

Summer 2014

Improvements on segment based contours method for DNA microarray image segmentation

Yang Li

Louisiana Tech University

Follow this and additional works at: <https://digitalcommons.latech.edu/dissertations>



Part of the [Applied Statistics Commons](#), [Bioinformatics Commons](#), and the [Mathematics Commons](#)

Recommended Citation

Li, Yang, "" (2014). *Dissertation*. 251.

<https://digitalcommons.latech.edu/dissertations/251>

This Dissertation is brought to you for free and open access by the Graduate School at Louisiana Tech Digital Commons. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Louisiana Tech Digital Commons. For more information, please contact digitalcommons@latech.edu.

**IMPROVEMENTS ON SEGMENT BASED CONTOURS METHOD
FOR DNA MICROARRAY IMAGE SEGMENTATION**

by

Yang Li, B.S., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements of the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2014

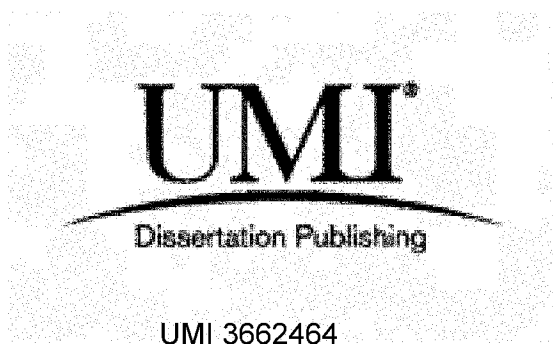
UMI Number: 3662464

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.

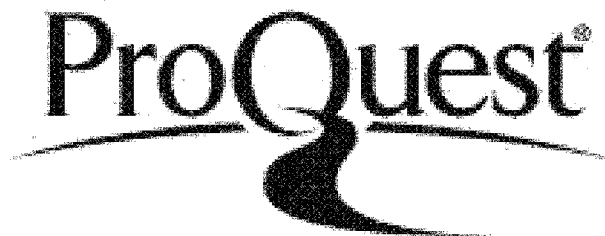


UMI 3662464

Published by ProQuest LLC 2015. Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

January 17, 2014

Date

We hereby recommend that the dissertation prepared under our supervision
by Yang Li

entitled Improvements on Segmentation Based Contour Method for DNA Microarray
Image Segmentation

be accepted in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy



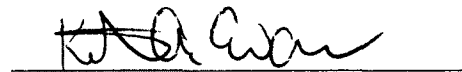
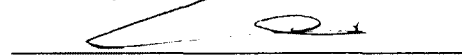
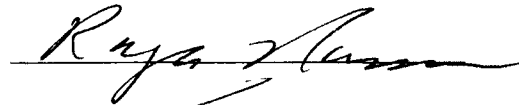
Supervisor of Dissertation Research



Head of Department

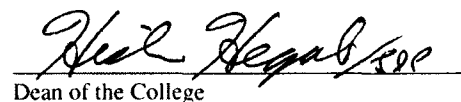
Mathematics and Statistics
Department

Recommendation concurred in:

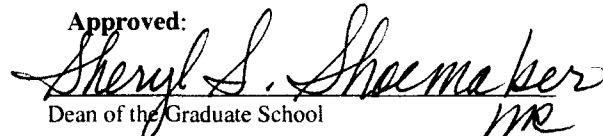


Advisory Committee

Approved:


Director of Graduate Studies
Dean of the College

Approved:


Dean of the Graduate School

ABSTRACT

DNA microarray is an efficient biotechnology tool for scientists to measure the expression levels of large numbers of genes, simultaneously. To obtain the gene expression, microarray image analysis needs to be conducted. Microarray image segmentation is a fundamental step in the microarray analysis process. Segmentation gives the intensities of each probe spot in the array image, and those intensities are used to calculate the gene expression in subsequent analysis procedures. Therefore, more accurate and efficient microarray image segmentation methods are being pursued all the time.

In this dissertation, we are making efforts to obtain more accurate image segmentation results. We improve the Segment Based Contours (SBC) method by implementing a higher order of finite difference schemes in the partial differential equation used in our mathematical model. Therefore, we achieved two improved methods: the 4th order method and the 8th order method. The 4th order method could be applied to segment both the cDNA microarray images and the Affymetrix GeneChips, while the 8th order method could be applied to segment only the cDNA microarray images, due to the limitation of the current image resolution.

The mathematical derivation shows that both our 4th order method and 8th order method are better approximating the C-V model [Chan & Vese, 2001] than the SBC method, which means they will offer more accurate segmentation results than the SBC

method. Besides mathematical proof, we do the practical experiments to double check the conclusion drawn from the mathematical derivation. Both the 4th order method and the 8th order method are used to segment microarray images, and the output segmentation results—the intensities of each probe cell in the microarray image—are being compared to the results from the SBC method and two other mainstream microarray image segmentation methods, the Globally Optimal Geodesic Active Contours (GOGAC) method and the GeneChip Operating System (GCOS) software, for more valid evaluation.

To give the ground true values of intensities as the standard for different segmentation methods comparison, a microarray image simulator is introduced to generate the simulated images used in our experiments. The simulated microarray images have all the characteristics that real microarray images have, and the true intensity values of each probe spot in the image are provided by this simulator. Intensity values segmented by those segmentation methods are compared to the true intensity values. Therefore, we could evaluate that one segmentation method is more accurate than the other methods if its intensity values are closer to the true values.

We conduct several analysis procedures in the segmentation results comparison part to convince our analysis results. Intensity analysis, paired t-test and Unweighted Pair Group Method with Arithmetic Mean (UPGMA) hierarchy cluster experiments are applied to analyze intensity values of those methods. The segmentation output analysis results show that our 4th order method and the 8th order method could offer more accurate segmentation than the SBC method, the GCOS method and the GOGAC method on some kinds of the microarray images. There are accuracy improvements achieved with the 8th order method over the 4th order method on the cDNA microarray image. On the Bovine

type Affymetrix GeneChip image, there is no significant difference between the 4th order method and the 8th order method.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Thesis. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis. Further, any portions of the Thesis used in books, papers, and other works must be appropriately referenced to this Thesis.

Finally, the author of this Thesis reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis.

Author Yang Li

Date 7/29/2014

DEDICATION

For my parents and husband, who have always been supportive and caring.

TABLE OF CONTENTS

ABSTRACT.....	iii
DEDICATION.....	vii
LIST OF TABLES.....	xi
LIST OF FIGURES	xiv
LIST OF ABBREVIATIONS.....	xvi
NOMENCLATURE	xvii
ACKNOWLEDGMENTS	xviii
CHAPTER 1 INTRODUCTION	1
1.1 Overview	1
1.2 Objective of the Research.....	2
1.3 Organization of the Dissertation.....	2
CHAPTER 2 BACKGROUND AND PREVIOUS WORK.....	4
2.1 DNA and RNA	4
2.2 DNA Microarray and DNA Microarray Image Analysis	8
2.2.1 cDNA Microarray	9
2.2.2 Affymetrix GeneChip Microarray	11
2.2.3 DNA Microarray Image Analysis.....	14
2.3 Previous Work on DNA Image Segmentation Methods.....	15
2.3.1 The GOGAC Segmentation Method.....	15
2.3.2 The GCOS Segmentation Software	17

CHAPTER 3 SIMULATED DNA MICROARRAY IMAGES	20
3.1 DNA Microarray Image Simulator	20
3.2 Dataset for Microarray Image Simulation	26
3.3 Simulation on cDNA Microarray Image and Affymetrix GeneChip Image.....	27
3.3.1 Simulation on cDNA Microarray Image	27
3.3.2 Simulation on Affymetrix GeneChip Image.....	28
CHAPTER 4 THE 4 TH ORDER METHOD AND THE 8 TH ORDER METHOD.....	32
4.1 Active Contours Without Edges Method.....	32
4.2 Segmentation Based Contours Method.....	36
4.3 The 4 th Order Method and the 8 th Order Method.....	38
4.3.1 The 4 th Order Method.....	41
4.3.2 The 8 th Order Method	43
4.4 The Comparison of the Performance between the Modified ACWE Method, the SBC Method, the 4 th Order Method and the 8 th Order Method	45
4.5 Application of the Algorithms on the Segmentation of Simulated DNA Microarray Images	46
CHAPTER 5 SEGMENTATION RESULTS ON SIMULATED CDNA MICROARRAY IMAGES	49
5.1 Segmentation Output Intensities Comparison	49
5.2 Statistical Analysis on Segmentation Output Intensities	52
CHAPTER 6 SEGMENTATION RESULTS ON SIMULATED AFFYMETRIX GENECHIP IMAGES	56
6.1 Segmentation Results on Simulated Good Affymetrix GeneChip Image	57
6.1.1 Segmentation Output Intensities Comparison on Simulated Good Affymetrix GeneChip Image	57
6.1.2 Statistical Analysis on Segmentation Output Intensities on Simulated Good Affymetrix GeneChip Image.....	59
6.2 Segmentation Results on Simulated Bad Affymetrix GeneChip Image.....	61

6.2.1 Segmentation Output Intensities Comparison on Simulated Bad Affymetrix GeneChip Image	61
6.2.2 Statistical Analysis on Segmentation Output Intensities on Simulated Bad Affymetrix GeneChip Image	63
6.3 Segmentation Results on Simulated Expanded Affymetrix GeneChip Images.....	65
6.3.1 Segmentation Results on Simulated Good Expanded Affymetrix GeneChip Image	65
6.3.1.1 Segmentation Output Intensities Comparison on Simulated Good Expanded Affymetrix GeneChip image	65
6.3.1.2 Statistical Analysis on Segmentation Output Intensities on Simulated Good Expanded Affymetrix GeneChip Image.....	68
6.3.2 Segmentation Results on Simulated Bad Expanded Affymetrix GeneChip Image	69
6.3.2.1 Segmentation Output Intensities Comparison on Simulated Bad Expanded Affymetrix GeneChip Image	70
6.3.2.2 Statistical Analysis on Segmentation Output Intensities on Simulated Bad Expanded Affymetrix GeneChip Image	72
CHAPTER 7 CONCLUSIONS AND FUTURE WORK	74
APPENDIX A CLASS CODE FOR THE 8 TH ORDER SEGMENTATION METHOD	77
REFERENCES	95

LIST OF TABLES

Table 2.1: Pixel matrix for one probe spot in an Affymetrix Genechip image	19
Table 3.1: List of noise parameters.....	22
Table 3.2: List of slide manufacturing parameters.	23
Table 3.3: List of hybridization parameters.....	24
Table 3.4: List of scanning parameters.	25
Table 4.1: Comparison in respect to the order of error term in the numerical equation of the C-V model between four segmentation methods.....	46
Table 5.1: Performance of the GOGAC method, the SBC method, the 4 th order method, and the 8 th order method on the simulated cDNA microarray image. The 4 th order method and the 8 th order method are with 100 iterations.....	50
Table 5.2: Performance of the GOGAC method, the SBC method, the 4 th order method, and the 8 th order method on the simulated cDNA microarray image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	51
Table 5.3: Performance of the GOGAC method, the SBC method, the 4 th order method, and the 8 th order method on the simulated cDNA microarray image. The 4 th order method and the 8 th order method are with 10,000 iterations.....	51
Table 5.4: Paired t-test result on the GOGAC method, the SBC method, the 4 th order method, and the 8 th order method on the simulated cDNA microarray image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	54
Table 6.1: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated good Affymetrix GeneChip image. The 4 th order method is with 100 iterations	58

Table 6.2: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated good Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	58
Table 6.3: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated good Affymetrix GeneChip image. The 4 th order method is with 10,000 iterations	58
Table 6.4: Paired t-test result on the GCOS method, the SBC method, and the 4 th order method on the simulated good Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	60
Table 6.5: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated bad Affymetrix GeneChip image. The 4 th order method is with 100 iterations	62
Table 6.6: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated bad Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	62
Table 6.7: Performance of the GCOS method, the SBC method, and the 4 th order method on the simulated bad Affymetrix GeneChip image. The 4 th order method is with 10,000 iterations	63
Table 6.8: Paired t-test result on the GCOS method, the SBC method, and the 4 th order method on the simulated bad Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	64
Table 6.9: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated good expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 100 iterations.....	66
Table 6.10: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated good expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	66
Table 6.11: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated good expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 10,000 iterations.....	67

Table 6.12: Performance of the SBC method, and the 4 th order method on the top-left corner of the original simulated good Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	67
Table 6.13: Paired t-test result on the SBC method, the 4 th order method, and the 8 th order method on the simulated good expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	68
Table 6.14: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated bad expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 100 iterations.....	70
Table 6.15: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated bad expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	71
Table 6.16: Performance of the SBC method, the 4 th order method, and the 8 th order method on the simulated bad expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 10,000 iterations.....	71
Table 6.17: Performance of the SBC method, and the 4 th order method on the top-left corner of the original simulated bad Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations	72
Table 6.18: Paired t-test result on the SBC method, the 4 th order method, and the 8 th order method on the simulated bad expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations.....	73

LIST OF FIGURES

Figure 2.1: Building block of DNA from [Alberts <i>et al.</i> , 2002]	4
Figure 2.2: The structure of DNA from Encyclopædia Britannica, Inc., 2007	5
Figure 2.3: DNA transcription from Pearson Education, Inc., 2012	7
Figure 2.4: From DNA to protein [DNA to protein from Northeastern University]	7
Figure 2.5: cDNA microarray (left) and cDNA microarray image (right)	10
Figure 2.6: cDNA microarray experiment process [cDNA microarray experiment by Jeremy Buhler, 1998]	11
Figure 2.7: Affymetrix GeneChip (left) and part of its image (right)	12
Figure 2.8: GeneChip expression array design from Affymetrix, Inc	13
Figure 2.9: GeneChip experiment process from Affymetrix, Inc	14
Figure 2.10: The Spot interface window	17
Figure 2.11: GCOS microarray image analysis flow	18
Figure 3.1: Block diagram of the microarray image simulation model from [Nykter <i>et al.</i> , 2006]	21
Figure 3.2: A simulated cDNA microarray image	28
Figure 3.3: The top-left corner of a simulated Affymetrix GeneChip image	29
Figure 3.4: A simulated expanded Affymetrix GeneChip image	31
Figure 5.1: UPGMA hierarchy cluster result of comparing the GOGAC method, the SBC method, the 4 th order method, and the 8 th order method on the simulated cDNA microarray image. The 4 th order method and the 8 th order method are with 1,000 iterations	55

Figure 6.1: UPGMA hierarchy cluster result of comparing the GCOS method, the SBC method, and the 4 th order method on the simulated good Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations.....	61
Figure 6.2: UPGMA hierarchy cluster result of comparing the GCOS method, the SBC method, and the 4 th order method on the simulated bad Affymetrix GeneChip image. The 4 th order method is with 1,000 iterations.....	64
Figure 6.3: UPGMA hierarchy cluster result of comparing the SBC method, the 4 th order method, and the 8 th order method on the simulated good expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations... ..	69
Figure 6.4: UPGMA hierarchy cluster result of comparing the SBC method, the 4 th order method, and the 8 th order method on the simulated bad expanded Affymetrix GeneChip image. The 4 th order method and the 8 th order method are with 1,000 iterations... ..	73

LIST OF ABBREVIATIONS

ACWE	Active contours without edges
SBC	Segment based contours
GCOS	GeneChip operation system
GOGAC	Globally optimal geodesic active contours
UPGMA	Unweighted pair group method with arithmetic mean

NOMENCLATURE

Ω	A bounded open subset of \mathbb{R}^2
$\bar{\Omega}$	Closure of Ω
$\partial\Omega$	Boundary of Ω
u_0	image
C	Evolving curve
ϕ	Level set function
$\phi_{i,j}^n$	$\phi_{i,j}^n = \phi(n\Delta t, x_i, y_j)$, numerical solution of ϕ
h	Space step
Δt	Time step
$\nabla\phi$	$\nabla\phi = \left\langle \frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y} \right\rangle$
$ \nabla\phi $	$ \nabla\phi = \sqrt{\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2}$
$div(\nabla\phi)$	$div(\nabla\phi) = \nabla \cdot (\nabla\phi) = \frac{\partial}{\partial x} \frac{\partial\phi}{\partial x} + \frac{\partial}{\partial y} \frac{\partial\phi}{\partial y}$
c_1, c_2	Averages of u_0 inside C and outside C
$\mu, \nu, \lambda_1, \lambda_2$	Constant coefficients
H	Heaviside function
δ_0	One dimensional Dirac measure, $\delta_0 = H'$
\vec{n}	The exterior normal to the boundary $\partial\Omega$
$\frac{\partial\phi}{\partial n}$	The normal derivative of ϕ at the boundary

ACKNOWLEDGMENTS

I would like to give special thanks to my advisor, Dr. Mihaela Paun, for her patience, knowledge and encouragement while helping me with my research. I also want to thank Dr. Weizhong Dai, Dr. Bogdan Strimbu, Dr. Raja Nassar, Dr. Katie Evans, Dr. Box Chokchai Leangsuksun, Dr. Andrew Paun, and Dr. Songming Hou for their guidance.

Finally, my research would have been impossible without the help of Yuan Cheng, Shenghua Ni, Yifan Wang, and Fei Han.

CHAPTER 1

INTRODUCTION

1.1 Overview

DNA microarray is an efficient biotechnology tool for scientists to measure the expression levels for a large number of genes, simultaneously. There is a collection of DNA spots attached to the surface of the microarray, and each spot contains a specific DNA sequence known as a probe. Labeled target DNA sequences are hybridized to these probes, and this probe-target hybridization is used to detect and quantify the associative gene expression [Roger, 2013].

To obtain the performance of the gene expression, we need to analyze the DNA microarray image. There are three principle steps for DNA microarray image analysis: Addressing, Segmentation, and Information extraction. The research in this dissertation is focused on the segmentation step, which is to identify the foreground of each DNA spot in the image from the background of each spot. To be more specific, segmentation is to detect the boundary of the bright part of each spot. The GeneChip Operating System (GCOS) by Affymetrix, Inc. and the Global Optimal Geodesic Active Countours (GOGAC) method [Appleton & Talbot, 2006] are the two most widely used microarray image segmentation methods in the world. More detailed descriptions of these two methods are described in Section 2.3.

Since segmentation is a fundamental step in the microarray image analysis and the segmentation accuracy has significant influence on the subsequent gene expression generation, more accurate and efficient segmentation algorithms are being pursued all the time. The SBC method [Ni *et al.*, 2009] is modified from the ACWE method [Chan & Vese, 2001], which has been shown to be more accurate in terms of segmentation than the GCOS method and the GOGAC method.

1.2 Objective of the Research

The objective of this research is to develop more accurate microarray image segmentation methods than the SBC method [Ni *et al.*, 2009] based on the current resolution of the microarray images. To achieve this objective, the following research plan is pursued:

- (1) Develop the numerical algorithms of our improved segmentation method and deduce the truncation errors for the numerical approximation.
- (2) Simulate cDNA microarray images and Affymetrix GeneChip images, which are used to evaluate our improved segmentation methods.
- (3) Apply our two improved DNA microarray image segmentation methods to the simulated images, and then compare the performance of our method to the SBC, the GCOS, and the GOGAC methods.

1.3 Organization of the Dissertation

Chapter 2 introduces the fundamental information about DNA microarray and DNA microarray image analysis procedures. Two widely used DNA microarray image

segmentation methods are mentioned. We use these discussions to help understand our research in the subsequent chapters.

Chapter 3 describes how we simulated the cDNA microarray image and Affymetrix GeneChip images that are used in our experiment. The image simulator [Nykter, 2006] can generate microarray images with all the realistic characteristics that a real microarray image contains. What is more important is that this simulator gives the ground true intensity values of each spot in the image. Therefore, we can compare our improved method with different segmentation algorithms.

In Chapter 4, we give a detailed description on how we improved the SBC method to obtain our two methods: the 4th order method and the 8th order method. The reason why we use the fourth order forward, backward, central finite difference schemes, and the eighth order forward, backward, central finite difference schemes to implement the C – V model [Chan & Vese, 2001] is discussed. The associated truncation errors in terms of space and time are deduced to evaluate the numerical approximation accuracy.

In Chapter 5, we apply the 4th order and the 8th order method to a simulated cDNA microarray image. Intensity analysis, paired t-test and UPGMA hierarchy cluster are implemented to compare our methods to the GOGCA method and the SBC method.

In Chapter 6, we apply the 4th order method to two simulated Affymetrix GeneChip images. To evaluate how the 4th order method and the 8th order method perform on Affymetrix arrays, we apply them to two simulated expanded Affymetrix GeneChip images. The same intensity analysis, paired t-test and UPGMA hierarchy cluster are implemented to compare our methods to the GCOS method and the SBC method. Conclusions and future work are addressed in Chapter 7.

Chapter 2

BACKGROUND AND PREVIOUS WORK

2.1 DNA and RNA

DNA is a double-stranded molecule of genetic material that stores information regarding its own replication and the order in which amino acids are to be joined to make a protein [Mader, 2010]. All living cells on Earth, without any known exception, store their hereditary information in DNA [Alberts *et al.*, 2002] [Sheeler & Bianchi, 1980].

DNA is a long unbranched pair of polymer chains always formed from the same four types of monomers: A, T, C, and G. These monomers are strung together in a long linear sequence that encodes the genetic information. Each molecule, that is, each nucleotide, consists of two parts: a sugar (deoxyribose) with a phosphate group attached to it, and a base, which may be either adenine (A), guanine (G), cytosine (C), or thymine (T), as shown in Figure 2.1.

