of stress and as a result the blood flow is affected. Consequently, we can compare the distribution of the face temperatures when the person is asked routine questions, where he or she is supposed to answer the truth, with the distribution of the temperatures when answering the questions of interest. The area surrounding the eyes is of particular concern, since it is the most affected from blood flow imbalances during anxious states. Therefore, it would be very useful if we could focus on the face.

6.2 FACE DETECTION USING K-MEANS

The human face has nearly ellipsoid shape and in the usual movements of the head the major axis is almost vertical, while the minor axis is almost horizontal. If we consider a column of the picture that contains the head, the number of skin pixels in this column will be greater than the one in a column that does not contain the head. On the other hand, it does not stand for the rows, since the rows that contain the shoulders will have more skin pixels. In



Figure 6.2.1 we may see a column that contains the head and two rows, one including the head and the other including the shoulders.

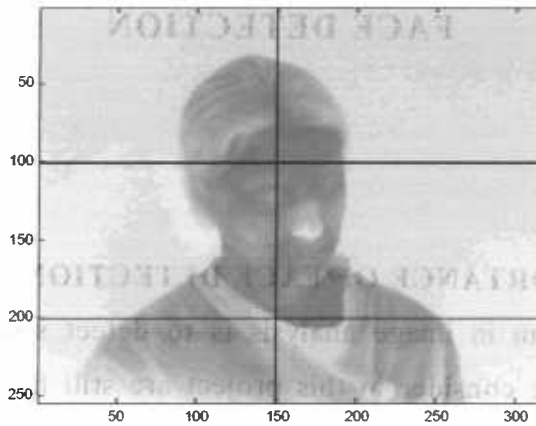


Figure 6.2.1: The blue column contains the head. The upper blue row contains the head, while the lower one contains the shoulders.

In other words, the number of the foreground pixels cannot be used to detect the face area, since the area of the neck and shoulders has a lot of foreground pixels as well. We can cope with this problem by using the K-means method introduced in chapter 2. The idea is to separate the columns into two groups (warm and cold), dividing the frame into regions, vertically.

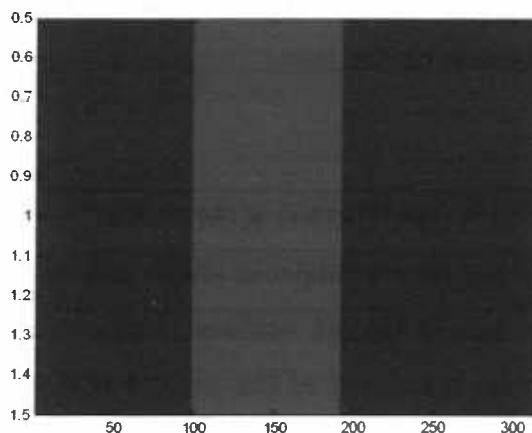


Figure 6.2.2: The frame has been divided into three vertical areas. The central area includes the face.



We consider the corresponding regions in the continuous image. The darker lines in Figure 6.2.3 show the region that has the highest average temperature and is the one that includes the face.

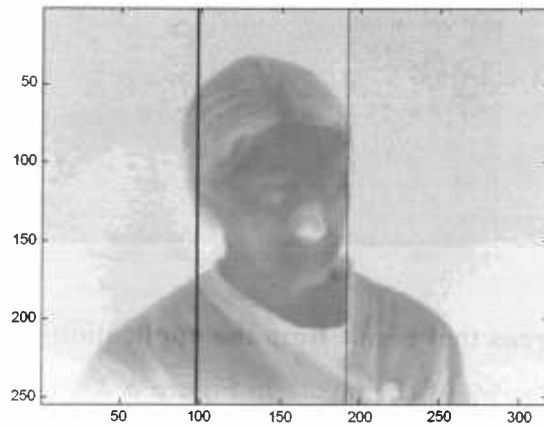


Figure 6.2.3: The head lies within the blue columns.

If we extract from the initial image the two regions that have lower mean temperature, we have the image in Figure 6.2.4. In Figure 6.2.4 there are non-skin pixels above the head and under the neck because of the clothes. If we apply the k-means algorithm for the rows of this frame, two clusters are formed, the one that contains the head and has rectangular shape (because it is defined by two vertical columns) and the other with the rest of the image.

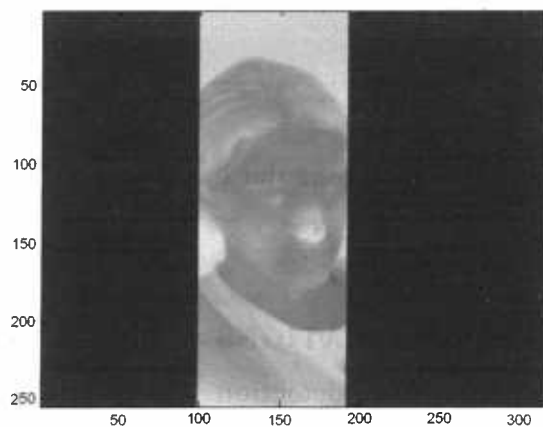


Figure 6.2.4: The rectangular region that contains the head.



