2004

New machine-learning-based techniques for DNA microarray image segmentation.

Li Qin
University of Windsor

New Machine-learning-based Techniques for DNA Microarray Image Segmentation

By

Li Qin

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

Your file Votre référence
ISBN: 0-612-96382-9
Our file Notre référence
ISBN: 0-612-96382-9

Abstract

Microarray technology, which provides detailed and abundant information about biological experiments, is a significant achievement in the history of biology. One of the key issues in the microarray processing is to extract quantitative information from the *spots*, which represent the *genes* in the experiments. The process of identifying the spots and separating the foreground from the background is known as *microarray image segmentation*. In this thesis, we present two methods for microarray image segmentation. First, we conduct an in-depth analysis of the influence of important factors on clustering-based microarray image segmentation algorithms. Based on our analysis, we present an optimized clustering-based algorithm for microarray image segmentation, which exploits more than one feature to gain better results comparing to the state-of-the-art clustering-based algorithms. We also consider the fact that most of the spots in a microarray image are ellipses in shape, and hence introduce a novel adaptive ellipse method. This method shows various advantages when compared to the adaptive circle method, one of the most used approaches in microarray image segmentation. The simulations on real-life microarray images show that our method is capable of extracting information from the images which is ignored by the traditional adaptive circle method, and hence showing more flexibility.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Luis Rueda, for his guidance, encouragement, and support at every stage of my graduate studies. I have learned a lot from the many talks both personally and academically. His broad knowledge in the area, deep insight into the problems, and serious attitude towards research has deeply impressed me. This thesis would not be possible without his guidance.

I am grateful to the Prof. Behnam Shahrrava and Prof. Dan Wu, who are in my thesis defense committee for their valuable comments and suggestions on the thesis drafts. I would like to thank course instructors Prof. Boubakeur Boufama, Prof. Ahmad Tawfik, and Prof. Alioune Ngom. Their courses provided the background and foundations for my thesis research. Although I can not name them all, I would like to take this opportunity to thank all the people who helped me and gave me support during my graduate studies.

Finally, I want to express my eternal thanks to my parents and my husband. I am grateful for their patience, encouragement, and generosity. Without their support, I could not finish my studies at University of Windsor.

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1 INTRODUCTION

1.1 Molecular Biology and Bioinformatics

Bioinformatics is an emerging interdisciplinary research area in which biology, computer science, and information technology merge into a single discipline. Bioinformatics uses the applications of computer technology to the analysis and management of biological data. It involves the production, extraction, storage, retrieval, and analysis of nucleic acid sequences, protein sequences and structural information. Various kinds of databases manage the sequences and structural information, and provide methods to access, search, visualize and retrieve the information.

On the other hand, the four basic types of molecules involved in life are *amino acids*, *proteins*, *DNA*, and *RNA*. The later three types of molecules are known collectively as *biological macromolecules*. The amino acids are small molecules that can either be regarded as the building blocks of the macromolecules or have independent functions. DNA, which carries the genomic information, is the basic molecule that directly controls the fundamental biology of life. A DNA genome consists of functional regions known as *genes*.

The process of expressing genes into proteins involves two steps, namely *transcription* and *translation*, which are described below.

(i) *Transcription*. DNA sequences are first transcribed into mRNA sequences. During this process, macromolecules can be coded as alphabet strings. A string that encodes a DNA/RNA sequence is called *nucleotide sequence*; each element of the string is called a *base*. A DNA macromolecule is composed of nucleic acid, which is constructed from each of the four kinds of bases, namely A, G, C, and T. An mRNA macromolecule is also composed of nucleic acid with four kinds of bases, namely A, G, C, and U.

(ii) *Translation*. The second step in gene expression is protein extraction. The string for protein molecules is called *protein sequence*, and each element in the sequence is called *amino acid*. Proteins are sequences of 20 different amino acids. The amino acid sequences of proteins perform various cellular functions. Protein and RNA molecules form three-dimensional structures that determine the function of the macromolecules. Nucleotide and protein sequences together are called *bio-sequences*, or simply *sequences*.

The discipline of bioinformatics has been developed because of the need to understand DNA, the *code of life*. DNA codes for genes, and genes code for proteins that determine the biological composition and functions of humans or any living organism. Simple organisms such as bacteria have genomes in 1-5 million bases, whereas human genomes are approximately in 3000 megabase [1]. A human genome contains more than 30,000 genes.

The ultimate goal of bioinformatics is to discover the biological information from the massive sequence data and obtain a clearer insight into the fundamental biology of

[1] One megabase is equivalent to 1,000,000 bases.

organisms [2Ca-04]. This information can be used to benefit the standard of life for mankind as well as to reveal the possible promising unifying principles in biology. The new knowledge will have profound impacts on fields such as human health, molecular medicine, agriculture, the environment, and energy. For instance, in the area of molecular medicine, bioinformatics technologies are used to produce more efficient and customized medicines to prevent or cure diseases. In environmental areas, they can be used to identify bacteria that consume waste. And in agriculture, they can be used for producing high yield, low maintenance crops.

While the genomic era is coming, a massive explosion in the amount of biological experimental data is available due to huge advances in the fields of molecular biology and genomics. One of the most influencing technologies in those fields is the microarray technology. The analysis of the data fell behind the discovery of the data. The greatest challenge that the molecular biology community faces today is to make sense of the large data that is being produced by the new technologies.

1.2 Microarrays

Microarray technology makes use of the sequence resources created by the genome projects and other sequencing efforts to find out which genes are expressed in a particular cell of an organism, at a certain time and under specific conditions [Bro-99]. Measuring gene expression levels in different conditions provides biologists better understanding of gene functions. For example, microarrays allow comparison of gene expression between normal and cancerous cells. There are different names referring to the same technology -- DNA microarrays, DNA arrays, DNA chips, gene chips, etc. In this thesis, we use the term

"microarrays". Microarrays have enabled simultaneous measurements of the expression levels of thousands of genes.

A microarray is typically a glass slide, onto which DNA molecules are attached at fixed locations, i.e. *spots*, each related to a single gene. Microarrays exploit the theory of preferential binding of complementary single-stranded nucleic acid sequences, i.e. complementary single stranded nucleic acid sequences tend to attract to each other and the longer the complementary parts, the stronger the attraction. Most of the microarray experiments compare gene-expression levels from two samples, one called *target level* and the other the *control level*. The two samples are labeled by synthesizing single stranded DNAs that are complementary to the extracted mRNA by a special enzyme [Sch-99].

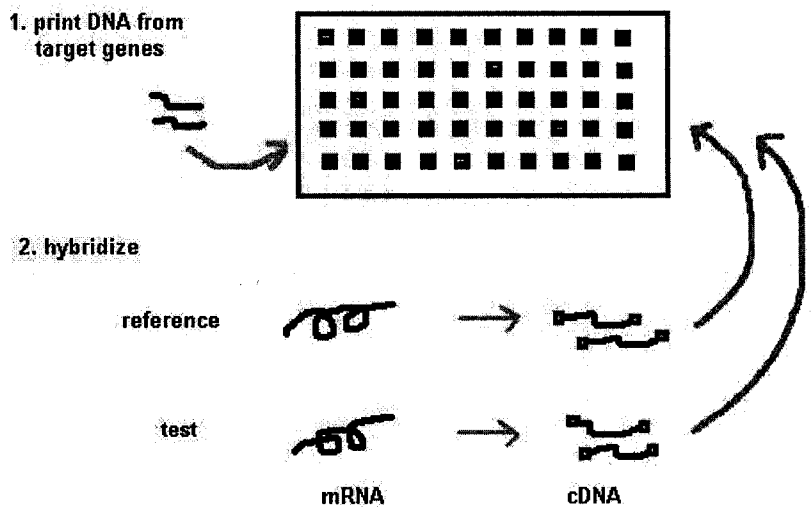


Figure 1. The production process of a microarray. The genes of interest are spotted on the surface of the microarray. Then, two samples of mRNA are converted into DNA segments, dyed with fluorescence, are eluted to the microarray surface. After the complementary sequences are combined, the microarray chip is washed.

A typical production process of a microarray is depicted in Figure 1. The spots are

either printed on the microarrays by a robot, or synthesized by photolithography or ink-jet printing. After the complementary DNA of the target genes is generated and laid out on the chip surfaces at defined positions, the DNA extracted from two samples, labeled with fluorescence dyes, is eluted over the surface, which will bind with complementary DNA. The result of the bounding of DNA is detected by fluorescence following laser excitation. In order to read out the abundance of the DNA, Cy5 and Cy3 fluorescent probes are prepared from two mRNA sources to be compared. The dyes enable the amount of sample bound to a spot to be measured by the level of fluorescence emitted when excited by a laser and detected with a scanning confocal microscope. The relative intensity of Cy5/Cy3 (red/green) probes is a reliable measure of the relative abundance of specific mRNA's in each sample. In order to obtain the intensity, the microarray image is processed so that each gene in the microarray is identified, and the intensity of the signal and its surrounding areas are extracted. The ratio between the signals in the two channels is then calculated for each spot. Based on quick napkin calculations [Ans-01], the number of DNA molecules in a microarray spot is approximately 107. For gene expression studies, each of these spots typically identifies one gene in the genome.

Figure 2 shows a sample microarray image that contains thousands of genes, each corresponding to a spot in the image. The physical dimension of such an array is about one inch or less, and the spot diameter is of the order of 0.1 mm, for some microarray types being even smaller. If the RNA from the sample in the condition that is dyed using Cy3 is in abundance, the spot will turn to be green. As opposed to this, if the RNA from the sample in the condition that is dyed using Cy5 is in abundance, it will appear red. If both are equal, the spot will be close to yellow, while if neither is present it will appear black. Thus, from

the fluorescence intensities and colors of each spot, the relative expression levels of the genes in both samples can be estimated. Figure 3 shows the color reflected by different abundance of fluorescence intensities in the two channels.

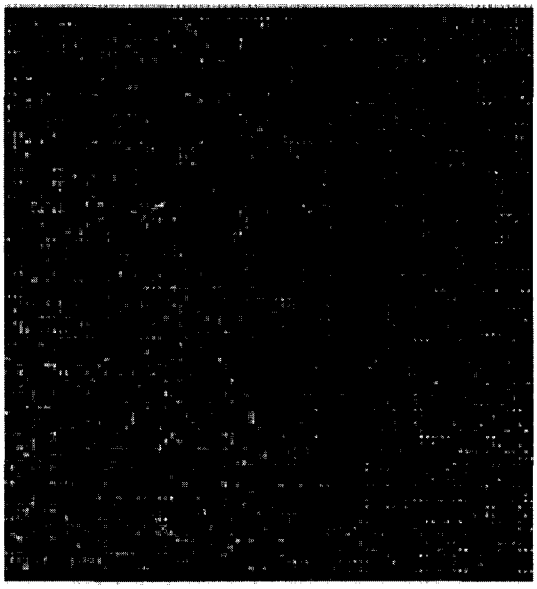


Figure 2. An enlarged illuminated microarray [2Ca-04]. The picture shows thousands of spots in a single microarray chip, each spot standing for a gene. The color of each spot reflects the relative abundance of the two fluorescence intensities.

The raw data produced from microarray experiments are called the *hybridized microarray images*. To obtain information about gene expression levels, these images have to be analyzed, each spot on the array identified, its intensity measured and compared to the background. This process is called *image quantization*. As microarray technology is still rapidly growing, it is natural that, at present, there are no established standards for microarray experiments and how the raw data should be processed. There are also no standard measurement units for gene expression levels.

Today's biological experiments of microarrays are producing massive amounts of data [Sta-04a] [Dow-04]. These data can help us to gain insights into underlying biological

processes, only if they are carefully extracted and stored in databases, where they can be retrieved and analyzed. Some of the published data can be accessed publicly, such as Stanford yeast data, which covers 6,000 genes corresponding to a series of experiments [Sta-04b], and the Cancer Genome Anatomy Project (CGAP) data at NCI/NCBI [CGA-04].

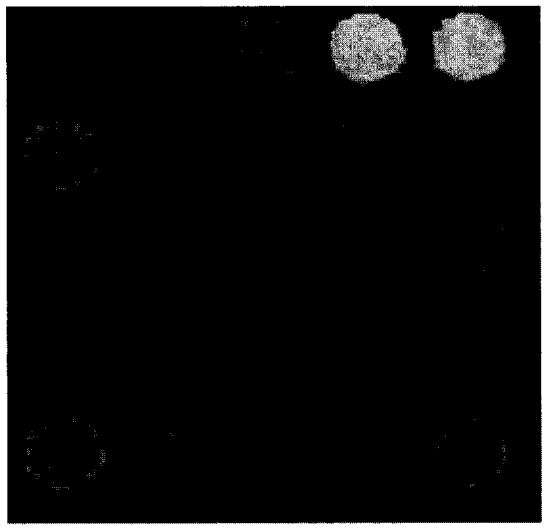


Figure 3. An enlarged subset of microarray image [NPL-04]. Each spot appears to be green or red if the RNA from the sample dyed by Cy3 or Ct5 (respectively) is in abundance. Yellow spots indicate that both are equal.

1.3 Applications of Microarrays

Microarray analysis is a significant achievement in the history of biology, because no other technology has used such sophisticated tools, combined expertise from many different disciplines, and provided such detailed and abundant information about bio-sequences.

As DNA microarray technologies emerge, they conform to simple, yet efficient tools for experimental explorations of genomic structures, gene expression programs, gene

function, and cell and organism biology. It is widely believed that gene expression data contain information that allows us to understand higher-order structures of organisms and their behavior. Besides their scientific significance, gene expression data have important applications in pharmaceutical and clinical research. For example, the comparison of the gene expression levels before and after cancer treatment helps identify the genes that the drug affects. Thus, this process helps provide a quicker and more accurate diagnosis and the subsequent treatment of the disease.

Although microarrays are a new emerging technology, they have already been widely adopted, and many users are now going beyond exploratory studies. Microarrays are being exploited in human diseases, drug discovery, and genetic screening and diagnostics. The most promising commercial application of microarrays is their potential use in clinical diagnostics. Its potential application goes from drug discovery to gene-based diagnostics. The most appropriate treatments can be reached by the study of changing the expression of genes over time, among tissues, and disease status. In addition, microarrays have a huge potential impact in the areas of preventative medicine, ability to diagnose accurately the disease, and design drugs that treat disease causes, rather than symptoms.

1.4 Microarray Data Processing

Measuring gene expression levels in different conditions provides biologists better understanding of the genes. In general, the analysis of DNA microarray gene expression data involves two steps. The first step is image quantization, i.e. the extraction of gene expression data. The microarray image is processed in order to obtain the ratios of the intensities of the two probes for each gene. This is a very important step as the accuracy of

the resulting data is essential in posterior analysis. The second step is gene expression data analysis. After the ratios of the intensities are obtained, various methods can be applied to cluster the genes into different function groups based on the ratios retrieved in the first step.

In the past few years, more than twenty [Fuc-04] commercial software and free packages have been devised, which are used to extract the intensities of each genes and conduct further analysis, being the most popular ones ScanAlyze [Eis-99], GenePix [Axo-99], ScanArray Express [GSI-99], and Spot [Buc-00] etc.

1.4.1 Scanning

The hybridized arrays are scanned to measure the red and green fluorescence intensities for each spot on the glass slide. These fluorescence intensities correspond to the level of hybridization of the two samples to the DNA sequences laid on the slide. The scanned original images are stored as a pair of 16-bit TIFF files, typically 2.5 -- 20 MB in size, where one channel is for the testing samples and the other for the reference.

Gene expression levels in an array can vary in a wide range of orders of magnitude. It is necessary to measure signals over a wide dynamic range, from saturated signals to signals that are lost in the background noise. Additionally, the integration and scaling of the two data sets involve difficult tasks. The two commonly used fluorescences have different physical properties. The spots are widely separated comparing to their individual size, so the spots may be hybridized, washed, and scanned. These principles of microarray production lead to the fact that spots are not of the same size and some spots have high or low intensity. In Figure 4, a sample image from the Apo AI data [Apo-04] shows spots that

have different sizes.

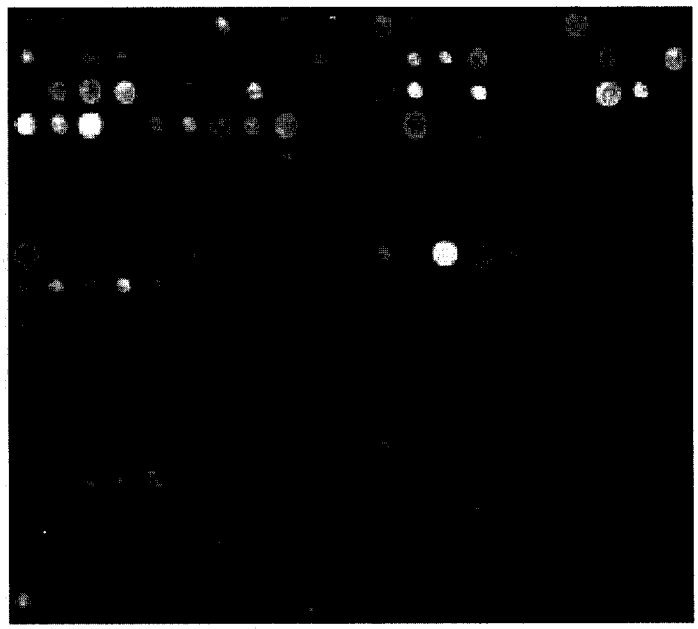


Figure 4. A sample microarray image of ApoA1 microarray data [Apo-04]. It is typical that the sizes and the intensities of the spots vary in a wide range in a microarray image.

Using a fluorescent signal, the diameter of the spots printed on glass slide range from 25 to 100 μm (1000 μm = 1 mm). There are two main kinds of microarray chips, traditional microarray and Affymetrix array. The size of a traditional microarray is 1x3 inches, i.e. 25x76x0.94 mm whereas the size for Affymetrix array is 0.5x0.5 inches. The typical resolution is 3-5 μm per pixel. The minimum resolution that this type of arrays can reach is 8 pixels/dimension (assuming the pixels lie on a two-dimensional space), i.e. 64 pixels/spot. The range of quotient of brightest and dimmest signals is in a 1000-fold dynamic range, while reducing the background can increase the dynamic range when detecting signals [Sch-02]. The most commonly used devices are laser scanners equipped with photomultiplier tubes, whereas the most advanced scanners can detect 0.1-0.01 fluors/μm^2.

1.4.2 Addressing

The process of *addressing*, which is also called *gridding*, refers to the identification of the center coordinates of each spot. The basic structure of a microarray image is often provided by the manufacturer. This includes the number of sub-grids in the image, the number of rows and columns in each sub-grid, and often, the coordinate of a marker position. After gridding, each spot is identified and linked to a unique identifier. In Figure 5, we show a gridded microarray image. For better results, the foreground region should be located at the center of the each grid.

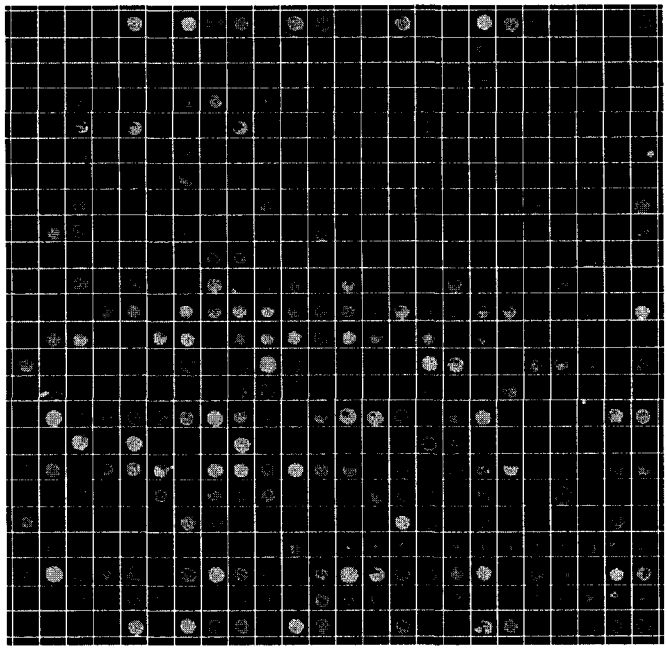


Figure 5. A gridded microarray image [Mic-04]. Horizontal and vertical lines separate the spots. For the best result, the foreground should locate at the center of the each grid.

Some issues need to be addressed when processing the image, including the misregistration of the red and green channels, the skewness of the sub-grids in the image, and the rotation of the grid axes relative to the image. The task of addressing is often done

manually or interactively, usually by drag-and-drop of the mouse, sometimes together with providing initial parameters. User intervention may make the process very slow. However, when automatic addressing is desired, the following problems also need to be considered: the contamination of the slide surface, the variety of the spot shape and size, and the highly dynamic signal intensities. Usually, an initial guess of the parameters is given as the input for automatic addressing techniques to start the process. Sample parameters include the number of sub-grids, columns and rows in a sub-grid, row spacing, row and column resolution, tip spacing, and spot width and height. A few recent studies focus on automatic microarray image analysis [Kat-02] [Jai-02] [Ste-01]. A reliable addressing procedure is desirable to ensure the accuracy of the subsequential process discussed below.

1.4.3 Segmentation

In general, segmentation of an image refers to the process of partitioning the image into several regions, each having its own properties [Soi-99]. In microarray image processing, segmentation refers to the classification of pixels as either the signal or the surrounding area, i.e. *foreground* or *background*. The microarray image contains noisy pixels that come from contamination during different stages when producing the chips. The process of segmentation should be able to distinguish noise pixels and true foreground pixels.