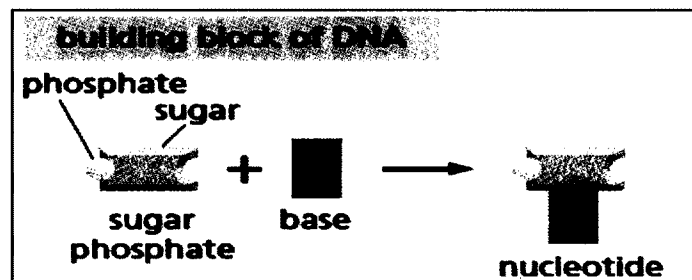


Figure 2.1: Building block of DNA from [Alberts *et al.*, 2002].

Each sugar is linked to the next via the phosphate group, creating a polymer chain composed of a repetitive sugar-phosphate backbone with a series of bases protruding from it. A single DNA strand is formed in this way. The bases protruding from the existing strand bind to the bases of another strand being synthesized, according to a strict rule defined by the complementary structures of the bases: A binds to T and C binds to G. In this way, a double-stranded structure is created, consisting of two exactly complementary sequences [Alberts *et al.*, 2002]. The two strands twist around each other, forming a double helix, as shown in Figure 2.2.

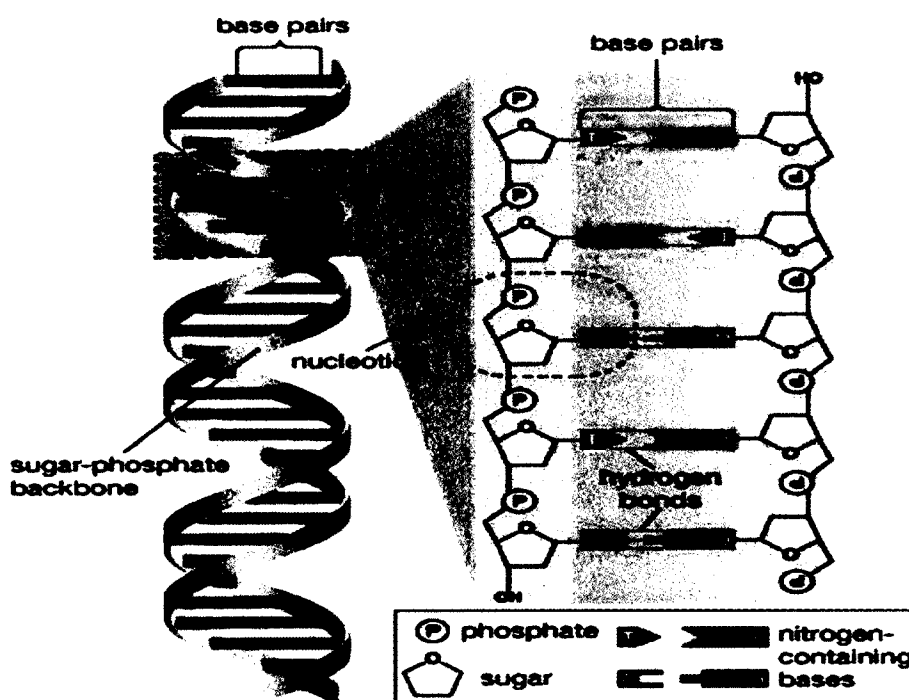


Figure 2.2: The structure of DNA from Encyclopædia Britannica, Inc., 2007.

The replication process allows the DNA to make a copy of itself. During the process the base pairs of the two strands in a DNA open and each strand acts as a

template. Two complement strands are reproduced which achieve the DNA duplication process.

In order to carry the genomic information, the DNA sequence must undergo the process of replication and transcription with the help of RNA (ribonucleic acid) and protein. RNA has the similar intermediary structure with the DNA strand stored in the cytoplasm. There are, however, some differences in RNA compared with DNA. In RNA, the backbone is formed by ribose instead of deoxyribose. In addition, those four bases are the same with one exception: U (uracil) replaces T (thymine) [Alberts *et al.*, 2002] [Sheeler & Bianchi, 1980]. Thus, in RNA, A is paired with U and C is paired with G.

This process starts from the transcription, as the DNA sequence is treated as the template for RNA synthesis. The genetic information in a specific sequence is transferred into a complementary special sequence of messenger RNA (mRNA) as seen in Figure 2.3. Three bases in RNA transcripts are considered as the genetic code called “codon.” Several of these triplet codons guide the synthesis of polymers of protein, which is the translation process. Thus, from DNA to protein, hereditary information is deciphered, as shown in Figure 2.4.

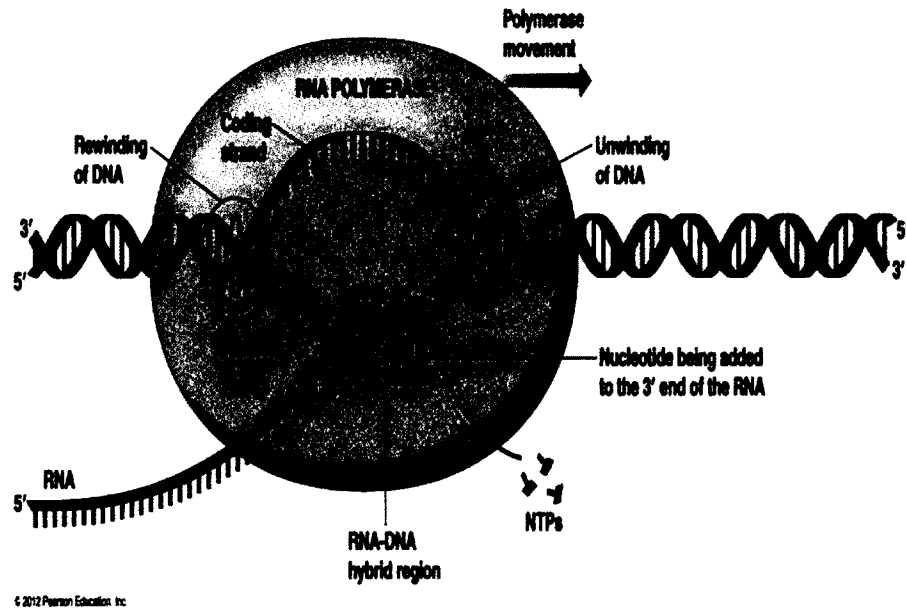


Figure 2.3: DNA transcription from Pearson Education, Inc., 2012.

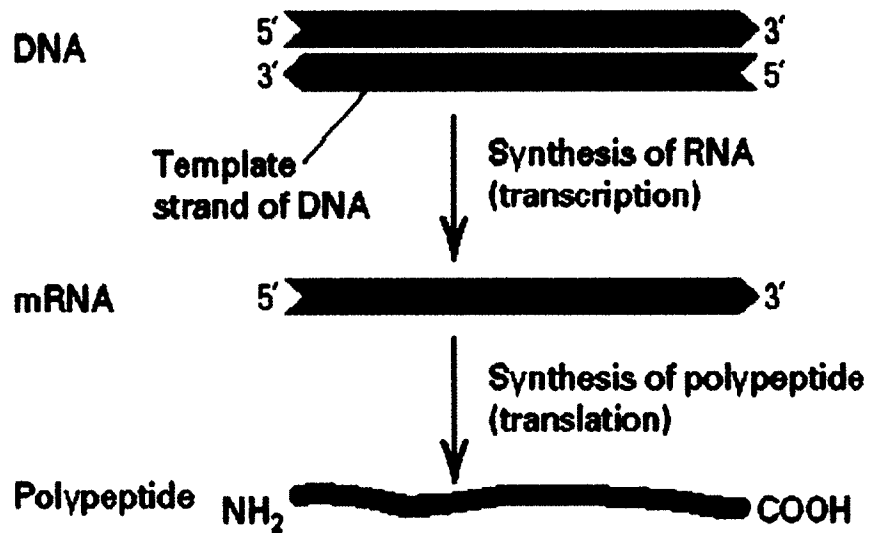


Figure 2.4: From DNA to protein [DNA to protein from Northeastern University].

Each DNA sequence experiences three stages: the replication, the transcription and the translation, and genetic information is passed down through this process. The

subsequence of DNA that is transferred into the protein is called a “gene” [Lakhotia, 1997]. Thus, this process is called the “gene expression”. In the genetics field, gene expression is the most significant and basic foundation for transforming the genotype to the phenotype. Different organism phenotype is caused by controlling the different properties of the gene expression [Rockman & Kruglyak, 2006]. By using DNA microarray technology, scientists are able to monitor and manage thousands of genes’ expressions, simultaneously.

2.2 DNA Microarray and DNA Microarray Image Analysis

DNA microarray is an efficient biotechnology tool for scientists to monitor thousands of genes’ expressions, simultaneously. It is a tiny silicon chip with a collection of DNA spots attached to its surface. Each spot contains a specific DNA sequence known as a probe. Labeled target DNA sequences are hybridized to these probes. This probe-target hybridization is used to detect and quantify the associative gene expression [Roger, 2013]. With the manufacturing method, there are two types of DNA microarrays: spotted microarray and oligonucleotide microarray.

Spotted microarrays are cheap and the polymerase chain reaction (PCR) method is used to produce the sequences on the array spots. The probes in the arrays are long cDNA sequences. Oligonucleotide microarrays are expensive and the spot probes in the arrays are short oligonucleotide sequences. For the spotted array, the DNA sequence may or may not be known, and there is little control of the amount of DNA in a spot. For the oligonucleotide array, the DNA sequence is known as a perfect match (PM) and mismatch (MM). PM and MM are paired and used as controls of DNA. Since an

oligonucleotide array has more probe controls in the microarray than that of the spotted array, the oligonucleotide microarray is more efficient than the spotted microarray; this is the same reason why the oligonucleotide microarray is more expensive than the spotted array.

2.2.1 cDNA Microarray

cDNA microarray is a kind of spotted microarray, as shown in Figure 2.5. To make a cDNA microarray, the RNA sequence from both the control sample (normal sample) and the experimental sample (diseased sample) are isolated. Next, reverse transcription process is operated, which allows it to convert the RNA sequences of interest into cDNAs. After the reverse transcription, the cDNAs will be labeled with fluorescent probes, Cy3 for the control sample, and Cy5 for the experiment sample. The Cy3 is in a green channel with 530 nm wave length, and Cy5 is in a red channel with 630 nm wavelength [Yang *et al.*, 2002]. When finishing the labeling process, cDNA microarray is scanned both at the ~540 nm and ~630 nm for each channel, respectively. Two 16-bit monochromatic images are generated after scanning, which are red and green images, as shown in Figure 2.6. In these two images, each spot represents a specific gene [Gohlmann & Talloen, 2009][cDNA microarray experiment from SQL, 2006].

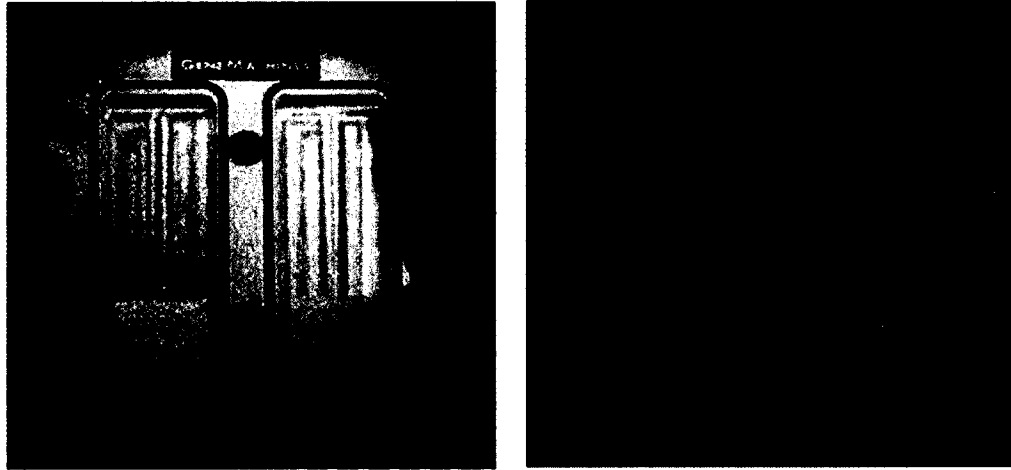


Figure 2.5: cDNA microarray (left) and cDNA microarray image (right).

Typically, a cDNA microarray experiment [cDNA microarray experiment from SQL, 2006][cDNA microarray experiment by Jeremy Buhler, 1998] includes the following six steps, as shown in Figure 2.6:

- (1) In the sample preparation step, a normal sample and a disease sample are selected.
- (2) In the nucleic acid isolation and purification step, the mRNA sequences of the two samples are extracted.
- (3) In the reverse transcription step, mRNAs are transcribed to cDNAs.
- (4) In the hybridization step, the cDNAs are tagged with fluorescent dye. Tagged cDNA sequences are hybridized to a microarray. The excess tagged cDNAs are washed away from the microarray.
- (5) In the laser scanning step, the microarray is scanned in two channels.
- (6) In the analysis step, the spot intensities are generated and the gene expression analysis is implemented.

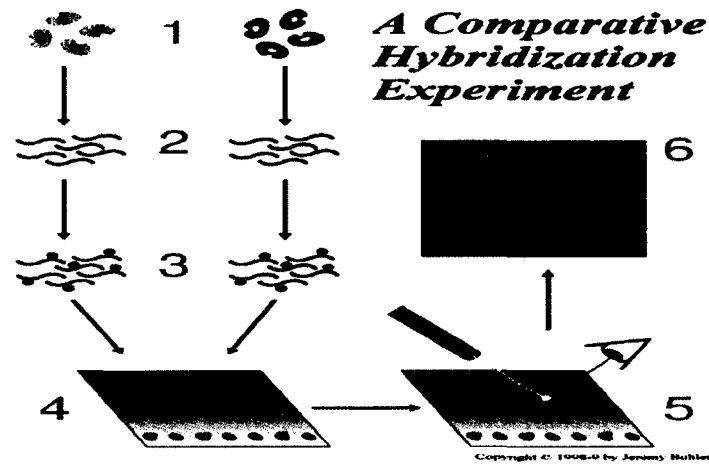


Figure 2.6: cDNA microarray experiment process [cDNA microarray experiment by Jeremy Buhler, 1998].

2.2.2 Affymetrix GeneChip Microarray

The Affymetrix GeneChip is a kind of oligonucleotide microarray, as shown in Figure 2.7. Mentioned in the late 1980s, Fodor *et al.* introduced the semi-conductor technique for a biological setting in the microarray fabrication process. This process helped to construct a system to measure more and more various mRNA sequences in one sample. In addition, Affymetrix microarray introduced small oligonucleotide sequences (probes) containing 25-nucleotides located variously in their sequence composition. These small probes could bring a better discrimination between similarly related transcripts over long oligonucleotides, especially when mRNAs are highly abundant [Cheng, 2013].

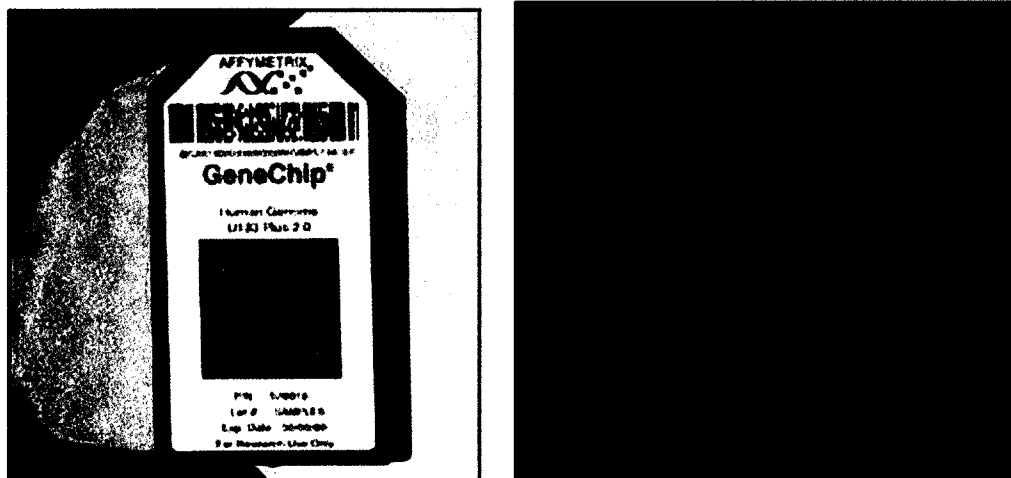


Figure 2.7: Affymetrix GeneChip (left) and part of its image (right).

The Affymetrix uses a probe that is paired with the Perfect Match (PM) and the Mismatch (MM), as shown in Figure 2.8. These two probes are exactly the same, except for the one base in the middle. To illustrate, PM has 25-nucleotides, which are perfectly hybridized to the mRNA sequences, whereas MM has the same 25-nucleotides, except the only one base in the middle of the 25 bases that is different from what the PM has. Each PM should be uniquely different from each other. In this case, false signals transcription caused by similar complete sequences were completely eliminated, and MM was used to help scientists to learn and control the unspecific signal and background signal.

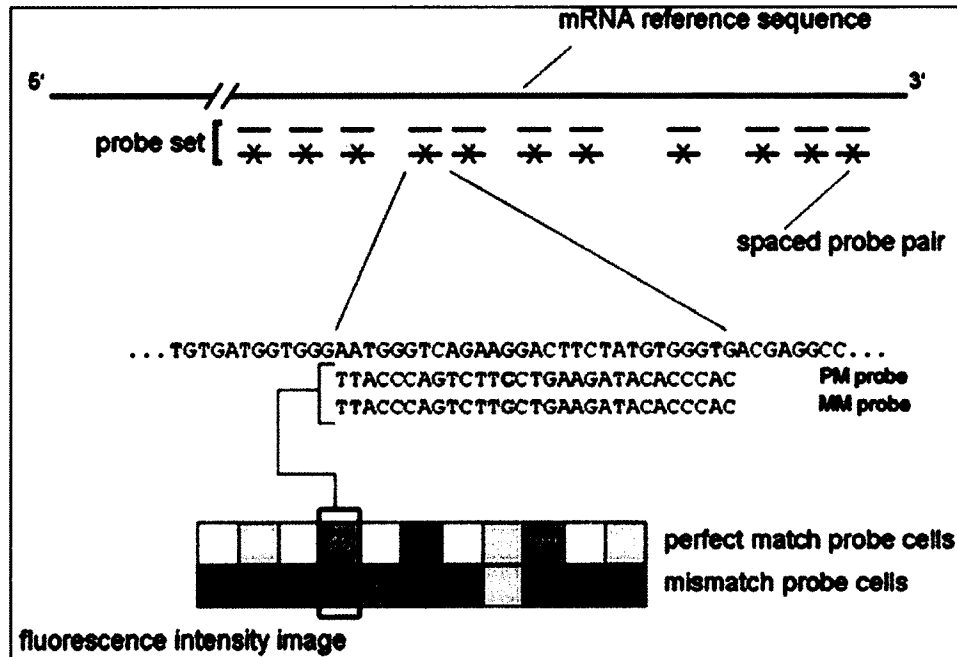


Figure 2.8: GeneChip expression array design from Affymetrix, Inc.

Normally, an Affymetrix experiment contains the following six steps, as shown in Figure 2.9:

- (1) First, a sample of interest is selected.
- (2) The RNA sequences are isolated and purified. After checking the quality of RNA sequences, good quality RNA sequences are labeled. These mRNAs experience the reverse transcription to cDNA.
- (3) In Vitro Transcription (IVT), the cDNA sequences are transcribed to cRNA sequences, and these cRNA sequences are labeled and fragmented to short pieces.
- (4) Hybridization is performed on the gene microarray platform under specific temperature and hours.
- (5) After complete hybridization, the microarray is scanned by a special laser, generating the Affymetrix GeneChip image in 16-bit gray level.

(6) The intensity of each pixel on the chip is recorded according to the emission of the fluorescent dye.

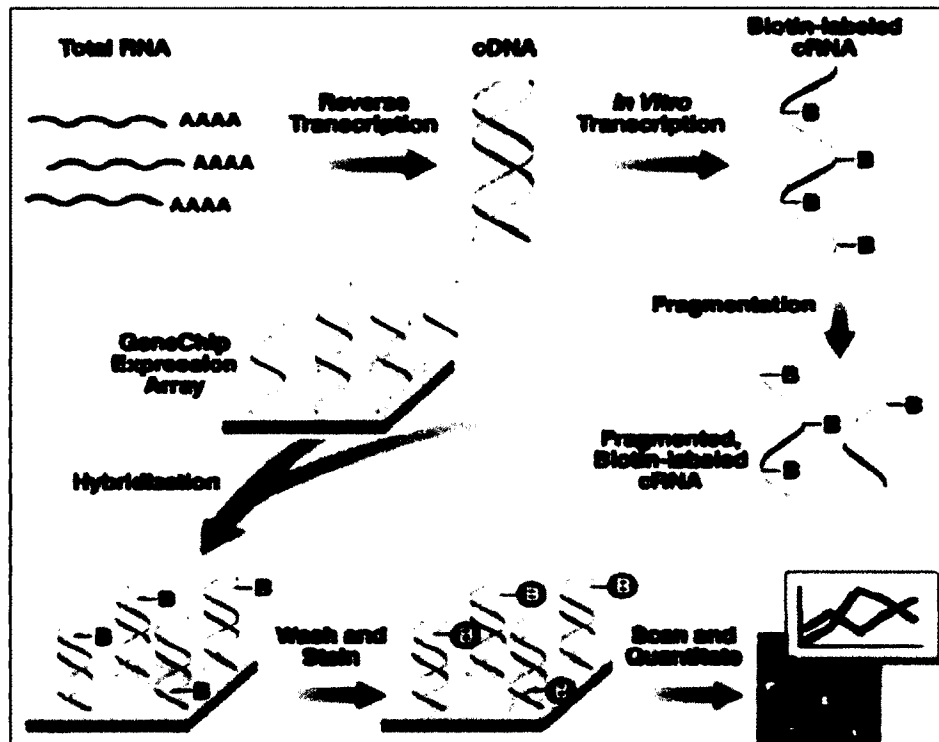


Figure 2.9: GeneChip experiment process from Affymetrix, Inc.