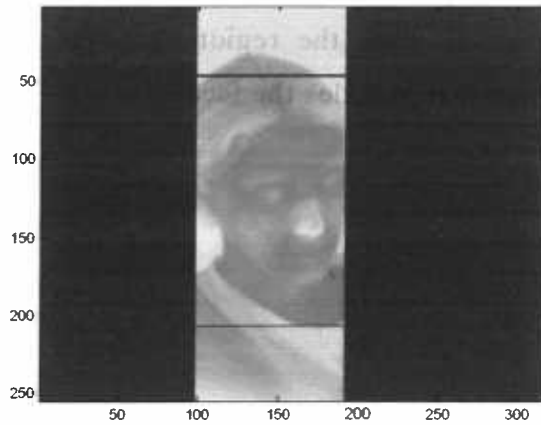


Figure 6.2.5: The areas that result from the application of K-means to the rows of the image in Figure 6.2.4

The dark lines in Figures 6.2.3 and 6.2.5 are considered to the respective rows and columns of the input image and a rectangular that contains the face is formed.

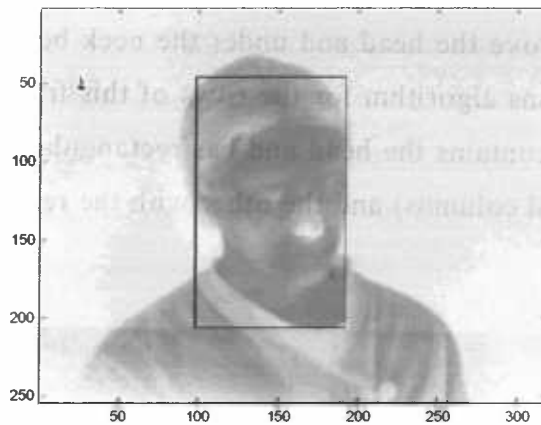
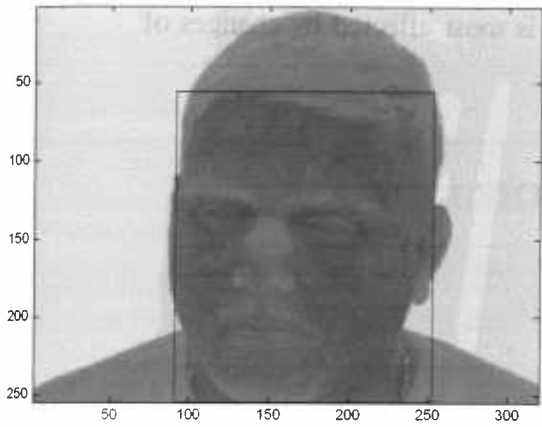


Figure 6.2.6: The rectangular has detected the face

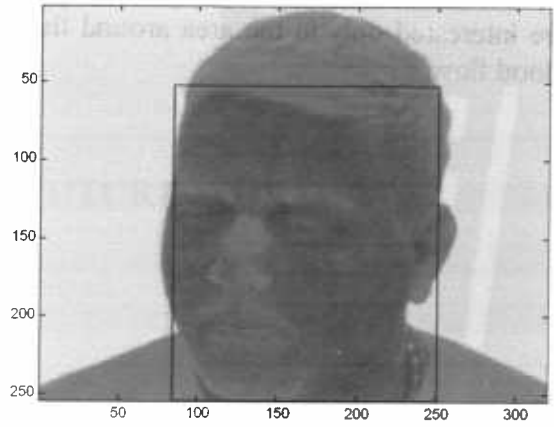
6.3 SOME MORE APPLICATIONS

We apply the same face detection algorithm for six images with different positions of the head. The method has worked with success in all six cases and it is effective in video, which means that the rectangular follows the head during the reproduction.

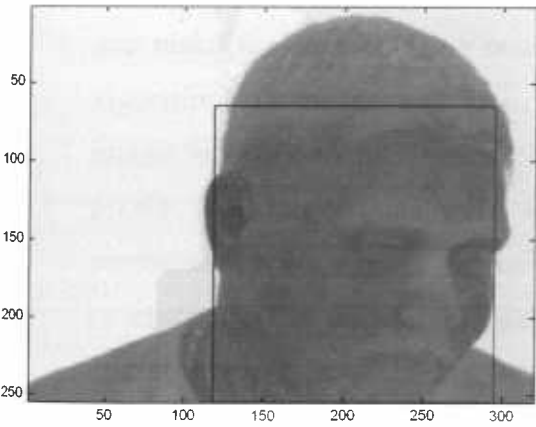




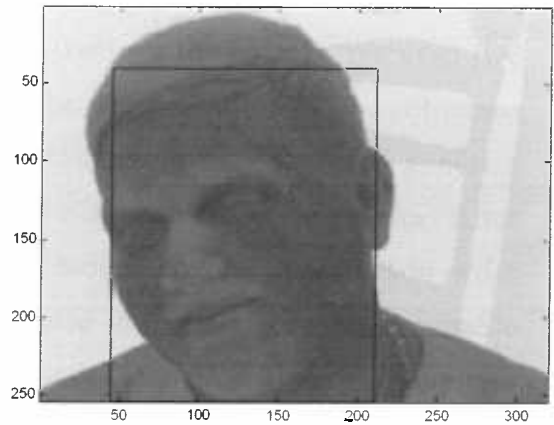
(a)