As the intensities of some spots are weak in nature, image processing of microarray data must be able to identify weak spots. The estimation of the signal background components is quite crucial, especially for weak spots, because weak signals often correspond to rare transcripts that are extremely important biologically. Some studies show

that the gene expression level of weak spots not as reliable as bright spots [Yue-01], [Jen-02]. In order to increase the accuracy of data, a general criterion for data extraction is that signals lower than 1.5-fold above the median background are considered unreliable and will be discarded.

1.4.4 Background Correction

After the identification of spots, or the foreground, the next step in processing a microarray image is background correction. The pioneering paper on the statistical robustness of background correction and normalization is [Che-97].

The measured fluorescence intensity of a spot includes a contribution that is not due to the hybridization of mRNA samples to the spotted DNA. Besides contamination, the background also contains information about scan and hybridization effects, which can be used to correct spot signals. To reduce bias, background correction is recommended. A common procedure is to subtract the background intensity from that of the foreground for each channel before calculating the ratio of the two channels, following the equation:

$$I_t = I_f - I_b, \tag{1.1}$$

where I_t is the true intensity of the spot, i.e. the intensity after eliminating the influence of the contamination and scan effects and hybridization effects. I_f is the foreground intensity, which is measured in the foreground region, and I_b is the background intensity, which is measured in the background region.

Equation (1.1) sometimes gives negative true intensity value for weak spots, thus makes the data for those spots unusable. An approach for improving background correction

for low intensity spots using a Bayesian based method can be found in [Koo-00].

Although recent experiments suggest that DNA on slides mask background, which means it is not additive [Wlt-04], local background correction is recommended for most microarray images because even small variations in the background signal can affect the ratio values. In fact, background correction is preformed by most of the microarray processing packages.

1.4.5 Normalization

It has been conjectured that the process of normalization contributes more than background correction [Gor-01]. Normalization is conducted either within a single microarray slide or among multiple slides. Usually, the normalization factor is independent of the location, while in some cases it is not. Thus, this should be checked for each experimental setup. Also, the identification of weak spots can improve the performance of normalization [Yan-01]. Various normalization methods exist, most of them falling into one of the following categories.

i) *Housekeeping genes:* Chen et al. [Che-97] proposed this method in 1997. Housekeeping genes are selected based on both a biological basis and their experimental behavior [Nih-04]. The ratio of housekeeping genes in the same group of experiments is close to 1.0. The main problem is how to identify such genes in each experiment. This method is useful when cells reach a stable steady state [Sch-00].

ii) *Control spots:* Exogenous RNA is added as housekeeping genes to both sample and

control conditions. The drawback of this approach is that an offset may be induced because all expression ratios are divided by an unknown factor, therefore, a systematic preference for incorporation of Cy5 or Cy3 may exist.

iii) *Constant majority:* Assume that the majority of genes do not change their expression level. Rigorous treatment for this process can be found in [Che-97]. The corresponding coefficient of variation does not need to be estimated, and does not rely on knowing the subset of genes that is constant with the condition that genes are not spotted on the array in any particular order.

iv) *Integral balance:* Assume that the total levels of gene expression in the sample and control conditions are the same, provided that cells will not significantly increase or decrease the total level of mRNA transcription.

After performing scanning, gridding, segmentation, background correction, and normalization, the ratio or logarithm of the ratio of the intensities for the two channels has to be extracted. Figure 6 shows the result of a normalization process using housekeeping genes. The systematic bias shown in Figure 6(a) and 6(b) is eliminated using the normalization process as shown in Figure 6(c). The reddish columns with low intensities are not unwanted artifacts but housekeeping genes or so called negative controls.

1.4.6 Gene Expression Data Analysis

While common *features*, or *patterns*, are regarded to be important for the biological functions of the macromolecules, a wide range of domains is involved in analyzing

sequence data, among which, the most prolific field is pattern recognition.

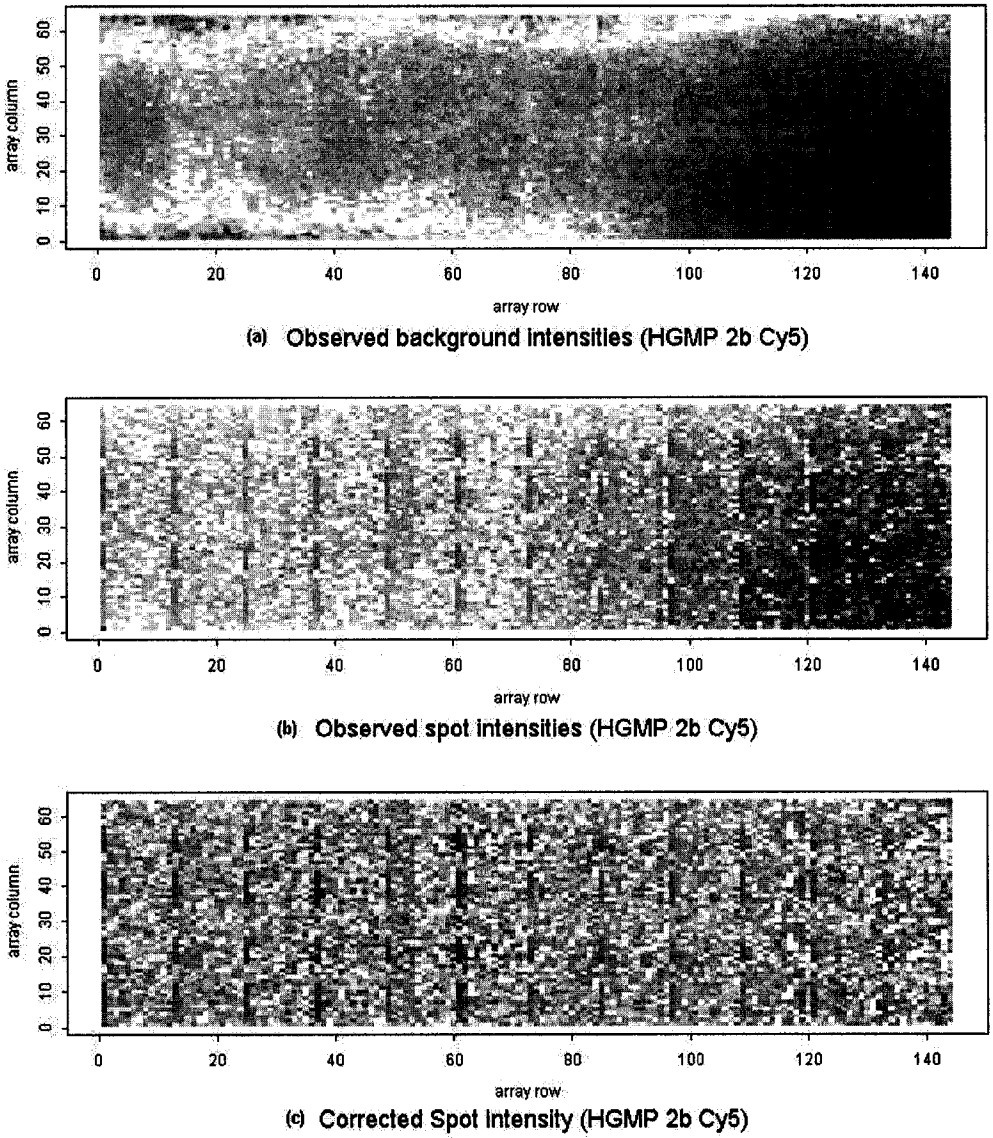


Figure 6. The effect of normalization [Vas-04]. The images are shown in false colors (blue=high, red=low). The image (a) and (b) illustrate a systematic shift of the background and foreground intensities dependent to the location of the spots, and (c) shows the result of bias eliminated after the process of normalization.

Whether the training samples are available or not, a pattern recognition system can fall into either of the two categories, namely, *supervised learning* or *unsupervised learning* respectively [Dua-00]. In supervised learning, each sample in the training set is labeled and

the cost of mislabeling samples is given. After the system is adjusted to achieve the minimum cost according to the result of classifying the training samples, it is used to assign labels to unknown objects. In unsupervised learning, no prior knowledge about the object is assumed, and hence the system attempts to group datasets into *natural groups*, or *clusters*.

In supervised learning, support vector machines (SVMs) are one of the most successful methods in gene expression data classification (see [Jaa-99], [Muk-98], [Spe-98], [Zie-00], [Cai-01], [Sch-01], [Din-01], [Cam-02], and [Guy-02]). Gene expression data classification is typically a high-dimensional problem, i.e. it involves an overwhelming number of features compared to the relatively small number of samples. Because of the powerful features of SVMs and the nature of gene-expression data, SVMs are well suited for high-dimensional classification problems.

In unsupervised learning, hierarchical clustering is one of the most widely used methods in gene expression analysis (see [Spe-98], [Wen-98], [Alo-99], [Tib-99], [Per-99], [Fur-00], [Bit-00], and [Bic-01]), even though Tamayo et al. [Tam-96] mentioned that hierarchical clustering lacks robustness and the solution may not be unique because the result is dependent on the order of the data. Apart from this drawback, hierarchical clustering has difficulties in dealing with noise.

Other pattern recognition techniques found in the literature include nearest-neighbor classifiers, aggregating classifiers, the naive Bayesian approach, self-organizing maps, hidden Markov models, *k*-means clustering, neural networks, and classifier combination or fusion.

1.5 Problem Formulation

In this thesis, we deal with the following problem, which is called microarray image segmentation. The aim is to partition the microarray image pixels into different regions or groups. As a result, the foreground pixels fall into one group, and background pixels fall into another group. There may exist other types of pixels, such as noisy pixels, which are contaminated pixels produced during microarray production and scanning, and should be excluded from the either the background or the foreground region during segmentation. Depending on the approaches to classify the pixels, another possible type of pixels includes the edge pixels surrounding the foreground region. Because the intensities of these pixels fall in between the foreground and the background, including or excluding them will lead to different signal to noise ratios.

In a few words, it can be said that the goal of segmentation is to obtain the foreground intensity and background intensity of each spot in the microarray image. The problem can be stated more formally as follows.

Let R be an m-by-n integer-valued matrix that represents the image corresponding to Cy5, $\{R(i,j) \mid i=1,2,\ldots,m; j=1,2,\ldots,n\}$. G be an m-by-n integer-valued matrix that represents the image corresponding to Cy3, $\{G(i,j) \mid i=1,2,\ldots,m; j=1,2,\ldots,n\}$. A pixel is an element of an image. We use $R(i,j)$ to refer to the pixel p_{ij} at row i column j of an image R. We define I as the image obtained after combining R and G using some arbitrary function $f(.,.)$, i.e. $I(i,j) = f(R(i,j), G(i,j))$.

Assume we deal with c clusters $\{\omega_1,\ldots, \omega_c\}$, each representing one of the c categories of pixel intensities. In general, it is assumed that there are two clusters of interest, namely ω_1 and ω_2, which represent foreground and background pixels respectively.

In our model, we use a real-valued, d-dimensional feature vector $\mathbf{x}=[x_1,\ldots,x_d]^t$ to represent the features (or information) that we can extract from a pixel p. The aim is to assign each pixel of R, G, or I, to one of the pre-defined classes, $\omega_1,\ldots\omega_c$. In particular, if we are dealing with the two-class problem, the result of the segmentation method will be a black and white or binary image B, $\{B(i,j) \mid i=1,2,\ldots,m; j=1,2,\ldots,n$ where $B(i,j)$ equals to either 0 or 255$\}$. After the label of classes is assigned to every pixel in the image, the foreground and background intensity can be computed using many different statistical measures of the two sets.

Part of the notation that we use in this thesis includes the following:

Number of background pixels: N_{bg}

Number of foreground pixels: N_{fg}

Number of noise pixels: N_n

Background intensity: I_{bg}

Foreground intensity: I_{fg}

Coordinate of a pixel: x,y

Feature matrix: $R^{l\,x\,d}$, where $l=m*n$, the number of pixels in the image, d equals to the number of features.

1.6 Contributions of the Thesis

In this thesis, we systematically introduce microarray technology, its background information, and microarray data processing. We review the existing approaches on microarray image segmentation, and discussed the advantages and drawbacks of each approach.

We analyze the different aspects of applying clustering techniques to microarray image segmentation, including the features, the number of clusters, and the clustering models. We conclude that a certain set of parameters taken as the input of the clustering algorithm generate better result. We, thus, propose a new approach for microarray image segmentation that involve more than one pixel feature, more than two clusters, and different clustering algorithm. We compare our result with the latest clustering-based microarray image segmentation methods. Our experiments show that our model is simpler and our algorithm is efficient and yielding better results.

We also developed a novel shape-based method, which we call *adaptive ellipse method.* Due to the fact that most of the spots in a microarray are circle or ellipse in shape, this method first use diagonalization to transform the feature space into another space where the shape of each spot is a circle. We then calculate a threshold to pre-classify the pixels into foreground and background, and subsequently use an algorithm to find the radius that defines the foreground region. The pixels whose radius is smaller than that radius are classified as foreground pixels. Our adaptive ellipse method produces quite good results compared to the adaptive circle methods, and can be applied to a much wider range of

microarray images.

1.7 Outline of the thesis

In this chapter, we introduce the microarray technology and its background information, and provide a formal description of the microarray image segmentation problem, which we will discuss later in detail. In Chapter 2, we introduce the existing approaches for microarray image segmentation and discuss the advantages and disadvantages of each approach. In Chapter 3, we discuss the preliminaries of clustering methods, and introduce new models for microarray image segmentation using different clustering methods and tuning the parameters of each method. In Chapter 4, we present a novel method for microarray image segmentation that we called the adaptive ellipse method. In Chapter 5, we conclude the thesis with contributions and future work.

CHAPTER 2 MICROARRAY IMAGE SEGMENTATION

In general, image segmentation is the process of distinguishing objects from the background [Asa-96]. Image segmentation is usually the first step in vision systems, and is the basis for further processing such as description or recognition. The goal of segmentation is to extract important features from images. Segmentation of an image can also be seen, in practice, as the classification of each image pixel to be assigned to one of the image compositions.

Most image segmentation approaches can be placed in one of five categories: clustering or threshold-based methods, boundary detection, region growing, shape-based methods, and hybrid methods. Many image segmentation approaches are intended for specific application domains to yield better results. For instance, real-time image segmentation, color image segmentation, 3-D image segmentation, motion image segmentation, etc.

Clustering or thresholding methods were one of the earliest image segmentation techniques (see [Puz-99] and [Wol-04]). In these methods, the information about the pixel and its neighbors is used to classify the pixel into one of the many regions.

Boundary detection or edge-based methods focus on contour detection. The image is segmented based on spatial discontinuity or edge finding and linking. This method is implemented as the convolution of mathematical gradient operators or template matching

operators that use multiple templates at different orientations of the image. Sobel, Prewitt and Laplacian operators are examples of edge detection operators.

The region growing method performs image segmentation based on spatial similarity among pixels. The image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving the homogeneity or sharpness of the region boundaries.

Shape-based methods utilize some knowledge of the shape of the object to be segmented, e.g., mathematical morphology and template matching.

Although many methods exist for general image segmentation, specialized methods have been designed for microarray image analysis. These methods are able to consider the characteristics of the microarray image. While there are quite a few, most of them discussed in this chapter, they are being perfected so as to maximize the information being extracted from the microarray image.

2.1 Fixed Circle Segmentation

Fixed circle segmentation was first used in ScanAlyze. It assignes all the spots the same size and shape. It uses a constant-diameter circle as the shape of all the spots in the image. GenePix and ScanArray Express also provide the option for fixed circle method.

Fixed circle method is simple to implement, and works well when all the spots are circular and have approximately the same size. But as the shape and size of spots varies in

practice, it clearly can not satisfy the needs. Figure 7 shows the resulting image after applying the fixed circle approach. We can see that some regions within the high intensity areas are left out of the foreground, and some regions within the low intensity areas are included in the foreground regions.

2.2 Adaptive Circle Segmentation

Roughly speaking, adaptive circle segmentation considers the shape of each spot as a circle, where the center and diameter of the circle is estimated for each spot. An implementation of this approach can be found in GenePix, ScanAlyze, ScanArray Express, Imagene [Ima-04], and Dapple [Buh-00].

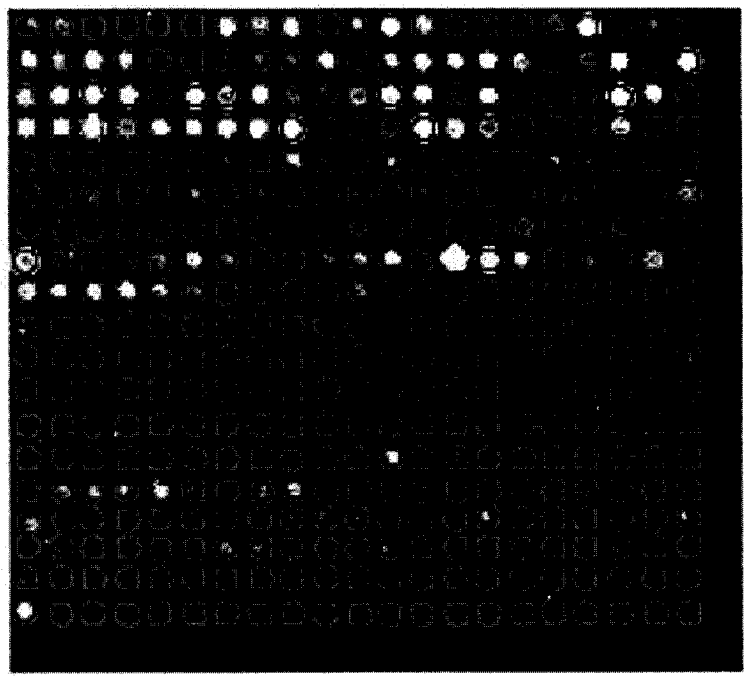


Figure 7. A sample microarray image from ApoA1 microarray data that shows the limitations of the fixed circle approach. Even though the actual size of spots varies, the radius of all the spots is the same.

Adaptive circle segmentation involves two steps. First, the center of each spot needs to

be estimated. Second, the diameter of the circle has to be adjusted. There are different implementations for this approach. A typical edge detection technique, e.g. Laplacian transformation, can be applied to automatically estimate the diameter of the circle [Buh-00]. Another algorithm considers all pixels above a user-specified threshold to be foreground and finds the circle with the highest percentage of pixels that are foreground [MAG-03]. As opposed to automatic approaches, some software packages allow the user to manually adjust the diameter of each spot.

Algorithm 1 Adaptive_Circle_Laplacian

Input: a pair of microarray images, G and R.
Output: a binary image, B.
Method:

```
I(i,j) ← G(i,j) + R(i,j)
k1 ← CreateFilter('ave', [3,3])
I2 ← Convolution(I, k1) // smooth the image
k2 ← CreateFilter('log', [5,5], 0.5)
I3 = Convolution(I2, k2)        // apply Laplacian of Gaussian filter
for i ← 1 to m do               // count zero crossings
   for j ← 1 to n do
      edge(i,j) ← CountZeroCrossing(I3)
   endfor
endfor
for n ← maxSpot                 // find the radius of the foreground region
   for i ← spotWidth
      for j ← spotHeight
         if edge(i, j) = 255 then
            distance(i, j) ← GetDistance(i, j)
         endif
      endfor
   endfor
```

```
    radius ← Mean(distance)
    for i ← spotWidth
      for j ← spotHeight
        if distance(i, j) < radius
        then  B(i, j) ← 255
        else  B(i, j) ← 0
        endif
      endfor
    endfor
  endfor
  end Algorithm Adaptive_Circle_Laplacian
```

The adopted algorithm for the adaptive circle segmentation in this thesis is shown in Algorithm 1. The centers of each spot are obtained as the center mean obtained from the weighted average of the pixel coordinates using the intensity of the pixel as the weight. Then, the Laplacian filter is applied in order to perform convolution on the image. Convolution provides a way of "multiplying together" two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality [Dai-04]. Equation (2.1) shows the function of convolution for two-dimensional images. In the equation, $m \times n$ is the size of the image, and $k \times l$ is the size of the filter. After the edge pixels are detected, the diameter of the spots can be estimated by computing the median of the radius of the edge pixels.

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l) \qquad (2.1)$$

One of the drawbacks of the Laplacian transformation is its extreme sensitivity to noise. Prior to the Laplacian transformation, the image has to be enhanced by applying a smoothing operator, such as the average filter, the Gaussian blur, or the Laplacian of

Gaussian (LoG) filter in a single step [HIP-04]. In our algorithm, we apply both the average filter and the LoG filter. Using the detected edges to estimate the diameter can avoid the drawback of disconnectivity of Laplacian filters. By smoothing the pictures and carefully selecting the parameters of the filters, the typical double edge problem of Laplacian can also be avoided. Figure 8 depicts the resulting image after applying our adaptive circle segmentation algorithm to the Apo AI image.