2.2.3 DNA Microarray Image Analysis

There are three principle procedures to analyze a DNA microarray image [Yang *et al.*, 2002]: Addressing, Segmentation, and Information extraction.

When we deal with an image, we need to transfer the visualized image into digital values for calculating and analyzing. Addressing is to find the exact geometry location of each probe spot in the microarray image, and then to arrange each spot into a grid with coordinates.

Since the target DNA sequences are labeled with fluorescent dyes, the hybridization part of each spot will be bright in the image, and the other part will remain dark. Therefore, segmentation is the procedure to identify the bright part, which is the foreground, from the dark part, which is the background. To be more specific, segmentation is to identify the pixels either as the foreground or as the background and get the exact boundary of the foreground pixels.

Information extraction is to generate the foreground and the background intensity value of each spot, and then summarize these intensity values into the signal values to analyze the associative gene expression.

2.3 Previous Work on DNA Image Segmentation Methods

There are two widely used DNA microarray image segmentation methods: one is the GOGAC method [Appleton & Talbot, 2006] for cDNA microarray images, and the other one is the GCOS software from Affymetrix, Inc. for Affymetrix GeneChip images.

2.3.1 The GOGAC Segmentation Method

Globally Optimal Geodesic Active Contours (GOGAC) was first proposed in [Appleton & Talbot, 2006]. It is a kind of adaptive shape segmentation technique and can be used on cDNA microarray image segmentation. GOGAC searches the geodesic active contours with globally minimal energy containing an internal point p_{int} . A general algorithm of GOGAC is presented with the following steps:

1. Initialization:

- Assign the root search cut node $R (P_{cut}^{root})$ with ∞ as the lower bound
- Mark P_{cut}^{root} as open
- Enqueue R

2. Priority First Search (infinite loop):

- Delete the search cut node n of the least lower bound from the priority queue
- If n is marked as closed:
 - Assign the minimal closed geodesic corresponding to n
 - Halt
- Calculate the surface of the minimal action U in the helical surface space S

from the start of set n

- Halt the calculation early when at least one element of each end set of χ_1

and χ_2 has been checked

- Find out the end of the geodesic: $p_{end} = \arg \inf \{U(p_{end}) | p_{end} \in P_{end}\}$
- Obtain the minimal geodesic C_{min} and the start point p_{start} for n by gradient

descent from p_{end} to p_{start}

3. For each child χ of the search tree:

- Assign P_{start} , P_{end} to be the start set and end set of χ
- Let χ be a lower bound $\min \{U(p_{end}) | p_{end} \in P_{end}\}$
- Mark χ as closed if p_{start} and p_{end} are both located in χ and are connected

in the discrete grid

- Enqueue χ

The proposed GOGAC algorithm was implemented using Spot software developed by CSIRO, Inc. Spot is a package installed in the R software. The interface of Spot is shown in Figure 2.10. The user needs to create the batch files and set up the parameters and a template by himself, according to the cDNA microarray image that is being segmented. The top left grid point needs to be pointed manually in the image by the user. After the segmentation, a SPOT format file and a JPG format file are generated. The former file contains the intensity value of each spot, and the latter file displays the grid finding and the segmentation results. The weakness of this method is that it prefers to produce circles and cannot prevent overlap.

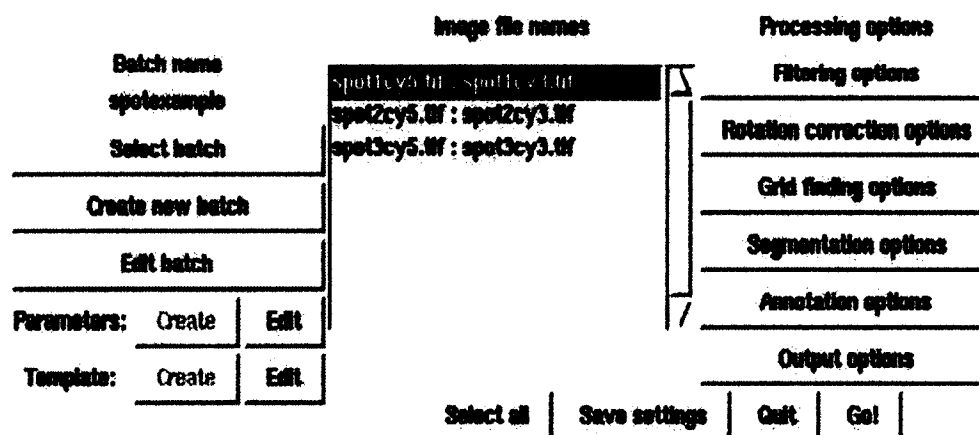


Figure 2.10: The Spot interface window.

2.3.2 The GCOS Segmentation Software

All the Affymetrix GeneChip images can be analyzed by the GeneChip Operating System (GCOS) software, which is developed by Affymetrix, Inc. It provides an intuitive set of tools for instrument control and data management used in the processing of GeneChip Arrays. The software summarizes probe cell intensity data, generates the gene

signal values, and enables sample and array registration, data management, and instrument control as well as automatic and manual image gridding.

The raw image information was stored in a DAT format file and we use the GCOS software to open this DAT format file. Alignment and addressing are automatically performed and intensity values of each probe spots are written into a CEL format file. With the intensity values obtained, the GCOS software implements the MAS5 algorithm to analyze the CEL format file and related CDF format file to calculate the gene signal value for each probe set. This gene signal value is stored in the CHP format file and the TXT format file. One was in a special format in the CHP file. The other one was in text format in the TXT file. The workflow of the GCOS software is illustrated in Figure 2.11.

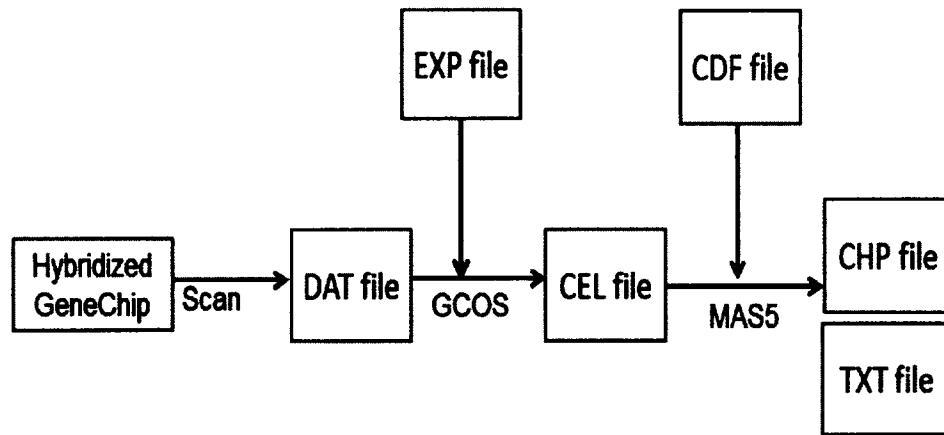


Figure2.11: GCOS microarray image analysis flow.

In a Affymetrix GeneChip image, each probe spot cell contains $n \times n$ pixels depending on the experiment design. For the microarray image segmentation and intensities extraction step, after identifying the position of each probe, the GCOS software omits the outer boundary pixels. Only the inner $(n-1) \times (n-1)$ pixels are

included and considered to be within the foreground area. The GCOS software chooses the 75th percentile of the inner pixels to represent the intensity for each probe.

We illustrate how the GCOS software computes the intensity of one spot in Table 2.1. Table 2.1 contains the pixels' matrix of one spot in the microarray image. The outer highlighted pixels are dropped off by the GCOS software. The remaining 75th percentile of the inner pixels is recorded as the intensity value for the spot. The reason why the GCOS software omits the outer pixels is that it is believed that such pixels are not reliable and may carry some noise and errors, for they may be located by the misalignment in the scanning process, or they may be influenced by the neighboring probes which have a large amount of emission.

Table 2.1: Pixel matrix for one probe spot in a Affymetrix GeneChip image.

256	166	413	301	309	473
294	256	166	234	204	286
166	204	166	256	196	174
196	369	279	458	219	264
181	166	241	286	451	376
234	219	249	376	166	219

Zuzan *et al.* [Zuzan *et al.*, 2001] showed in their research that with the increasing pixel values, the variance would become unstable when choosing the 75th percentile as the probe intensity. It is not robust enough when dealing with different qualities of cells.

CHAPTER 3

SIMULATED DNA MICROARRAY IMAGES

In this chapter, we use a DNA microarray image simulator to simulate the cDNA microarray images and the Affymetrix GeneChip images. The simulated images have all the characteristics that the real microarray images have. More importantly, we can have the ground true intensity values for each spot, which enable us to evaluate the performances of different segmentation methods.

3.1 DNA Microarray Image Simulator

The DNA microarray image simulator, proposed by Nykter *et al.* in 2006, is used to validate different kinds of data analysis algorithms. It can simulate both spotted two-channel and oligonucleotide one-channel microarrays, by using the true intensity values of each probe spot as an input. This simulator contains all the steps that affect the quality of real microarray data. To illustrate, those steps include the simulation of biological ground truth data, applying biological measurement technology specific error models, and simulating the microarray slide manufacturing and hybridization. With this in mind, the simulated data has realistic biological and statistical characteristics. Therefore, we use this simulator to simulate the microarray images used in our experiment.

The simulation model [Nykter *et al.*, 2006] contains six main modules, which are data input, slide manufacturing, biological noise, slide hybridization, slide scanning, and image reading, as shown in Figure 3.1. Each module is independent of the others, and can be easily replaced or modified. Each module has several parameters that can be established. By operating the module parameters, the simulation process will provide three different quality images, which are high, normal, and bad. It should be pointed out that the simulator is written into a Matlab program, with each module as a separate function file.

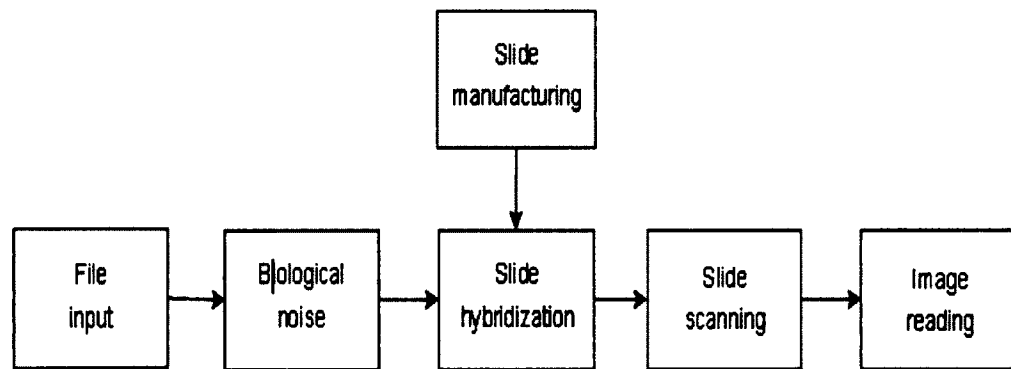


Figure 3.1: Block diagram of the microarray simulation model from [Nykter *et al.*, 2006].

Before we write the input information into the file, we need to set up the module parameters according to the type and quality of the image we want. The available parameters and their values are listed in Tables 3.1- 3.4. Table 3.1 lists the noise options that can help the users control the statistical properties of the data. Table 3.2 lists the slide manufacturing parameters, which have an effect on how the simulated slide looks. For instance, we can set up the number of blocks we want in an image, or the number of

pixels in each direction we want to arrange in each spot. Table 3.3 lists the slide hybridization parameters that control the quality of the slide. We can control the background noise by setting the parameters in this table. Table 3.4 lists the scanning parameters that are used to set up the virtual scanner. For the detailed description of each parameter, we refer the reader to [Nykter, 2006].

Table 3.1: List of noise parameters.

Kernel	Kernel used to model the population effect.
Copies	Number of times the population effect is applied.
Error model	Error model to be used: each error model has its own parameters
Simple noise model	(0.01, 0.001)
SNR noise model	(0, 10)
Dror noise model	(1, 0.01, 0, 36, 13, 0.76, 0, 0.21)
Hartemink noise model	(0.2, 0.01, 1)
Hierarchical error model	(0.012, 0.010, 0.085, 0.094, 0.011)
Rocke noise model	(5, 0.1, 1, 1)
Hein noise model	(0.341, 0.335, 0, 50, 0.5, 1, 0.5, 10)

Table 3.2: List of slide manufacturing parameters.

	Good	Normal	Bad	Affymetrix
Stype	cdna	cdna	cdna	oligo
Sspot	circle	gussian	gaussian	
Spix	12	12	12	10
Smovprob	0.01	0.1	0.5	0.1
Smov	0	1	2	1
S_μ	5	5	5	4
S_{σ^2}	0.001	0.01	0.1	0.01
P	0	1	1	
Pp	0.0	0.5	0.9	
Ph	0	3	3	
Pw	0	2	2	
Pb	0	1	2	
Cprob	0	0.1	0.25	
Cnum	0	4	8	
Ccut	0	3	6	
B	[4,2]	[4,2]	[4,2]	[1,1]
Bspace	50	50	50	
Bcurve	0	1	2	
Bmaxc	0	3	10	

Table 3.3: List of hybridization parameters.

	Good	Normal	Bad	Affymetrix
H_{σ^2}	0.001	0.01	0.1	0.01
Herrors	1	1	1	1
Hbgnoise	10	30	50	20
Hbgvar	0.001	0.01	0.03	
Hbggrad	1	1	1	1
Hnoscratch	0	1	3	0
HSlength	0	0.3	0.9	
HSwidth	0	3	5	
Hnoair	0	1	3	
$H_{\sigma_{air}}$	0	15	30	
H_{σ}	1	10	20	
Hbleed	0	2	10	
Hbleedsize	0	5	10	
Hbleeddist	0	0.4	0.4	

Table 3.4: List of scanning parameters.

	Good	Normal	Bad
Rpower	1	10	20
Rb	16	16	10
Req	0	0	0
Rth	7	5	3
RRch	2	2	2
RGch	1	1	1
Rerrors	0	1	1
Rangle	0	0.1	1
Rmm	0	0	1

After all the parameters are set, we need to prepare the input data as required. There are seven different variables that are required for the input data: data (intensity matrix), time, name, info.genes, info.spots, type, and scale [Nykter *et al.*, 2006]. After these seven variables are saved, the simulator could be run to generate the microarray images as required. Next, we will discuss these variables one by one.

1. **Data:** this variable contained the intensity matrix of each probe spot in the microarray image. Each column corresponds to one sample tissue of the microarray. These intensities are the ground true values that are used to evaluate different segmentation algorithms in Chapter 4 and Chapter 5.

2. Time: this variable contains the time instants for different microarray experiments. Time scale can vary. The total length of this vector should equal the number of rows in the Data matrix. In our experiment, we set time as 1.

3. Name: this string variable indicates the name of the experiment dataset.

4. Info.genes: this array variable stores the name of the genes/probes.

5. Info.spots: this matrix variable contains the locations of each spot in the slide.

6. Type: this string variable indicates the type of the input data: ratios, expression or intensity. Ratios represent the gene expression type. Expression represents the cDNA microarray type. Intensity represents the Affymetrix microarray type.

7. Scale: this string variable indicates the scale of the input data: linear or log. These two options indicate whether the input data is in log scale or linear scale.

3.2 Dataset for Microarray Image Simulation

The original cDNA microarray images can be downloaded from the public website for Stanford University's Yeast Cell Cycle Analysis Project. The cDNA microarray image file and grid file used in our simulation are from Elutriation Experiments, at 390 minutes. The downloaded image is in TIFF format and the grid file contains the location of each spot in the order of left, top, right, and bottom.

The original Affymetrix GeneChip images can be downloaded from the data resource center of the Affymetric Company's website. The Affymetrix GeneChip image file used in our simulation is from Bovine Data file, which contains replicate probe array files for the Bovine Genome Array. The download file is in DAT format.

3.3 Simulation on cDNA Microarray Image and Affymetrix GeneChip Image

3.3.1 Simulation on cDNA Microarray Image

We rewrite the downloaded grid file into a TXT format file with the same order of each spot's location. This TXT file and the downloaded cDNA TIFF format image file are taken as the input of the SBC segmentation method. After the segmentation by the SBC method, we obtained the intensity values of each spot in the image. These intensities are written into the variable data as the input data of the simulator. It should be pointed out that these intensities are also the ground true values for the future evaluation analysis in Chapter 4 and Chapter 5.

In the meantime, the other six input variables are written according to the downloaded grid file and the characteristics of the cDNA microarray. Parameters in different modules are set in the type of cDNA microarray image simulation, with values of good slide quality. After the simulation, we obtain a simulated cDNA microarray image in .tiff format, with the ground true intensity values we know. Figure 3.2 is an example of the simulated cDNA microarray image. Since the simulation will lay the spots in the locations where the associated parameters are set, we can write a new grid file for the simulated cDNA microarray image for further use.

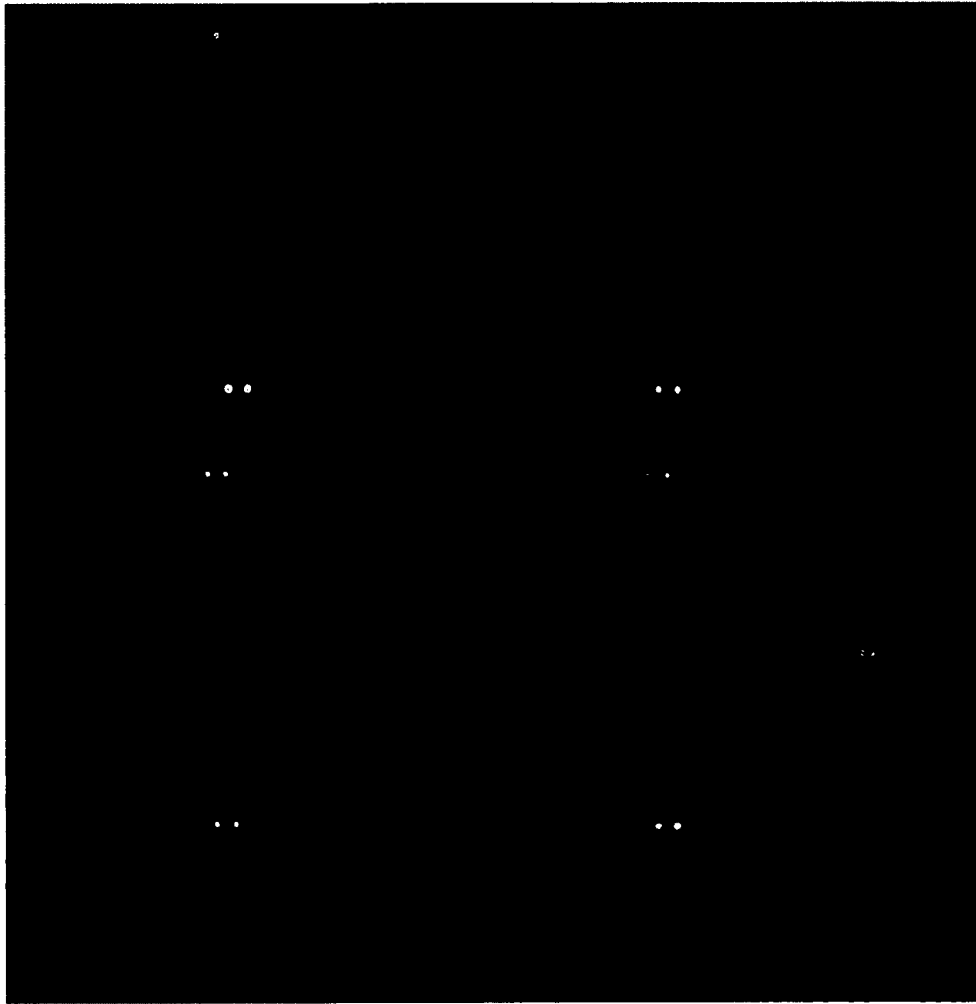


Figure 3.2: A simulated cDNA microarray image.

3.3.2 Simulation on Affymetrix GeneChip Image

We transfer the DAT file downloaded from Affymetrix Company's website into the GCOS software. Therefore, we obtain a DAT file and a CEL file. The DAT file contains the Affymetrix GeneChip image's information like the pixel values of the image. Also, the CEL file contains the intensity values of each probe spot. These intensities are written into the variable Data as the input data of the simulator. A Matlab program written

by Ni is used to extract each probe spot's location information from the DAT file, and to write the location information into a TXT grid file for further use [Ni, 2009].

Similar to the argument for cDNA microarray image simulation, the other six input variables are written according to the grid file and the characteristics of the Affymetrix GeneChip microarray. Parameters in different modules are set in the type of oligonucleotide microarray image simulation, with values of good slide quality. After the simulation, we obtain a simulated Affymetrix microarray image in TIFF format, with the ground true intensity values that we know. Figure 3.3 is an example of the simulated Affymetrix GeneChip image.