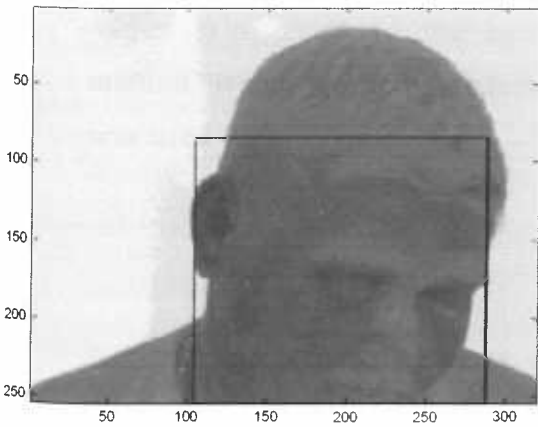
(b)



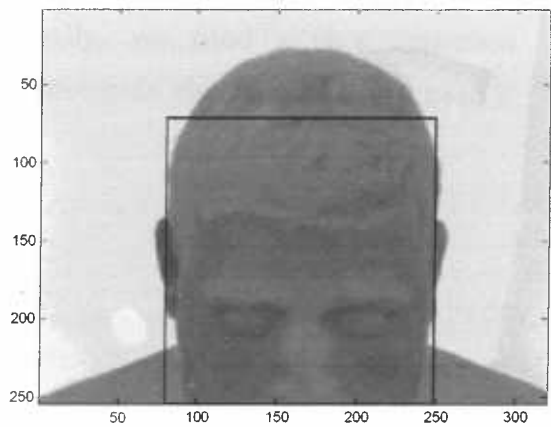
(c)



(d)



(e)



(f)

Figure 6.3.1: Face detection for different orientations of the head.



The rectangular does not contain the hair area. This is not a problem, since we are interested only in the area around the eyes, which is most affected by changes of blood flow.



CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

The problem we were dealing with was to perform segmentation in pictures that have been obtained from thermal camera frames of individuals and make the segmentation results smoother (remove the noise). We used two algorithms (k-means and Otsu's method) for segmentation of a continuous image in order to produce a binary image, containing only skin and non-skin pixels. Both algorithms were stable and gave almost the same results. After segmentation we applied to the binary image morphological operators in order to eliminate the undesirable gaps and eliminate the edges. The most suitable operator was binary closing. We also used spatial statistics' theory to improve the image, assuming the Ising model for the data. The results were not satisfactory, since the Ising model increased the noise for more than 5 iterations. The MCMC methods also added more noise and in particular they added skin pixels to the background. Finally, we used a face detection method, which worked satisfactory; since it detected the face in every case it was used.

7.2 FUTURE WORK

Image processing has a wide range of tools that we can use. Some of the techniques applied in this thesis did not work properly, while others did. In the future, all the above methods can be modified in order to give better results and be used for more applications, such as relative biomedical respiration, detection of perspiration, sleep studies, etc. Image morphology can also be used for continuous data, which means that we can perform it in



the original image using grayscale operators. Furthermore, we can improve the MCMC applications, assuming more complicated models than the Ising model. The face detection can lead to better results and be used in cases that we need to detect more specific areas, like the eyes of the subject. This would be very useful, since the blood vessels around the eyes appear to have increasing temperatures when a person experiences high level of anxiety and thus a consistent (noise free) segmentation it could be very useful in non-invasive deception detection techniques. Moreover, the face detection algorithm can be expanded when more than one individual appear in the same frame.



APPENDIX

The segmentation k-means algorithm:

```
% A is the original thermal image  
A1=reshape(A,size(A,1)*size(A,2),1);  
A1=kmeans(A1,2);  
A1=reshape(A1,size(A,1),size(A,2));
```