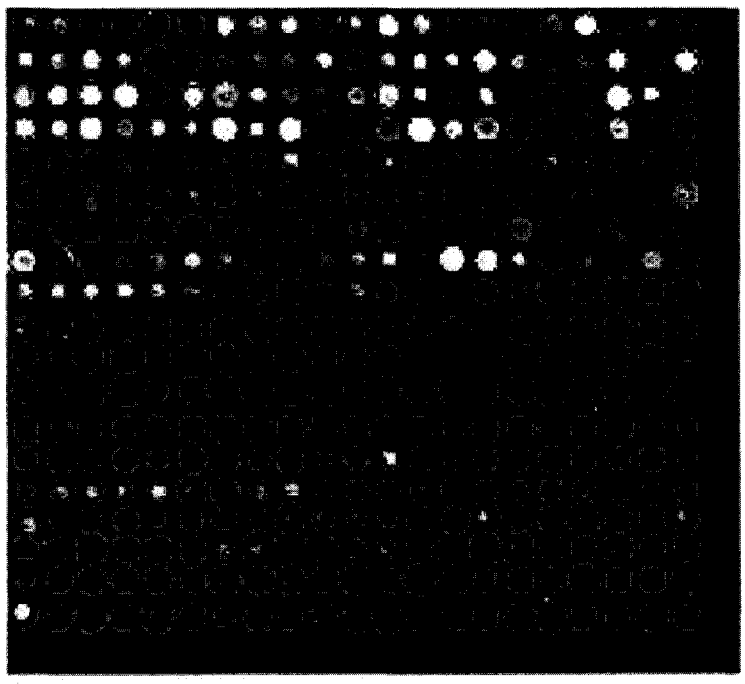


Figure 8. The result of our proposed adaptive circle method to an ApoA1 microarray image. The center of each spot is calculated as the weighted mean of coordinates, using intensity as the weight, and radius of the spots is estimated by applying Laplacian transformation.

The main drawback of the adaptive circle segmentation algorithm is that it restricts the shape of the spots to be circles. However, the spots in a microarray image can take shapes including ellipses, donuts, and other irregular shapes. As it will be seen later, assuming that the shapes of the spots are ellipses is advantageous.

2.3 Adaptive Shape Segmentation

Seeded region growing (SRG) and watershed are common techniques that deal with different shapes in image segmentation. In SRG, the regions grow outwards from the seed points, preferentially, based on the difference between the pixel value and the running mean of values in an adjoining region [Ada-94]. These two methods require an initial point to be known, which is called the *seed*. One advantage of using SRG in microarray image processing is that the location of foreground pixels and background pixels can be estimated.

SRG has been implemented in microarray processing package Spot. In this package, the foreground seed is chosen as the center pixel of the horizontal and vertical grid line. To avoid the situation when the spot is small and the grid center is slipped out of the spot foreground, a small numbers of $n \times n$ square pixels, whose center has the maximum intensity in a small area around the grid center, are taken as foreground seeds. The background seed is chosen as the point in which the grid lines intersect. After obtaining the seeds, the process is repeated simultaneously for both foreground and background regions until all the pixels are assigned to either foreground or background. Those pixels that are adjacent to a region are assigned first according to its intensity [Yan-02].

Although SRG can be applied to microarray images containing spots of any shape and size, it has the disadvantage that the spots generated are generally smaller than the actual size. Refer to Figure 9 in which, the software package Spot is used to process the same Apo AI microarray image as in Figure 7 and Figure 8. We can observe that the resulting shapes

of the spots are irregular.

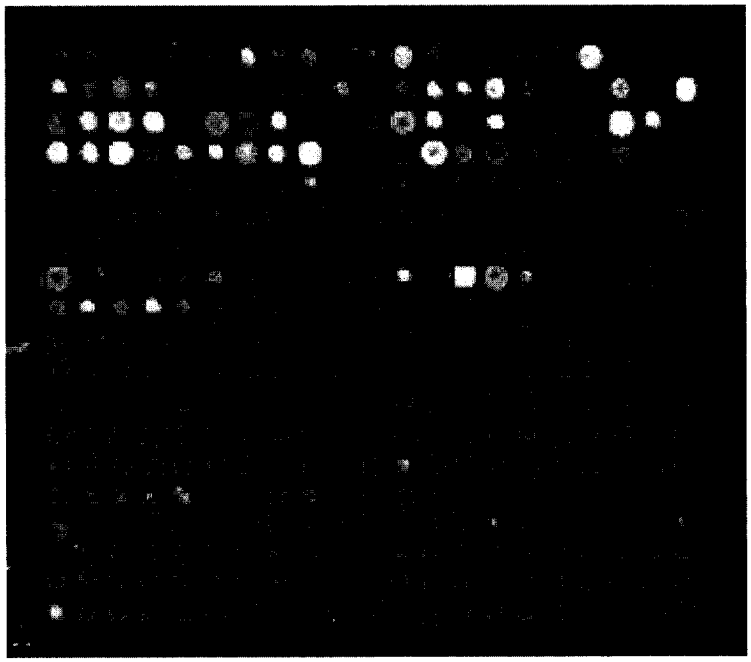


Figure 9. Spot's result of seeded region growing on the ApoA1 microarray data, which is the same image as we use in Figure 7 and 8. The shapes of spots are not restricted to circles.

2.4 Histogram-based Segmentation

Using histograms to classify a pixel into either foreground or background is a simple and intuitive idea. Chen et al. introduced a method that uses a circular target mask to cover all the foreground pixels, and computes a threshold using Mann-Whitney test [Che-97]. If the pixel intensity is greater than a certain threshold, it is assigned to the foreground region; otherwise it is assigned to the background region. They use a statistical test to find sets of signal pixels with intensities significantly different from the local background. The application of Chen's method can be found in ScanArray Express, which also provides another version of a histogram method that uses a square target mask, and defines the ratio of foreground and background as the mean intensities between predefined percentile values,

usually 5%-20% for background, and 80%-95% for foreground.

Histogram methods are simple in concept, but a suitable size of the mask is difficult to choose in comparison with the spot size. When the mask is too small, it may not cover all the foreground pixels. In contrary, when it is too large, it may overlap with neighboring spots. According to Yang et al., Chen's method is not as accurate as other recently introduced segmentation methods [Yan-02].

2.5 Clustering-based Segmentation

Clustering is one of the pioneering approaches in image segmentation. The idea of clustering can be summarized as follows. Consider a dataset $D=\{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i=[x_{i1},x_{i2},\ldots x_{id}]'$ is a d-dimensional feature vector representing the *features* of each pixel in an image. Clustering in the feature space attempts to find an indicator of similarity of image regions, and has been successfully used for segmentation purposes.

Clustering methods have some advantages when applied for microarray image segmentation, since they are not restricted to a particular shape and size for the spot. Although a clustering method has been recently proposed in microarray image analysis [Wu-03], no commercial microarray processing software has adopted this method yet.

Wu et al. used a k-means cluster algorithm in microarray image segmentation, which we refer to as single-feature k-means microarray image segmentation (SKMIS). They attempt to cluster the pixels into two groups, one for foreground, and the other for background. The first step of SKMIS method consists of initializing the class label for each

pixel and calculating the means for each cluster. Let x_{min} and x_{max} be the minimum and maximum value of intensity in the spot. If $| x_i - x_{min}| > | x_i - x_{max}|$, assign x_i belongs to foreground, or equivalently the label of the pixel x_i is set to 1. Otherwise, x_i belongs to background, thus x_i is labeled 0. After this process, the mean (or centroid) for each class, foreground or background, is calculated as

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad x_i \in \omega_j \tag{2.2}$$

This method is quite fast because the first step, i.e. the initialization, is effective. The second step of the algorithm is the re-calculation of the means and the adjustment of the label of each pixel by the following criteria. Assign $\omega_i = 2$ for all the x_i whose label is 1, if

$$\frac{N_1 * | x_i - \mu_1 |}{N_1 - 1} > \frac{N_2 * | x_i - \mu_2 |}{N_2 + 1}, \tag{2.3}$$

otherwise assign $\omega_i = 1$.

This step is repeated until no change in the means has been observed. Figure 10 shows the result of an Apo A1 microarray image after applying SKMIS.

After implementing SKMIS, we found out that using this method alone can not deal well enough with noisy data and weak spots, and improvements can be made to achieve better accuracy. There are many techniques that can be used for clustering. By choosing a suitable method and tuning the parameters, we show in the next chapter that some models are suitable for microarray image segmentation.

One of the properties of SKMIS is that for a large number of pixels, its clustering

method is equivalent to the standard *k*-means. This result is show below.

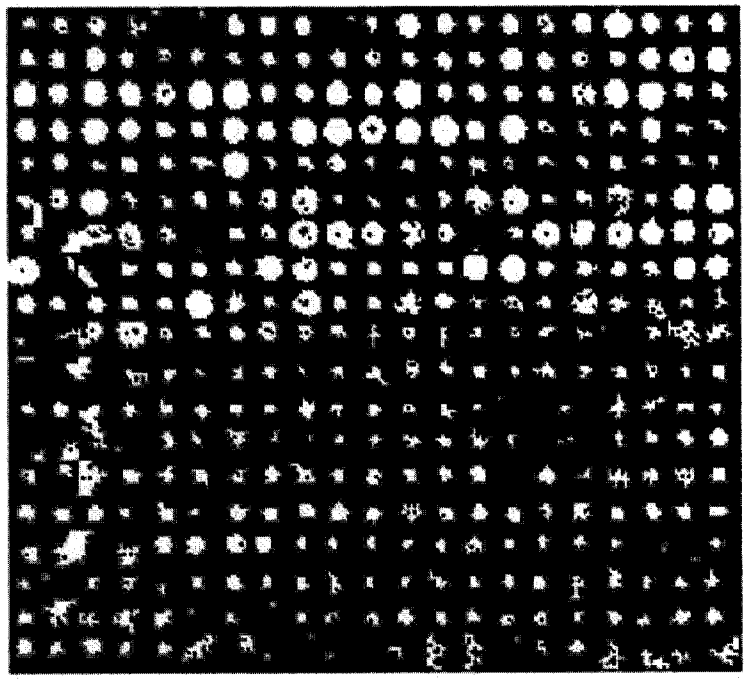


Figure 10. The result of SMIS on an ApoA1 microarray image. The foreground and the edges after eliminating the noisy pixels using LCR method show that the shapes of the spots are not restricted to circles.

Theorem 2.1: Let $D=\{x_1,\ldots,x_n\}$, which has to be clustered into two classes. If $n \rightarrow \infty$, SKMIS produces the same results as *k*-means.

Proof: Let $n=N_1+N_2$. Since empty clusters are not allowed, then it is true that as $n \rightarrow \infty$, it implies $N_1, N_2 \rightarrow \infty$. We can then write the asymptotic behavior of (2.3) as follows:

$$\lim_{N_1,N_2\rightarrow\infty} \frac{N_1 \mid x_i-\mu_1 \mid}{N_1-1} > \frac{N_2 \mid x_i-\mu_2 \mid}{N_2+1} \tag{2.4}$$

$$= \mid x_i-\mu_1 \mid > \mid x_i-\mu_2 \mid \tag{2.5}$$

Additionally, it is straightforward that, in the one-dimensional Euclidean space, (2.5) is equivalent to $(x_i-\mu_1)^2 > (x_i-\mu_1)^2$.Clearly, (2.5) is the criterion used by the standard *k*-means,

and thus, the result follows.

It is important to remark that the result of Theorem 2.1 is also applicable to the case in which N_1 and N_2 are large, but not necessarily 8 . For some microarray images N_1 and N_2 can take values, for example, above 200, and, thus, (2.4) results in

$$1.005 \mid x_i - \mu_1 \mid > 0.995 \mid x_i - \mu_2 \mid \quad \equiv \quad \mid x_i - \mu_1 \mid > 0.990 \mid x_i - \mu_2 \mid$$

2.6 Conclusion

Microarray image segmentation is a specific sub-field in image segmentation. Although many methods exist for image segmentation, in general, custom-designed methods for microarray image segmentation are desirable to achieve better accuracy by considering the characteristics of the microarray image.

Methods found in the literature can be grounded into five categories: the fixed-circle method, the adaptive circle approach, adaptive shape techniques, histogram-based methods, and clustering-based methods. The first two methods are shape-based segmentations techniques. While the first one is too simple and naive to produce good results, the adaptive circle method achieves better results for circle-shaped spots. In this thesis, we present a more general shape-based method that we call the adaptive ellipse method. The histogram, adaptive shape, and clustering-based methods do not restrict to the shape and size of the spots. The adaptive shape methods, in general, produce smaller foreground area than the actual spots. The histogram methods have been found to obtain good results, but they suffer from the difficulty in choosing a suitable mask size. In the next chapter, we discuss in detail the application of different clustering methods to microarray image segmentation.

CHAPTER 3 CLUSTERING-BASED MICROARRAY IMAGE SEGMENTATION

Image segmentation is the process of distinguishing objects from the image background. The goal of segmentation is to extract important features from images in order to assign each image pixel to one of the image parts. Microarray image segmentation aims to partition the microarray image pixels into different regions or groups, i.e. foreground and background.

Most of the image segmentation approaches fall into one of the five categories: clustering or threshold-based methods, boundary detection, region growing, shape-based techniques, and hybrid methods. Clustering based methods are one of the pioneering image segmentation techniques. Clustering based methods are built on the idea of grouping the pixels into natural groups, using the information about the pixels and their neighbors as the features.

Clustering-based algorithms have various advantages over other methods. The former does not depend on the shape of the objects in the image, albeit they can exploit the shape information as one of the many features to achieve a better segmentation for the image. Unlike region growing methods, they do not require an initial state of pixels be known. As a matter of fact, clustering-based algorithms do not need any prior knowledge about the labels of the objects be known. Common edge detection approaches have major drawbacks, such as the sensitivity to noisy pixels, the thickness of the detected edges, the

disconnectivity of the edges, and the need of post processing to assign each pixel to one part of the image after edge detection. In this sense, clustering-based methods are more flexible in dealing with the aforementioned problems.

In the domain of microarray image segmentation, the shape of a spot in a microarray image can take the form of a *circle* or an *ellipse* in most of the cases, while for others it can look like *donut* or other *irregular* shape. Clustering has many advantages over other microarray image segmentation methods, since it can deal with irregular shapes, and it can extract more information from the image, including the pixel intensity, the coordinates of the pixels, the distribution of the intensities, the characteristics of the surrounding pixels, etc. We first discuss some of the existing clustering-based techniques before applying them to process microarray images.

3.1 Clustering Methods

Clustering, which belongs to the sub-field of *unsupervised learning*, organizes a set of objects into natural groups, or *clusters*, so that each object within a group is more closely related to the objects in that group than the objects assigned to other groups. The goal of clustering is to maximize the similarity within each group, or the dissimilarity among the groups.

Clustering algorithms fall into two well-defined models, *hierarchical* and *partitional*. Hierarchical algorithms produce a tree-structured representation whose leaves are the input data and each layer of non-leaf nodes is a different level of grouping. The partitional approach is also called *flat clustering*. The aim of the partitional approach is to cluster the

data into a predefined number of clusters. In the applications of image segmentation, partitional clustering is a more suitable approach. An overview of partitional clustering methods and the preliminaries for clustering are given below.

3.1.1 Maximum Likelihood Estimation

The aim of a generic classification algorithm is to extract statistical information about the input data, which is used later to design a classifier. It is typically assumed that the data obeys a certain parametric probabilistic distribution, and hence the intent of the exercise is to estimate the *parametric* form of that distribution. Many estimation techniques have been proposed for this purpose, being *maximum likelihood estimation* (MLE) one of the most widely used method, due to its simplicity and efficiency. Suppose that we are given a dataset $D=\{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_n\}$. Assuming that we know that the data conforms to a distribution of type f, and we do not know the parameters of f, MLE can be used to estimate the parameters of f in both supervised and unsupervised learning [Dud-00].

In the case when no prior knowledge of the class labels is available, i.e. the unsupervised learning case, the MLE approach assumes the density function of the class-conditional probabilities of the dataset D of each cluster has a parametric form, typically, a *mixture* of density functions. The aim of the MLE is to estimate the parameters θ of a predetermined statistical distribution. Roughly speaking, MLE finds the value of θ that gives the density function that best represents D, i.e. it maximizes the function $P(D/\theta)$, given by the following equation:

$$p(D|\theta) = \prod_{j=1}^{n} p(\mathbf{x}_j|\theta) \tag{3.1}$$

Many statistical distributions can be assumed, being the most important one the normal distribution, because of its many properties of theoretical importance, and its application to many real-life problems.

When considering the normal distribution, the parametric form of the *probability density function* (pdf) is assumed to be known. Suppose we are dealing with a certain (probably unknown) number of clusters $\{\omega_1,\ldots, \omega_c\}$, the aim is to find the parameters of a pdf, which are obtained as a sum of c normally distributed density functions as follows:

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{c} p(\mathbf{x}|\omega_i,\theta_i)P(\omega_i) \tag{3.2}$$

This function is known as the *mixture density function*. The parameters are given by $\theta=[\theta_1,\ldots, \theta_c]^t$, where for normal distribution each $\theta_i=[\mu_i, \Sigma_i]^t$, with μ_i and Σ_i being the mean and covariance of a multivariate normally distributed random variable (or a random vector).

The aim of MLE is to maximize the likelihood of (3.1). There are many cases, depending on the information we know about the problem. A typical case is to assume that the number of clusters is known and that the parameters are unknown. The MLE solution for this case is to start with arbitrary initial values for μ_i and Σ_i, and compute $p(\omega_i|\mathbf{x}_j, \theta)$ as follows.

$$p(\omega_i \mid \mathbf{x}_j,\theta) = \frac{p(\mathbf{x}_j \mid \omega_i,\theta_i)P(\omega_i)}{\sum_{i=1}^{c} p(\mathbf{x}_j \mid \omega_i,\theta_i)P(\omega_i)} \tag{3.3}$$

Then, we compute μ_i and Σ_i, using the following equations:

$$\mu_i = \frac{\sum_{j=1}^{n} p(\omega_i \mid \mathbf{x}_j, \theta)\mathbf{x}_j}{\sum_{j=1}^{n} p(\omega_i \mid \mathbf{x}_j, \theta)} \tag{3.4}$$

$$\Sigma_i = \frac{\sum_{j=1}^{n} p(\omega_i \mid \mathbf{x}_j, \theta)(\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^t}{\sum_{j=1}^{n} p(\omega_i \mid \mathbf{x}_j, \theta)} \tag{3.5}$$

The next step is to re-compute $p(\omega_i \mid \mathbf{x}_j, \theta)$ as in (3.3) and repeat this process until there is a small change in μ_i and Σ_i. This process is implemented in algorithm **MLE_Normal_Distribution,** which is given below. In the algorithm, μ_{init} and Σ_{init} are the arbitrary values which are assigned for μ_i and Σ_i respectively.

To apply the MLE to microarray segmentation, we need to assume a pre-specified number of clusters (e.g. foreground and background), and that the distribution of the pixel intensities obeys the normal distribution. For this purpose, the features corresponding to each pixel have to be obtained.

Algorithm 2 MLE_Normal_Distribution

Input: A dataset, *D*, and the number of classes, *c*.

Output: The label vector, *label*, the parameters μ_i and Σ_i for i=1,..., c

Method:

for $i \leftarrow 1$ **to** c

$\mu_i \leftarrow \mu_{init}$

$\Sigma_i \leftarrow \Sigma_{init}$

endfor

repeat

for $i \leftarrow 1$ **to** c

```
        for j ← 1 to n
            re-compute p(ω_i | x_j, θ) as in (3.3)
        endfor
      endfor
    for i ← 1 to c
      re-compute μ_i as in (3.4)
      re-compute Σ_i as in (3.5)
    end
  until small change in μ_i and Σ_i
  for j ← 1 to n
     label[j] ← index i of max{ p(ω_i | x_j, θ)}
  endfor
end Algorithm MLE
```

3.1.2 k-Means Clustering

The most well known *k*-means algorithm was first introduced by MacQueen in 1967 [Mac-67], which can be seen as a particular case of the unsupervised MLE approach. From a statistical point of view, when we assume that the probability of each sample belonging to a certain cluster can be estimated by a mixture of normal distributions with different means but identical variance and zero covariance, the clusters obtained by *k*-means coincide with the MLE solution.

The aim of *k*-means is to form groups by specifying a desired number of clusters, say, *k*, and then assign each object to one of the *k* clusters so as to maximize the dispersion between the clusters. Each object is assigned to a cluster by using a certain "closeness", or "similarity" criterion, which are measured by using a "metric". The metric can be the distance to the centroids from the object, the sum of the variances over all clusters, the sum

of the average distances to the centroids over all clusters, the total distance between all objects and their centroids, or other measurements. The distance can be any function of the feature vectors and the clusters, being the most widely used ones the Euclidean distance, the Mahalanobis distance, the Manhattan distance, and the Minkovsky distance. The *k*-means algorithm ensures that there are always *k* clusters, and that at least one sample is assigned to each cluster.

As discussed in Chapter 2, the central idea of *k*-means is to find *k* mean vectors, $\mu_1,\ldots,$ μ_k, in such a way that $\mathbf{x}_j$ belongs to cluster *i* if μ_i is the mean vector *closest* to $\mathbf{x}_j$. The algorithm, which is shown in Algorithm **k_Means**, starts with an initial guess of the partition, i.e. the centroids are chosen randomly. The second step is to calculate the distance from each sample to all the centroids and assign each object to its closest centroid. All the centroids are re-computed as follows:

$$\mu_i = \frac{1}{n} \sum_{x_j \in \omega_j} \mathbf{x}_j \tag{3.7}$$

This step is repeated until no change in the centroids is observed.

Although *k*-means can start with random values for the initial centroids, i.e. μ_{init}, a good initialization can improve the progress of convergence, while certain initialization procedures may lead to poor results, which do not converge to the true centroids but to a *local optimum*.

Algorithm 3 k_Means

Input: a dataset *D*, the number of clusters, *k*.