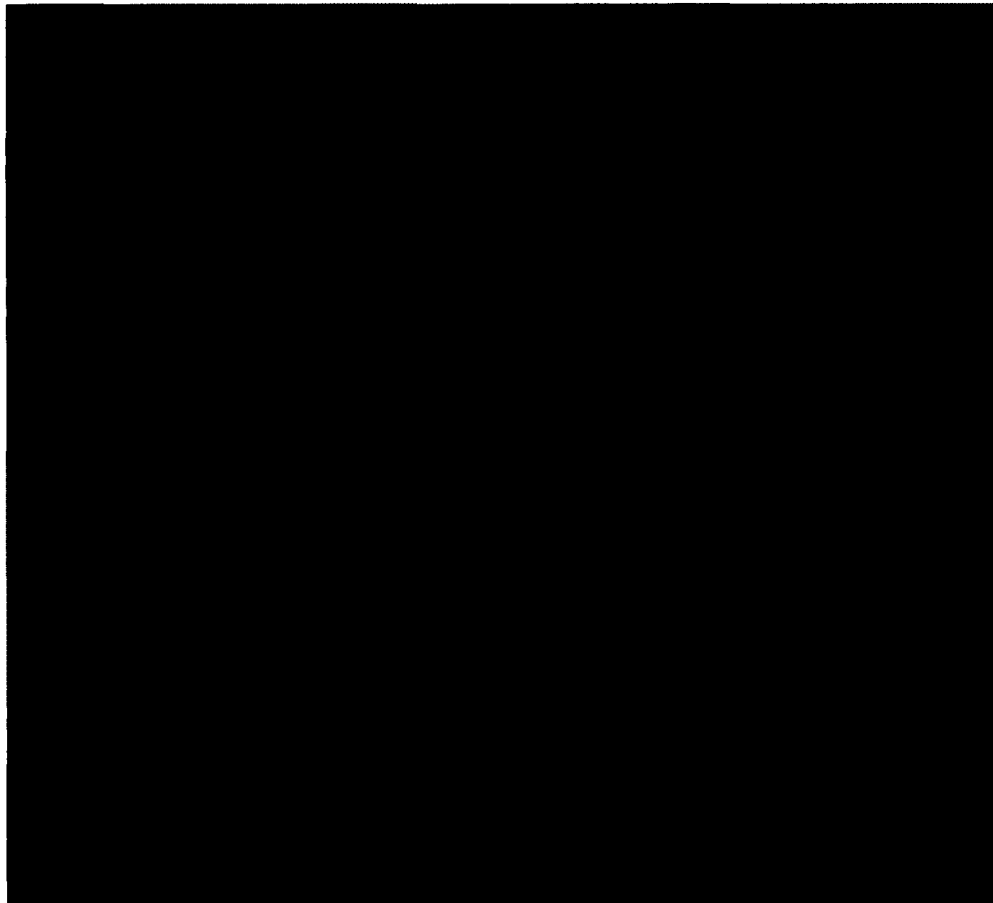


Figure 3.3: The top-left corner of a simulated Affymetrix GeneChip image.

In order to compare the segmentation output intensities by the GCOS method to the true intensity values, we need to use the GCOS software to analyze the simulated GeneChip image. However, the Affymetrix GCOS software cannot analyze the microarray image in the format of JPG or TIFF directly. In order to allow the Affymetrix GCOS software to analyze the simulated image, we need to rewrite the simulated TIFF microarray image file into a DAT file that the GCOS software can read. The DAT file contains the 16-bit grey level image pixel data matrix, header information, layout information, and so on. The pixel data matrix is stored as a 16-bit unsigned integer value at byte 512 following the header. The DAT file for the simulated microarray image contains the same information as the original DAT file except for the image pixel data matrix of the simulated image. Thus, we extract the new simulated pixels data and write them into the original DAT file and keep any other layout information the same. In this case, we get a new DAT file corresponding to the new simulated microarray image. Therefore, we can use the GCOS software to analyze this DAT file and obtain the corresponding CEL file, which contains the segmentation intensities of the simulated GeneChip image.

Due to the resolution of the Affymetrix GeneChip images, we could only apply the 4th order method. In order to compare the performance of the 4th order method and the 8th order method on the Affymetrix Genechip images, we simulate two expanded GeneChip images. We take the one-sixteenth top-left corner of the simulated GeneChip image, and copy each pixel four times in the expanded image to make it twice the resolution than the original Genechip image. Therefore, the two expanded GeneChip

images are 12×12 each. Figure 3.4 is an example of a simulated expanded Affymetrix GeneChip image.

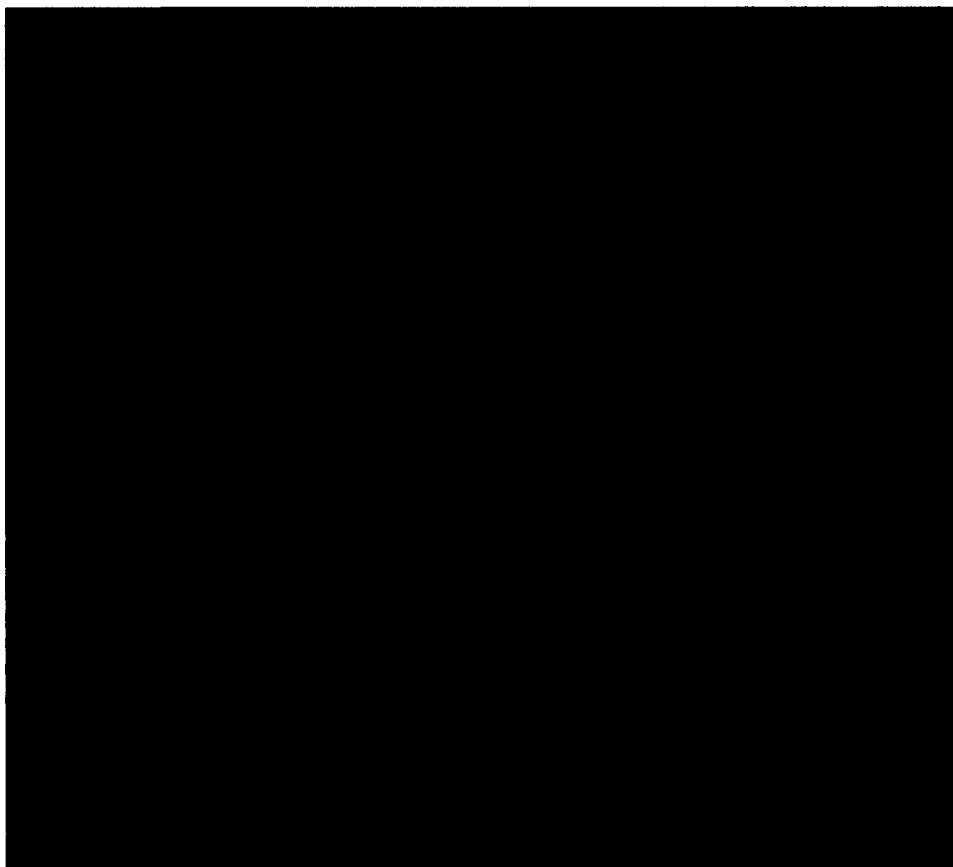


Figure 3.4: A simulated expanded Affymetrix GeneChip image.

CHAPTER 4

THE 4TH ORDER METHOD AND

THE 8TH ORDER METHOD

In this chapter, we first introduce the ACWE method in Section 4.1 and the SBC method in Section 4.2 that we use to obtain our two new methods: the 4th order method and the 8th order method. In Section 4.3, we state the detailed procedures of how we achieve our two methods by improving the SBC method. Improvements of our methods over the SBC method on mathematical derivation are listed in Section 4.4. Parameters definitions and application instruction are stated in Section 4.5.

4.1 Active Contours Without Edges Method

The Active Contours Without Edges (ACWE) model [Chan & Vese, 2001] is a new model for active contours to segment objects in a given image. This method is based on the techniques of the curve evolution, the Mumford-Shah functional for segmentation, and level sets. The authors give a numerical algorithm for the model using finite differences.

The conventional active contours model needs a stopping edge function, which depends on the image gradient, to detect the boundary of an object in an image. The

involving curve stops when the edge function identifies the points where the image brightness changes sharply or there is discontinuity. However, most of these edge functions are sensitive to noise and inaccurate segmentation over the boundary [Lakshmi & Sankaranarayanan, 2010].

The ACWE method Chan and Vese proposed does not depend on the gradient of the image for the stopping process. Its stopping term is based on Mumford–Shah segmentation techniques [Mumford & Shah 1989]. Therefore, the ACWE model can detect contours either with or without gradient. This makes the ACWE method capable of detecting objects with very smooth boundaries or even with discontinuous boundaries. In addition, the ACWE model has a level set formulation, interior contours are automatically detected, and the initial curve can be anywhere in the image [Chan & Vese, 2001].

With all these advantages, the ACWE method is now a frequently used segmentation method for general images. Moelich and Chanin developed a tracking algorithm based on the ACWE segmentation algorithm that is able to handle changes that result from deformations in the object that is tracked [Moelich & Chanin 2003]. Almhdie *et al.* presented a method based on ACWE algorithm as a segmentation method used for mouse brain MRI images [Almhdie *et al.*, 2009], and Salman introduced an image segmentation algorithm based on the ACWE used to extract individual components from a medical image [Salman, 2006] [Yuan, 2013].

The basic idea of the ACWE method [Chan & Vese, 2001] is to assume that the image u_0 can be divided by two regions of approximately piecewise-constant intensities of distinct values: one region inside the object is denoted by the region with the value u_0^i , and the other region outside the object is with the value u_0^o . The defined C_0 is the

boundary of the object. Then inside the objects there exists $u_0 \approx u_0^i$ (or $inside(C_0)$), and outside the objects there exists $u_0 \approx u_0^o$ (or $outside(C_0)$). The fitting term is as follows:

$$F_1(C) + F_2(C) = \int_{inside(C)} |u_0(x, y) - c_1|^2 dx dy + \int_{outside(C)} |u_0(x, y) - c_2|^2 dx dy, \quad (4.1)$$

where C is the variable curve and constants c_1, c_2 , depending on C , are the averages of u_0 inside C , and outside C , respectively. C_0 is the minimizer of the fitting term

$$\inf_C \{F_1(C) + F_2(C)\} \approx 0 \approx F_1(C_0) + F_2(C_0). \quad (4.2)$$

Adding some regularizing terms like the length of the curve and the area inside the C , the energy function $F(c_1, c_2, C)$ in the ACWE model is defined as

$$F(c_1, c_2, C) = \mu.Length(C) + \nu.Area(inside(C)) + \lambda_1 \int_{inside(C)} |u_0(x, y) - c_1|^2 dx dy + \lambda_2 \int_{outside(C)} |u_0(x, y) - c_2|^2 dx dy, \quad (4.3)$$

where $\mu \geq 0, \nu \geq 0, \lambda_1, \lambda_2 > 0$ are constants.

The ACWE model with $\nu = 0, \lambda_1, \lambda_2 = \lambda$ is a particular case of the Mumford–Shah minimal partition problem, in which the best approximation u of u_0 is pursued.

$$u = \begin{cases} average(u_0) \text{ inside } C \\ average(u_0) \text{ outside } C \end{cases} \quad (4.4)$$

This particular case of the minimal partition problem can be formulated and solved using the level set method [Osher & Sethian, 1988]. In this level set method, $C \subset \Omega$ is represented by the level set function $\phi : \Omega \rightarrow \mathbb{R}$.

$$\begin{cases} C = \partial\omega = \{(x, y) \in \Omega : \phi(x, y) = 0\}, \\ inside(C) = \omega = \{(x, y) \in \Omega : \phi(x, y) > 0\} \\ outside(C) = \Omega \setminus \bar{\omega} = \{(x, y) \in \Omega : \phi(x, y) < 0\} \end{cases} \quad (4.5)$$

By using the Heaviside function H and Dirac function δ_0 as follows:

$$H(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases} \quad \delta_0(z) = \frac{d}{dz} H(z), \quad (4.6)$$

the terms in the energy function $F(c_1, c_2, C)$ can be rewritten as follows:

$$\begin{aligned} F(c_1, c_2, \phi) = & \mu \int_{\Omega} \delta(\phi(x, y)) |\nabla \phi(x, y)| dx dy + \nu \int_{\Omega} H(\phi(x, y)) dx dy \\ & + \lambda_1 \int_{\Omega} |u_0(x, y) - c_1|^2 H(\phi(x, y)) dx dy + \lambda_2 \int_{\Omega} |u_0(x, y) - c_2|^2 (1 - H(\phi(x, y))) dx dy \end{aligned}, \quad (4.7)$$

where the variable C is replaced by variable ϕ .

Let H_ε and δ_ε be the regularization of H and δ . Keeping c_1, c_2 fixed, we minimize $F_\varepsilon(c_1, c_2, \phi)$ with respect to ϕ . By computing the Gateaux derivative and using the Riesz Representation Theorem, we obtain the associated Euler-Lagrange equation for ϕ as follows:

$$\begin{aligned} \frac{\partial \phi}{\partial t} = \delta_\varepsilon(\phi) \left[\mu \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \nu - \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \right] &= 0 \quad \text{in } (0, \infty) \times \Omega \\ \phi(0, x, y) = \phi_0(x, y) &\quad \text{in } \Omega, \\ \frac{\delta_\varepsilon(\phi)}{|\nabla \phi|} \frac{\partial \phi}{\partial \bar{n}} &= 0 \quad \text{on } \partial\Omega, \end{aligned} \quad (4.8)$$

where \bar{n} is the exterior normal to the boundary $\partial\Omega$, and $\frac{\partial \phi}{\partial \bar{n}}$ is the normal derivative of ϕ at the boundary. This partial differential equation is called the C-V model. When we implement this model using the finite difference schemes, we can obtain a numerical algorithm of the model.

The algorithm of the ACWE method is as follows:

1. Initialize ϕ^0 by ϕ_0 , $n = 0$.
2. Compute $c_1(\phi^n)$ and $c_2(\phi^n)$ by

$$c_1(\phi) = \frac{\int_{\Omega} u_0(x, y) H(\phi(x, y)) dx dy}{\int_{\Omega} H(\phi(x, y)) dx dy}, \quad (4.9)$$

$$c_2(\phi) = \frac{\int_{\Omega} u_0(x, y) (1 - H(\phi(x, y))) dx dy}{\int_{\Omega} (1 - H(\phi(x, y))) dx dy}, \quad (4.10)$$

where Equations (4.8) and (4.9) are obtained by calculating the partial derivative of $F_{\epsilon}(c_1, c_2, \phi)$ on c_1, c_2 , respectively.

3. Solve the PDE in ϕ from Equation (4.8) to obtain ϕ^{n+1} .
4. Reinitialize ϕ locally to the signed distance function to the curve (this step is optional).
5. Check whether the solution is stationary. If not, $n = n+1$ and repeat.

4.2 Segmentation Based Contours Method

The ACWE method does not depend on the edge function to determine the object's boundary, can avoid the evolving curve passing through the objects' boundary, can place the initial curve at any position within an image, and can segment objects in a very noisy image [Chan & Vese, 2001]. With all these advantages, the ACWE method can be more useful when compared to the current segmentation methods in the DNA microarray segmentation [Ni, 2009]. To apply the ACWE method to the DNA microarray image segmentation, Ni implements some adjustments on the ACWE method as follows:

1. Make the ACWE to segment each spot patch one at a time. A DNA microarray may contain a half million spots. If we would apply the ACWE method to segment the image as a whole picture, then it would not give the correct segmentation result and it would use a lot of memory. Also, since the ACWE will segment all the spots as a whole region, it is very difficult to extract each spot intensity value if using the whole image for segmentation.

2. Use the grid file as an input which gives the approximate spot locations. This will help to save some computation time since some areas in the image will be neglected because there are no spots in these areas.

3. Decrease the number of iterations 100 times to make computing fast.

4. Adjust the μ value and find more tiny spots.

Having made the ACWE method applicable on the DNA microarray images, Ni improves the accuracy of the ACWE method by using a higher order of finite difference schemes in the numerical algorithm of the ACWE method. The second order forward and backward finite difference schemes are used to replace the first order finite difference schemes in the ACWE method, and the fourth order central finite difference schemes are used to replace the second order finite difference schemes in the ACWE method. The detailed finite difference schemes Ni used are given in the next section.

This modified ACWE method is called the Segmentation Based Contours (SBC) method. Experimental results show that the SBC method can segment more accurately than the GOGAC method for cDNA microarray image and GCOS method for Affymetrix GeneChip image [Ni, 2009].

4.3 The 4th Order Method and the 8th Order Method

Chan and Vese use the following $C^\infty(\bar{\Omega})$ regulation of H and δ_0 to make the algorithm capable of computing a global minimizer of the energy [Chan & Vese, 2001].

In this way, the algorithm is independent of the position of the initial curve.

$$H_\varepsilon(z) = \frac{1}{2} \left(1 + \frac{2}{\pi} \arctan\left(\frac{z}{\varepsilon}\right) \right) \quad (4.11)$$

$$\delta_\varepsilon(z) = \frac{d}{dz} H_\varepsilon(z) \quad (4.12)$$

To discrete the partial differential equation in ϕ in Equation (4.8), we use the finite difference schemes. First, we recall some usual notations: Let h be the space step, Δt be the time step, and $(x_i, y_i) = (ih, jh)$ be the grid points for $1 \leq i, j \leq M$. Let $\phi_{i,j}^n = \phi(n\Delta t, x_i, y_i)$ be an approximation of $\phi(t, x, y)$ with $n \geq 0, \phi^0 = \phi_0$.

To implement the finite differences in the C-V model, we expand the partial differential equation in the model to get its detailed form.

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \delta_\varepsilon(\phi) \left[\mu \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - v - \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \right] \\ &= \delta_\varepsilon(\phi) \left[\mu \operatorname{div} \left(\frac{\left\langle \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y} \right\rangle}{\sqrt{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2}} \right) - v - \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \right] \\ &= \delta_\varepsilon(\phi) \left[\mu \left(\frac{\partial}{\partial x} \left(\frac{\frac{\partial \phi}{\partial x}}{\sqrt{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2}} \right) + \frac{\partial}{\partial y} \left(\frac{\frac{\partial \phi}{\partial y}}{\sqrt{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2}} \right) \right) \right. \\ &\quad \left. - v - \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \right] \end{aligned}$$

$$= \delta_\varepsilon(\phi) \left[\mu \left[\frac{\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}}{\sqrt{(\frac{\partial \phi}{\partial x})^2 + (\frac{\partial \phi}{\partial y})^2}} - \frac{(\frac{\partial \phi}{\partial x})^2 \frac{\partial^2 \phi}{\partial x^2} + (\frac{\partial \phi}{\partial y})^2 \frac{\partial^2 \phi}{\partial y^2}}{\sqrt{[(\frac{\partial \phi}{\partial x})^2 + (\frac{\partial \phi}{\partial y})^2]^3}} - 2 \frac{\frac{\partial \phi}{\partial x} \frac{\partial \phi}{\partial y} \frac{\partial \phi}{\partial x \partial y}}{\sqrt{[(\frac{\partial \phi}{\partial x})^2 + (\frac{\partial \phi}{\partial y})^2]^3}} \right] \right. \\ \left. - v - \lambda_1(u_0 - c_1)^2 + \lambda_2(u_0 - c_2)^2 \right] \quad (4.13)$$

The forward and backward finite difference schemes Chan and Vese use in Equation (4.13) are as follows:

$$\frac{\partial \phi(n)}{\partial x} = \frac{\phi_{i+1,j}^n - \phi_{i,j}^n}{h} + O(h), \quad (4.14)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{\phi_{i,j+1}^n - \phi_{i,j}^n}{h} + O(h), \quad (4.15)$$

$$\frac{\partial \phi(n)}{\partial x} = \frac{\phi_{i,j}^n - \phi_{i-1,j}^n}{h} + O(h), \quad (4.16)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{\phi_{i,j}^n - \phi_{i,j-1}^n}{h} + O(h). \quad (4.17)$$

The central finite differences Chan and Vese use are as follows:

$$\frac{\partial \phi(n)}{\partial x} = \frac{\phi_{i,j+1}^n - \phi_{i,j-1}^n}{2h} + O(h^2), \quad (4.18)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{\phi_{i+1,j}^n - \phi_{i-1,j}^n}{2h} + O(h^2). \quad (4.19)$$

After substituting Equation (4.14-4.19) into Equation (4.13), we get

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} + O(\Delta t) = \delta_h(\phi_{i,j}^n) + \frac{\mu}{h^2} \Delta_-^x \left(\frac{\Delta_+^x \phi_{i,j}^{n+1}}{\sqrt{(\Delta_+^x \phi_{i,j}^n)^2 / (h^2) + (\phi_{i,j+1}^n - \phi_{i,j-1}^n)^2 / (2h)^2}} \right) + \frac{\mu}{h^2} \Delta_-^y \left(\frac{\Delta_+^y \phi_{i,j}^{n+1}}{\sqrt{(\phi_{i+1,j}^n - \phi_{i-1,j}^n)^2 / (2h)^2 + (\Delta_+^y \phi_{i,j}^n)^2 / (h^2)}} \right) + O(h^{1/2}). \quad (4.20)$$

$$-v - \lambda_1(u_{0,i,j} - c_1(\phi^n))^2 + \lambda_2(u_{0,i,j} - c_2(\phi^n))^2$$

The SBC method aims to reach a more accurate approximation, so Ni *et al.* use the second-order forward and backward finite difference schemes and the fourth-order central finite difference schemes to replace the lower ones in the ACWE method [Ni *et al.*, 2009]. The forward and backward finite differences are as follows:

$$\frac{\partial \phi(n)}{\partial x} = \frac{-\phi_{i+2,j}^n + 4\phi_{i+1,j}^n - 3\phi_{i,j}^n}{2h} + O(h^2), \quad (4.21)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{-\phi_{i,j+2}^n + 4\phi_{i,j+1}^n - 3\phi_{i,j}^n}{2h} + O(h^2), \quad (4.22)$$

$$\frac{\partial \phi(n)}{\partial x} = \frac{\phi_{i-2,j}^n - 4\phi_{i-1,j}^n + 3\phi_{i,j}^n}{2h} + O(h^2), \quad (4.23)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{\phi_{i,j-2}^n - 4\phi_{i,j-1}^n + 3\phi_{i,j}^n}{2h} + O(h^2), \quad (4.24)$$

and the central finite differences are as follows:

$$\frac{\partial \phi(n)}{\partial x} = \frac{8(\phi_{i,j+1}^n - \phi_{i,j-1}^n) - (\phi_{i,j+2}^n - \phi_{i,j-2}^n)}{12h} + O(h^4), \quad (4.25)$$

$$\frac{\partial \phi(n)}{\partial y} = \frac{8(\phi_{i+1,j}^n - \phi_{i-1,j}^n) - (\phi_{i+2,j}^n - \phi_{i-2,j}^n)}{12h} + O(h^4), \quad (4.26)$$

which in the end gives Equation (4.13) the first-order accuracy in space and first-order accuracy in time as follows:

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} + O(\Delta t) = \delta_h(\phi_{i,j}^n) + \left[\begin{aligned} & \frac{\mu}{h^2} \Delta_-^x \left(\frac{\Delta_+^x \phi_{i,j}^{n+1}}{\sqrt{(\Delta_+^x \phi_{i,j}^n)^2 / (h^2) + (\phi_{i,j+1}^n - \phi_{i,j-1}^n)^2 / (2h)^2}} \right) \\ & + \frac{\mu}{h^2} \Delta_-^y \left(\frac{\Delta_+^y \phi_{i,j}^{n+1}}{\sqrt{(\phi_{i+1,j}^n - \phi_{i-1,j}^n)^2 / (2h)^2 + (\Delta_+^y \phi_{i,j}^n)^2 / (h^2)}} \right) \\ & - v - \lambda_1 (u_{0,i,j} - c_1(\phi^n))^2 + \lambda_2 (u_{0,i,j} - c_2(\phi^n))^2 \end{aligned} \right] + O(h). \quad (4.27)$$

When we apply the numerical algorithm to the DNA microarray images, we take each pixel point in the image as the grid point for the finite difference schemes. Therefore, the resolution of the microarray image is the determining factor for the order of finite differences used in the C-V model. Given the current resolution of the microarray images, we can further improve the accuracy of the SBC method by implementing a higher order of finite difference schemes in the C-V model.