The segmentation Otsu's algorithm (function):

```
function otsu=otsus_thr(A)  
A1=reshape(A,size(A,1)*size(A,2),1);  
A1=sort(A1);  
n=max(size(A1,1),size(A1,2));  
  
j=1;  
for i=1:n-1  
    if A1(i)~=A1(i+1)  
        Q(j)=A1(i);  
        j=j+1;  
    end  
end  
if A1(i+1)~=A1(i)  
    Q(j)=A1(i+1);  
end  
m=max(size(Q,1),size(Q,2));  
  
% ypologismos tw n pi  
i=1;
```



```
for j=1:m
    N(j)=0;
    flag=true;
    while i<=n & flag==true
        if Q(j)==A1(i)
            N(j)=N(j)+1;
            i=i+1;
        else
            flag=false;
        end
    end
    p(j)=N(j)/n;
end

% total mean value
mean_total=dot(p,Q);

% total variance
var_total=dot((Q-m).*(Q-m),p);
for k=2:m-1
    w_0=sum(N(1:k))/n;
    w_1=sum(N((k+1):m))/n;
    d=p.*Q;
    m_0=sum(d(1:k))/w_0;
    m_1=sum(d(k+1:m))/w_1;
    var_between=w_0*w_1*(m_0-m_1)^2;
    criterion(k)=var_between/var_total;
end
t=find(criterion==max(criterion));
threshold=Q(t);

for i=1:size(A,1)
    for j=1:size(A,2)
        if A(i,j)<threshold
```



```
otsu(i,j)=0;  
else  
otsu(i,j)=1;  
end  
end  
end  
for i=1:size(criterion,2)  
    V(i)=i+1;  
end  
figure  
imagesc(otsu)  
figure  
plot(V)  
return;
```

The morphological operators algorithm:

```
% -----Dilation-----
```

```
% se=structuring element  
% A1=the image obtained from segmentation of A  
R=input('Give the radial')  
se=strel('diamond',R);  
D=imdilate(A1,se);  
figure  
Imagesc(D)
```

```
R=input('Give the radial')  
se=strel('octagon',R);  
D=imdilate(A1,se);  
Figure  
Imagesc(D)
```



```
M=input('Give number of rows')
N=input('Give the number of columns')
se=stel('rectangle',[M N]);
D=imdilate(A1,se);
Figure
Imagesc(A1)
```

% -----Erosion-----

```
R= input('Give the radial')
se=strel('diamond',R);
D=imerode(A1,se);
figure
Imagesc(D)
```

```
R= input('Give the radial')
se=strel('octagon',R);
D=imerode(A1,se);
Figure
Imagesc(D)
```

% -----Opening-----

```
R= input('Give the radial')
se=strel('diamond',R);
D=imopen(A1,se);
figure
Imagesc(D)
```

```
R= input('Give the radial')
se=strel('octagon',R);
D=imopen(A1,se);
```



Figure

Imagesc(D)

```
% -----Closing-----
```

```
R= input('Give the radial')
```

```
se=strel('diamond',R);
```

```
D=imclose(A1,se);
```

```
figure
```

```
Imagesc(D)
```

```
R= input('Give the radial')
```

```
se=strel('octagon',R);
```

```
D=imclose(A1,se);
```

```
figure
```

```
Imagesc(D)
```

```
M= input('Give the number of rows')
```

```
N= input('Give the number of columns')
```

```
se=stel('rectangle',[M N]);
```

```
D=imclose(A1,se);
```

```
Figure
```

```
Imagesc(A1)
```

Ising model algorithm:

```
% isb=Ising constant
```

```
% M=number of iterations
```

```
% b=number of black neighbors of pixel (i,j)
```



```

% w=number of white neighbors of pixel (i,j)
% p1=probability calculated from the ising model
% u=random value from uniform(0,1)

M=input('Give number of iterations')
isb=0.4;
for k=1:M
tmpA=A1;
for i=1:size(A1,1)
for j=1:size(A1,2)
    b=0;
    if i>1 & A1(i-1,j)==1
        b=b+1;
    end
    if i<size(A1,1) & A1(i+1,j)==1
        b=b+1;
    end
    if j>1 & A1(i,j-1)==1
        b=b+1;
    end
    if j<size(A1,2) & A1(i,j+1)==1
        b=b+1;
    end
    if i>1 & j>1 & i<size(A1,1) & j<size(A1,2)
        w=4-b;
    end
    if (i==1 & j==1) | (i==size(A1,1) & j==1) | (i==1 & j==size(A1,2)) |
(i==size(A1,1) & j==size(A1,2))
        w=2-b;
    end
    if (i==1 & j~=1 & j~=size(A1,2)) | (i==size(A1,1) & j~=1 &
j~=size(A1,2)) | (j==1 & i~=1 & i~=size(A1,1)) | (j==size(A1,2) & i~=1 &
i~=size(A1,1))
        w=3-b;

```



```
end
p1=exp(isb*b)/(exp(isb*b)+exp(isb*w));
if p1>=0.5
    B1(i,j)=1;
else
    B1(i,j)=0;
end
end
end
A1=B1;
end
```