Output: the label vector, **lables**, the centroids μ_i, for i=1,...,k

Method:

for $i \leftarrow 1$ **to** k

$\mu_i \leftarrow \mu_{init}$

ENDFOR

repeat

for $j \leftarrow 1$ **to** n

for $i \leftarrow 1$ **to** k

dist[i] $\leftarrow$ distance from $\mathbf{x}_j$ to μ_i

endfor

labels[j] $\leftarrow$ index i of min{*dist*[i]}

$\mu_i \leftarrow$ re-compute μ_i as in (3.6)

endfor

until no change in μ_i

end Algorithm k_Means

Another potential problem of *k*-means is how to choose a value of *k* that leads to an optimal grouping of the data. This problem is out of the scope of this dissertation. We choose the value of *k* based on the specific problem domain of microarray image processing. We will discuss later the consequences of using different values of *k* in this context.

3.1.3 Fuzzy k-Means Clustering

Fuzzy *k*-means can be seen as a generalization of the *k*-means algorithm. Instead of assigning an object to a particular class, fuzzy *k*-means assigns the class label by using the probability that the object belongs to that class. As in *k*-means, fuzzy *k*-means requires that the value of *k* be known beforehand.

The algorithm starts with an initial value for the mean for each cluster and maintains a membership matrix **M**. The subsequent task is to calculate the probability that $\mathbf{x}_j$ belongs cluster i, referred to as m_{ij}, as follows:

$$m_{ij} = \frac{(1/d_{ij})^{1/(b-1)}}{\Sigma_{i=1}^{k}(1/d_{ij})^{1/(b-1)}} \quad where \quad d_{ij} = |\mathbf{x}_j - \mu_i|^2 \tag{3.7}$$

The means are then recalculated using the following equation.

$$\mu_i = \frac{\Sigma_{j=1}^{n} m_{ij}{}^{b}\mathbf{x}_j}{\Sigma_{j=1}^{n} m_{ij}{}^{b}} \tag{3.8}$$

The process is repeated until a small change in the means is observed. The fuzzy k-means method is implemented in Algorithm **Fuzzy_k_Means**. As in **k_Means**, μ_{init} is the initial value for all the cluster centroids. Note that in (3.7) the Euclidean distance is used. As mentioned earlier, other distance functions can be used, producing different results depending on the nature of the problem.

Algorithm 4 Fuzzy_k_Means

Input: A dataset, D, the number of clusters, k, a blending factor, b

Output: A membership matrix, **M**

Method:

```
for i ← 1 to k
    μ_i ← μ_init
endfor
repeat
  for j ← 1 to n
    for i ← 1 to k
      d_ij ← | x_j - μ_i|^2
      m_ij ← re-compute m_ij as in (3.7)
    endfor
```

endfor

for $i \leftarrow 1$ **to** k

$\mu_i \leftarrow$ re-compute μ_i as in (3.8)

endfor

until small change in μ_i, i=1,...,k

end algorithm Fuzzy_k_Means

In (3.7) and (3.8), b is a *blending* factor that indicates how much the classes overlap: $b = 1$ indicates that the fuzzy k-means algorithm is equivalent to the k-means algorithm, while $b > 1$ means each $\mathbf{x}_j$ belongs to multiple clusters. As the value of b increases, the "fuzziness" of the classification increases until all or the majority of the objects belong to each cluster equally, resulting in too much "fuzziness". There is currently no established theory regarding the optimal choice of this parameter [Bez-84]. We use the value of b = 1.25 in our experiments, which has been shown to be efficient in a wide range of clustering problems.

3.2 Clustering in Microarray Image Segmentation

In this section, we discuss the application of the above-mentioned clustering methods to microarray image segmentation. We investigate the effects of adopting different features and tuning the parameters. The first step in segmentation is to apply a clustering algorithm to cluster (or classify) the pixels in a spot into foreground and background. The most suitable clustering algorithms for this purpose are the non-hierarchical ones, such as k-means, fuzzy k-means, or MLE. We compare the effect of applying these methods, the use of different features, and the application of different parameters. Our experiments are performed on the ApoA1 microarray data, obtained from [Apo-04].

3.2.1 Single Feature Clustering

Knowledge about the problem domain is usually applied to guide the process of feature selection. If the features are not sufficient to express the most representative information about the data, then no amount of computational effort will produce good clusters. However, if the feature space is too large, then expensive computations are needed to cluster the data, or when the data are incorrectly characterized, no good results can be produced at all.

As discussed in Chapter 2, SKMIS uses one feature for clustering, i.e. it calculates the square root of the intensities of the two channels and takes the maximum of these two. We implemented SKMIS and studied the effect of different measurements of the intensity based on the algorithm. The results also apply to other clustering models without losing generality.

Besides the square root of the intensity, we also used the original 16-bit intensity. In addition to the maximum of the two channels, we also used the sum of the two channels. We tried different combinations and concluded that the sum of the two square roots of the intensities is a better measurement than any other combination when the feature space is one-dimensional, i.e. it only contains the intensity. In Figure 11, we show the results of applying SKMIS and different measurements for the feature to the microarray image contained in files "1230c1G" and "1230c1R" from the ApoA1 data. We observe that if the noise intensity in a spot is too high, as in spot No. 11, using the maximum of the two 16-bit intensities finds only the noise. Table 1 shows the effect of different calculations for the

feature to the clustering result of spot No. 8 from the 1230c1G/R image. Table 2 includes a comparison of the results obtained after different measurements are used to cluster the whole 1230c1G/R image. N_{fg}, I_{fg}, and I_{bg} are defined as in Section 1.5.

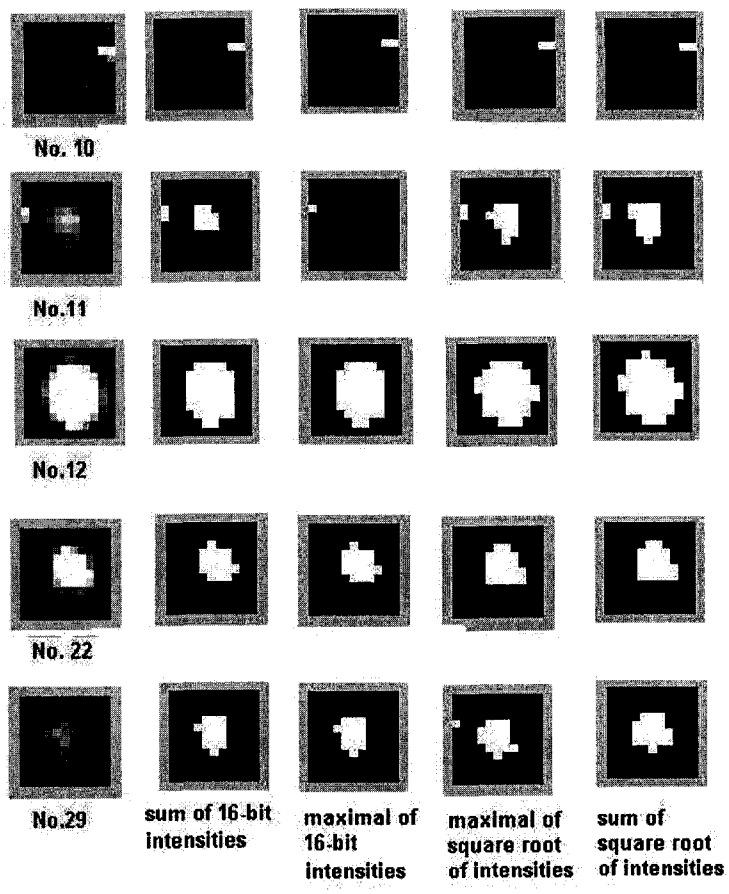


Figure 11. Sample results obtained after applying SKMIS using different calculations for the single feature to the 1230c1G/R microarray image. Using the sum of the square roots of the two channels gives a foreground that is closest to the real spot.

From Tables 1 and 2, we observe that the total number of foreground pixels is the largest using the sum of square roots of the intensities for the two channels. This corroborates the results shown in Figure 11, in which this measurement generates the largest foreground region.

Feature calculation	Square root		16-bit Intensity	
	Maximal	Sum	Maximal	Sum
N_{fg}	28	28	22	27
I_{fg} green/red	8336.0/4604.4	8336.0/4604.4	9358/4834.5	8486.1/4662.5
I_{bg} green/red	926.8925/1393.4	926.8925/1393.4	1148.8/1536.9	962.5851/1410.9

Table 1. Results obtained after applying SKMIS using different measurements for the feature for sample spot No. 8 from 1230c1G/R microarray image.

To conclude this subsection, we observe that the sum of square roots of the intensities generates the largest foreground region. These regions, which are observed in Figure 11, are the closest to the actual spot size. Since they result in larger foreground regions, the mean foreground intensity and the background intensity produce the smallest spots among the four methods.

Feature calculation	Square root		16-bit Intensity	
	Maximal	Sum	Maximal	Sum
Total N_{fg}	8,620	11,516	8,996	9,447
Total I_{fg} green/red	2.0609/1.7723 $*10^6$	2.0137/1.7014 $*10^6$	2.1576/1.8992 $*10^6$	2.1518/1.8379 $*10^6$
Total I_{bg} green/red	3.6299/5.7540 $*10^5$	3.3930/5.6170 $*10^5$	3.8647/5.8740 $*10^5$	3.7086/5.8195 $*10^5$

Table 2. Results obtained after applying different calculations for the feature using SKMIS to 1230c1G/R microarray image. The sum of square roots produces the largest foreground region among the four measurements.

3.2.2 Double Feature Clustering

Traditional image processing algorithms have been developed based on the information of the intensity of the pixel only. In the microarray image segmentation problem, we encountered that the position of the pixel, for example, could also influence the result of the clustering, and subsequently that of the segmentation. The use of these

kinds of features has also been successfully applied, for segmentation of nature pictures in [Car-02]. They showed that using the position of the pixels as additional information for the segmentation algorithm improves the quality of the results. Their results, based on human observations, are very good when compared to the traditional segmentation approaches. We have been motivated by this work, and utilized more features from the images in addition to the pixel intensity.

If we consider the shape of the spot, the pixels whose distance to the center of the spot is smaller are more likely to be foreground pixels. We can thus take this *spatial* information about the pixels into account, and construct different features. For example, we can take the Manhattan distance as one of the features, i.e. the distance from the pixel to the center of the spot in the x-axis direction and in the y-axis direction. Alternatively, we can consider the Euclidean distance from the pixel to the spot center as one of the features. In this case, the spot center refers to the weighted-mean of the coordinates using the intensity as the weight. The coordinates of the spot center is computed as follows:

$$\mathbf{c} = \frac{\sum_{i,j} I_{ij}\mathbf{p}_{ij}}{\sum_{i,j} I_{ij}} \quad , \tag{3.9}$$

where I_{ij} is the intensity of the pixel and $\mathbf{p}_{ij} = [p_x, p_y]$ contains its coordinates.

When we consider the fact that a pixel with most of its surrounding pixels belonging to the same cluster is likely to belong to the same group, we take into account the mean of the surrounding pixels within a certain distance and the variance of the intensities. Adjusting the size of the surrounding regions, we obtain different values of mean and variance for the

pixel as its features.

One of the problems of feature extraction is the fact that different features come from different sources, and thus they lie on different ranges. For example, the intensity is in a range between 0 and 65,535, while the radius is usually less than 20. When clustering a dataset with these characteristics, some features will dominate over others. Thus, different scaling methods can lead to different clustering results. The dependency among each feature, e.g. the *correlation*, should also be taken into consideration. We would like a representation of the data that reveals the natural grouping of the pixels.

The distance from the pixel to the spot center is calculated as follows:

$$r_{ij} = [(\mathbf{p}_{ij} - \mathbf{c})^t (\mathbf{p}_{ij} - \mathbf{c})]^{1/2}, \tag{3.10}$$

The feature vector is given by $\mathbf{x} = [x_1, x_2]'$, where x_1 is the pixel intensity, and x_2 is the distance from $\mathbf{c}$ to $\mathbf{p}_{ij}$ as obtained in (3.10). We call the *k*-means algorithm that uses $\mathbf{x}$ as the feature vector *double-feature k-means microarray image segmentation* (DKMIS).

In order to compare DKMIS and SKMIS, where the latter uses the intensity only, we run both methods on the 1230c1G/R microarray image. The resulting image obtained from DKMIS is shown in Figure 12. For most of the spots, the noisy pixels are excluded from the foreground when considering the information of pixel coordinates. More importantly, for some spots, using intensity alone leads to poor results. This case is shown in Figure 13, in which it is clear that the two features obtained from the intensity and distance from the center can retrieve the true foreground from the background of the spots.

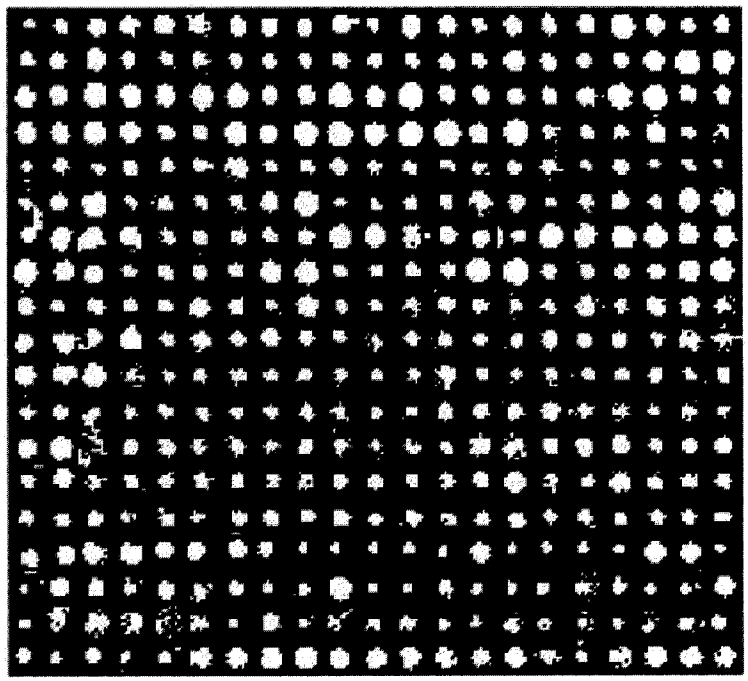


Figure 12. Using pixel intensity and distance from the pixel coordinates to the center as the features to cluster the 1230c1G/R microarray image.

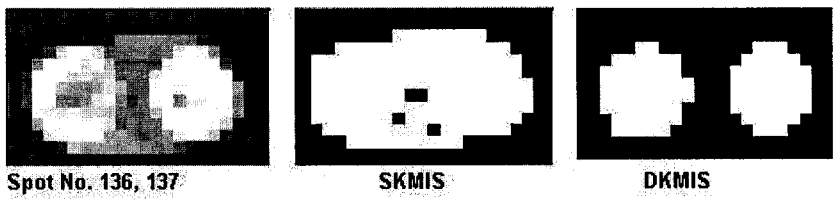


Figure 13. The result of applying SKMIS and DKMIS to spots No. 136 and 137, extracted from the 1230c1G/R microarray image. It is clear that using intensity and distance from the center as the features can reveal the true foreground for these spots.

It is important to highlight that it is not always true that using more features leads to better results. More elaborated feature extraction and normalization schemes, such as *principal component analysis* (PCA), should be used in order to expect better results. This by itself constitutes a future research avenue for the approach we propose in this thesis.

3.2.2 Varying Number of the Clusters

Apart from increasing the number of features, we also studied the result of using different numbers of the clusters. In SKMIS, by setting k to 2, the pixels are clustered into two groups, one is for foreground, and the other is for background. We tested the number of clusters based on our implementation of the k-means algorithm, described in Algorithm **k_Means**, using the intensity as the feature. Our testing consists of the segmentation of the 1230c1G/R microarray image. The results for a sample spot (No 8) and for the whole image are shown in Tables 3 and 4 respectively. In order to further analyze our methods, in Figure 14, we show the results obtained for different values of k, and for various spots of the 1230c1G/R image. From the figure, we observe that when k=2, the clusters can be classified as foreground and background. When k=3, the pixels are grouped into three clusters, one for foreground, one for background, and the other contains the pixels whose intensities are in between these two. In most of the cases, the latter constitutes the edge of the foreground region. However, in some cases, such as in spot No. 10, the third cluster represents the true foreground regions. When calculating the ratio of intensities, the edge pixels can be either included as the foreground or excluded from both.

When k=4, the results depend largely on how the clusters are grouped. In Table 4, we consider the two clusters of higher intensities as the foreground region, and the other two groups as the background region. In some cases, it generates spots which are similar to those of k=3 when the edge is considered as foreground (see Spot No. 10). In other cases, taking three clusters for the highest intensities as the foreground is a better representation

for the spot (see Spot No. 29). It is quite difficult to predict which classes correspond to foreground and which ones should be assigned to background. From our experiments we conclude that it is impractical to use k=4 when automatic microarray image segmentation is required. This is a scenario that deserves further investigation.

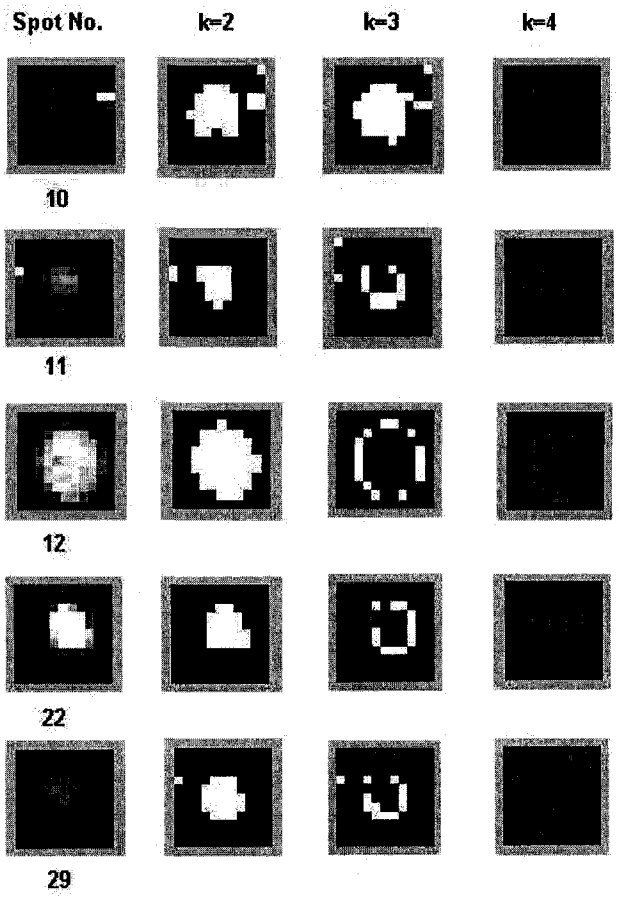


Figure 14. Resulting spots obtained from k-means with different values of k on the 1230c1G/R microarray image. While both k=2 and k=3 give reasonable results, the resulting spots for k=3 are larger than those of k=2 and are closer to the original spots.

To analyze the results from another perspective, in Figure 15, we show the scatter plot of the foreground and the background intensity of each spot. We observe that the intensities for the foreground are larger for k=3, especially for low intensity spots. Notice that the noisy pixels are excluded after applying a post-processing noise-removal method, which

we call *largest continuous region* (LCR), discussed later in this chapter. We notice that *k*-means with k=3, where the edges are included in the foreground, is the best choice for the number of clusters.

K	2	3	4
N_{fg}	28	45	28
N_{bg}	93	76	93
I_{fg} g/r	8336.0/4604.4	5854.8/3557.2	8336.0/4604.4
I_{bg} g/r	926.8925/1393.4	738.6842/1295.3	926.8925/1393.4

Table 3. Comparison for the segmentation of spot No. 8 of the 1230c1G/R microarray image using *k*-means with different values of *k*.

k	2	3	4
Total N_{fg}	10240	19608	11724
Total N_{bg}	35538	24252	32747
Total N_n	2501	4419	3808
Total I_{fg} g/r	1.9435/1.5955*10^6	1.4571/1.2567*10^6	1.7897/1.4927*10^6
Total I_{bg} g/r	3.3197/5.5679*10^6	3.3981/5.6803*10^6	3.2063/5.4963*10^6
Total I_{bg} before LCR	3.4416/5.6636*10^6	3.6107/5.8076*10^6	3.3753/5.6147*10^6

Table 4. Comparison for segmentation of 1230c1G/R microarray image using *k*-means with different values of *k*. The last row shows the background intensity before noise post-processing

If we use the 1.5-fold threshold (which is typically used by bioinformaticians) to verify whether a spot is too weak to be reliable for the analysis, the SRG method produces 49 such spots, while *k*-means with k=2 produces 45; *k*-means with k=3, instead, produces 70 of those spots when the edge is considered as foreground. The case in which k=4 produces 46 such spots. If we exclude the noisy pixels using LCR, k=2, k=3, and k=4 give 3, 0, and 5 weak spots respectively. We can, thus, safely conclude that k=3 can distinguish weak spots as good as k=2 and k=4. Based on the comparison, and the clustering results depicted in

Figure 14, we draw the conclusion that k=3 is the best number of clusters, and a more natural way of performing microarray image segmentation.