4.3.1 The 4th Order Method

The higher the order of finite difference schemes are used, the more accurate the approximation. When we choose the order of finite difference schemes to approximate the partial differential equation in the C-V model, we should pay attention to the relationship between the resolution of the microarray images and the number of difference points of the finite difference schemes used in the C-V model. If the number of difference points is larger than the number of pixel points in each spot patch, it will cause overlapping or error in the algorithm, so it is meaningless to use that segmentation method.

For Affymetrix microarray images, the common resolutions are 6×6 , 7×7 and 8×8 pixels in each spot patch in the array images. For cDNA microarray images, there are usually 10×10 pixels in each spot patch. According to the order of the finite difference schemes used in the SBC method, we need at least three and at most four difference points in the computing process, which means that we need at least 4×4 pixels in each spot in the microarray images. Since both Affymetrix GeneChip images and cDNA array images meet this requirement, the SBC segmentation method can be applied to both images.

Depending on the resolution of the common microarray image, there are upper limitations of the order of the finite difference schemes that we can use to approximate Equation (4.13). So for Affymetrix microarray images, we choose the fourth-order forward, backward and central finite difference schemes to replace the lower ones that are used in the SBC method. We name the new method the 4th order method. Therefore, the finite difference schemes used in the 4th order method are as follows:

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{1}{12h} [-25\phi_{i,j}^{n+1} + 48\phi_{i+1,j}^{n+1} - 36\phi_{i+2,j}^{n+1} + 16\phi_{i+3,j}^{n+1} - 3\phi_{i+4,j}^{n+1}] + O(h^4), \quad (4.28)$$

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{1}{12h} [25\phi_{i,j}^{n+1} - 48\phi_{i-1,j}^{n+1} + 36\phi_{i-2,j}^{n+1} - 16\phi_{i-3,j}^{n+1} + 3\phi_{i-4,j}^{n+1}] + O(h^4), \quad (4.29)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{1}{12h} [-25\phi_{i,j}^{n+1} + 48\phi_{i,j+1}^{n+1} - 36\phi_{i,j+2}^{n+1} + 16\phi_{i,j+3}^{n+1} - 3\phi_{i,j+4}^{n+1}] + O(h^4), \quad (4.30)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{1}{12h} [25\phi_{(i,j)}^{(n+1)} - 48\phi_{(i,j-1)}^{(n+1)} + 36\phi_{(i,j-2)}^{(n+1)} - 16\phi_{(i,j-3)}^{(n+1)} + 3\phi_{(i,j-4)}^{(n+1)}] + O(h^4), \quad (4.31)$$

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{8(\phi_{i,j+1}^n - \phi_{i,j-1}^n) - (\phi_{i,j+2}^n - \phi_{i,j-2}^n)}{12h} + O(h^4), \quad (4.32)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{8(\phi_{i+1,j}^n - \phi_{i-1,j}^n) - (\phi_{i+2,j}^n - \phi_{i-2,j}^n)}{12h} + O(h^4). \quad (4.33)$$

After substituting Equation (4.28-4.33) into Equation (4.13), we get the second-order accuracy in space and the first-order accuracy in time as follows:

$$\begin{aligned} & \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} + O(\Delta t) \\ &= \delta_h(\phi_{i,j}^n) + \left[\begin{aligned} & \frac{\mu}{h^2} \Delta_x^- \left(\frac{\Delta_x^+ \phi_{i,j}^{n+1}}{\sqrt{(\Delta_x^+ \phi_{i,j}^n)^2 / (h^2) + (\phi_{i,j+1}^n - \phi_{i,j-1}^n)^2 / (2h)^2}} \right) \\ & + \frac{\mu}{h^2} \Delta_y^- \left(\frac{\Delta_y^+ \phi_{i,j}^{n+1}}{\sqrt{(\phi_{i+1,j}^n - \phi_{i-1,j}^n)^2 / (2h)^2 + (\Delta_y^+ \phi_{i,j}^n)^2 / (h^2)}} \right) \\ & - v - \lambda_1 (u_{0,i,j} - c_1(\phi^n))^2 + \lambda_2 (u_{0,i,j} - c_2(\phi^n))^2 \end{aligned} \right] + O(h^2). \quad (4.34) \end{aligned}$$

For the finite difference schemes used in the 4th order method, at least four and at most five difference points are needed, so we need at least 5×5 pixels in each spot in the microarray images. Similar to the discussion in the SBC method, the 4th order method can be used both in the segmentation of the Affymetrix GeneChip images and the cDNA microarray images.

4.3.2 The 8th Order Method

The cDNA microarray images have a higher resolution, as a 10×10 in each spot patch, so we can choose a higher order of finite difference schemes to approximate Equation (4.13) to get better segmentation. We increase the order of forward, backward and central finite difference schemes used to discrete Equation (4.13) to the eighth order of finite differences, which are as follows:

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{1}{h} \left[\begin{aligned} &-\frac{761}{280} \phi_{i,j}^{n+1} + 8 \phi_{i+1,j}^{n+1} - 14 \phi_{i+2,j}^{n+1} + \frac{56}{3} \phi_{i+3,j}^{n+1} - \frac{35}{2} \phi_{i+4,j}^{n+1} \\ &+ \frac{56}{5} \phi_{i+5,j}^{n+1} - \frac{14}{3} \phi_{i+6,j}^{n+1} + \frac{8}{7} \phi_{i+7,j}^{n+1} - \frac{1}{8} \phi_{i+8,j}^{n+1} \end{aligned} \right] + O(h^8), \quad (4.35)$$

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{1}{h} \left[\begin{aligned} &\frac{761}{280} \phi_{i,j}^{n+1} - 8 \phi_{i-1,j}^{n+1} + 14 \phi_{i-2,j}^{n+1} - \frac{56}{3} \phi_{i-3,j}^{n+1} + \frac{35}{2} \phi_{i-4,j}^{n+1} \\ &-\frac{56}{5} \phi_{i-5,j}^{n+1} + \frac{14}{3} \phi_{i-6,j}^{n+1} - \frac{8}{7} \phi_{i-7,j}^{n+1} + \frac{1}{8} \phi_{i-8,j}^{n+1} \end{aligned} \right] + O(h^8), \quad (4.36)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{1}{h} \left[\begin{aligned} &-\frac{761}{280} \phi_{i,j}^{n+1} + 8 \phi_{i,j+1}^{n+1} - 14 \phi_{i,j+2}^{n+1} + \frac{56}{3} \phi_{i,j+3}^{n+1} - \frac{35}{2} \phi_{i,j+4}^{n+1} \\ &+ \frac{56}{5} \phi_{i,j+5}^{n+1} - \frac{14}{3} \phi_{i,j+6}^{n+1} + \frac{8}{7} \phi_{i,j+7}^{n+1} - \frac{1}{8} \phi_{i,j+8}^{n+1} \end{aligned} \right] + O(h^8), \quad (4.37)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{1}{h} \left[\begin{aligned} &\frac{761}{280} \phi_{i,j}^{n+1} - 8 \phi_{i,j-1}^{n+1} + 14 \phi_{i,j-2}^{n+1} - \frac{56}{3} \phi_{i,j-3}^{n+1} + \frac{35}{2} \phi_{i,j-4}^{n+1} \\ &-56 \phi_{i,j-5}^{n+1} + \frac{14}{3} \phi_{i,j-6}^{n+1} - \frac{8}{7} \phi_{i,j-7}^{n+1} + \frac{1}{8} \phi_{i,j-8}^{n+1} \end{aligned} \right] + O(h^8), \quad (4.38)$$

$$\frac{\partial \phi(n+1)}{\partial x} = \frac{1}{h} \left[\begin{aligned} &\frac{4}{5} (\phi_{i+1,j}^{n+1} - \phi_{i-1,j}^{n+1}) - \frac{1}{5} (\phi_{i+2,j}^{n+1} - \phi_{i-2,j}^{n+1}) \\ &+ \frac{4}{105} (\phi_{i+3,j}^{n+1} - \phi_{i-3,j}^{n+1}) - \frac{1}{280} (\phi_{i+4,j}^{n+1} - \phi_{i-4,j}^{n+1}) \end{aligned} \right] + O(h^8), \quad (4.39)$$

$$\frac{\partial \phi(n+1)}{\partial y} = \frac{1}{h} \left[\begin{aligned} &\frac{4}{5} (\phi_{i,j+1}^{n+1} - \phi_{i,j-1}^{n+1}) - \frac{1}{5} (\phi_{i,j+2}^{n+1} - \phi_{i,j-2}^{n+1}) \\ &+ \frac{4}{105} (\phi_{i,j+3}^{n+1} - \phi_{i,j-3}^{n+1}) - \frac{1}{280} (\phi_{i,j+4}^{n+1} - \phi_{i,j-4}^{n+1}) \end{aligned} \right] + O(h^8). \quad (4.40)$$

Therefore, we call this new improved method the 8th order method. After we implement Equation (4.35-4.40) into Equation (4.13), we get the fourth order accuracy approximation in space and the first order accuracy in time as follows:

$$\begin{aligned} & \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} + O(\Delta t) \\ &= \delta_h(\phi_{i,j}^n) + \left[\begin{aligned} & \frac{\mu}{h^2} \Delta_x^x \left(\frac{\Delta_x^x \phi_{i,j}^{n+1}}{\sqrt{(\Delta_x^x \phi_{i,j}^n)^2 / (h^2) + (\phi_{i,j+1}^n - \phi_{i,j-1}^n)^2 / (2h)^2}} \right) \\ & + \frac{\mu}{h^2} \Delta_y^y \left(\frac{\Delta_y^y \phi_{i,j}^{n+1}}{\sqrt{(\phi_{i+1,j}^n - \phi_{i-1,j}^n)^2 / (2h)^2 + (\Delta_y^y \phi_{i,j}^n)^2 / (h^2)}} \right) \\ & - \nu - \lambda_1 (u_{0,i,j} - c_1(\phi^n))^2 + \lambda_2 (u_{0,i,j} - c_2(\phi^n))^2 \end{aligned} \right] + O(h^4) \quad (4.41) \end{aligned}$$

According to the finite difference schemes used in the 8th order method, we need at least eight and at most nine difference points, which means at least 9×9 pixels in each spot are required in the microarray images. Thus, we can only apply the 8th order method into the segmentation of cDNA microarray images based on the existing image resolution.

4.4 The Comparison of the Performance between the Modified ACWE Method, the SBC Method, the 4th Order Method and the 8th Order Method

We summarize the orders of the truncation error term in space and the order of the truncation error term in time in the numerical approximation of the C-V model in the ACWE method, the SBC method, the 4th order method, and the 8th order method, respectively. Table 4.1 gives a clearer realization of the accuracy of each segmentation method in approximating the C-V model.

Table 4.1: Comparison in respect to the order of error term in the numerical equation of the C-V model between four segmentation methods.

Method	The ACWE	The SBC	The 4 th order	The 8 th order
	method	method	method	method
The order of error term in space	1/2	1	2	4
The order of error term in time	1	1	1	1

In Table 4.1, the powers of the order of the error term present how well the numerical equation approximates the partial differential equation in the C-V model in [Chan & Vese, 2001]. The larger the power is, the more accurate the numerical approximation. So according to Table 4.1, the 4th order method and the 8th order method are better than the modified ACWE method and the SBC method in giving the microarray image segmentation. Furthermore, the 8th order method should be more accurate than the 4th order method.

4.5 Application of the Algorithms on the Segmentation of Simulated DNA Microarray Images

The algorithms of the SBC method, the 4th order method and the 8th order method are written into executable programs in Java language. We choose the parameters as $\lambda_1 = \lambda_2 = 1, \nu = 0, h = 1, \Delta t = 0.1$ [Chan & Vese, 2001]. For the length parameter μ , it has a scale role. If we want to detect as many objects as possible, then μ should be small. However, if we want to detect only larger objects, and not detect small objects, μ should

be larger. In our experiment, we take $\mu = 0.01 * 255 * 255$ for segmenting cDNA microarray images, and $\mu = 0.025 * 255 * 255$ for segmenting Affymetrix GeneChip microarray images [Ni *et al.*, 2009]. Once all these parameters are set up in the program, it is not necessary to adjust them during the segmentation.

We also increase iterations of the algorithms of the 4th order method and the 8th order method from 100 times, that is, the iterations in the SBC algorithm to 1000 times. This iteration number is chosen after we compare the segment results with 100, 1,000, and 10,000 iterations on all five microarray images used in our experiment. The results with 100 iterations are quite with the results as with 1,000 iterations, while the results with 1,000 iterations have the same characteristics with the results with 10,000 iterations. Therefore, the algorithms of the 4th order method and the 8th order method with at least 1,000 iterations can be considered as time indepent.

Intensities generated by the segmentation methods discussed in our research are compared with the true intensity values. Therefore, we can evaluate the performance of each method by how close their intensities are to the true values.

For a cDNA microarray image, we take the mean of the pixel values within the segmentation boundary of each spot as the intensity value for that spot. For an Affymetrix GeneChip image, we take the 75 percentile of the pixel values within the segmentation boundary of each spot as the intensity for that spot.

Each probe spot in a microarray image consists of a relative small fixed number of pixels. For instance, in our experiment, there are 6×6 pixels in each spot of an Affymetrix GeneChip image, and 10×10 pixels in each spot of a cDNA image. When

the image is being segmented, those spots are being processed one by one. Therefore, the algorithm is linear in complexity with a large constant.

CHAPTER 5

SEGMENTATION RESULTS ON SIMULATED CDNA

MICROARRAY IMAGES

As we introduced in Chapter 2, the simulated cDNA microarray image has the same characteristics as the proto cDNA image. Each probe spot in the simulated cDNA image has 10×10 pixels, and each pixel in the horizontal direction and the vertical direction can be used as a difference point when being segmented. Since the 4th order method needs five difference points in each direction and the 8th order method needs nine difference points in each direction, both methods can be used to segment the simulated cDNA image. In the meantime, the GOGAC method and the SBC method are also being applied on the simulated cDNA image. All of the output intensities from the above four methods will be compared to the true intensities of the image and therefore evaluated.

5.1 Segmentation Output Intensities Comparison

There are 7,744 probe spots total in the simulated cDNA image. For each segmentation method mentioned above, we calculated the absolute difference between the output intensity and the true intensity of each probe spot. We then compare any two above segmentation methods by counting the number of spot in one method that has a larger, equal, and smaller difference than the number of spots in the other method,

respectively. The percentages of that number among the total spot number are as well calculated for a more clear view. As we discussed in Section 4.5, we perform the 4th order method and the 8th order method with three different iterations: 100 times, 1,000 times, and 10,000 times. The data are summarized in Tables 5.1-5.3.

Table 5.1: Performance of the GOGAC method, the SBC method, the 4th order method, and the 8th order method on the simulated cDNA microarray image. The 4th order method and the 8th order method are with 100 iterations.

Total spots number:7744	d1>d2	d1=d2	d1<d2
SBC vs GOGAC	3177(41.03%)	0 (0%)	4567(58.97%)
4th order vs GOGAC	3174(40.99%)	0 (0%)	4570(59.01%)
8th order vs GOGAC	3174(40.99%)	0 (0%)	4570(59.01%)
4th order vs SBC	234(3.02%)	7119 (91.93%)	391(5.05%)
8th order vs SBC	245(3.16%)	7038 (90.88%)	461(5.95%)
8th order vs 4th order	135(1.74%)	7427 (95.91%)	182(2.35%)

Table 5.2: Performance of the GOGAC method, the SBC method, the 4th order method, and the 8th order method on the simulated cDNA microarray image. The 4th order method and the 8th order method are with 1,000 iterations.

Total spots number:7744	d1>d2	d1=d2	d1<d2
SBC vs GOGAC	3177(41.03%)	0(0%)	4567(58.97%)
4th order vs GOGAC	3149(40.66%)	0(0%)	4595(59.34%)
8th order vs GOGAC	3152(40.70%)	0(0%)	4592(59.30%)
4th order vs SBC	945(12.20%)	4891(63.16%)	1908(24.64%)
8th order vs 4th order	960(12.40%)	4845(62.56%)	1939(25.04%)
8th order vs 4th order	112(1.45%)	7489(96.71%)	143(1.85%)

Table 5.3: Performance of the GOGAC method, the SBC method, the 4th order method, and the 8th order method on the simulated cDNA microarray image. The 4th order method and the 8th order method are with 10,000 iterations.

Total spots number:7744	d1>d2	d1=d2	d1<d2
SBC vs GOGAC	3177(41.03%)	0(0%)	4567(58.97%)
4th order vs GOGAC	3127(40.38%)	0(0%)	4617(59.62%)
8th order vs GOGAC	3130(40.42%)	0(0%)	4614(59.58%)
4th order vs SBC	1181(15.25%)	4294(55.45%)	2269(29.30%)
8th order vs SBC	1183(15.28%)	4281(55.28)	2281(29.46%)
8th order vs 4th order	79(1.02%)	7568(97.73%)	98(1.27%)

In Tables 5.1-5.3, d1 stands for the absolute difference between the spot intensity from the former segmentation method listed in the first column of each row and the true spot intensity. Similarly, d2 stands for the absolute difference between the spot intensity

from the latter segmentation method listed in the first column of each row and the true spot intensity.

When we compare the results in Tables 5.1-5.3, we observe that the results with 1,000 iterations are consistent with the results with 10,000 iterations, which means that the algorithms of the 4th order method and the 8th order method can be considered time independent with at least 1,000 iterations. Therefore, we use the segmentations results with 1,000 iterations for analysis.

Since we expect our 4th order method and 8th order method are better than the GOGAC method and the SBC method, we would like the percentage in the fourth column to be larger than the one in the second column. Table 5.2 shows that SBC method, the 4th order method, and the 8th order method have more spot intensities that are closer to the true intensities than the GOGAC method, respectively; the 4th order method and the 8th order method have more spot intensities that are closer to the true intensities than the SBC method, respectively; the 8th order method has more spot intensities that are closer to the true intensities than the 4th order method.

5.2 Statistical Analysis on Segmentation Output Intensities

In order to have a more comprehensive understanding of the performances of the 4th order method and the 8th order method, we implement statistical analysis on the segmentation output intensities from the four segmentation methods and the true intensity values.

One-tailed paired t-test is selected in our research. Therefore, in each experiment we could test whether the two methods are equally close to the true values, or one method

is closer to the true values than the other method. All the segmentation methods are tested pairwise.