The Ising model algorithm with terminating condition:

```
% A1=the input image
% isb=the ising constant beta
% numb= is the total number of pixels changing at each iteration
% ratio is the ratio of pixels changing at each iteration
% the iterations stop when over 5% of the total number of pixels is changed
% b=number of black neighbors of pixel (i,j)
% w=number of white neighbors of pixel (i,j)

prob=input('Give probability')
isb=input('Give the Ising constant')
numb=0;
ratio=0;
while ratio<=0.05
    % the program chooses randomly which pixel to update
    tmp=rand(size(A1,1),size(A1,2));
    [i1,j1,v]=find(tmp);
    [v1,k]=sort(v);
    R=i1(k);
    C=j1(k);
    M=max(size(R,1),size(R,2));
```



```

for i2=1:M
    i=R(i2);
    j=C(i2);
    % b is the number of neighbors that correspond to foreground
    b=0;
    if i>1 & A1(i-1,j)==1
        b=b+1;
    end
    if i<size(A1,1) & A1(i+1,j)==1
        b=b+1;
    end
    if j>1 & A1(i,j-1)==1
        b=b+1;
    end
    if j<size(A1,2) & A1(i,j+1)==1
        b=b+1;
    end
    % w is the number of neighbors that correspond to background
    if i>1 & j>1 & i<size(A1,1) & j<size(A1,2)
        w=4-b;
    end
    if (i==1 & j==1) | (i==size(A1,1) & j==1) | (i==1 & j==size(A1,2)) |
(i==size(A1,1) & j==size(A1,2))
        w=2-b;
    end
    if (i==1 & j~=1 & j~=size(A1,2)) | (i==size(A1,1) & j~=1 &
j~=size(A1,2)) | (j==1 & i~=1 & i~=size(A1,1)) | (j==size(A1,2) & i~=1 &
i~=size(A1,1))
        w=3-b;
    end
    p1=exp(isb*b)/(exp(isb*b)+exp(isb*w));
    if p1>=prob
        B1(i,j)=1;
    else

```



```
        B1(i,j)=0;
    end
end
F=find(B1-A1~=0);
numb=numb+max(size(F,1),size(F,2));
ratio=numb/(size(A1,1)*size(A1,2))
A1=B1;
end
figure
imagesc(A1)
```

The Gibbs sampler for the Ising model:

```
% Initialize X
X=zeros(254,318);
% isb=Ising parameter
% M=number of iterations
% b=number
% b=number of black neighbors of pixel (i,j)
% w=number of white neighbors of pixel (i,j)
```

```
M=input('Give number of iterations')
```

```
isb=input('Give the Ising constant')
```

```
for t=1:M
```

```
    % Choose randomly one pixel
```

```
    tmp=rand(size(A,1),size(A,2));
```

```
    [i1,j1,v]=find(tmp);
```

```
    [v1,k]=sort(v);
```

```
    R=i1(k);
```

```
    C=j1(k);
```

```
    M=max(size(R,1),size(R,2));
```

```
    for i2=1:M
```

```
        i=R(i2);
```



j=C(i2);

% Count for pixel (i,j) the number of black neighbors(b)

b=0;

if i>1 & X(i-1,j)==1

b=b+1;

end

if i<size(X,1) & X(i+1,j)==1

b=b+1;

end

if j>1 & X(i,j-1)==1

b=b+1;

end

if j<size(X,2) & X(i,j+1)==1

b=b+1;

end

% Count for pixel (i,j) the number of white neighbors(w)

if i>1 & j>1 & i<size(X,1) & j<size(X,2)

w=4-b;

end

if (i==1 & j==1) | (i==size(X,1) & j==1) | (i==1 & j==size(X,2)) |
(i==size(X,1) & j==size(X,2))

w=2-b;

end

if (i==1 & j~=1 & j~=size(X,2)) | (i==size(X,1) & j~=1 & j~=size(X,2))
| (j==1 & i~=1 & i~=size(X,1)) | (j==size(X,2) & i~=1 & i~=size(X,1))

w=3-b;

end

% Generate a uniform(0,1) value

u=rand;

% Calculate the probability p and compare with u



```
p=exp(isb*b)/(exp(isb*b)+exp(isb*w));  
if u<p  
    X(i,j)=1;  
else  
    X(i,j)=0;  
end  
end  
end  
figure  
imagesc(X)
```

The Gibbs sampler for the posterior of interest:

```
% iter=the number of iterations  
% isb=the ising beta parameter  
% M=number of rows  
% N=number of columns  
% prob=probability that we have recorded the wrong value  
% Y=the input image  
  
isb=0.4;  
iter=input('Give the number of iterations');  
M=254;  
N=318;  
prob=input('Give the probability that we have recorded the wrong value')  
X=zeros(M,N);  
for t=1:iter  
    for i=1:M  
        for j=1:N  
            if Y(i,j)==X(i,j)  
                I=1;  
            else  
                I=0;  
            end  
            h=I*log((1-prob)/prob);
```



```

%Count for pixel (i,j) the number of black neighbors(b)
    b=0;
    if i>1 & X(i-1,j)==1
        b=b+1;
    end
    if i<size(X,1) & X(i+1,j)==1
        b=b+1;
    end
    if j>1 & X(i,j-1)==1
        b=b+1;
    end
    if j<size(X,2) & X(i,j+1)==1
        b=b+1;
    end
%Count for pixel (i,j) the number of white neighbors(w)
    if i>1 & j>1 & i<size(X,1) & j<size(X,2)
        w=4-b;
    end
    if (i==1 & j==1) | (i==size(X,1) & j==1) | (i==1 & j==size(X,2)) |
(i==size(X,1) & j==size(X,2))
        w=2-b;
    end
    if (i==1 & j~=1 & j~=size(X,2)) | (i==size(X,1) & j~=1 & j~=size(X,2))
| (j==1 & i~=1 & i~=size(X,1)) | (j==size(X,2) & i~=1 & i~=size(X,1))
        w=3-b;
    end
    u=rand;
    p=exp(isb*b+h)/(exp(isb*b)+exp(isb*w));
    if u<p
        X(i,j)=1;
    else
        X(i,j)=0;
    end
end
end