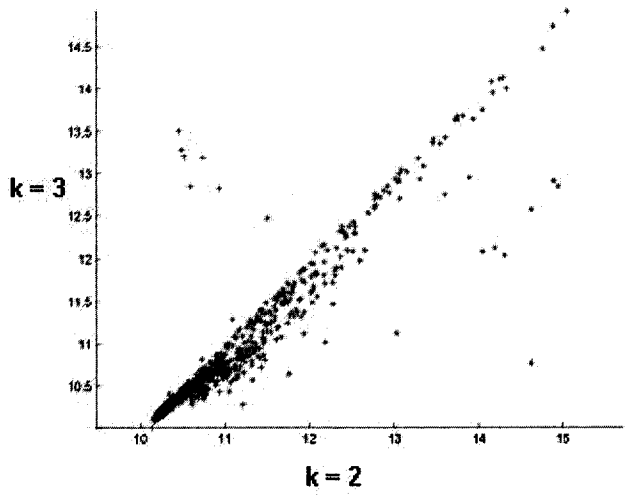


Figure 15. Scatter plot of the foreground and background intensities obtained after applying k-means to the 1230c1G/R microarray image. The y-axis represents k=3, where the edges are excluded from both foreground and background. The red and blue dots correspond to foreground and background respectively.

3.2.3 Using Different Clustering Models

The use of different clustering models can also affect the result of the microarray image segmentation. We study three common clustering models, namely k-means, fuzzy k-means, and MLE. In this section, we consider the intensity as the only feature. The algorithms are referred as SKM, SFKM, and SMLE respectively. The analysis using multi-dimensional feature space constitutes a future work for our current research.

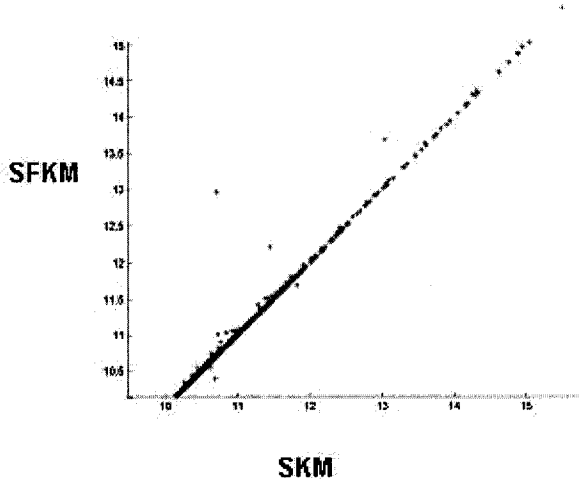


Figure 16. The spot intensities obtained after applying SKM and SFKM to the 1230c1G/R microarray image, where k=2.

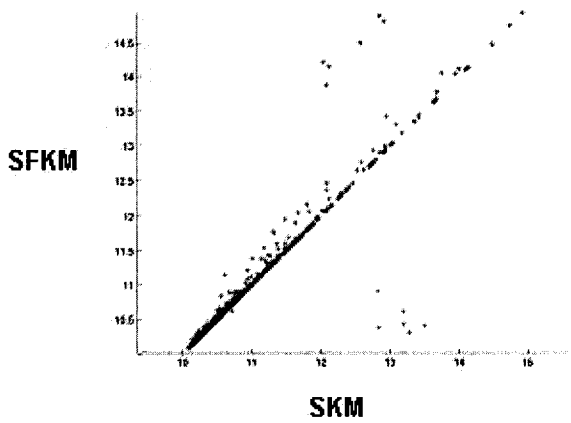


Figure 17. The spot intensities obtained after applying SKM and SFKM to the 1230c1G/R microarray image, where k=3.

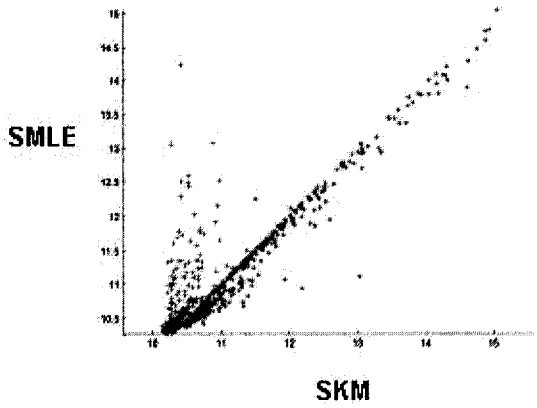


Figure 18. The spot intensities obtained after applying SKM and MLE to the 1230c1G/R microarray image, where k=2.

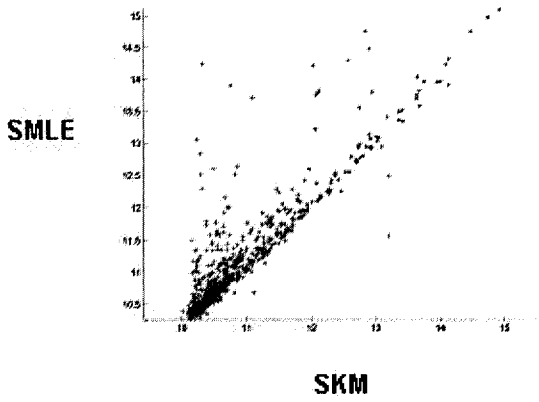


Figure 19. The spot intensities obtained after applying SKM and MLE, k=3.

In SKMIS, the distance function is computed by using Equation (2.3). As shown in Theorem 2.1, this algorithm can be regarded as the Euclidean distance in the one-dimensional space, i.e. equivalent to the Manhattan distance, when the number of pixels becomes arbitrarily large. We adopted another distance metric, which is the standard Euclidean distance, in our k-means model. The cluster membership consists of computing

the Euclidean distance from the feature vector to the center mean of the clusters, and choosing the label of the cluster that is closest to the feature vector. We also implemented the fuzzy k-means and the MLE method based on the algorithms described in section 3.1. We refer to these algorithms as SKM, SFKM, and SMLE respectively.

We now analyze the application of clustering models from anther perspective, and use scatter plots. Scatter plots show the relationship between the resulting intensities for different models. Each scatter plot compares the results for a pair of models, one axis is used for each model. Each point in the figure represents a spot, where the value of the x-axis and y-axis corresponds to the resulting spot intensity for each model. Red dots (in the figure '*') represent the foreground, and blue ones ('+') represent the background.

We ran the three clustering models mentioned above on the 1230c1G/R microarray image. Figures 16 and 17 show the comparison of the results of SKM and SFKM, where k=2 and k=3 respectively. When k=3, the edge is considered as foreground. We observe that SKM and SFKM generate almost identical results for k=2. When k=3, SFKM results in larger foreground intensities for some spots, implying that the foreground region produced is smaller.

Figures 18 and 19 show the comparison of the results of SKM and SFKM, where k=2 and k=3 respectively. When k=2, the SMLE model results in lower intensities than SKM and SFKM, because it generates larger foreground regions than the actual foreground (see Figure 18). When k=3, SMLE produces substantially large foreground regions, if the edges are included in the foreground. When the edges are considered as background, SMLE

results in much smaller foreground regions than SKM and SFKM.

In Figure 20, we also show the results obtained after applying the three methods to some spots drawn from the 1230c1G/R microarray image. We observe that SMLE produces the largest foreground region among the three models when k=2, especially for low intensity spots. When k=3, SMLE produces edges that include many background pixels. However, when excluding those edges from the foreground, it generates a foreground region that is too small.

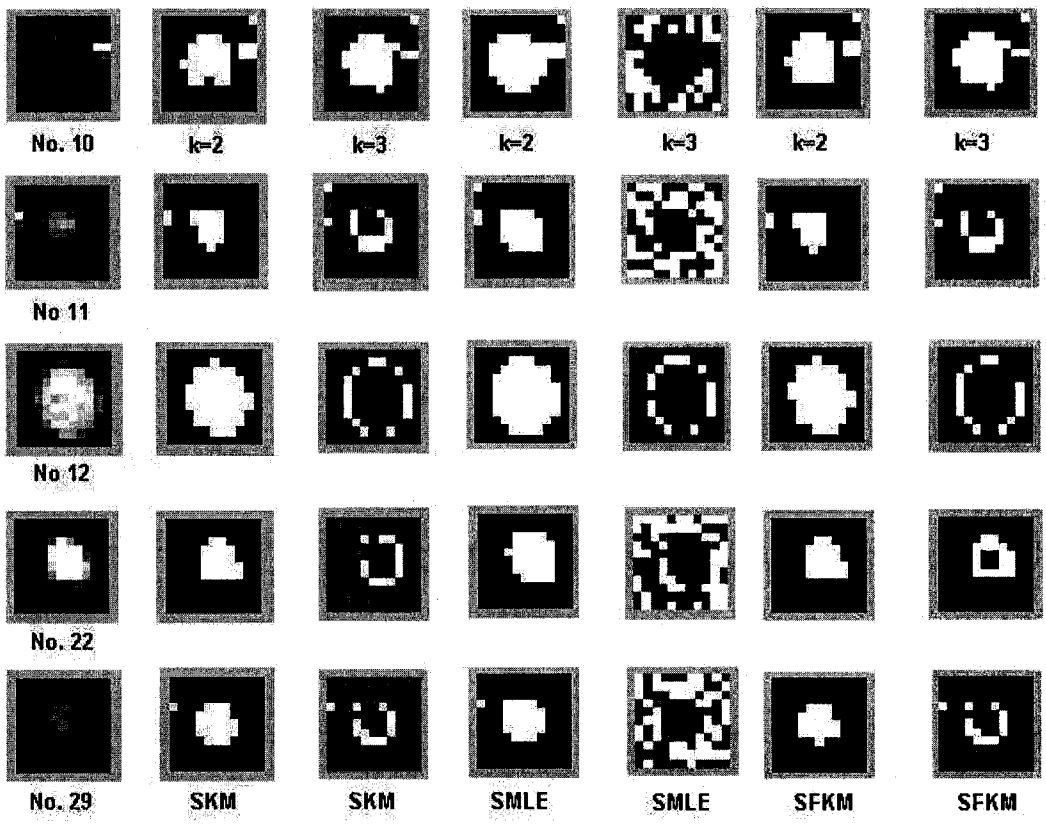


Figure 20. The results of SKM, SFKM and SMLE for some typical spots on an ApoA1 microarray data.

Apart from the poor results of SMLE, it is sensitive to initial values, which makes it difficult to be applied in practice. From our experiments, we conclude that k-means is a

more suitable model for microarray image segmentation than fuzzy *k*-means, because the former produces better results and the latter involves a slower algorithm.

3.3 Post-processing of Noisy Pixels

After applying a clustering algorithm, the resulting spots may contain true foreground pixels as well as noisy pixels due to the contamination introduced during the microarray experiments. Although different models and parameters can be chosen so that the foreground region contains very few noisy pixels, as it will be seen later, some post-processing methods may still be desirable to eliminate even more noisy pixels. The pixels labeled "foreground" contain not only the high-intensity true-foreground pixels, but also high-intensity noisy pixels scattered in the background region. Without post-processing of noisy pixels, the resulting spots may contain noise that will affect the accuracy of the subsequent steps during the experiment analysis. While it is hard to eliminate noisy pixels that reside in the foreground region, discarding noisy pixels in the background region is desired.

Noisy pixel processing can be achieved by applying many different noise removal algorithms. In the next two subsections, we introduce the different techniques: the k-means algorithm re-applied to the labeled pixels, and the largest continuous region (LCR) method.

3.3.1 Re-applying Clustering

We can apply *k*-means a second time in order to discard noisy pixels. Unlike the previous stage, in which we use the intensity as the feature to cluster foreground and background pixels, we use the distance from the center to the pixel coordinates as the

feature in the SKM algorithm to eliminate the noisy pixels in the background region.

Instead of using the coordinates of the pixels, using the distance avoids the problem of determining how many noisy clusters are in a spot. The noisy pixels can all be erased in one step and the shape of the spot is trimmed towards a circle. This shows again that it is very important to choose good features in the problem of clustering.

3.3.3 Largest Continuous Region Method

Our alternative LCR method consists of calculating the largest continuous region using the pixels that compose the foreground, assuming the spot foreground is larger than any noisy area. Since it is expected that the spot foreground is the largest cluster compared to the noisy clusters, by assuming that the foreground is a continuous region, we can easily identify the cluster by finding the area with the largest number of pixels.

The procedure that implements the LCR is shown in Algorithm LCR. The algorithm first marks each continuous region with different labels by involving a recursive function called Paint. After the first step, the algorithm obtains a mask for the spot, in which each label stands for a different continuous region. Then the algorithm counts the number of pixels for each label of the mask. The spot foreground is the region with the largest number of pixels. Finally, the algorithm clears all the other labels except those of the foreground pixels. In the algorithm, p_{ij} represent the pixel at row i, column j, and m_{ij} is the label for pixel p_{ij}.

Algorithm 5 LCR

Input: Labeled pixel matrix, **P**; the size of the image, m, n

Output: A matrix labeled only the foreground, **M**

Method:

for i ←1 **to** *n*

for j ← 1 **to** *m*

if p_{ij} is a foreground pixel **and** m_{ij} <> 0 **then**

M ← Paint(**P**, **M**, *i*, *j*, *k*); //Label the pixel and its 4 neighboring pixels with the same value

k ← *k*+1;

endif

endfor

endfor

labels ← 0;

for *i* ← 1 **to** *n*

for *j* ← 1 **to** *m*

if m_{ij} <> 0 **then**

labels[m_{ij}] ← *labels*[m_{ij}]+1; // find the number of pixels of each value of mark

endif

endfor

endfor

maxMark ← the index of max(*labels*)

clear *mask* except *maxMark*

Procedure Paint(Labeled pixel matrix, **P**; a matrix labeled only the foreground, **M**; coordinate *i*, *j*; the mark, *k*) **return** A matrix labeled only the foreground, **M**

Begin

if p_{ij} is a foreground pixel **then**

m_{ij} ← *k*;

if *j*>1 **and** $\mathbf{p}_{i,j-1}$ is a foreground pixel **and** $m_{i,j-1}$ = 0 **then**

M ← Paint(**P**, **M**, *i*, *j*-1, *k*);

endif

if *j*<*n* **and** $\mathbf{p}_{i,j+1}$ is a foreground pixel **and** $m_{i,j+1}$ = 0 **then**

M ← Paint(**P**, **M**, *i*, *j*+1, *k*);

endif

if *i*>1 **and** $\mathbf{p}_{i-1,j}$ is a foreground pixel **and** $m_{i-1,j}$ = 0 **then**

M ← Paint(**P**, **M**, *i*-1, *j*, *k*);

endif

```
  if i<m and p_{i+1,j} is a foreground pixel and m_{i+1,j} = 0 then
    M ←Paint(P, M, i+1, j, k);
  endif
 ENDIF
END
end Algorithm LCR
```

In the worst case, when all the pixels are foreground, every pixel is visited by Paint only once. The complexity of LCR is $O(n^2)$, when the size of the image is n by n.

Theorem 3.1. The worst-case time complexity of LCR is $O(n^2)$, where the size of the input image is $n \times m$, and $n = O(m)$.

Sketch of proof. In the two for-loops in the main module, every pixel is tested once for the if-statement. This includes 2 cases:

Case I: p_{ij} is a background pixel. In this case, Paint is not invoked.

Case II: p_{ij} is foreground pixel. In this case, Paint is invoked.

In Case II, if p_{ij} has a neighbor that is foreground and has not being marked, Paint will be recursively invoked. This implies that Paint will be invoked for every pixel only once. This implies that the complexity is $O(nm) = O(n^2)$, where $n = O(m)$. From Case I, we see that the worst case will occur when all pixels belong to background, in which case, the worst-case complexity is $O(n^2)$ as well.

LCR generates almost the same result as the previous SKM algorithm that uses the distance as the feature, shown in Figure 21. For spots No. 26 and 27, although they are

weak spots, the results are quite good.

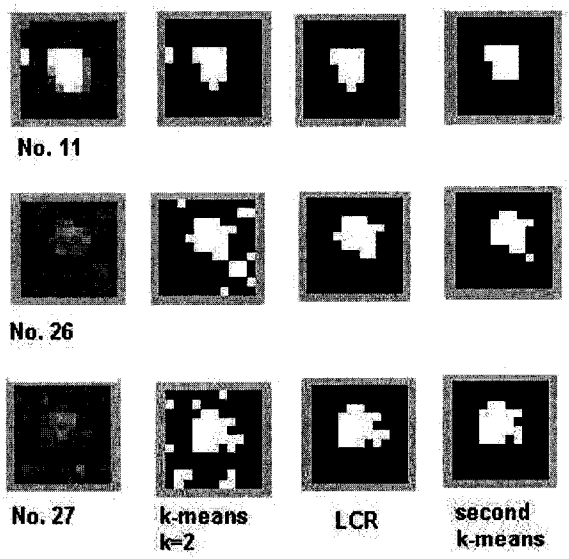


Figure 21. The post-processing of noisy pixels on the 1230c1G/R microarray image. The second *k*-means uses the distance as the feature.

3.4 Optimized Clustering-based Microarray Image Segmentation

Based on our previous analysis of the most important factors in a clustering algorithm, including the feature space, the number of clusters, and the clustering model, we present a clustering-based algorithm for microarray image segmentation, which we call *optimized k-means for microarray image segmentation* (OKMIS). OKMIS is a segmentation algorithm that uses *k*-means, and sets *k* to 3. The feature space contains two features: one being the sum of the square root of the intensities, and the other the distance from the pixel coordinates to the spot center.

In general, evaluating an image segmentation algorithm is not an easy task. Since microarray technology is still rapidly growing, there are no established standards for the analysis of microarray images. In this regard, there are no standard measurements for

evaluating the results of microarray image segmentation. Subjective methods (based on human being observation) are used in literature to evaluate the results. In this section, we combine our subjective judgment and an objective measurement to compare the SKMIS method and our OKMIS method. Because the SKMIS method generates a foreground with a significant number of noisy pixels, Wu *et al.* applied a further mathematical morphological process, which they called *foreground correction*, to eliminate the noise [Wu-03]. In our experiments, we apply our LCR to perform the foreground correction to SKMIS and OKMIS. Thus, in the simulations, we compare SKMIS, SKMIS after LCR, our OKMIS, and OKMIS after LCR. Figure 22 illustrates the result of applying OKMIS to the 1230c1G/R microarray image. It shows OKMIS resulting in a much "cleaner" foreground region than SKMIS.

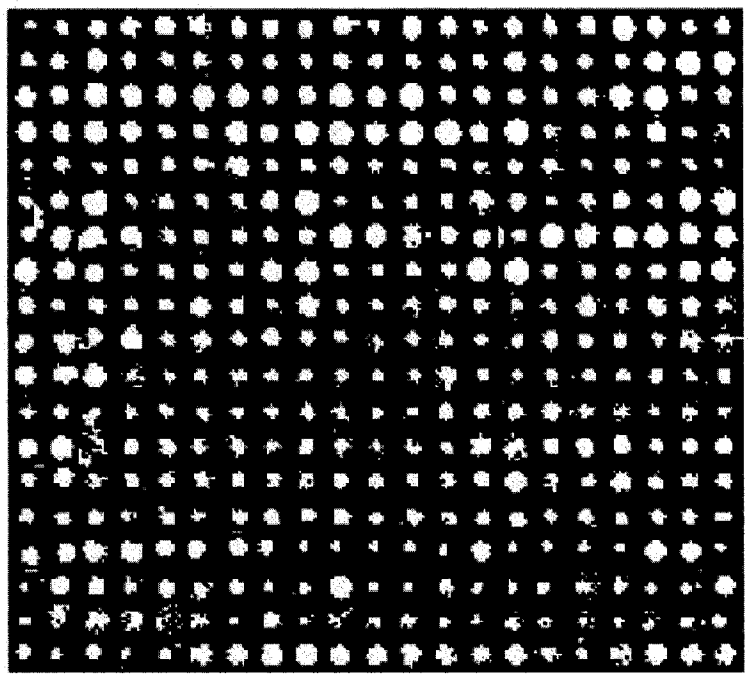


Figure 22. The result of applying OKMIS to the segmentation of the 1230c1G/R microarray image.

3.5 Experiments on Real-life Microarray Data

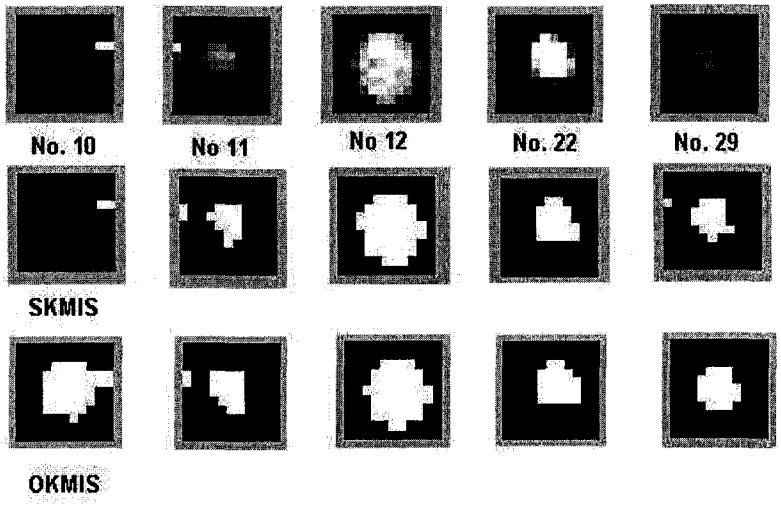


Figure 23. Comparison of SKMIS and OKMIS for some typical spots from the 1230c1G/R microarray image. For spots with high intensity noisy pixels, such as No.10, OKMIS can reveal the true spot foreground instead of the noise produced by SKMIS.