The UPGMA hierarchy cluster is selected because it is the most widely used cluster method in microarray image analysis. Hierarchy cluster is an approach to group the items with the most similarities. The UPGMA hierarchy cluster, which is an average linkage hierarchical clustering method, takes the average distance between two groups as the standard to measure that similarity, and the two groups with the smallest average distance would be grouped into one cluster [Rencher, 2002]. The dendrogram given by the cluster analysis presents both steps of grouping and the distances at which each grouping happened. Paired t-test and UPGMA hierarchy cluster results are applied in Chapters 5 and 6.

In the paired t-test experiment, we first calculate the absolute difference d_1 and d_2 as we explained in Table 5.2, which d_1 stands for the absolute difference between the spot intensity from the former segmentation method listed in the first column of each row and the true spot intensity, and d_2 stands for the absolute difference between the spot intensity from the latter segmentation method listed in the first column of each row and the true spot intensity. Those differences are the samples of our test. We let μ_1 represent the difference between the former method and the true value, while μ_2 represents the difference between the latter method and the true value. The null hypothesis is $H_0 : \mu_2 \geq \mu_1$, and the alternative hypothesis is $H_1 : \mu_2 < \mu_1$. The significance level is $\alpha = 0.05$. Therefore, the rejection of the null hypothesis will show that the latter segmentation method has the spot intensity closer to the true intensity than the former

segmentation method, which means that the latter method is more accurate than the former method.

Table 5.4 shows that the SBC method, the 4th order method and the 8th order method has spot intensities closer to true intensities than GOGAC method, respectively, and the 4th order method has spot intensities closer to true intensities than the SBC method. In addition, there is no significant difference between the performance of the SBC method and the 8th order method, and the performance of the 4th order method and the 8th order method.

Table 5.4: Paired t-test result on the GOGAC method, the SBC method, the 4th order method, and the 8th order method on the simulated cDNA microarray image. The 4th order method and the 8th order method are with 1,000 iterations.

Two methods on t-test	p-value	Null hypothesis	Alternative hypothesis	Reject null hypothesis
GOGAC vs SBC	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
GOGAC vs 4 th order	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
GOGAC vs 8 th order	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
SBC vs 4 th order	0.00028	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
SBC vs 8 th order	0.00028	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
4 th order vs 8 th order	0.4053	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	No

The UPGMA hierarchy cluster results in Figure 5.1 show that the GOGAC method generates spot intensities closer to true intensities than the other three methods. The SBC method has very little improvement on the accuracy than the 4th order method

and the 8th order method. There is no significant difference between the 4th order method and the 8th order method.

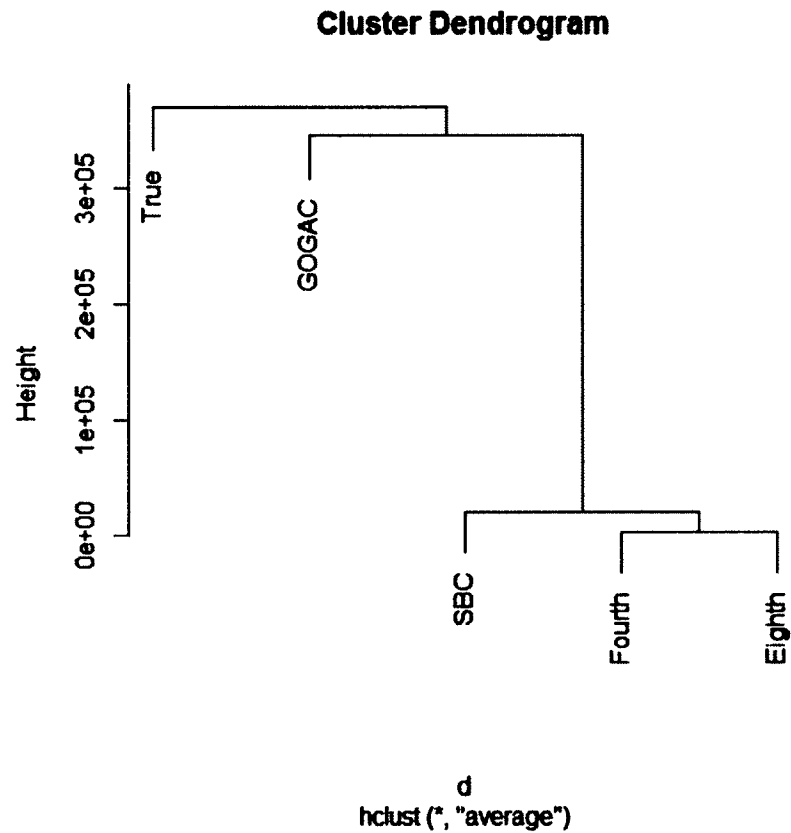


Figure 5.1: UPGMA hierarchy cluster result of comparing the GOGAC method, the SBC method, the 4th order method and the 8th order method on the simulated cDNA microarray image. The 4th order method and the 8th order method are with 1,000 iterations.

CHAPTER 6

SEGMENTATION RESULTS ON SIMULATED

AFFYMETRIX GENECHIP IMAGES

Similar with what we implemented in Chapter 5 on cDNA microarray image, we simulate two Affymetrix GeneChip images using the simulator introduced in Chapter 3. The simulated Affymetrix GeneChip images have the same characteristics as the original image, which has at least 6×6 pixels in each spot in the image. Since the 4th order method needs five difference points in each direction and the 8th order method needs nine difference points in each direction, and limited by the pixel number in each spot, we can only apply the 4th order method on Affymetrix Genechip images.

Even so, with the development of the image scanning technology, the resolution of the DNA microarray image will be higher. Based on the discussion in Chapter 4, the 8th order method should be more accurate than the 4th order method. Therefore, we implement the 8th order method on Affymetrix GeneChip image as well to compare the performance of both the two method at the same condition. Therefore, we simulate two pixel-expanded Affymetrix Genechip images, as stated in Chapter 3, which have 12×12 pixels in each probe spot in both images

6.1 Segmentation Results on Simulated Good Affymetrix GeneChip Image

In the research [Cheng, 2013], Cheng applies the SBC method on 50 simulated Affymetrix GeneChip images, and the SBC method performs better on some of the images than the GCOS, but worse on the remaining images. Therefore, we apply the 4th order method on two simulated Genechip images: one is the image from Cheng's research for which the SBC method is better, and the other is the image from Cheng's research for which the SBC is worse. In the meanwhile, the SBC method and the GCOS method are also applied to offer comparisons

6.1.1 Segmentation Output Intensities Comparison on Simulated Good Affymetrix GeneChip Image

Like the discussion in Section 5.1, for each segmentation method mentioned above, we calculated the absolute difference between the output intensity and the true intensity of each probe spot. There are 535,824 probe spots total in the simulated good Affymetrix GeneChip image. We then compare any two above segmentation methods by counting the number of spots in one method that have larger, equal, and smaller differences than one of the other methods, respectively. As the same argument we did in Section 5.1, we we perform the 4th order method and the 8th order method with three different iterations: 100, 1,000 and 10,000 times. The data is summarized in Tables 6.1-6.3. And we take the data with 1,000 iterations for the following analysis procedures.

Table 6.1: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated good Affymetrix GeneChip image. The 4th order method is with 100 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
SBC vs GCOS	114365(21.34%)	2562(0.48%)	418897(78.17%)
4th order vs GCOS	112964(21.08%)	2585(0.48%)	420275(78.44%)
4th order vs SBC	6687(1.25%)	511326(95.42%)	17811(3.32%)

Table 6.2: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated good Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
SBC vs GCOS	114365(21.34%)	2562(0.48%)	418897(78.17%)
4th order vs GCOS	113506(21.12%)	2570(0.48%)	419748(78.34%)
4th order vs SBC	8570(1.60%)	512980(95.74%)	14274(2.66%)

Table 6.3: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated good Affymetrix GeneChip image. The 4th order method is with 10,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
SBC vs GCOS	114365(21.34%)	2562(0.48%)	418897(78.17%)
4th order vs GCOS	113644(21.21%)	2579(0.48%)	419601(78.31%)
4th order vs SBC	9568(1.79%)	511952(95.54%)	14304(2.67%)

In Tables 6.1-6.3, d1 stands for the absolute difference between the spot intensity from the former segmentation method listed in the first column of each row and the true

spot intensity. Similarly, d2 stands for the absolute difference between the spot intensity from the latter segmentation method listed in the first column of each row and the true spot intensity.

Table 6.2 shows that the SBC method and the 4th order method have more spot intensities that are closer to the true intensities than the GCOS method, respectively, and the 4th order method has more spot intensities that are closer to the true intensities than the SBC method.

6.1.2 Statistical Analysis on Segmentation Output Intensities on Simulated Good Affymetrix GeneChip Image

Paired t-test and UPGMA hierarchy cluster results are applied on the segmentation output intensities from the three segmentation methods, and the true intensity values aim to give a more comprehensive understanding of the performances of the 4th order method.

In the paired t-test experiment, we first calculated the absolute difference d1 and d2, which d1 stands for the absolute difference between the spot intensity from the former segmentation method listed in the first column of each row and the true spot intensity, and d2 stands for the absolute difference between the spot intensity from the latter segmentation method listed in the first column of each row and the true spot intensity. Those differences are the samples of our test. We let μ_1 represent the difference between the former method and the true value, while μ_2 represents the difference between the latter method and the true value. The null hypothesis is $H_0 : \mu_2 \geq \mu_1$, and the alternative hypothesis is $H_1 : \mu_2 < \mu_1$. The significance level is $\alpha = 0.05$. Therefore, the rejection of the null hypothesis will show that the latter segmentation method has the spot intensity

closer to the true intensity than the former segmentation method, which means that the latter method is more accurate than the former method.

Table 6.4 shows that the SBC method and the 4th order method has spot intensity closer to true intensity than the GCOS method respectively, and the 4th order method has spot intensity closer to the true intensity than the SBC method. The UPGMA hierarchy cluster result in Figure 6.1 shows that the 4th order method and the SBC method generate spot intensities closer to true intensities than the GCOS method.

Table 6.4: Paired t-test result on the GCOS method, the SBC method, and the 4th order method on the simulated good Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Two methods on t-test	p-value	Null hypothesis	Alternative hypothesis	Reject null hypothesis
GCOS vs SBC	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
GCOS vs 4 th order	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes
SBC vs 4 th order	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes

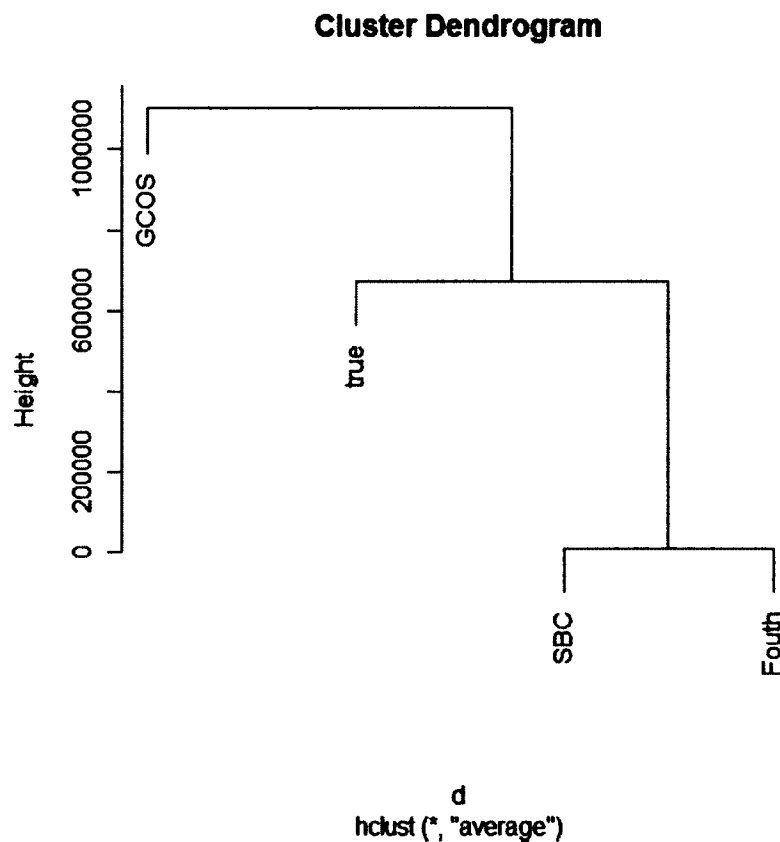


Figure 6.1: UPGMA hierarchy cluster result of comparing the GCOS method, the SBC method, and the 4th order method on the simulated good Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

6.2 Segmentation Results on Simulated Bad Affymetrix GeneChip Image

6.2.1 Segmentation Output Intensities Comparison on Simulated Bad Affymetrix GeneChip Image

Like the discussion in Section 6.1.1, for each segmentation method mentioned above, we calculate the absolute difference between the output intensity and the true intensity of each probe spot. There are 535,824 probe spots total in the simulated bad Affymetrix GeneChip image. We compare any two above segmentation methods by counting the number of spots in one method that has larger, equal, and smaller differences

than one of the other methods, respectively. We then perform the 4th order method and the 8th order method with three different iterations: 100, 1,000 and 10,000 times. The data are summarized in Tables 6.5-6.7. Furthermore, we take the data with 1,000 iterations for the following analysis procedures.

Table 6.5: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated bad Affymetrix GeneChip image. The 4th order method is with 100 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
SBC vs GCOS	360199(67.22%)	13144(2.45%)	162481(30.32%)
4th order vs GCOS	358450(66.90%)	13389(2.50%)	163985(30.60%)
4th order vs SBC	6644(1.24%)	514795(96.08%)	14385(2.68%)

Table 6.6: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated bad Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
SBC vs GCOS	360199(67.22%)	13144(2.45%)	162481(30.32%)
4th order vs GCOS	359071(67.01%)	13292(2.48%)	163461(30.51%)
4th order vs SBC	6894(1.29%)	516959(96.48%)	11971(2.23%)

Table 6.7: Performance of the GCOS method, the SBC method, and the 4th order method on the simulated bad Affymetrix GeneChip image. The 4th order method is with 10,000 iterations.

Total spots number: 535824	d1>d2	d1=d2	d1<d2
SBC vs GCOS	360199(67.22%)	13144(2.45%)	162481(30.32%)
4th order vs GCOS	359300(67.06%)	13260(2.47%)	163264(30.47%)
4th order vs SBC	7357(1.37%)	516572(96.41%)	11895(2.22%)

In Table 6.6, d1 and d2 have exactly the same meanings as those in Table 6.2. Table 6.6 shows that the SBC method and the 4th order method have less spot intensities that are closer to the true intensities than GCOS method, respectively. However, the last row in the table shows that the 4th order method has more spot intensities that are closer to the true intensities than the SBC method.

6.2.2 Statistical Analysis on Segmentation Output Intensities on Simulated Bad Affymetrix GeneChip Image

Exactly the same paired t-test and UPGMA hierarchy cluster experiments utilized on simulated good Affymetrix GeneChip image are applied on the segmentation output intensities from the three segmentation methods and the true intensity values of simulated bad Affymetrix GeneChhip image.

Table 6.8 shows that the GCOS method has spot intensity closer to the true intensity than the SBC method and the 4th order method, respectively. However, the rejection of the null hypothesis on the last row shows that the 4th order method has spot intensities closer to true intensities than the SBC method. The UPGMA hierarchy cluster

result in Figure 6.2 shows that the GCOS method generates spot intensities closer to true intensities than the 4th order method and the SBC method.

Table 6.8: Paired t-test result on the GCOS method, the SBC method, and the 4th order method on the simulated bad Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Two methods on t-test	p-value	Null hypothesis	Alternative hypothesis	Reject null hypothesis
GCOS vs SBC	1	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	No
GCOS vs 4 th order	1	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	No
SBC vs 4 th order	<0.0001	$H_0 : \mu_2 \geq \mu_1$	$H_1 : \mu_2 < \mu_1$	Yes

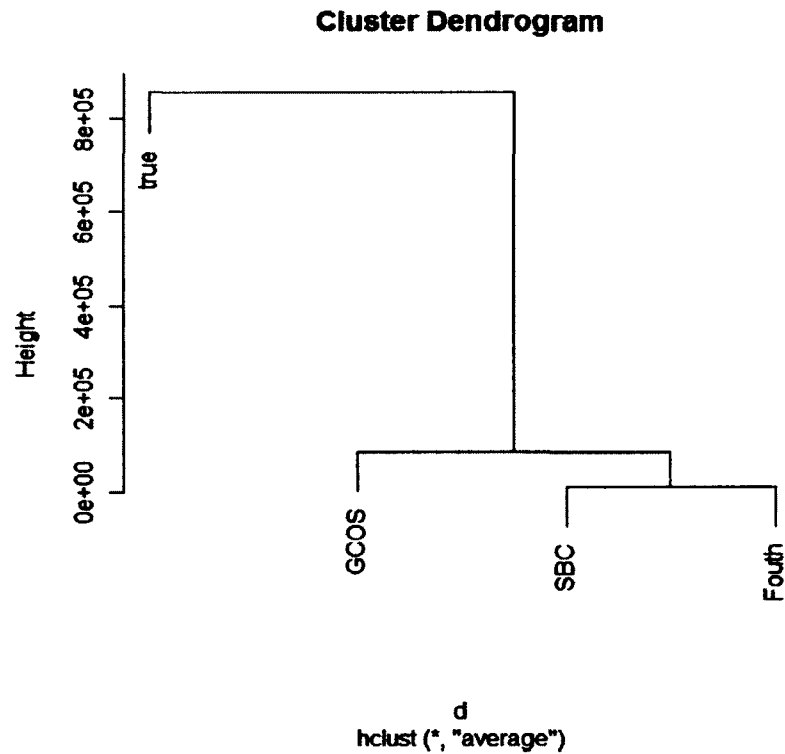


Figure 6.2: UPGMA hierarchy cluster result of comparing the GCOS method, the SBC method, and the 4th order method on the simulated bad Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

6.3 Segmentation Results on Simulated Expanded Affymetrix GeneChip Images

As performed in Sections 6.1 and 6.2, we evaluate in this section the performance of the the 8th order method, the 4th order method, and the SBC method on expanded Affymetrix GeneChip images. The two simulated images used in this section are expanded from the same images used in Sections 6.1 and 6.2, respectively. But only the top-left corners of the original GeneChip images are expanded as described in Section 3.3.2. Hence, each simulated expanded image has 33,489 probe spots total.

Specifically, there is one thing that needs to be claimed here. The GCOS software can only read and segment the type of GeneChip images that have already existed in its library. Since the expanded GeneChip images have different attributes from the original images, we cannot use the GCOS software to segment the simulated expanded images. In this section, we are comparing the 4th order method and the 8th order method to the SBC method.

6.3.1 Segmentation Results on Simulated Good Expanded Affymetrix GeneChip Image

The same segmentation output intensity analysis procedures that were implemented in Section 6.1 are also implemented in Sections 6.3.1, including intensity analysis, paired t-test analysis, and UPGMA cluster analysis.

6.3.1.1 Segmentation Output Intensities Comparison on Simulated Good Expanded Affymetrix GeneChip Image. For each segmentation method mentioned above, we calculated the absolute differences between the output intensity and the true intensity of each probe spot. There are 33,489 probe spots total in the simulated good expanded Affymetrix GeneChip image. We compare any two above segmentation methods by

counting the number of spots in one method that have larger, equal, and smaller differences than one of the other methods, respectively. Next we perform the 4th order method and the 8th order method analysis with three different iterations: 100, 1,000 and 10,000 times. The data is summarized in Tables 6.9-6.11. Also, we take the data with 1,000 iterations for the following analysis procedures.

Table 6.9: Performance of the SBC method, the 4th order method, and the 8th order method on the simulated good expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 100 iterations.

Total spots number: 33489	d1>d2	d1=d2	d1<d2
4th order vs SBC	9300(27.77%)	16204(48.39%)	7985(23.85%)
8th order vs SBC	9468(27.27%)	16206(48.39%)	7815(23.34%)
8th order vs 4th order	1412(4.22%)	30941(92.39%)	1136(3.39%)

Table 6.10: Performance of the SBC method, the 4th order method, and the 8th order method on the simulated good expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

Total spots number: 33489	d1>d2	d1=d2	d1<d2
4th order vs SBC	9165(27.37%)	16245(48.51%)	8079(24.12%)
8th order vs SBC	9351(27.92%)	16215(48.42%)	7923(23.66%)
8th order vs 4th order	1333(3.98%)	31108(92.89%)	1048(3.13%)

Table 6.11: Performance of the SBC method, the 4th order method, and the 8th order method on the simulated good expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 10,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 33489			
4th order vs SBC	9204(27.48%)	16202(48.38%)	8083(24.14%)
8th order vs SBC	9389(28.04%)	16166(48.27%)	7934(23.69%)
8th order vs 4th order	1333(3.98%)	31119(92.92%)	1037(3.10%)

Table 6.10 shows that there is no significant difference in the segmentation accuracy between the SBC method, the 4th order method, and the 8th order method. For comparison, we take out the intensities of the spots in the top-left corner that used to expand in the original image, and calculate the performance of each method on these spots in Table 6.12. Comparing Table 6.10 and Table 6.12, we find that the 4th order method performs better than the SBC method on the good bovine image than on the expanded good bovine image.

Table 6.12: Performance of the SBC method, and the 4th order method on the top-left corner of the original simulated good Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
4th order vs. SBC	414(1.24%)	32010(95.58%)	1065(3.18%)

6.3.1.2 Statistical Analysis on Segmentation Output Intensities on Simulated Good Expanded Affymetrix GeneChip Image. Table 6.13 summarizes the paired t-test results on the SBC method, the 4th order method, and the 8th order method. The results show that the spot intensities generated by the SBC method, the 4th order method and the 8th order method are significantly close to the true intensities. There is no significant difference of the segmentation accuracy between the above three methods on good expanded bovine image.