```



```
end  
end
```

The Metropolis-Hastings sampler for the Ising model:

```
% Fill x randomly with 1s and 0s  
X=zeros(254,318);  
isb=input('Give the Ising parameter')  
M=input('Give number of iterations')  
for t=1:M  
    % Choose randomly one pixel  
    tmp=rand(size(A,1),size(A,2));  
    [i1,j1,v]=find(tmp);  
    [v1,k]=sort(v);  
    R=i1(k);  
    C=j1(k);  
    M=max(size(R,1),size(R,2));  
    for i2=1:M  
        i=R(i2);  
        j=C(i2);  
        % Calculate the supplemental value  
        X1(i,j)=1-X(i,j);  
        % Count number of neighbors of (i,j) equal to X(i,j)  
        a=0;  
        if i>1 & X(i-1,j)==X(i,j)  
            a=a+1;  
        end  
        if i<size(X,1) & X(i+1,j)==X(i,j)  
            a=a+1;  
        end  
        if j>1 & X(i,j-1)==X(i,j)  
            a=a+1;  
        end  
        if j<size(X,2) & X(i,j+1)==X(i,j)
```



```

    a=a+1;
end

% Calculate the numerator of ratio R
d=exp(isb*a);
% Count number of neighbors of (i,j) equal to X1(i,j)
if i>1 & j>1 & i<size(X,1) & j<size(X,2)
    b=4-a;
end
if (i==1 & j==1) | (i==size(X,1) & j==1) | (i==1 & j==size(X,2)) |
(i==size(X,1) & j==size(X,2))
    b=2-a;
end
if (i==1 & j~=1 & j~=size(X,2)) | (i==size(X,1) & j~=1 &
j~=size(X,2)) | (j==1 & i~=1 & i~=size(X,1)) | (j==size(X,2) & i~=1 &
i~=size(X,1))
    b=3-a;
end
end
end

% Calculate the denominator of R
d1=exp(isb*b);
% Generate uniform(0,1) , calculate p and compare
u=rand;
p=min(1,d1/d);
if u<p
    X(i,j)=X1(i,j);
end
end
end

figure
imagesc(X)

```



The Metropolis-Hastings sampler for the posterior:

```
isb=input('Give the Ising constant')
prob=input('Give probability p')
T0=input('Give the initial temperature T0')
M=input('Give the number of iterations')
for t=1:M
    % Chose at random pixel (i,j)
    tmp=rand(size(A,1),size(A,2));
    [i1,j1,v]=find(tmp);
    [v1,k]=sort(v);
    R=i1(k);
    C=j1(k);
    M=max(size(R,1),size(R,2));

    for i2=1:M
        i=R(i2);
        j=C(i2);
        % find the complemental value
        X1(i,j)=1-X(i,j);
        % Calculate  $h_i(x_i, x_{1i})$  and  $h_i(x_i', x_{1i})$ 
        if Y(i,j)==X(i,j)
            I=1;
        else
            I=0;
        end
        h(i,j)=I*log((1-prob)/prob));
        h1(i,j)=I*log(prob/(1-prob));
        % Count the number of neighbors equal to X(i,j): b
        b=0;
        if i>1 & X(i-1,j)==X(i,j)
            b=b+1;
        end
        if i<size(A,1) & X(i+1,j)==X(i,j)
```



```

    b=b+1;
end
if j>1 & X(i,j-1)==X(i,j)
    b=b+1;
end
if j<size(A,2) & X(i,j+1)==X(i,j)
    b=b+1;
end
% Count the number of neighbors equal to X(i,j): w
if i>1 & j>1 & i<size(A,1) & j<size(A,2)
    w=4-b;
end
if (i==1 & j==1) | (i==size(A,1) & j==1) | (i==1 & j==size(A,2)) |
(i==size(A,1) & j==size(A,2))
    w=2-b;
end
if (i==1 & j~=1 & j~=size(A,2)) | (i==size(A,1) & j~=1 & j~=size(A,2)) |
(j==1 & i~=1 & i~=size(A,1)) | (j==size(A,2) & i~=1 & i~=size(A,1))
    w=3-b;
end
% Generate a uniform(0,1) value and calculate R and p
d=isb*b-h(i,j);
d1=isb*w-h1(i,j);
p=exp(min(d1-d,0)/T);
u=rand;
if u<p
    X(i,j)=X1(i,j);
end
end
% Change temperature T(t) for the next iteration
T=T0*0.99^(t-1);
end
figure
imagesc(X)