In order to obtain a more consistent assessment about our segmentation methods, and their comparison with other approaches, we performed some simulations on real-life microarray images obtained from the ApoA1 data [Apo-04]. We ran our experiments on an Intel Pentium III (633MHz) computer under Windows 2000, and using the well-known Matlab software. First of all, we compare the resulting binary images of the two clustering methods to the original microarray image. A few spots from the 1230c1G/R microarray image are shown in Figure 23. We observe that in general OKMIS achieves better results. In some cases (spot No.10), OKMIS reveals the true foreground region while SKMIS finds only the noisy pixels. In some cases (spot No.29), OKMIS can result in a foreground region that contains fewer noisy pixels. In other cases (spots No. 11, 12, 22), both OKMIS and SKMIS can obtain a reasonable foreground region, where OKMIS generates a region that is closer in size to the real spot. In particular, Figure 24 shows the comparison of OKMIS

with SKMIS for weak spots. In order to easily visualize the results, the intensities of the original spots have been enlarged 20 times. We observe that OKMIS results in larger, closer to the actual spot, and cleaner foreground regions. As it can be seen in the figures, OKMIS automatically removes most of the noisy pixels, and is more efficient than SKMIS because SKMIS must perform an additional nosy removal procedure.

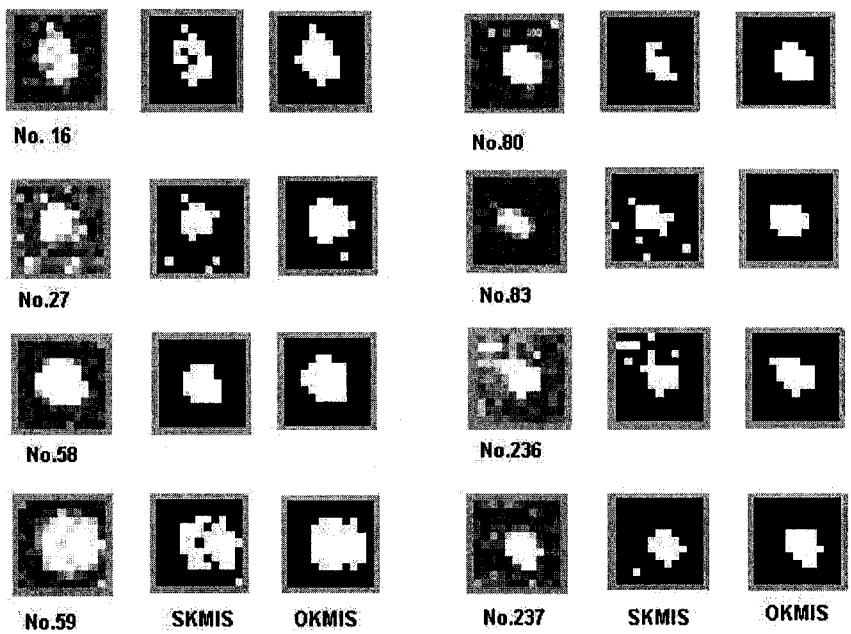


Figure 24. The comparison of SKMIS and OKMIS for some weak spots of the 1230c1G/R microarray image. The intensities of the original spots have been enlarged 20 times to achieve better visualization.

After demonstrating that OKMIS generates better results than the SKMIS method, we now provide an objective measurement for a batch of real-life microarray images. To achieve this analysis, we compare the size of the resulting foreground region for both methods. The results are shown in Table 5. The first column for each method contains the total foreground intensity of the green channel, I_{fg}, and the second column represents the number of pixels in the foreground region, N_{fg}. In the first two columns, we note that the foreground region generated by SKMIS contains many noisy pixels. Thus, a post-

processing method has to be applied in order to eliminate the noise. In the last two columns, we observe that OKMIS eliminates most of the noisy pixels -- only 1,307 noisy pixels are left for the 8 images, totaling 3,192 spots. After the LCR process is applied, the changes to the size of the foreground and the foreground intensities are so minor that can be neglected. Therefore, in these cases, post-processing is not necessary at all. As opposed to this, SKMIS generates 23,575 noisy pixels, and hence a process to eliminate the noise must be performed. Full images resulting from the adaptive ellipse method can be found in Appendix A.

	SKMIS		SKMIS after LCR		OKMIS		OKMIS after LCR	
Images	I_{fg}	N_{fg}	I_{fg}	N_{fg}	I_{fg}	N_{fg}	I_{fg}	N_{fg}
1230ko1G/R	1,013,600	14,016	1,025,300	11,508	1,049,100	11,388	1,049,000	11,234
1230ko2G/R	1,420,000	11,323	1,439,400	7,520	1,365,900	9,843	1,366,100	9,701
1230ko3G/R	1,593,100	14,502	1,618,600	10,826	1,541,900	11,102	1,541,100	10,903
1230ko4G/R	1,549,200	10,085	1,559,100	8,141	1,520,300	9,159	1,518,300	9,046
1230c1G/R	2,039,300	10,455	2,060,900	8,620	1,928,900	10,292	1,929,100	10,128
1230c2G/R	2,495,800	11,650	2,530,300	8,179	2,364,100	9,733	2,362,900	9,510
1230c3G/R	2,435,200	11,182	2,446,300	8,446	2,368,700	9,587	2,367,100	9,436
1230c4G/R	1,815,700	12,287	1,836,700	8,685	1,816,900	9,560	1,816,800	9,399
Total	14,361,900	95,500	14,516,600	71,925	13,955,800	80,664	13,950,400	79,357

Table 5. Comparison of SKMIS and OKMIS for a batch of images from the ApoA1 dataset, where the first sub-grid of each image is analyzed.

In addition to a nearly-noisy-free foreground, OKMIS generates a larger foreground region, which is closer to the real spot foreground (see Figures 23 and 24). When comparing OKMIS with SKMIS after LCR, the resulting foreground regions are larger than those of the latter for all the images except for the first pair. Thus, in most of the cases, OKMIS results in much better results than SKMIS, even after the foreground correction process. We observe that, for the weak spots, OKMIS is superior to SKMIS, in the sense that the resulting foreground contains less noisy pixels and the resulting foreground regions

are closer to the real spots. In addition, OKMIS does not require an extra post-processing stage for removing the noise.

3.6 Conclusion

In this chapter, we analyze the most important factors that influence the accuracy of a clustering algorithm for microarray image segmentation, including the feature space, the number of clusters, and the clustering model. We propose a new microarray image segmentation method based on a clustering algorithm, which we call OKMIS, and sets k to 3. Its feature space has two features: the sum of the square root of the intensities, and the distance from the pixel coordinates to the true spot center. As shown in the experiments, our algorithm performs microarray image segmentation more accurately than the previous clustering-based microarray image segmentation methods, SKMIS, and does not need a post-processing stage to eliminate the noisy pixels.

The proposed algorithm, which generates quite satisfying results, still has room for improvements. More elaborated feature extraction and normalization schemes can improve the accuracy of a clustering algorithm. When considering more than one feature, the normalization process is very important, not only by scaling, but also analyzing the correlation between each pair of features. In this regard, PCA is a widely used method that could be used to produce even better results. This problem constitutes a possible avenue for future research.

When the number of clusters is more than two, an extra step is needed to determine which clusters correspond to foreground and which ones belong to background. Although

this is not an easy task, the refined classification may lead to more interesting results. An automated clustering algorithm is desired to evaluate the best number of clusters for each spot and classify the pixels into foreground and background.

CHAPTER 4 ADAPTIVE ELLIPSE METHOD

As we discussed in Chapter 3, clustering methods work well in the problem of microarray image segmentation. If we choose the clustering model and features carefully, the clustering algorithm can achieve very good results. Yet, the clustering methods often need an extra step to erase the noise pixels from the foreground region. As opposed to this, shape-based segmentation methods have many advantages in this regard, because most of the spots in microarray images are regular in shape. In general, the shape of a microarray spot is a circle or an ellipse. Other shapes like donuts and irregular shapes can also be presented in a microarray image. In this chapter, we propose an adaptive ellipse method that works well for many microarray images.

The method that we introduce in this chapter is in its infancy and very promising. It is seen as a generalization of the adaptive circle technique. Although our results are preliminary, they show that a huge amount of work is worth to be done in this direction. We notice that the word "adaptive" usually refers to the property that an algorithm can improve itself in the procedure of learning. In microarray literature, however, it is used in adaptive circle method to refer to the fact that the algorithm results in different diameters for different sizes of spots. We inherit the meaning of the word "adaptive" as it is used in microarray literature, and call our approach *adaptive ellipse method.*

4.1 Diagonalization Transformation

Diagonalization is the process of transforming data from a multivariate normal

distribution $N(\mu, \Sigma)$ to $N(0, I)$, where μ is the mean vector and Σ is the covariance matrix, by performing linear and whitening transformations [Rue-02]. After applying diagonalization, the normally distributed data with arbitrary mean and covariance matrix is transformed to a distribution in which the covariance is the identity matrix.

Diagonalization involves two steps. For a normally distributed random vector $x \sim N(\mu, \sigma)$, first, the following linear transformation transforms x into another vector $y = \Phi^t x$, where Φ is a $d \times d$ orthogonal matrix that contains the d eigenvectors of σ, $\Phi_1 \ldots \Phi_d$. The next step is the whitening transformation, which transforms y into a new matrix $z = \Lambda^{-1/2} y$, where its random variables are uncorrelated and their variances are equal to unity, i.e. the

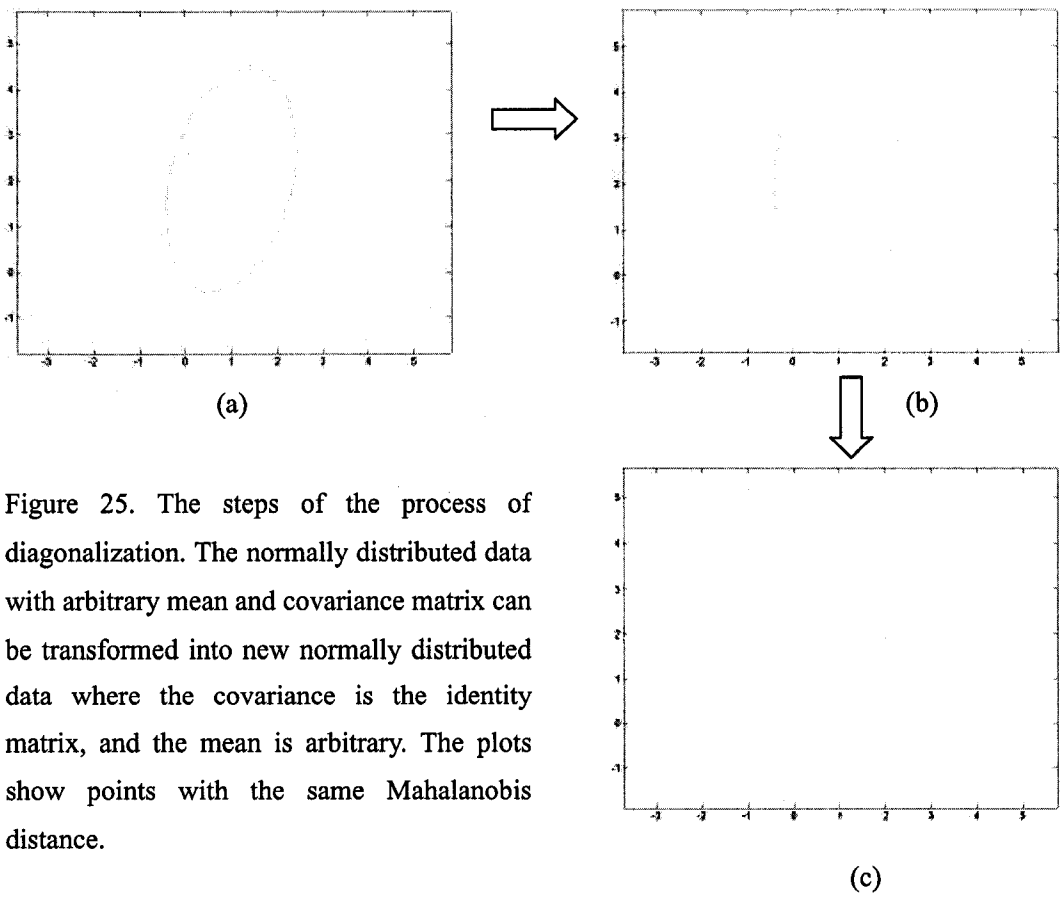


Figure 25. The steps of the process of diagonalization. The normally distributed data with arbitrary mean and covariance matrix can be transformed into new normally distributed data where the covariance is the identity matrix, and the mean is arbitrary. The plots show points with the same Mahalanobis distance.

identity matrix. $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the eigenvalues of σ, $\lambda_1,\ldots\lambda_d$. The covariance matrix of z in the transformed space is the identity matrix. Figure 25(a) shows the points with the same Mahalanobis distance in the original distribution having arbitrary mean and covariance. Figure 25(b) shows the points with the same Mahalanobis distance in a new distribution in which the data are uncorrelated. Figure 25(c) shows the points with the same Mahalanobis distance in the new distribution, whose covariance is the identity covariance matrix. In our segmentation approach, we use two-dimensional random vectors, a particular case of the general scenario discussed above. The points that we show in the plots have the same Mahalanobis distance and a property in common: they all have the same probability.

4.2 Adaptive Ellipse Method

As shown in the previous subsection, diagonalization has an appealing propriety that can transform the normally distributed data with arbitrary mean and covariance matrix into a new distribution where the covariance is the identity matrix, and the mean vector is arbitrary. This, in certain cases, simplifies the analysis of the data. We noticed that the data points with the same Mahalanobis distance in the original distribution lie on an ellipse, while the data points with the same Mahalanobis distance in the transformed distribution lie on a circle. The model that considers spots to be ellipses is discussed in detail below.

In microarray image segmentation, the intensities of the pixels in the spot area can be regarded as a two-dimensional normally distributed random vector. The 3-D shape determined by the intensities is considered as a histogram, which is, in turn, used to estimate the parameters of the two-dimensional normally distributed random vector. The

probability density function (pdf) of the two-dimensional normally distributed random vector is approximated using that histogram. After applying the diagonalization process to the underlying random vector, the data points with the same Mahalanobis distance, which lie on an ellipse in the original space, will lie on a circle in the new space. After this transformation has taken place, the aim is to exploit the properties of the circular shape of the spot to simplify the subsequent analysis. Note that the traditional circle-based segmentation methods can not be applied to the transformed pixels, because the pixel coordinates are given in terms of real numbers in the transformed space. Figure 26 shows a sample spot that takes the shape of an ellipse in the original space. After the diagonalization process is applied, the coordinates of the pixels are real numbers, as can be observed in the plot on the right hand side of the figure. For example, the pixel whose coordinates in the original space is [1,1] is represented as [-0.4139, 0.4934] in the transformed space.

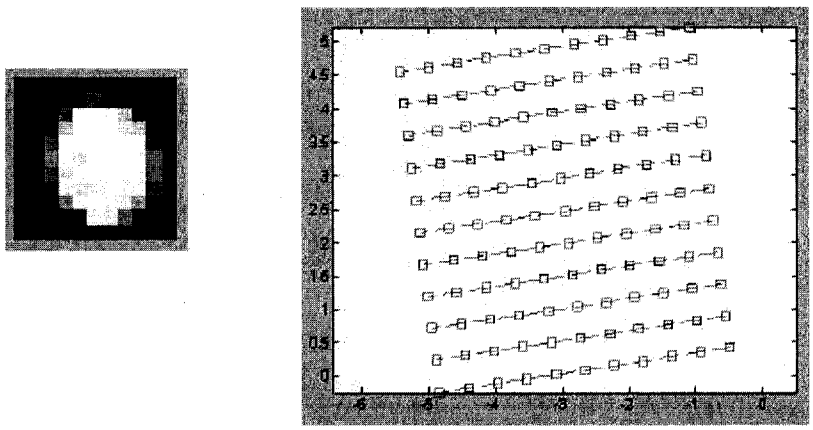


Figure 26. Change of coordinates after diagonalization. The left image shows a sample spot (Spot No.12 of 1230c1G/R microarray image) that takes the shape of an ellipse, instead of a circle. The plot on the right hand side shows the coordinates of each pixel in the new distribution after applying the diagonalization process.

The adaptive ellipse method involves three steps. First, the parameters of the dataset are computed using the maximum likelihood estimation. The mean and the covariance

matrix are then computed, followed by the diagonalization transformation. The coordinates of the pixels in the transformed space are computed using the eigenvalues and eigenvectors of the covariance matrix. Finally, after the dataset is transformed into the new distribution, the next task of the adaptive ellipse method is to compute the radius that defines the foreground region. For this purpose, different approaches can be applied. For example, we can compute the slope of the pdf for each pixel, and then find the radius that generates the largest slope average. In this thesis, we use another approach instead, and leave the former for future research.

4.2.1 Parameters Estimation

We consider the dataset $D = \{\mathbf{x}_{ij} \mid i = 1...m, j = 1...n\}$, where $\mathbf{x}_{ij}$ is the coordinate of a pixel. We assume that the dataset D conforms to normal distribution $x \sim N(\mu, \Sigma)$. μ is computed using the following equation:

$$\mu = \frac{1}{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}} \sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij} x_{ij} \qquad (4.1)$$

The covariance matrix Σ is computed using:

$$S = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(x_{ij} - \mu)(x_{ij} - \mu)^t}{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}} \qquad (4.2)$$

4.2.2 Diagonalization

After the two parameters, μ and Σ, of dataset D are computed, we apply diagonalization by seeking the eigenvalues, Λ, and eigenvectors, Φ, of Σ. The coordinates

for each point in the transformed space is calculated as follows:

$$y_{ij} = \Lambda^{-1/2} \Phi^t x_{ij} \tag{4.3}$$

This result can be verified by calculating μ' and Σ' in the new distribution: the mean obtained using the transformed points is the same as the mean obtained using the points in the original distribution. And Σ' is the identity matrix. After the diagonalization process, the dataset is transformed into a new space, where the data points which have the same Mahalanobis distance in the original space, lie on a circle.

4.2.3 Computing the Radius

In the adaptive ellipse method, we adopt a statistical method to compute the radius of the foreground region. First, we use Mann-Whitney test to estimate a threshold. A more detailed discussion of Mann-Whitney test can be found in [Dou-90]. Pixels from the predefined positions, i.e. the 4 corners and 4 middle-points in the edges in a gridded spot, are considered to be $Y_1, Y_2, \ldots, Y_8$. The pixels from the other region of the spot are sorted and the lowest 8 pixels are chosen as $X_1, X_2, \ldots, X_8$. We need a parameter α to compute the rank-sum statistic, namely W. If the null hypothesis is not rejected, we discard the pixel with the lowest intensity from the target set, and choose the next lowest intensity pixel from the remaining pixels. The process is repeated until the null hypothesis is rejected. The lowest intensity of the 8 pixels is the threshold that we are looking for. The pixels whose intensity is above the threshold are considered to be foreground pixels.

In the next stage, we sort all the pixels by their distance to the spot center μ in an

increasing order. Starting from the smallest distance pixel, we count the number of foreground pixels and background pixels for the next $2n+1$ pixels. The foreground and background pixels are grouped according to the threshold obtained in the Mann-Whitney test. The algorithm stops when the majority in the testing set is a background pixel. Otherwise, we move the starting pixel to the next one in the sorted pixels and use the next $2n+1$ pixels. The average distance of the $2n+1$ pixels is the radius that defines the foreground region. All the pixels whose distance to the spot center μ is smaller than the radius are marked as foreground; otherwise they are assigned to the background. The complete process is shown in Algorithm **Adaptive_Ellipse**. In the algorithm, we set the size of the testing set to 3 pixels.

Algorithm 6 Adaptive_Ellipse

Input: The spot image, **P**; the size of the image, m, n.
Output: A labeled matrix, *label*
Method:

```
Compute μ and Σ using equations (4.1) and (4.2)
Compute eigenvalues Λ and eigenvectors Φ of matrix Σ.
for i ←1 to n      //compute the new coordinates, y
 for j ← 1 to m
      y_ij =Λ^1/2 * transpose(Φ) * x_ij
 ENDFOR
endfor
threhd = MWT(P, α)  // use Mann-Whitney test to find the threshold
for i ← 1 to n       // compute the distance from the pixels to the spot center
 for j ← 1 to m
   radius[i, j] ← distance from y_ij to μ
 endfor
endfor
sortedr ← sort(reshape(radius))
```

```
idxr ← index of sort(radius)
for i ←1 to n*m
 countOn ← 0
 countOff ← 0
 for j ← i to i+2
  if P[idxr[j]] > threhd
    countOn ← countOn +1
   else
    countOff ← countOff +1
   endif
 endfor
  if countOn < countOff
   radiusFg ← sortedr[i]
  endif
  i ← i+1
endfor
for i ←1 to n
 for j ←1 to m
   if radius[i, j] < radiusFg
     label[i, j] ← 1
    else
     label[i, j] ←0
    ENDIF
 endfor
endfor
```

Procedure MWT(dataset , **D**; the significance level, α) **return** threshold, v

Begin

w ← rank sum value corresponding to α

set_y ← 8 pixels of the corner and the middle point of the edges //the background set

set_x ← 8 lowest intensity pixels from the remaining region // the foreground set

wmt ←MAX_VALUE

```
while wmt < w
  merge the two sets, and sort the resulting set in increasing order
  wmt ← the summation of the index of the foreground set
  discard the lowest intensity pixel from set_x
  add the lowest intensity pixel from the remaining pixels
endwhile
v ←min(set_x)
END
return v
END ALGORITHM ADAPTIVE_ELLIPSE
```

4.3 Experimental Results

In order to evaluate our adaptive ellipse method, we performed some simulations on real-life microarray images obtained from the ApoA1 data [Apo-04], and compared the results with the widely used adaptive circle method. In our experiments, the significance level α is set to 0.01. This value has been formed to give good result in most of our experiments.