Table 6.13: Paired t-test result on the SBC method, the 4th order method, and the 8th order method on the simulated good expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

Two methods on t-test	p-value	Null hypothesis	Alternative hypothesis	Reject null hypothesis
SBC vs 4 th order	1	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No
SBC vs 8 th order	1	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No
4 th order vs 8 th order	1	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No

Figure 6.3 is the UPGMA cluster result on the SBC method, the 4th order method and the 8th order method. It shows that there is no significant difference of the segmentation accuracy between the above three methods on good expanded bovine image.

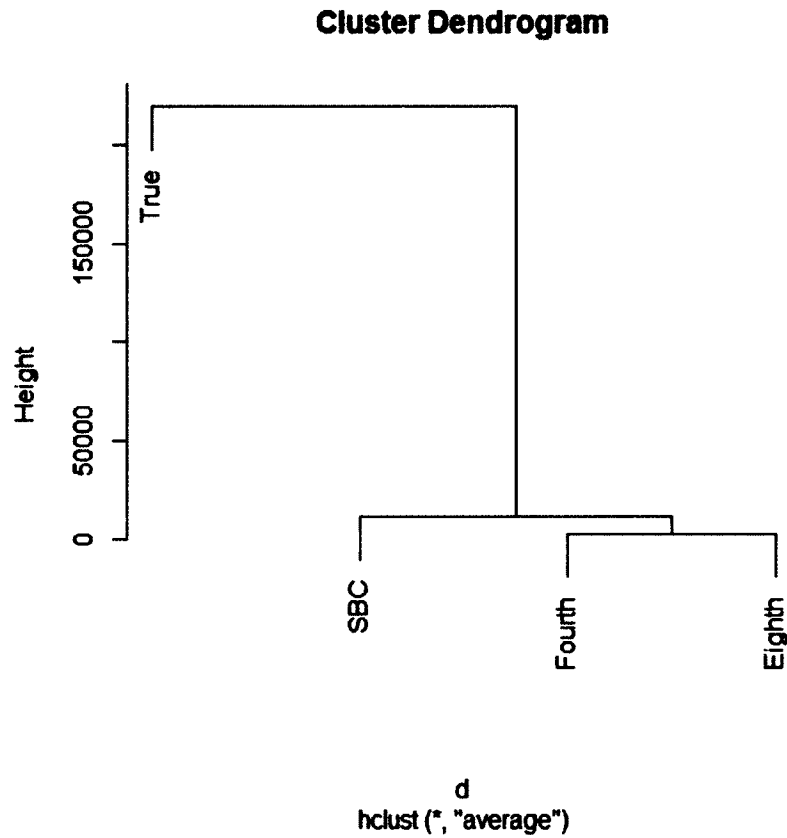


Figure 6.3: UPGMA hierarchy cluster result of comparing the SBC method, the 4th order method, and the 8th order method on the simulated good expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

6.3.2 Segmentation Results on Simulated Bad Expanded Affymetrix GeneChip Image

The same segmentation output intensity analysis procedures performed in Section 6.2 are implemented in Sections 6.3.2, including intensity analysis, paired t-test analysis, and UPGMA cluster analysis.

6.3.2.1 Segmentation Output Intensities Comparison on Simulated Bad Expanded Affymetrix GeneChip Image. For each of the SBC method, the 4th order method, and the 8th order method, we calculate the absolute difference between the output intensity and the true intensity of each probe spot. There are 33,489 probe spots total in the simulated bad expanded Affymetrix GeneChip image. Then we compare any two above segmentation methods by counting the number of spots in one method that have larger, equal, and smaller differences than one of the other methods, respectively. We perform the 4th order method and the 8th order method with three different iterations: 100, 1,000 and 10,000 times. The data is summarized in Tables 6.14-6.16. Furthermore, we take the data with 1,000 iterations for the following analysis procedures.

Table 6.14: Performance of the SBC method, the 4th order method, and the 8th order method on the simulated bad expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 100 iterations.

Total spots number: 33489	d1>d2	d1=d2	d1<d2
4th order vs SBC	1244(3.71%)	31038(92.68%)	1207(3.61%)
8th order vs SBC	963(2.88%)	31217(93.21%)	1309(3.91%)
8th order vs 4th order	799(2.39%)	31520(94.12%)	1170(3.49%)

Table 6.15: Performance of the the SBC method, the 4th order method, and the 8th order method on the simulated bad expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

Total spots number: 33489	d1>d2	d1=d2	d1<d2
4th order vs SBC	1353(4.04%)	30467(90.98%)	1669(4.98%)
8th order vs SBC	1078(3.22%)	30714(91.71%)	1697(5.07%)
8th order vs 4th order	819(2.45%)	31543(94.19%)	1127(3.36%)

Table 6.16: Performance of the the SBC method, the 4th order method, and the 8th order method on the simulated bad expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 10,000 iterations.

Total spots number: 33489	d1>d2	d1=d2	d1<d2
4th order vs SBC	1390(4.15%)	30376(90.70%)	1723(5.15%)
8th order vs SBC	1115(3.33%)	30621(91.44%)	1753(5.23%)
8th order vs 4th order	821(2.45%)	31536(94.17%)	1132(3.38%)

Table 6.15 shows that there is no significant difference of the segmentation accuracy between the SBC method, the 4th order method, and the 8th order method.

Table 6.17 offers a comparison analysis to the same spots in the top-left corner of the original bad bovine image. Comparing Table 6.15 and Table 6.17, we find that the 4th order method performs better than the SBC method both on the bad bovine image and expanded bad bovine image.

Table 6.17: Performance of the SBC method, and the 4th order method on the top-left corner of the original simulated bad Affymetrix GeneChip image. The 4th order method is with 1,000 iterations.

Total spots	d1>d2	d1=d2	d1<d2
number: 535824			
4th order vs SBC	443(1.32%)	32097(95.84%)	949(2.83%)

6.3.2.2 Statistical Analysis on Segmentation Output Intensities on Simulated Bad Expanded Affymetrix GeneChip Image. We implement paired t-test to give a pairwise comparison on the performance of the SBC method, the 4th order method and the 8th order method, following with a UPGMA hierarchy cluster experiment to make the comparison more thoughtful.

Table 6.18 summarizes the paired t-test results on the SBC method, the 4th order method, and the 8th order method. The results show that the spot intensities generated by the SBC method, the 4th order method and the 8th order method are significantly close to the true intensities. There is no significant difference of the segmentation accuracy between the above three methods on good expanded bovine image. The UPGMA hierarchy cluster results in Figure 6.4 show consistent findings with the paired t-test experiment result. There is no significant difference of the segmentation accuracy between the above three methods on good expanded bovine image.

Table 6.18: Paired t-test result on the SBC method, the 4th order method, and the 8th order method on the simulated bad expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

Two methods on t-test	p-value	Null hypothesis	Alternative hypothesis	Reject null hypothesis
SBC vs 4 th order	0.8777	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No
SBC vs 8 th order	0.8006	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No
4 th order vs 8 th order	0.3115	$H_0 : \mu_2 \geq \mu_1$	$H_0 : \mu_2 < \mu_1$	No

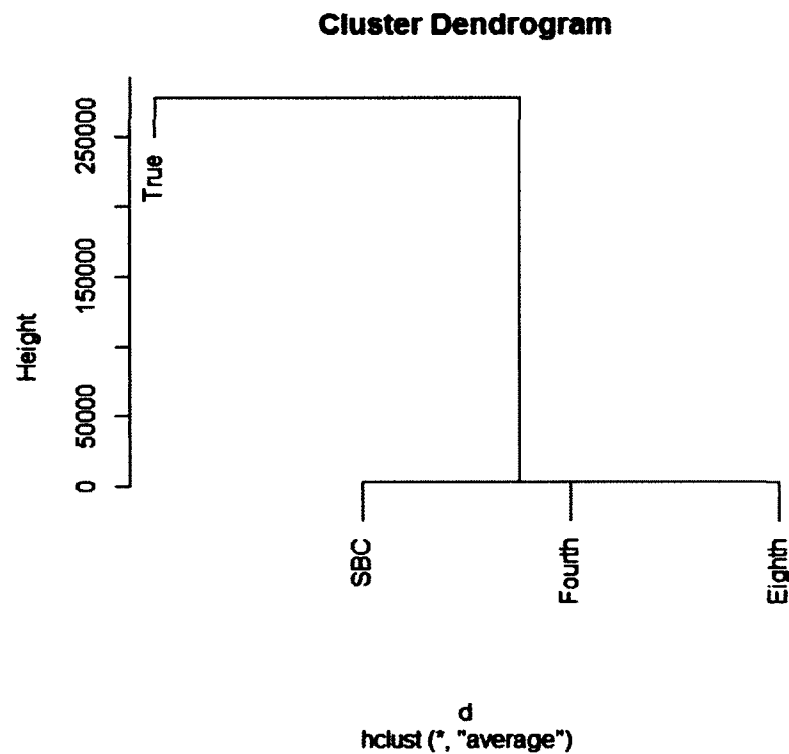


Figure 6.4: UPGMA hierarchy cluster result of comparing the SBC method, the 4th order method, and the 8th order method on the simulated bad expanded Affymetrix GeneChip image. The 4th order method and the 8th order method are with 1,000 iterations.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this dissertation, we have improved the SBC method to get two more accurate microarray image segmentation methods, the 4th order method and the 8th order method. In Chapter 4, it is proven that by implementing a higher order of finite difference schemes to discrete the C-V model provided in the ACWE method, the 4th order method and the 8th order method have higher order error terms with respect to space than the SBC method, which means that the 4th order method and the 8th order method should perform more accurate segmentations when applied on the microarray images.

In Chapter 5 and Chapter 6, we apply the 4th order method and the 8th order to the simulated cDNA microarray image and the simulated Affymetrix GeneChip images, which have all the characteristics that real microarray images do. Therefore, the performance of the 4th order method and the 8th order method could be compared with the SBC method and two other mainstream methods: the GOGAC method for cDNA microarray image segmentation, and the GCOS software for Affymetrix GeneChip image segmentation.

True intensity values of each probe spot in the simulated microarray images are provided. For valid evaluation, all the segmentation methods mentioned above are

conducted to segment the same image at each time. Their output intensity values of each spot are compared to the true intensity values, using statistical tools. The segmentation method whose spot intensity values are closer to the true value is more accurate than the other method.

A simulated cDNA microarray image, two simulated Affymetrix GeneChip images, and two simulated expanded Affymetrix GeneChip images are used in our experiment. Paired t-test and UPGMA hierarchy cluster are implemented to analyze the intensities output by those segmentation methods.

For the simulated cDNA microarray image, the intensity comparison and paired t-test show that the 4th order method and the 8th order method provide more accurate segmentation than the GOGAC method and the SBC method. The 8th order method performs better than the 4th order method on this microarray image. The UPGMA hierarchy cluster result shows that the GOGAC method is more accurate on segmentation.

For the simulated good Affymetrix GeneChip image, the intensity comparison, paired t-test and UPGMA hierarchy cluster results all show that the 4th order method is more accurate than the GCOS method and the SBC method, while the GCOS method is more accurate than the 4th order method on the simulated bad Affymetrix GeneChip image.

For the simulated expanded good and bad Affymetrix GeneChip images, the intensity analysis results show that there is no big significant difference of the segmentation accuracy between the SBC method, the 4th order method, and the 8th order method.

The Affymetrix GeneChip images we used in our research are the Bovine type GeneChip images. There are three other types of GeneChip images are available to implement our segmentation methods. They are Canine, Yeast, and Vitis. In the future, we will apply more segmentation on these three types of GeneChip images to see the performance our segmentation methods.

In the work of DNA microarray image analysis, even a small improvement in the image segmentation process could lead to a significant influence on gene expression analysis. Therefore, we have some future directions that we could work on. We could change the value of parameter μ to an even smaller one to check the subsequent influence. For the t-test analysis, since the sample size is relatively large in our experiment, it may cause a large variance that could affect our analysis results. We may figure out a way to break our microarray image into several smaller pieces to do the paired t-test. In addition, we would try to implement the 4th order method and the 8th order method on Affymetrix Exon arrays, which is a new kind of microarray that is able to detect both the gene-level expression and the exon-level expression at the same time.

APPENDIX A

CLASS CODE FOR THE 8TH ORDER SEGMENTATION METHOD

/*
 Class Code for the 8th order method. This program is an improvement of the SBC method proposed by Shenghua Ni and Yuan Cheng. Modified by Yang Li. The class code for the 4th order method is similar to this one.

*/

class segment

```
{
    // initial variables
    int  xpels, ypels ;
    int  startx, starty ;
    int  lastx, lasty ;
    double c1, c2 ;
    int  n_toreinit, n_doreinit ;
    double [] sign_d ;
    double [] area_mapping ;
    double [] gridcombine_mapping ;
    double [] forw_dx, back_dx, forw_dy, back_dy, cent_dx, cent_dy ;
    double [] intensity ;
```

```

    double h ;
    double dt ;
    double e ;
    double w ;
```

```

    // The Dirac delta funtion
    double dirac(double d)
    {
        double result=1/(Math.PI*e*(1+(d/e)*(d/e)));

        return result;
    }
```

```

    void initsigned_dist(double h,int  m,int  n,int  a)
    {
        double [] center ;
        center = new double[2];
        double r ;
        int  i, j ;
        center[0]=0;
        center[1]=0;
        center[0]=Math.floor(m/2*h);
        center[1]=Math.floor(n/2*h);
        r=Math.min((m*h-center[0]-a*h),(n*h-center[1]-a*h));
```

```

r=Math.max(r,0);

for (j=0; j<ypels;j++)
    for (i=0;i<xpels;i++)

        sign_d[i+xpels*j]=r-Math.sqrt(Math.pow((center[0]-i*h),2)+Math.pow((center[1]-j*
h),2));
    }

//compute c1,c2 value using average
void meanc1_c2()
{
    int  i, j, counter ;
    double sum1, sum2 ;
    sum1=0;
    sum2=0;
    counter=0;

    for (j=0;j<ypels;j++)
        for (i=0;i<xpels;i++)
        {

            if (sign_d[i+xpels*j] >= 0)
            {
                counter=counter+1;
                sum1=sum1+intensity[i+xpels*j];

            }
            else
                sum2=sum2+intensity[i+xpels*j];
        }
    if (counter != 0)
        c1=sum1/counter;
    if ((xpels*ypels-counter) != 0)
        c2=sum2/(xpels*ypels-counter);
}

void get_diff_results()
{
    int  i, j ;

    for (j=1; j<ypels-1;j++)
        for (i=1;i<xpels-1;i++)
        {
            forw_dx[i+xpels*j]=(sign_d[i+1+xpels*j]-sign_d[i+xpels*j])/h;
            if (forw_dx[i+xpels*j] == 0)

```

```

        forw_dx[i+xpels*j]=Math.pow(2,-23);
        back_dx[i+xpels*j]=(sign_d[i+xpels*j]-sign_d[i-1+xpels*j])/h;
        if (back_dx[i+xpels*j] == 0)
            back_dx[i+xpels*j]=Math.pow(2,-23);

        cent_dx[i+xpels*j]=(sign_d[i+1+xpels*j]-sign_d[i-1+xpels*j])/(2*h);
        if (cent_dx[i+xpels*j] == 0)
            cent_dx[i+xpels*j]=Math.pow(2,-23);

        forw_dy[i+xpels*j]=(sign_d[i+xpels*(j+1)]-sign_d[i+xpels*j])/h;
        if (forw_dy[i+xpels*j] == 0)
            forw_dy[i+xpels*j]=Math.pow(2,-23);

        back_dy[i+xpels*j]=(sign_d[i+xpels*j]-sign_d[i+xpels*(j-1)])/h;
        if (back_dy[i+xpels*j] == 0)
            back_dy[i+xpels*j]=Math.pow(2,-23);

        cent_dy[i+xpels*j]=(sign_d[i+xpels*(j+1)]-sign_d[i+xpels*(j-1)])/(2*h);
        if (cent_dy[i+xpels*j] == 0)
            cent_dy[i+xpels*j]=Math.pow(2,-23);

    }

    for (j=8; j<ypels-8;j++)
        for (i=8;i<xpels-8;i++)
        {

            forw_dx[i+xpels*j]=(-1522/560*sign_d[i+xpels*j]+8*sign_d[i+1+xpels*j]-14*sign_d[i+
            2+xpels*j]+56/3*sign_d[i+3+xpels*j]-35/2*sign_d[i+4+xpels*j]+56/5*sign_d[i+5+xpels
            *j]-14/3*sign_d[i+6+xpels*j]+8/7*sign_d[i+7+xpels*j]-1/8*sign_d[i+8+xpels*j])/(h);
            if (forw_dx[i+xpels*j] == 0)
                forw_dx[i+xpels*j]=Math.pow(2,-23);

            back_dx[i+xpels*j]=(1522/560*sign_d[i+xpels*j]-8*sign_d[i-1+xpels*j]+14*sign_d[i-2+
            xpels*j]-56/3*sign_d[i-3+xpels*j]+35/2*sign_d[i-4+xpels*j]-56/5*sign_d[i-5+xpels*j]+
            14/3*sign_d[i-6+xpels*j]-8/7*sign_d[i-7+xpels*j]+1/8*sign_d[i-8+xpels*j])/(h);
            if (back_dx[i+xpels*j] == 0)
                back_dx[i+xpels*j]=Math.pow(2,-23);
        }

```

```
cent_dx[i+xpels*j]=(4/5*(sign_d[i+1+xpels*j]-sign_d[i-1+xpels*j])-1/5*(sign_d[i+2+xpels*j]-sign_d[i-2+xpels*j])+4/105*(sign_d[i+3+xpels*j]-sign_d[i-3+xpels*j])-1/280*(sign_d[i+4+xpels*j]-sign_d[i-4+xpels*j]))/(h);
```

```
    if (cent_dx[i+xpels*j] == 0)
        cent_dx[i+xpels*j]=Math.pow(2,-23);
```

```
forw_dy[i+xpels*j]=(-1522/560*sign_d[i+xpels*j]+8*sign_d[i+xpels*(j+1)]-14*sign_d[i+xpels*(j+2)]+56/3*sign_d[i+xpels*(j+3)]-35/2*sign_d[i+xpels*(j+4)]+56/5*sign_d[i+xpels*(j+5)]-14/3*sign_d[i+xpels*(j+6)]+8/7*sign_d[i+xpels*(j+7)]-1/8*sign_d[i+xpels*(j+8)])/(h);
```

```
    if (forw_dy[i+xpels*j] == 0)
        forw_dy[i+xpels*j]=Math.pow(2,-23);
```

```
back_dy[i+xpels*j]=(1522/560*sign_d[i+xpels*j]-8*sign_d[i+xpels*(j-1)]+14*sign_d[i+xpels*(j-2)]-56/3*sign_d[i+xpels*(j-3)]+35/2*sign_d[i+xpels*(j-4)]-56/5*sign_d[i+xpels*(j-5)]+14/3*sign_d[i+xpels*(j-6)]-8/7*sign_d[i+xpels*(j-7)]+1/8*sign_d[i+xpels*(j-8)])/(h);
```

```
    if (back_dy[i+xpels*j] == 0)
        back_dy[i+xpels*j]=Math.pow(2,-23);
```

```
cent_dy[i+xpels*j]=(4/5*(sign_d[i+xpels*(j+1)]-sign_d[i+xpels*(j-1)])-1/5*(sign_d[i+xpels*(j+2)]-sign_d[i+xpels*(j-2)]))+4/105*(sign_d[i+xpels*(j+3)]-sign_d[i+xpels*(j-3)]))-1/280*(sign_d[i+xpels*(j+4)]-sign_d[i+xpels*(j-4)]))/(h);
```

```
    if (cent_dy[i+xpels*j] == 0)
        cent_dy[i+xpels*j]=Math.pow(2,-23);
```

```
    }
}
```

```
void set_dt_e_w(double p_dt,double p_e,double p_w)
{
    dt=p_dt;
    e=p_e;
    w=p_w*255*255;
}
```

```
void set_init_curve(int a)
{
    initsigned_dist(h,xpels,ypels,a);
}
```

```
void create(int x, int y,int sx1,int sy1,int sx2,int sy2,double [] data_intensity)
```

```

{
    int i,j ;
    dt=0.1;
    e=1;
    w=0.01*255*255;
    h=1;
    n_toreinit=40;
    n_doreinit=8;
    xpels=x;
    ypels=y;
    startx=sx1;
    starty=sy1;
    lastx=sx2;
    lasty=sy2;

    area_mapping=new double[xpels*ypels];
    sign_d=new double[xpels*ypels];
    forw_dx=new double[xpels*ypels];
    forw_dy=new double[xpels*ypels];
    back_dx=new double[xpels*ypels];
    back_dy=new double[xpels*ypels];
    cent_dx=new double[xpels*ypels];
    cent_dy=new double[xpels*ypels];
    intensity=new double[xpels*ypels];
    initsigned_dist(h,xpels,ypels,4);
    for (j=0;j<ypels;j++)
        for (i=0;i<xpels;i++)
            intensity[i+xpels*j]=data_intensity[i+xpels*j];
}

void segment()
{
    int i,j ;
    double t ;
    double[] ea, fa, ga, ha ;
    int t_max=10 ;

    ea = new double[xpels*ypels];
    fa = new double[xpels*ypels];
    ga = new double[xpels*ypels];
    ha = new double[xpels*ypels];

    //sign_d = new double[xpels*ypels];

    t=0;