```



The MPM estimator:

```
%sum counts for every pixel i in how many iterations it is black
% M=input('Give number of iterations')
sum=zeros(size(A,1),size(A,2));
for t=1:M
%randomize the order that we chose the pixels to update
tmp=rand(size(A,1),size(A,2));
[i1,j1,v]=find(tmp);
[v1,k]=sort(v);
R=i1(k);
C=j1(k);
M=max(size(R,1),size(R,2));
for i2=1:M
i=R(i2);
j=C(i2);
X1(i,j)=1-X(i,j);
if Y(i,j)==X(i,j)
    I=1;
else
    I=0;
end
h(i,j)=I*log((1-prob)/prob);
h1(i,j)=I*log(prob/(1-prob));
%b=number of neighbors equal to X(i,j)
b=0;
    if i>1 & X(i-1,j)==X(i,j)
        b=b+1;
    end
    if i<size(A,1) & X(i+1,j)==X(i,j)
        b=b+1;
    end
    if j>1 & X(i,j-1)==X(i,j)
        b=b+1;
```



```

end
if j<size(A,2) & X(i,j+1)~=X(i,j)
    b=b+1;
end
%w=number of neighbors that are different from X(i,j)
if i>1 & j>1 & i<size(A,1) & j<size(A,2)
    w=4-b;
end
if (i==1 & j==1) | (i==size(A,1) & j==1) | (i==1 & j==size(A,2)) |
(i==size(A,1) & j==size(A,2))
    w=2-b;
end
if (i==1 & j~=1 & j~=size(A,2)) | (i==size(A,1) & j~=1 & j~=size(A,2)) |
(j==1 & i~=1 & i~=size(A,1)) | (j==size(A,2) & i~=1 & i~=size(A,1))
    w=3-b;
end
d=isb*b-h(i,j);
d1=isb*w-h1(i,j);
p=exp(min(d1-d,0));
u=rand;
if u<p
    X(i,j)=X1(i,j);
end
if X(i,j)==1
    sum(i,j)=sum(i,j)+1;
end
end
Y=X;
end
%determine the final value of (i,j)
for i=1:size(X,1)
    for j=1:size(X,2)
        if sum(i,j)>=t/2
            X(i,j)=1;

```



```
    else
        X(i,j)=0;
    end
end
end
end
```

The MAP estimator:

```
% A1=the binary image obtained from segmentation
% Set the initial values
T0=input('Give the initial temperature T0')
T=T0;
isb=input('Give the Ising constant')
prob=input('Give the probability p')
iter=input('Give the number of iterations')
% initialize X
X=randsrc(size(A1,1),size(A1,2),[0,1]);
for i=1:size(A1,1)
    for j=1:size(A1,2)
        if Y(i,j)==1
            Y(i,j)=0;
        else
            Y(i,j)=1;
        end
    end
end
end

for t=1:iter
% choose a pixel at random

tmp=rand(size(A1,1),size(A1,2));
[i1,j1,v]=find(tmp);
[v1,k]=sort(v);
R=i1(k);
```



```
C=j1(k);
M=max(size(R,1),size(R,2));
for i2=1:M
i=R(i2);
j=C(i2);
X1(i,j)=1-X(i,j);
if Y(i,j)==X(i,j)
    I=1;
else
    I=0;
end
h(i,j)=I*log((1-prob)/prob);
h1(i,j)=I*log(prob/(1-prob));
% count the number of neighbours of (i,j) equal to X(i,j)
b=0;
    if i>1 & X(i-1,j)==X(i,j)
        b=b+1;
    end
    if i<size(A1,1) & X(i+1,j)==X(i,j)
        b=b+1;
    end
    if j>1 & X(i,j-1)==X(i,j)
        b=b+1;
    end
    if j<size(A1,2) & X(i,j+1)==X(i,j)
        b=b+1;
    end
% count the number of neighbours of (i,j) equal to 1-X(i,j)
    if i>1 & j>1 & i<size(A1,1) & j<size(A1,2)
        w=4-b;
    end
    if (i==1 & j==1) | (i==size(A1,1) & j==1) | (i==1 & j==size(A1,2)) |
(i==size(A1,1) & j==size(A1,2))
        w=2-b;
```



```
end
if (i==1 & j~=1 & j~=size(A1,2)) | (i==size(A1,1) & j~=1 &
j~=size(A1,2)) | (j==1 & i~=1 & i~=size(A1,1)) | (j==size(A1,2) & i~=1 &
i~=size(A1,1))
    w=3-b;
end
%calculate the probabilities
d=isb*b-h(i,j);
d1=isb*w-h1(i,j);
p=exp(min(d1-d,0)/T);
u=rand;
if u<p
    X(i,j)=X1(i,j);
end
end
Y=X;
T=T0*0.99^(t-1);
end
figure
imagesc(X)
```

The face detection algorithm:

The following function detects the face:

```
function orth=orthogwnio(A)
%vertical k means
A1=(kmeans(A',2))';
D1=diff(A1);
F=find(D1~=0);
n=max(size(F,2),size(F,2));

%find the mean temperature of each area
mesh_temp(1)=sum(sum(A(:,1:F(1))))/(F(1)*size(A,1));
```



```
if n~=1
    for j=2:n
        mesh_temp(j)=sum(sum(A(:,F(j-1)+1:F(j))))/((F(j)-F(j-1))*size(A,1));
    end
end
mesh_temp(n+1)=sum(sum(A(:,F(n):size(A,2))))/((size(A,2)-
F(n)+1)*size(A,1));

% find the maximum of the mean values
Maxx=max(mesh_temp);
M=find(mesh_temp==Maxx);
if M==1
    y1=1;
    y2=F(M);
elseif M==n+1
    y1=F(n);
    y2=size(A,2);
else
    y1=F(M-1);
    y2=F(M);
end

% B= the first rectangular area containing the face
for j=y1:y2
    B(:,j-y1+1)=A(:,j);
end

%horizontal k means in B
B1=kmeans(B,2);
flag=true;
i=1;
while flag==true & i<=size(B1,1)-1
    if B1(i)~=B1(i+1)
        flag=false;
    end
end
```



```
        k1=i;
    else
        i=i+1;
    end
end
i=size(B1,1);
flag=true;
while flag==true & i>=k1
    if B1(i)~=B1(i-1)
        flag=false;
        k2=i;
    else
        i=i-1;
    end
end

mesh_temp2(1)=sum(sum(B(1:k1,:)))/(k1*size(B,2));
mesh_temp2(2)=sum(sum(B(k1+1:k2,:)))/((k2-k1)*size(B,2));
mesh_temp2(3)=sum(sum(B(k2+1:size(B,1),:)))/((size(B,1)-k2+1)*size(B,2));

% find which of the areas has maximum mean temperature
Maxx1=max(mesh_temp2);
M1=find(mesh_temp2==Maxx1);
if M1==1
    x1=1;
    x2=k1;
elseif M1==2
    x1=k1;
    x2=k2;
else
    x1=k2;
    x2=size(B,1);
end
```



```
% making the rectangular
orth=A;
for i=x1:x2
    orth(i,y1)=0;
    orth(i,y2)=0;
end
for j=y1:y2
    orth(x1,j)=0;
    orth(x2,j)=0;
end

return;
```



REFERENCES

- Besag, J., E. (1986).** On the statistical analysis of dirty pictures, *Journal of the Royal Statistic Society. Series B (Methodological)*, Vol. 48, No.3,259-302
- Besag, J.E. (1974).** Spatial interaction and the statistical analysis of lattice Systems. *Journal of the Royal Statistical Society B*, 36(2): 192-236
- Bloomberg, D.S. (2002).** Implementation efficiency of binary morphology, *International Symposium For Mathematical Morphology IV, Sydney, Australia, April 3-5, 2002*
- Bouman, C.A. (1995).** Markov random fields and stochastic image models. *Presented at 1995 IEEE International Conference*
- Chipman, H. and George, E.I and McCulloch, R.E. (2001).** The practical implementation of Bayesian model selection. *IMS lecture notes*, University of Waterloo, University of Texas at Austin, University of Chicago
- Geman, S. and Geman, D. (1984).** Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, 6:721-745
- Hanson, M.K. (2006).** A tutorial on Markov Chain Monte Carlo
- Hastings, W.K. (1970).** Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97-109
- Hurn M.A., Husby O.K. and Rue H. (2001).** A tutorial on image Analysis, *In spatial statistics and computational methods*, 87-141, J.Moller
- MacQueen, J. B. (1967).** Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1:281-297
- Morse, B.S. (2000).** Binary morphology. *Lecture notes for computer vision from the Brigham University- <http://bryan.cs.byu.edu/650/home/index.php>*
- Otsu, N. (1979).** A threshold selection method from gray-level histograms, *IEEE Trans. Sys., Man., Cyber.*, vol. 9, pp. 62-66
- Ripley, B.D (1988).** Statistical inference for spatial data. Cambridge University Press, Cambridge



Rosenthal, J.S and Roberts, G.O. (2004). General state space Markov chains and MCMC algorithms, *Probability surveys, vol.1, 20-71*

Sahu, S. (2000). Tutorial on MCMC I

Sarat, C., Dass and Anil, K., Jain and Xiaoguang, Lu (2002). Face detection and synthesis using Markov random field models, *16th International Conference on Pattern Recognition, Vol. 4, p. 40201*

Ungar L,H. and Foster D.P. (1998). Clustering Methods for collaborative filtering, *Proceedings of the Workshop on Recommendation Systems - citeseer.ist.psu.edu*

Walsh, B. (2004). Markov chain Monte Carlo and Gibbs sampling. *Lecture notes for EEB 581, <http://www.stat.columbia.edu/~liam/teaching/neurostat-spr07/papers/mcmc/mcmc-gibbs-intro.pdf>*

Wayne, Lin Wei-Cheng (2000). Mathematical morphology and its applications on image segmentation, Dept. of Computer Science and Information Engineering, National Taiwan University, <http://www.cmlab.csie.ntu.edu.tw/~wclin/download/thesis.pdf>

2





Δωρεά