4.3.1 Experiments Based on Visual Observation

Figure 27 shows the different results of the adaptive ellipse method and the adaptive circle method for some spots obtained from the 1230c1G/R microarray image. For those ellipse-shaped spots, we observe that the adaptive ellipse method generates a foreground region that is closer to the original spot in both shape and size than adaptive circle method. Consider spot No. 49, for example. The foreground region generated by the adaptive ellipse method is an ellipse in shape, while the foreground region generated by the adaptive circle

looks like a circle. The same situation occurs as in spot No. 65, but in this case, the axes of the resulting ellipse are not coincident with the coordinates of the system. Full images resulting from the adaptive ellipse method can be found in Appendix A.

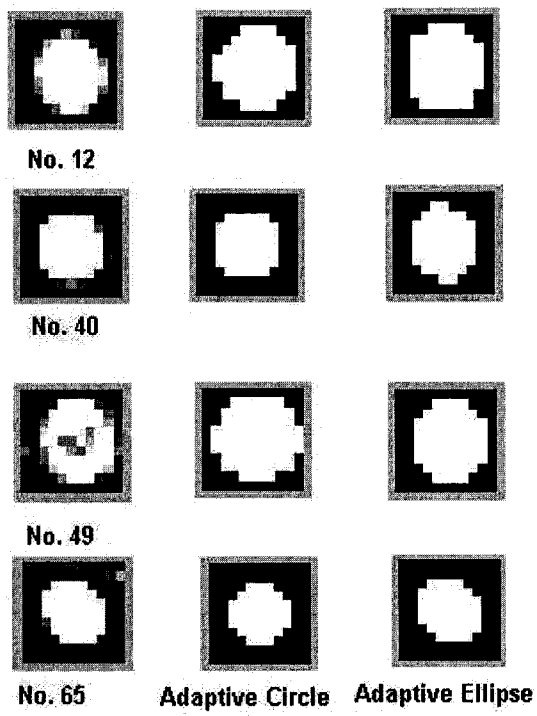


Figure 27. Different results obtained from the adaptive circle and adaptive ellipse methods for ellipse-shaped spots taken from 1230c1G/R microarray image.

Figure 28 shows the comparison of the two methods for some spots whose shape is similar to a circle. Because circle is a particular case of ellipse, the adaptive ellipse method also works well for circular spots. We observe that these two methods generate almost identical results for these spots.

Based on the above observations, we conclude that the adaptive ellipse method generates better results when dealing with spots that are ellipses in shape. Meanwhile, it generates results as good as the adaptive circle method when dealing with circular spots. In

general, the adaptive ellipse method is suitable for a wider range of spots, and generates better results. This argument could be shown theoretically, and is observed in the experiments below. The former constitutes an open problem, and proposes a future avenue for research.

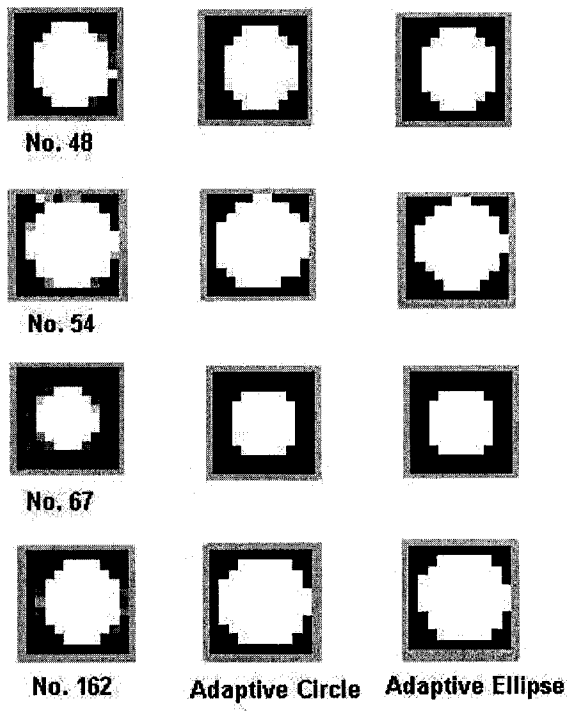


Figure 28. Adaptive circle and adaptive ellipse methods showing almost identical results for circular spots taken from 1230c1G/R microarray image

4.3.2 Experiments Based on Numerical Comparison

After illustrating that adaptive ellipse method generates better results than adaptive circle method using some sample spots, we now provide a numerical comparison for a batch of real-life microarray images drawn from the Apo A1 experiments. The measurement that we adopt for the comparison is the intensity of the foreground region for each spot, and the number of pixels belonging to that region. The results are shown in Table

6. The first column of each method contains the number of pixels in the foreground region, N_{fg}, and the second column represents the total average foreground intensity of the green channel, I_{fg}. In the first two columns, we notice that the average foreground intensity generated by the adaptive ellipse is higher than adaptive circle in general. Meanwhile the number of pixels generated by the former is approximately in the same range as the latter, but slightly larger. This can be easily justified by fact that adaptive ellipse finds a foreground region that represents the spot foreground better, which means that, it includes more foreground pixels and less background pixels than the adaptive circle method. Thus, it results in higher foreground intensity even though it contains more pixels in general. In our experiments, 7 out of the 9 images result in higher foreground intensity.

	Adaptive Circle Method			Adaptive Ellipse Method		
	N_{fg}	I_{fg}	I_{bg}	N_{fg}	I_{fg}	I_{bg}
1230c1G	28.24	3,652	846	28.34	3,670	843
1230c2G	25.36	4,301	1,120	25.80	4,314	1,123
1230c3G	28.17	4,178	595	28.29	4,181	592
1230c4G	24.37	3,248	818	24.57	3,235	816
1230c5G	26.88	2,914	459	26.91	2,961	459
1230ko1G	33.86	2,018	396	33.81	2,022	387
1230ko2G	20.69	2,353	532	20.54	2,373	531
1230ko3G	29.05	2,884	577	28.85	2,902	576
1230ko4G	24.56	2,735	564	24.64	2,729	563
Average	26.80	3,143	656	26.86	3,154	655

Table 6. Comparison of Adaptive Circle method and adaptive ellipse method for a batch of images from the ApoA1 dataset, where the first sub-grid of each image is analyzed.

In order to enhance the quality of our assessment about the experiments, we compare the number of "*hits*", i.e. the number of the pixels that are incorrectly labeled by the two algorithms. Because there are no standard solutions for microarray image segmentation and

classifying the pixels manually is still subjective and error-prone, we choose a histogram-based algorithm as the reference method to classify the pixels into foreground and background. Then, we apply the two methods to the same spots, and count the number of hits. The spots are obtained randomly from the Apo A1 image. Table 7 shows the results. We observe that, in most of the cases, the adaptive ellipse method generates fewer hits, which implies that it generates a foreground region that is more similar to that of the histogram-based approach.

Filename	Spot Number	Hits (adaptive ellipse)	Hits (adaptive circle)
1230c1	12	25	29
	24	8	8
	36	7	7
1230c2	12	47	48
	24	21	22
	36	24	24
1230c3	12	9	12
	24	14	12
	36	9	11
1230c4	12	36	36
	24	10	13
	36	16	17
1230c5	12	18	21
	24	15	17
	36	23	23
1230ko1	12	14	11
	24	18	18
	36	9	9
1230ko2	12	29	29
	24	41	41
	36	19	19
1230ko3	12	15	15
	24	18	17
	36	15	15
1230ko4	12	17	17
	24	16	16
	36	25	25
Total		518	532

Table 7. Comparison of the adaptive ellipse method and the adaptive circle method with a histogram-based approach. In most of the cases, the adaptive ellipse method shows more similar results than the adaptive circle method.

4.4 Conclusion

In this chapter, we introduce a new microarray image segmentation method, which we call the *adaptive ellipse method.* The advantage of this method is that it results in a foreground region that better represents the actual spots, and can be used for a wider range of microarray images than the traditional adaptive circle method. We view each spot in the microarray image from another perspective: the intensity represents the pdf of a normal distribution. Doing this enable us to extract statistical information from the images that is ignored by the traditional adaptive circle method, and hence showing more flexibility.

The method consists of three steps. First, we assume that the dataset is normally distributed. The two parameters, the center mean and the covariance matrix, are computed using the *maximum likelihood estimation.* Then, the diagonalization transformation is performed. The coordinates of the pixels in the transformed space are calculated using the eigenvalue and eigenvector of the covariance matrix. Finally, after the dataset is transformed into the new distribution, we use a statistical approach to estimate the threshold and find the radius of the foreground region. The results show that the adaptive ellipse method can reveal the true shape of the spots, and works better than adaptive circle method.

The adaptive ellipse method, which generates quite satisfying results, still has room for improvements. After the dataset is transformed to the new distribution, various methods can be applied to obtain the radius that defines the foreground region. A possible approach is to compute the slope of the pdf for each pixel, and then find the radius that generates the

largest slope average. This problem constitutes a possible avenue for future research. More work can also be done in more elaborated experiments to seek for better parameters of the present approach in finding the foreground radius.

CHAPTER 5 CONCLUSIONS AND FUTURE WORK

Microarray image segmentation is an important step in the process of microarray analysis. It extracts numerical information from the image files, thus, provides fundamental data for the subsequent analysis. Various methods exist in the literature, including the fixed circle method, the adaptive circle method, the seeded region growing, histogram approaches, and a recently proposed clustering-based algorithm. In this thesis, we provide a literature review and comparison on microarray image segmentation techniques. Then based on an in-depth analysis of clustering-based microarray image segmentation technique, we propose a novel algorithm, called *optimized k-means for microarray image segmentation* (OKMIS). In a subsequent chapter, we propose another shape-based algorithm, called adaptive ellipse method.

5.1 Contributions

As we investigated in Chapter 3, clustering algorithms can be used efficiently in microarray image segmentation, providing we choose the correct features and parameters. OKMIS is a segmentation algorithm that uses *k*-means, the feature space contains two features: one being the sum of the square root of the intensities, and the other the distance from the pixel coordinates to the spot center.

From the experiments depicted in the chapter, OKMIS is superior to previous clustering-based microarray image segmentation method. First, the resulting foreground regions after applying OKMIS are closer to the actual spots. Second, in some cases,

OKMIS reveals the true foreground region while SKMIS finds only the noisy pixels. More importantly, the resulting foreground obtained after applying OKMIS method contains less noisy pixels, and in most of the cases, it does not require an extra post-processing stage for removing the noise.

The adaptive ellipse method views the image data as the probability density function (pdf) of a dataset. By assuming that the dataset conforms to normal distribution, we apply diagonalization to transform the dataset from an arbitrarily mean and covariance matrix to a new space where the covariance matrix of the resulting random vector is the identity matrix. After the transformation, the data points of the same Mahalanobis distance lie on a circle. Then, we exploit the simple property of a circle to find the radius that defines the foreground region.

The adaptive ellipse method is compared to the state-of-art adaptive circle method. In our experiments, the adaptive ellipse method performs better than the adaptive circle method. First, the adaptive ellipse method is suitable for a wider range of microarray images than adaptive circle method. It deals with not only circular spots, but also ellipse-shaped spots. Second, the adaptive ellipse method generates more accurate results than the adaptive circle method. It generates better results than the adaptive circle method for ellipse-shaped spots, while being comparable to the adaptive circle method when dealing with circular spots. Based on our numerical analysis, we conclude that the adaptive ellipse method results in a foreground region that is closer to the actual spots.

5.2 Future Work

In the microarray image segmentation, it is not enough just grouping the pixels into foreground and background. In some cases, the noisy pixels may also need to be identified. We suggest that techniques such as Adaline, or SVMs, can be used to detect noise. In the process of background correction, an adaptive formula could be used to reveal the true foreground intensity instead of using the rigid formula described in Equation (1.1). Neural networks could also be applied to obtain the true foreground intensity. When conducting the gene expression data analysis, although hierarchical clustering is a widely used method, it suffers from drawbacks such as dealing with noise and providing a non-unique solution.

The OKMIS algorithm, which generates quite satisfying results, still has room for improvements. More elaborated feature extraction and normalization schemes can improve the accuracy of a clustering algorithm. When considering more than one feature, the normalization process has a high influence on the results of clustering. Normalization includes not only scaling, but also considering the correlation between each pair of features. In this regard, principle component analysis (PCA) is a widely used method that could be used to produce even better results. This problem also constitutes a possible avenue for future research.

When the number of clusters is more than two, an extra step is needed to determine which clusters correspond to foreground and which ones belong to background. Although this is not an easy task, the refined classification may lead to more interesting results. An automated clustering algorithm is desired to evaluate the best number of clusters for each

spot and classify the pixels into foreground and background.

The adaptive ellipse method, which generates quite accurate results, still in its infancy, and much work can be done for improvements. Our future work includes theoretically prove that the adaptive ellipse method is better than adaptive circle method. After the dataset is transformed into a new distribution, various methods can be applied to obtain the radius that defines the foreground region. A possible approach is to compute the slope of the pdf for each pixel, and then find the radius that generates the largest slope average. This problem constitutes a possible avenue for future research as well. More work can also be done in more exhaustive experiments to seek for better parameters for the present approach.

Another possible future work constitutes of a numerical comparison that includes incorporating white noise to the images, and subsequently tests the adaptive ellipse method, and compares it with the adaptive circle method. This approach is typically used in signal processing.

References

[2Ca-04] 2Can Bioinformatics Educational Resource. http://www.ebi.ac.uk/2can/bioinformatics/

[Ada-94] R. Adams and L. Bishop. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641-647, 1994.

[Alo-99] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745--6750, 1999.

[Ans-01] Quick napkin calculations. W. Ansorge and J. Quackenbush in Schnookeloch in Heidelberg on 4 October 2001.

[Apo-04] Terry Speed's Microarray Data Analysis Group Page.
http://www.stat.berkeley.edu/users/terry/zarray/Html/apodata.html

[Asa-96] T. Asano, D. Chen, N. Katoh, T. Tokuyama. Polynomial-time solutions to image segmentation, *Proc. of the 7th Ann. SIAM-ACM Conference on Discrete Algorithms*, 104-113, 1996.

[Axo-99] Axon Instruments, Inc. GenePix 4000A User's manual. 1999.

[Bez-84] J.C. Bezdek,, R. Ehrlich, and W. Full. The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10, 191-203. FCM, 1984.

[Bic-01] S. Bicciato, M. Pandin, G. Didone, and C. Di Bello. Analysis of an associative memory neural network for pattern identification in gene expression data. *In Proceedings of BIOKDD'01*, 2001.

[Bit-00] M. Bittner, X, P. Meltzer, X, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dork, N. Sampas, E. Dougherty, E. Wang , F. Marincola , C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, V. Sondak, N. Hayward & J. Trent. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406, 536-540, 2000.

[Bro-99] P. Brown, D. Botstein. Exploring the new world of the genome with DNA microarrays. *Nat Genet* Jan; 21(1 Suppl):33-7, 1999.

[Buc-00] M. Buckly. The Spot user's guide. CSIRO Mathematical and Information Sciences. 2000.

[Buh-00] J. Buhler, T. Ideker, and D. Haynor. Dapple: Improved Techniques for Finding Spots on DNA Microarrays. *Technical Report UWTR 2000-08-05*, University of Washington, 2000.

[Cai-01] Y. Cai, X. Liu, X. Xu, and G. Zhou. Support Vector Machines for predicting protein structural class ,*BMC Bioinformatics* 2:3, 2001.

[Cam-02] R. Campanini, D. Dongiovanni, N. Lanconelli, G. Palermo, A. Riccardi. A Support Vector Machine Classifier based on Recursive Feature Elimination for Microarray Data in Breast

Cancer Characterization. 2002.

[Car-02] C. Carson, S. Belongie, H. Greenspan, J. Malik. Blobworld -- Image segmentation using expectationmaximization and its application to image querying. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(8): 1026--1038, 2002.

[CGA-04] The Cancer Genome Anatomy Project. http://cgap.nci.nih.gov/

[Che-97] Y. Chen, E. Dougherty, and M. Bittner. Ratio-based decisions and the quantitative analysis of cDNA microarray images. *Journal of Biomedical Optics*, 2:364-374,1997.

[Dai-04] Hypermedia Image Processing Reference. http://www.dai.ed.ac.uk/HIPR2/convolve.htm

[Din-01] C. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks, *Bioinformatics*, 17:349-358, 2001.

[Dou-90] E.R. Dougherty, Probability and statistics for the engineering, computing, and physical sciences, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[Dow-04] Lymphoma/Leukemia Molecular Profiling Project.
http://llmpp.nih.gov/lymphoma/data/rawdata/

[Dud-00] R. Duda, P. Hart, D. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, ISBN: 0471056693, 2000.

[Eis-99] M. Eisen. ScanAlyze User Manual. http://www.microarrays.org/software.html 1999.

[Fuc-04] Y.F. Leung's Functional Genomics website.
http://ihome.cuhk.edu.hk/~b400559/arraysoft_image.html

[Fur-00] T.furey, N.Cristianini, N. Duffy, D. Bednarski, M. Schummer and D. Haussler. Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expressioin Data, *Bioinformatics*, 2000.

[Gor-01] A. Goryachev, P. Macgregor, A. Edwards. Unfolding of microarray data. *Journal of computational biology*. Volume 8, number 4, 443-461, 2001.

[GSI-99] GSI Lumonics. QuantArray Analysis Software, Operator's Manual. 1999.
http://www.perkinelmer.com/

[Guy-02] I. Guyon, J. Weston, S. Barnhill and V. Vapnik. Gene Selection for Cancer Classification using Support Vector Machines, *Machine Learning* 46(1/3): 389-422, January 2002.

[HIP-04] Hypermedia Image Processing Reference. http://www.dai.ed.ac.uk/HIPR2/log.htm

[Ima-04] Biodiscovery Inc. http://www.biodiscovery.com

[Jaa-99] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In T. Lengauer et al., editors, *Proc. 7th Int. Conf. on Intelligent Syst. for Molecular Biology (ISMB-99)*, 1999.

[Jai-02] A. Jain, T. Tokuyasu, A. Snijders, R. Segraves, D. Albertson, and D. Pinkel. Fully

Automatic Quantification of Microarray Image Data. *Genome Res.*,12(2):325–332, 2002.

[Jen-02] T.K. Jenssen, M. Langaas, W.P. Kuo, B. Smith-Sørensen, O. Myklebost, and E. Hovig. Analysis of repeatability in spotted cDNA microarrays. *Nucleic Acid Res,* 30:32353244, 2002.

[Kat-02] M. Katzer, F. Kummert, and G. Sagerer. Robust Automatic Microarray Image Analysis. In *Proceedings of the International Conference on Bioinformatics:North-South Networking*, Bangkok, 2002.

[Koo-00] C. Kooperberg, T. Fazzio, T. Tsukiyama. Improved background correction for spotted DNA microarrays, 2000.

[Mac-67] J. MacQueen, "Some methods for classification and analysis of multi-variate observations". *In: Proc. of the Fifth Berkeley Symp. on Math., Statistics and Probability*, pp 281-297, 1967.

[MAG-03] MicroArray Genome Imaging and Clustering Tool MAGIC Tool User Guide, Department of Biology, Davidson College.

[Mic-04] Microarray Image Analysis and Data Mining.
http://pearl.cs.pusan.ac.kr/~hyjung/pub/Microarray/

[Muk-99] S. Mukherjee, P. Tamayo, J. Mesirov, D. Slonim, A. Verri, and T. Poggio. Support vector machine classification of microarray data. *Technical Report* 182, AI Memo 1676, CBCL, 1999.

[Nih-04] National human genome research Institute. Image analysis.
http://research.nhgri.nih.gov/microarray/image_analysis.html

[NPL-04] National Physical Laboratory. http://www.npl.co.uk/biotech/validfluo.html

[Per-99] C. Perou, S. Jeffrey, M. van de Rijn, C. Rees, M. Eisen, D. Ross, A. Pergamenschikov, C. Williams, S. Zhu, J. Lee, D. Lashkari, D. Shalon, P. Brown and D. Botstein. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proc. N*atl Acad. Sci. USA, 96, 9212--9217 1999.

[Puz-99] J. Puzicha, J. Buhmann, T. Hofmann. Histogram Clustering for Unsupervised Image Segmentation, *Computer Vision and Pattern Recognition*-Volume 2, June 23 - 25, 1999.

[Rue-02] L. Rueda and B. J. Oommen, "On Optimal Pairwise Linear Classifiers for Normal Distributions: The Two-Dimensional Case". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 2, pp. 274-280, 2002.

[Sch-99] DNA microarrays : a practical approach. edited by Mark Schena.Oxford ; New York : Oxford University Press.ISBN 0-19-963777-6, 1999.

[Sch-00] J. Schuchhardt, D. Beule, A. Malik, E. Wolski, H. Eickho, H. Lehrach, and H. Herzel. Normalization strategies for cDNA microarrays. *Nucleic Acids Res*, 28:E47, 2000.

[Sch-01] B. Scholkopf, I. Guyon, and J. Weston. Statistical learning and kernel methods in

bioinformatics. *In proceedings NATO Advanced Studies Inst. on Artificial Intelligence and Heuristics Methods for Bioinformatics.* San Miniato, Italy October 1-11, to appear. 2001.

[Sch-02] M. Schena. Microarray analysis. Published by John Wiley & Sons, Inc., isbm 0-47-41443-3, 2002.