```

```

while (t <= t_max)
{
    meanc1_c2();
    get_diff_results();

    for (j=1;j<ypels-1;j++)
        for (i=1;i<xpels-1;i++)
            {

ea[i+xpels*j]=dt*dirac(sign_d[i+xpels*j])*w/(h*h*Math.sqrt(forw_dx[i+xpels*j]*forw_
dx[i+xpels*j]+cent_dy[i+xpels*j]*cent_dy[i+xpels*j]));

fa[i+xpels*j]=dt*dirac(sign_d[i+xpels*j])*w/(h*h*Math.sqrt(back_dx[i+xpels*j]*back_d
x[i+xpels*j]+cent_dy[i+xpels*j]*cent_dy[i+xpels*j]));

ga[i+xpels*j]=dt*dirac(sign_d[i+xpels*j])*w/(h*h*Math.sqrt(forw_dy[i+xpels*j]*forw_
dy[i+xpels*j]+cent_dx[i+xpels*j]*cent_dx[i+xpels*j]));

ha[i+xpels*j]=dt*dirac(sign_d[i+xpels*j])*w/(h*h*Math.sqrt(back_dy[i+xpels*j]*back_
dy[i+xpels*j]+cent_dx[i+xpels*j]*cent_dx[i+xpels*j]));
            }

        for (j=1;j<ypels-1;j++)
            for (i=1;i<xpels-1;i++)
                {
                    sign_d[i+xpels*j]=(sign_d[i+xpels*j]+ea[i+xpels*j]*sign_d[i+1+xpels*j]
+fa[i+xpels*j]*sign_d[i-1+xpels*j]+ga[i+xpels*j]*sign_d[i+xpels*(j+1)]
+ha[i+xpels*j]*sign_d[i+xpels*(j-1)]+dt*dirac(sign_d[i+xpels*j])

*(-(intensity[i+xpels*j]-c1)*(intensity[i+xpels*j]-c1)+(intensity[i+xpels*j]-c2)*(intensity
[i+xpels*j]-c2)))
                    /(1+ea[i+xpels*j]+fa[i+xpels*j]+ga[i+xpels*j]+ha[i+xpels*j]));
                }

        for (i=0;i<xpels;i++)
            {
                sign_d[i]=sign_d[i+xpels];
                sign_d[i+xpels*(ypels-1)]=sign_d[i+xpels*(ypels-2)];
            }

        for (j=0;j<ypels;j++)
            {
                sign_d[0+xpels*j]=sign_d[1+xpels*j];

```

```

        sign_d[xpels-1+xpels*j]=sign_d[xpels-2+xpels*j];
    }

    // if ((Math.floor(t/dt)%n_toreinit == 0) && (t != 0))
    //     reinitial(n_doreinit);

    t=t+dt;

    }
    for (j=0;j<ypels;j++)
        for (i=0;i<xpels;i++)
            area_mapping[i+xpels*j]=sign_d[i+xpels*j];

    ea=null;
    fa=null;
    ga=null;
    ha=null;

}

void reinitial(int n)
{
    int i, j, k;
    double[] grad_d;

    grad_d = new double[xpels*ypels];
    for (k=1;k<n+1;k++)
        for (j=1;j<ypels-1;j++)
            for (i=1;i<xpels-1;i++)
                {

                    grad_d[i+xpels*j]=Math.sqrt((((sign_d[i+1+xpels*j]-sign_d[i-1+xpels*j])/(2*h))*((sign_d[i+1+xpels*j]-sign_d[i-1+xpels*j])/(2*h))+((sign_d[i+xpels*(j+1)]-sign_d[i+xpels*(j-1)])/(2*h))*((sign_d[i+xpels*(j+1)]-sign_d[i+xpels*(j-1)])/(2*h))));

                    sign_d[i+xpels*j]=sign_d[i+xpels*j]+dt*(sign(sign_d[i+xpels*j])*(1-grad_d[i+xpels*j]));
                }

    }

double sign(double arg1)
{
    double result;

```



```

    if (arg1 < 0)
        result = -1;
    else if (arg1 > 0)
        result = 1.0;
    else
        result = 0.0;

    return result;
}

void adjust_boundary(int direct,double step,int sel_startx,int sel_starty,int
sel_lastx,int sel_lasty)
{
    int i, j ;
    int x, y ;
    int x1, x2, y1, y2 ;
    if ((sel_startx > startx))
        x1=sel_startx;
    else
        x1=startx;
    if ((sel_starty > starty))
        y1=sel_starty;
    else
        y1=starty;
    if ((sel_lastx < lastx))
        x2=sel_lastx;
    else
        x2=lastx;
    if ((sel_lasty < lasty))
        y2=sel_lasty;
    else
        y2=lasty;

    x=x2-x2+1;
    y=y2-y1+1;

    for (j=0;j<ypels;j++)
        for (i=0;i<xpels;i++)
            area_mapping[i+xpels*j]=sign_d[0];

    if (direct == -1)
    {
        for (j=0;j<y;j++)
            for (i=0;i<x;i++)
            {

```

```

        if (sign_d[i+x1-startx+xpels*(j+y1-starty)] < step)
            area_mapping[i+x1-startx+xpels*(j+y1-starty)]=-1;
        else
            area_mapping[i+x1-startx+xpels*(j+y1-starty)]=1;
    }
}
else if (direct == 1)
    for (j=0;j<y;j++)
        for (i=0;i<x;i++)

            if (sign_d[i+x1-startx+xpels*(j+y1-starty)] > step)
                area_mapping[i+x1-startx+xpels*(j+y1-starty)]=1;
            else
                area_mapping[i+x1-startx+xpels*(j+y1-starty)]=-1;
}

double areainfo(int sel_startx, int sel_starty, int sel_lastx, int sel_lasty)
{
    double result;
    int i, j ;
    int x, y ;
    int x1, x2, y1, y2 ;
    if ((sel_startx > startx))
        x1=sel_startx;
    else
        x1=startx;
    if ((sel_starty > starty))
        y1=sel_starty;
    else
        y1=starty;
    if ((sel_lastx < lastx))
        x2=sel_lastx;
    else
        x2=lastx;
    if ((sel_lasty < lasty))
        y2=sel_lasty;
    else
        y2=lasty;

    x=x2-x1+1;
    y=y2-y1+1;
    result=0;

    for (j=0;j<y;j++)
        for (i=0;i<x;i++)

```

```

        {

            if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) > 0)
                result=Math.round(result+1);
        }
        return result;
    }
}

double area_intensitymean(int sel_startx, int sel_starty, int sel_lastx, int sel_lasty)
{
    double result;
    int i, j ;
    int x, y ;
    int x1, x2, y1, y2 ;
    int n ;
    if ((sel_startx > startx))
        x1=sel_startx;
    else
        x1=startx;
    if ((sel_starty > starty))
        y1=sel_starty;
    else
        y1=starty;
    if ((sel_lastx < lastx))
        x2=sel_lastx;
    else
        x2=lastx;
    if ((sel_lasty < lasty))
        y2=sel_lasty;
    else
        y2=lasty;

    x=x2-x1+1;
    y=y2-y1+1;

    result=0;
    n=0;

    for (j=0;j<y;j++)
        for (i=0;i<x;i++)
        {

            if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) > 0)
            {

                result=result+intensity[i+x1-startx+xpels*(j+y1-starty)];
            }
        }
    }
}

```

```

        n=n+1;
    }
}
if (n != 0)
    result=Math.round(result/n);
return result;
}

void initialize(int x, int y, int sx1, int sy1, int sx2, int sy2,double []
data_intensity)
{
    int i,j ;
    dt=0.1;
    e=1;
    w=0.025*255*255;
    h=1;
    n_toreinit=40;
    n_doreinit=8;
    xpels=x;
    ypels=y;
    startx=sx1;
    starty=sy1;
    lastx=sx2;
    lasty=sy2;

    area_mapping = new double[xpels*ypels];
    sign_d=new double[xpels*ypels];
    forw_dx=new double[xpels*ypels];
    forw_dy=new double[xpels*ypels];
    back_dx=new double[xpels*ypels];
    back_dy=new double[xpels*ypels];
    cent_dx=new double[xpels*ypels];
    cent_dy=new double[xpels*ypels];
    intensity=new double[xpels*ypels];
    initsigned_dist(h,xpels,ypels,4);
    for (j=0; j< ypels;j++)
        for (i=0; i<xpels;i++)
            intensity[i+xpels*j]=data_intensity[i+xpels*j];
}

double area_intensity_75pvalue(int sel_startx,int sel_starty,int sel_lastx,int
sel_lasty)
{
    double result;

```

```

int i,j,k;
int x,y;
int x1,x2,y1,y2;
int n;
double [] data;

if ((sel_startx > startx))
    x1=sel_startx;
else
    x1=startx;
if ((sel_starty > starty))
    y1=sel_starty;
else
    y1=starty;
if ((sel_lastx < lastx))
    x2=sel_lastx;
else
    x2=lastx;
if ((sel_lasty < lasty))
    y2=sel_lasty;
else
    y2=lasty;

x=x2-x1+1;
y=y2-y1+1;

n=0;
for (j=0;j<y;j++)
    for (i=0;i<x;i++)
    {

        if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) > 0)
            n=n+1;
    }

k=0;
if (n != 0)
{
    data=new double[n];

    for (j=0;j<y;j++)
        for (i=0;i<x;i++)
        {

            if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) > 0)

```

```

        {
            data[k]=intensity[i+x1-startx+xpels*(j+y1-starty)];
            k=k+1;
        }
    }

    quicksort(data);

    result=data[(int) Math.round(n*0.75)-1];

}
else
{
    data=new double[x*y];

    for (j=0;j<y;j++)
        for (i=0;i<x;i++)
            {
                data[k]=intensity[i+x1-startx+xpels*(j+y1-starty)];
                k=k+1;
            }

    quicksort(data);

    result=data[(int) Math.round(x*y*0.75)-1];

}
return result;
}

public static void quicksort(double[] a) {
    shuffle(a); // to guard against worst-case
    quicksort(a, 0, a.length - 1);}

// quicksort a[left] to a[right]
public static void quicksort(double[] a, int left, int right) {
    if (right <= left) return;
    int i = partition(a, left, right);
    quicksort(a, left, i-1);
    quicksort(a, i+1, right);
}

```

```

// partition a[left] to a[right], assumes left < right
private static int partition(double[] a, int left, int right) {
    int i = left - 1;
    int j = right;
    while (true) {
        while (a[++i] < a[right])    // find item on left to swap
            ;                        // a[right] acts as sentinel
        while (a[right] < a[--j])    // find item on right to swap
            ;
        if (j == left) break;        // don't go out-of-bounds
        if (i >= j) break;           // check if pointers cross
        exch(a, i, j);               // swap two elements into place
    }
    exch(a, i, right);               // swap with partition element
    return i;
}

```

```

// exchange a[i] and a[j]
private static void exch(double[] a, int i, int j) {

```

```

    double swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}

```

```

// shuffle the array a[]
private static void shuffle(double[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        int r = i + (int) (Math.random() * (N-i));    // between i and N-1
        exch(a, i, r);
    }
}

```

```

double    background_intensitymedian(int    sel_startx,int    sel_starty,int
sel_lastx,int sel_lasty)
{
    double result;
    int i,j,k;
    int x,y;
    int x1,x2,y1,y2;
    int n;
    double [] data;

```

```

if ((sel_startx > startx))
    x1=sel_startx;
else
    x1=startx;
if ((sel_starty > starty))
    y1=sel_starty;
else
    y1=starty;
if ((sel_lastx < lastx))
    x2=sel_lastx;
else
    x2=lastx;
if ((sel_lasty < lasty))
    y2=sel_lasty;
else
    y2=lasty;

x=x2-x1+1;
y=y2-y1+1;

result=0;
n=0;
for (j=0;j<y;j++)
    for (i=0;i<x;i++)
    {

        if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) <= 0)
            n=n+1;
    }

k=0;
if (n != 0)
{
    data=new double[n];
    for (j=0;j<y;j++)
        for (i=0;i<x;i++)
        {

            if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) <= 0)
            {
                data[k]=intensity[i+x1-startx+xpels*(j+y1-starty)];
                k=k+1;
            }
        }
}

```



```

        quicksort(data);
        if ((n%2) == 0)
            result=Math.round(data[n/2]);
        else
            result=Math.round((data[Math.round(n/2)]+data[(n-1)/2])/2);
    }
    return result;
}

double background_intensity_75pvalue(int sel_startx,int sel_starty,int
sel_lastx,int sel_lasty)
{
    double result;
    int i,j,k;
    int x,y;
    int x1,x2,y1,y2;
    int n;
    double [] data;

    if ((sel_startx > startx))
        x1=sel_startx;
    else
        x1=startx;
    if ((sel_starty > starty))
        y1=sel_starty;
    else
        y1=starty;
    if ((sel_lastx < lastx))
        x2=sel_lastx;
    else
        x2=lastx;
    if ((sel_lasty < lasty))
        y2=sel_lasty;
    else
        y2=lasty;

    x=x2-x1+1;
    y=y2-y1+1;

    n=0;
    for (j=0;j<y;j++)
        for (i=0;i<x;i++)
            {

```

```

        if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) <= 0)
            n=n+1;
        }

        k=0;
        if (n != 0)
        {
            data=new double[n];

            for (j=0;j<y;j++)
                for (i=0;i<x;i++)
                {
                    if (sign(area_mapping[i+x1-startx+xpels*(j+y1-starty)]) <= 0)
                    {
                        data[k]=intensity[i+x1-startx+xpels*(j+y1-starty)];
                        k=k+1;
                    }
                }

            quicksort(data);
            result=data[(int) Math.round(n*0.75)-1];
        }
        else
        {
            data=new double[x*y];

            for (j=0;j<y;j++)
                for (i=0;i<x;i++)
                {
                    data[k]=intensity[i+x1-startx+xpels*(j+y1-starty)];
                    k=k+1;
                }

            quicksort(data);

            result=data[(int) Math.round(x*y*0.75)-1];
        }
        return result;
    }
}

```

REFERENCES

- [Alberts *et al.*, 2002] Bruce Alberts *et al.*, *Molecular biology of the cell (4th ed)*. Garland Science, New York, 2002.
- [Almhdie *et al.*, 2009] A. Almhdie, P. L. Pereira, S. Meme, C. Colombier, V. Brault, F. Szeremeta, B. T. Doan, R. Ledee, R. Harba, Y. Herault, J. C. Belaeil and C. Leger, "Chan-vese based method to segment mouse brain MRI images: application to cerebral malformation analysis in Trisomy 21," *17th European Signal processing Conference*, Glasgow, 2009.
- [Appleton & Talbot, 2006] B. Appleton and H. Talbot, "Globally optimal geodesic active contours," *Journal of Mathematical Imaging and Vision*, Vol. 24, pp.83-130, July 2006.
- [Aubert & Vese, 1997] Gilles Aubert and Luminita Vese, "A variational method in image recovery," *SIAM Journal of numerical analysis*, Vol. 34, No. 5, pp. 1948-1979, October 1997.
- [cDNA microarray experiment by Jeremy Buhler, 1998]
<http://www.cs.wustl.edu/~jbuhler/research/array/>
- [cDNA microarray experiment from SQL, 2006]
<http://www.scq.ubc.ca/spot-your-genes-an-overview-of-the-microarray/>
- [Chan & Vese, 2001] T. F. Chan and L. A. Vese, "Active Contours Without Edges," *IEEE Transactions On Image Processing*, Vol. 10, Issue 2, pp. 266-277, February 2001.
- [Cheng, 2013] Yuan Cheng, *New Microarray Image Segmentation Using Segmentation Based Contours Method*, Ph. D dissertation, Louisiana Tech University, Ruston, LA, 2013.
- [Datta, 2006] Susmita Datta and Somnath Datta, "Evaluation of clustering algorithms for gene expression data," *BMC Bioinformatics*, Vol. 7, Suppl 4, S17, 2006.
- [DNA microarrays]
http://en.wikipedia.org/wiki/DNA_microarray
- [DNA to protein from Northeastern Universtiy]
http://nuweb.neu.edu/bbarbiellini/CBIO3580/DOGMA/DNA_CenDog.html

[DNA transcription]

http://www.phschool.com/science/biology_place/biocoach/transcription/tcproc.html

[Exon 1.0 ST array sample dataset from Affymetrix, Inc.]

http://www.affymetrix.com/support/technical/sample_data/exon_array_data.affx

[Gohlmann & Talloen, 2009] H. Gohlmann and W. Talloen, “Gene expression studies using Affymetrix microarrays,” *Mathematical and Computational Biology*, pp. 1-100, 2009.

[Kass *et al.*, 1988] Michael Kass, Andrew Witkin and Demetri Terzopoulos, “Snakes: Active Contour Models,” *International Journal of Computer Vision*, pp. 321-331, 1988.

[Lakhotia, 1997] S.C. Lakhotia, “What is a Gene,” *Resonance*, Vol. 2, pp. 44-53, 1997.

[Lakshmi & Sankaranarayanan, 2010] S. Lakshmi and V. Sankaranarayanan, “A study of edge detection techniques for segmentation computing approaches,” *IJCA Special Issue on Computer Aided Soft Computing Techniques for Imaging and Biomedical Applications*, Vol. CASCT, pp. 35-41, 2010.

[Lele, 1991] Sanjiva K. Lele, “Compact finite difference schemes with spectral-like resolution,” *Journal of Computational Physics* 103, pp. 6-42, 1992.

[Mader, 2010] Sylvia S. Mader, *Biology (10th ed)*. McGraw – Hill, New York, 2010.

[Moelich & Chan, 2003] M. Moelich and T. Chan, “Tracking objects with the Chan-Vese algorithm,” *Computational Applied Mathematics*, Los Angeles, CA, 2003.

[Mumford & Shah, 1989] David Mumford and Jayant Shah, “Optimal approximation by piecewise smooth functions and associated variational problems,” *Communications on Pure and Applied Mathematics*, Vol. 42, pp. 577-685, 1989.

[Ni, 2009] Shenghua Ni, *DNA microarray Image Segmentation Using Active Contours Without Edges method*, Ph. D dissertation, Louisiana Tech University, Ruston, LA, 2009.

[Ni *et al.*, 2009] Shenghua Ni, Pan Wang, Mihaela Paun, Weizhong Dai, Andrei Paun. “Spotted cDNA microarray image segmentation using ACWE,” *Romanian Journal Of Information Science And Technology*, Vol. 12, pp. 249-263, 2009.

[Nykter, 2006] Matti Nykter, “Microarray simulation model user guide and parameter reference,” May 2006. <http://www.cs.tut.fi/sgn/csb/mamodel/>

[Nykter *et al.*, 2006] Matti Nykter, Tommi Aho, Miika Ahdesmäki, Pekka Ruusuvuori, Antti Lehmussola and Olli Yli-Harja, “Simulation of microarray data with realistic characteristics,” *BMC Bioinformatics*, pp. 7-349, July 2006.

[Osher & Sethian, 1988] S. Osher and J.A.Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation," *Journal of Computational Physics*, Vol. 79, pp. 12-49, 1988.

[Rencher, 2002] Alvin C. Rencher, *Methods of multivariate analysis (2nd ed.)*, John Wiley & Sons, Inc., 2002.

[Rockman & Kruglyak, 2006] M. V. Rockman and L. Kruglyak, "Genetic of global gene expression," *Nature Review Genetics*, Vol. 7, pp. 862-872, 2006.

[Roger, 2013] Roger Bumgarner, "DNA microarrays: Types, Applications and Their Future," *Current Protocols in Molecular Biology*, 0 22: Unit-22.1, Jan 2013.

[Salman, 2006] N. Salman, "Image Segmentation and Edge detection based on Chan-Vese Algorithm," *The International Arab Journal of Information Technology*, Vol. 3, No. 1, 2006.

[Sample data for GeneChip arrays from Affymetrix, Inc.]
http://www.affymetrix.com/support/technical/sample_data/demo_data.affx

[Sheeler & Bianchi, 1980] Phillip Sheeler and Donald E. Bianchi, *Cell Biology: Structure, Biochemistry, and Function*. John Wiley & Sons, New York, 1980.

[Sussman *et al.*, 1994] Mark Sussman, Peter Smereka, and Stanley Osher, "A Level Set Approach for Computing Solutions to Incompressible Two-phase Flow," Department of Mathematics, University of California, Los Angeles, CA, 1994.

[The Affymetrix Genechip data file format in Matlab]
<http://www.mathworks.com/products/demos/bioinfo/affydemo/affydemo.html>

[The guide book for the Spot software]
http://www.hca-vision.com/Spot_Documentation/Spot.pdf

[Yang *et al.*, 2002] Y. H. Yang, M. J. Buckley, S. Dudoit and T. P. Speed, "Comparison of Methods for Image Analysis on cDNA Microarray Data," *Journal of Computational & Graphical Statistics*, Vol. 11, pp. 108-136(29), November 2002.

[Weaver 2011] Robert F. Weaver, *Molecular Biology (5th ed.)*. McGraw – Hill, New York, 2011.

[Zuzan *et al.*, 2001] H. Zuzan, C. Blanchette, H. Dressman, E. Huang, S. Ishida, J. R. Marks, J. R. Nevins, R. Spang, M. West, V. E. Johnson, "Estimation of Probe Cell Locations in High-density Synthetic-oligonucleotide DNA Microarrays," *Journal of American Statistical Association*, pp. 611-631, 2001.