[Soi-99] P. Soille. Morphological Image Analysis: Principles and Applications. Sprinter, 1999.

[Spe-98] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, B. Futcher. Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. *Molecular Biology of the Cell* 9, 3273-3297, 1998.

[Sta-04a] Stanford microarray database. http://genome-www5.stanford.edu/MicroArray/SMD/

[Sta-04b]Yeast cell cycle analysis project.
http://genome-www.stanford.edu/cellcycle/data/rawdata/individual.html

[Ste-01] M. Steinfath, W. Wruck, and H. Seidel. Automated image analysis for array hybridization experiments. *Bioinformatics -OXFORD-*, 2001, Vol. 17, T. 7,S. 634-641, 2001.

[Tam-96] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander and T. Golub. Interpreting patterns of gene expression with selforganizing maps: methods and application to hematopoietic differentiation. *Proc. Natl Acad. Sci. USA*, 2907-2912, 1996.

[Tib-99] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. *Technical report,* Department of Statistics, Stanford University, 1999.

[Vas-04]Multi microarray normalisation, Keith Vass and Ernst Wit.
http://genome1.beatson.gla.ac.uk/Rweb/anova.html

[Wen-98] X. Wen, S. Fuhrman, G. Michaels, D. Carr, S. Smith, J. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development, *Proc. Natl. Acad. Sci.*, USA, vol. 95, pp. 334-339, 1998.

[Wol-04] Fort Collins, Colorado
http://documents.wolfram.com/applications/digitalimage/UsersGuide/7.5.html,

[Wlt-04] K.Vass, pers. Comm.; Lou et al. Lake Tahoe Center. See The Use of Statistics in Microarray Studies, by Ernst Wlt. University of Glasgow.

[Wu-03] H. Wu and H. Yan. Microarray Image Processing Based on Clustering and Morphological Analysis. *Proc. of First Asia Pacific Bioinformatics Conference*, Adelaide, Australia, 111-118 2003.

[Yan-01] M. Yang, Q. Ruan, J. Yang, S. Eckenrode, S. Wu, R. McIndoe, and J. She. A statistical procedure for flagging weak spots greatly improves normalization and ratio estimates in microarray

experiments. *Physiol Genomics,* Aug 8, 2001.

[Yan-02] Y. Yang, M. Buckley, S. Dudoit, and T. Speed. Comparison of Methods for Image Analysis on cDNA Microarray Data. *Journal of Computational and Graphical Statistics*, 11:108--136, 2002.

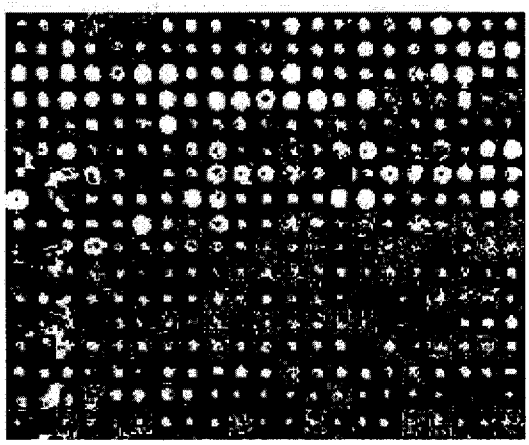
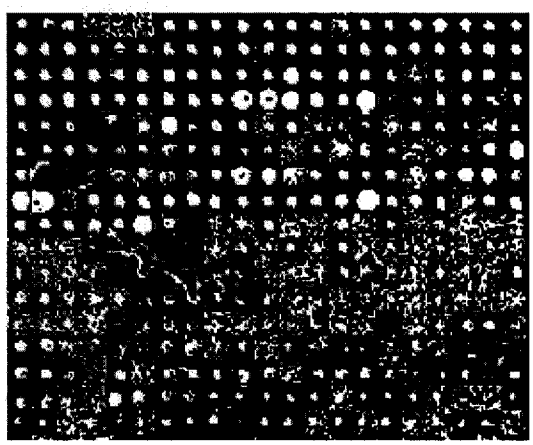
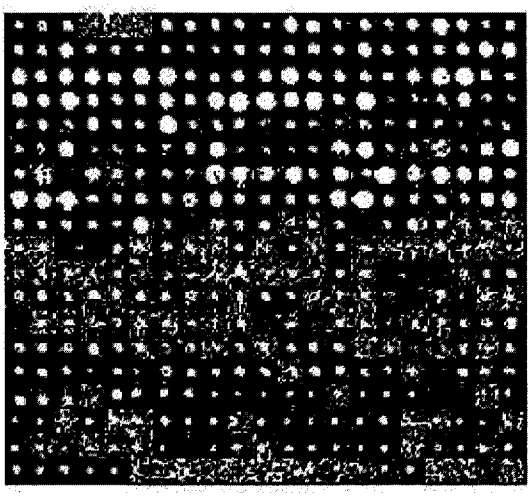
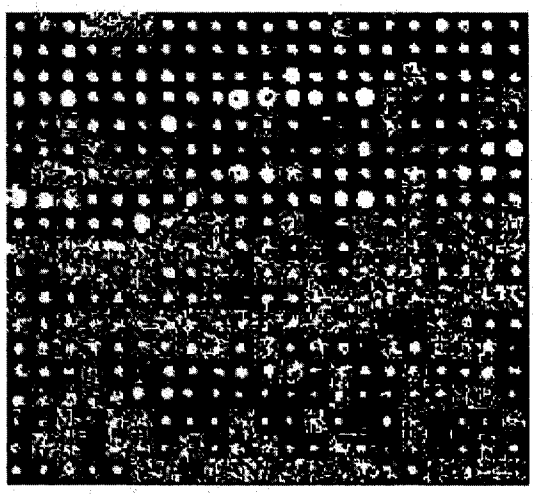
[Yue-01] H. Yue, P.S. Eastman, B.B. Wang, J. Minor, M.H. Doctolero, R.L. Nuttall, R. Stack, J.W. Becker, J.R. Montgomery, M. Vainer, and R. Johnston. An evaluation of the performance of cDNA microarrays for detecting changes in global mRNA expression. *Nucleic Acids Res*, 29:e41, 2001.

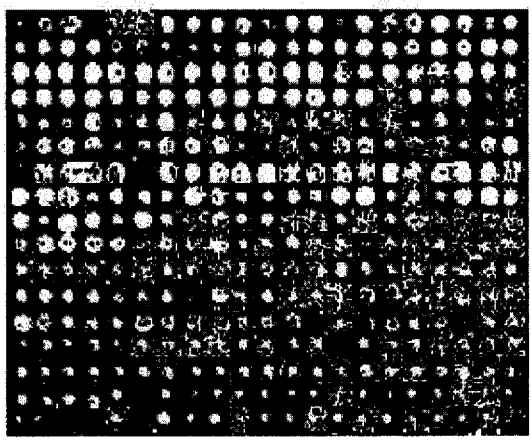
[Zie-00] A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K.-R. Muller. Engineering support vector machine kernels that recognize translation initiation sites, *Bioinformatics*, 16(9):799-807, 2000.

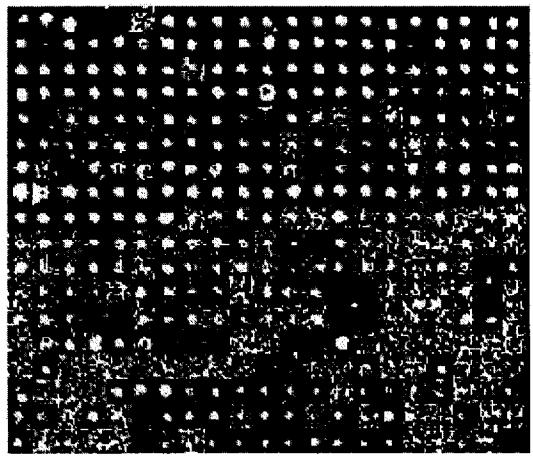
Appendix A Experimental Results

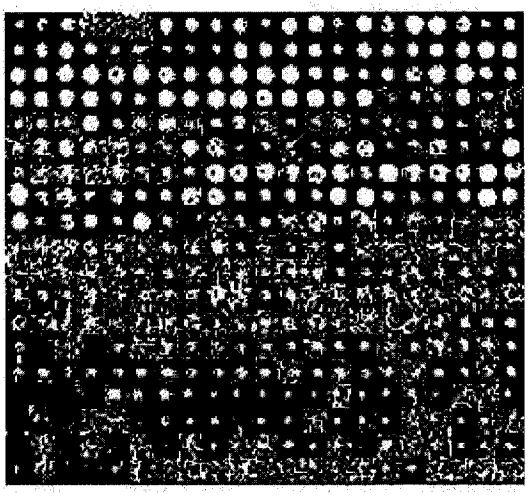
1. Results after applying SKMIS

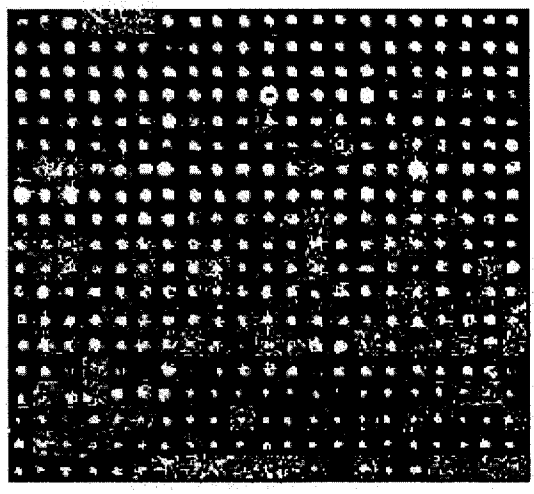
The images are 1230c1R/G, 1230c2R/G, 1230c3R/G, 1230c4R/G, 1230ko1R/G, 1230ko2R/G, 1230ko3R/G, 1230ko4R/G.



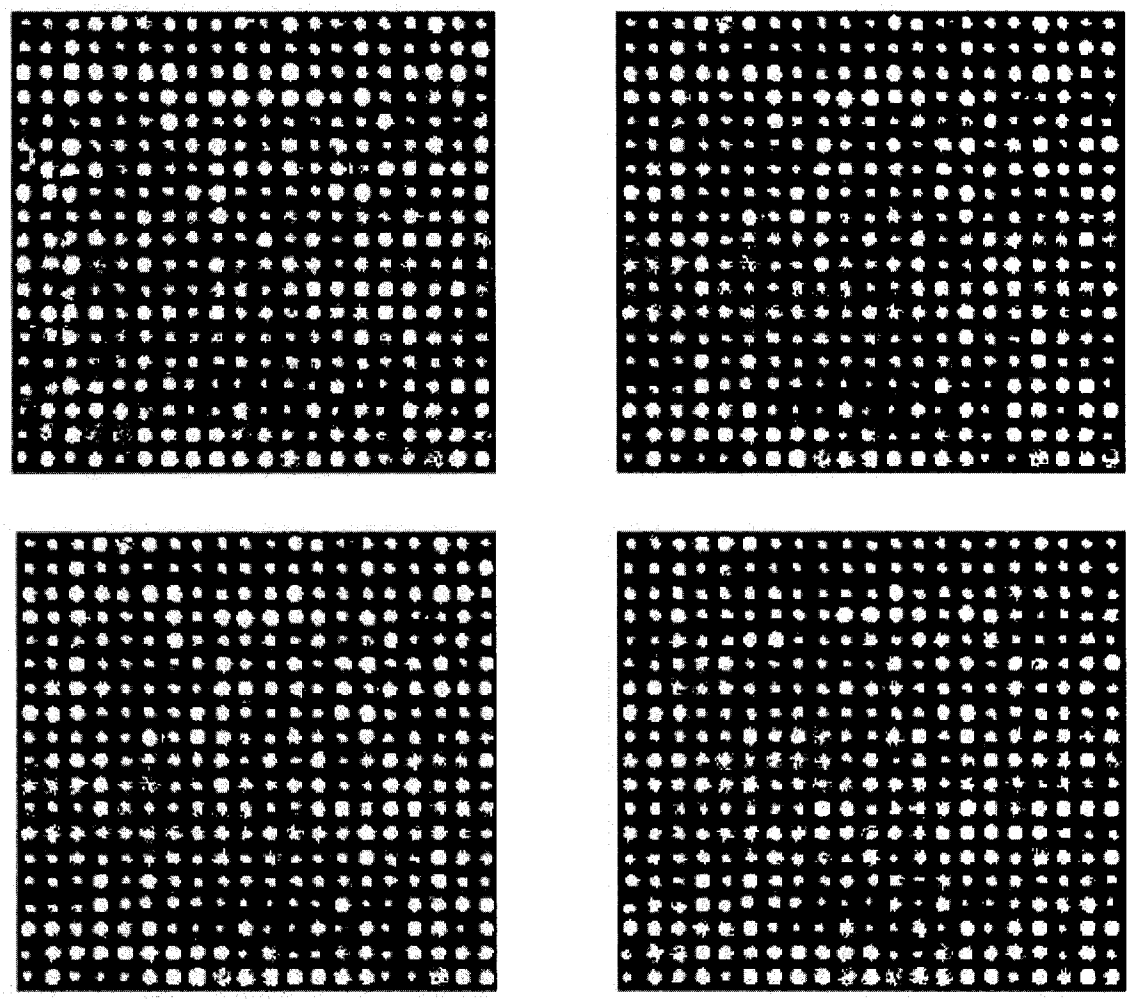


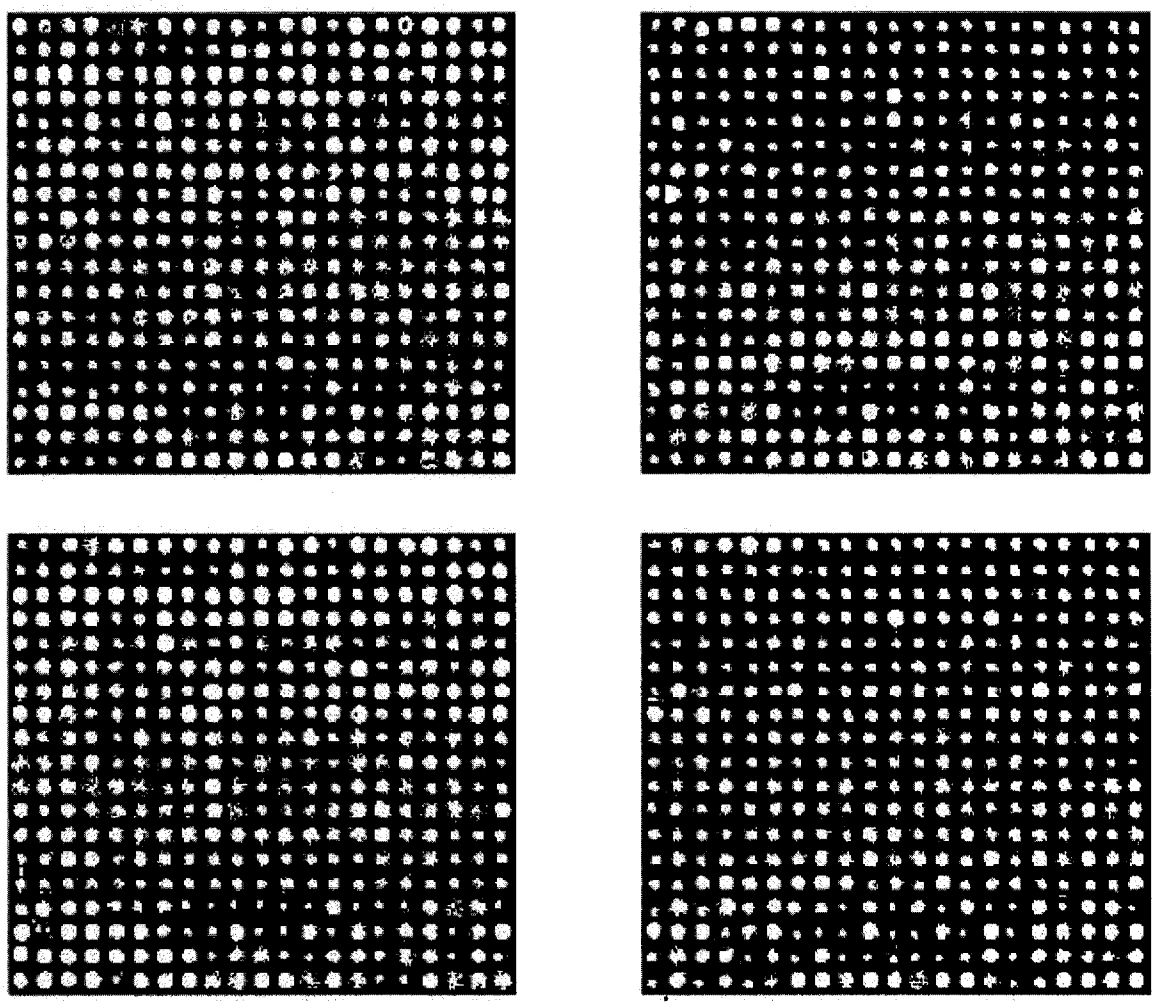




2. Results after applying OKMIS

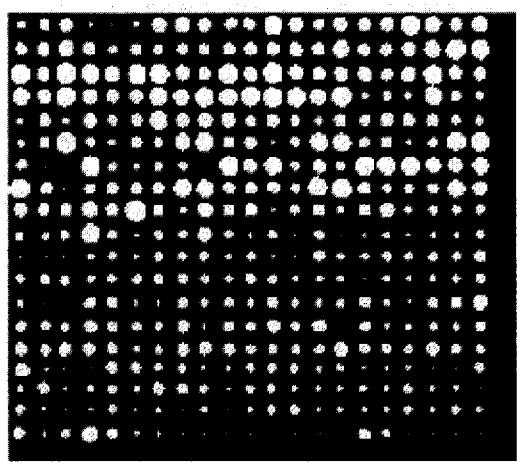
The images are 1230c1R/G, 1230c2R/G, 1230c3R/G, 1230c4R/G, 1230ko1R/G, 1230ko2R/G, 1230ko3R/G, 1230ko4R/G.

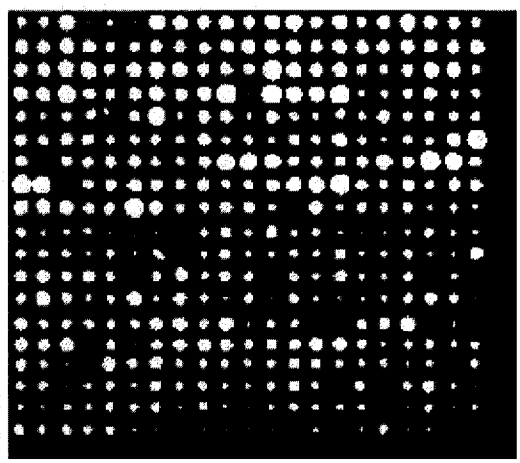


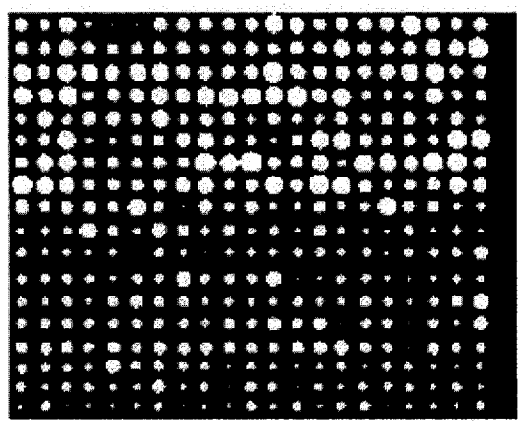


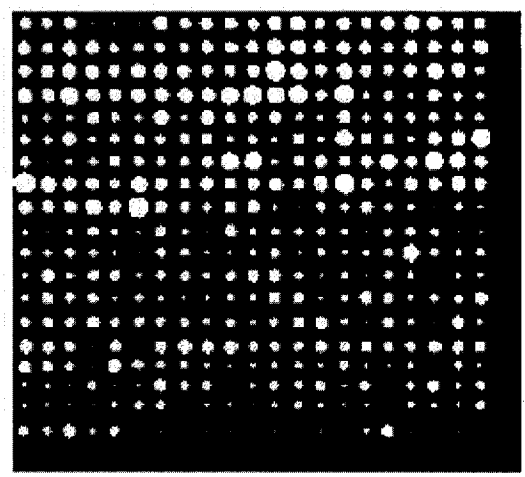
3. Results after applying the adaptive ellipse method

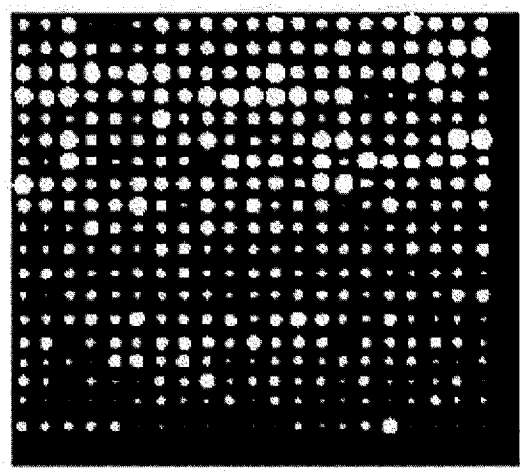
The images are 1230c1R/G, 1230c2R/G, 1230c3R/G, 1230c4R/G, 1230c5R/G, 1230ko1R/G, 1230ko2R/G, 1230ko3R/G, 1230ko4R/G.

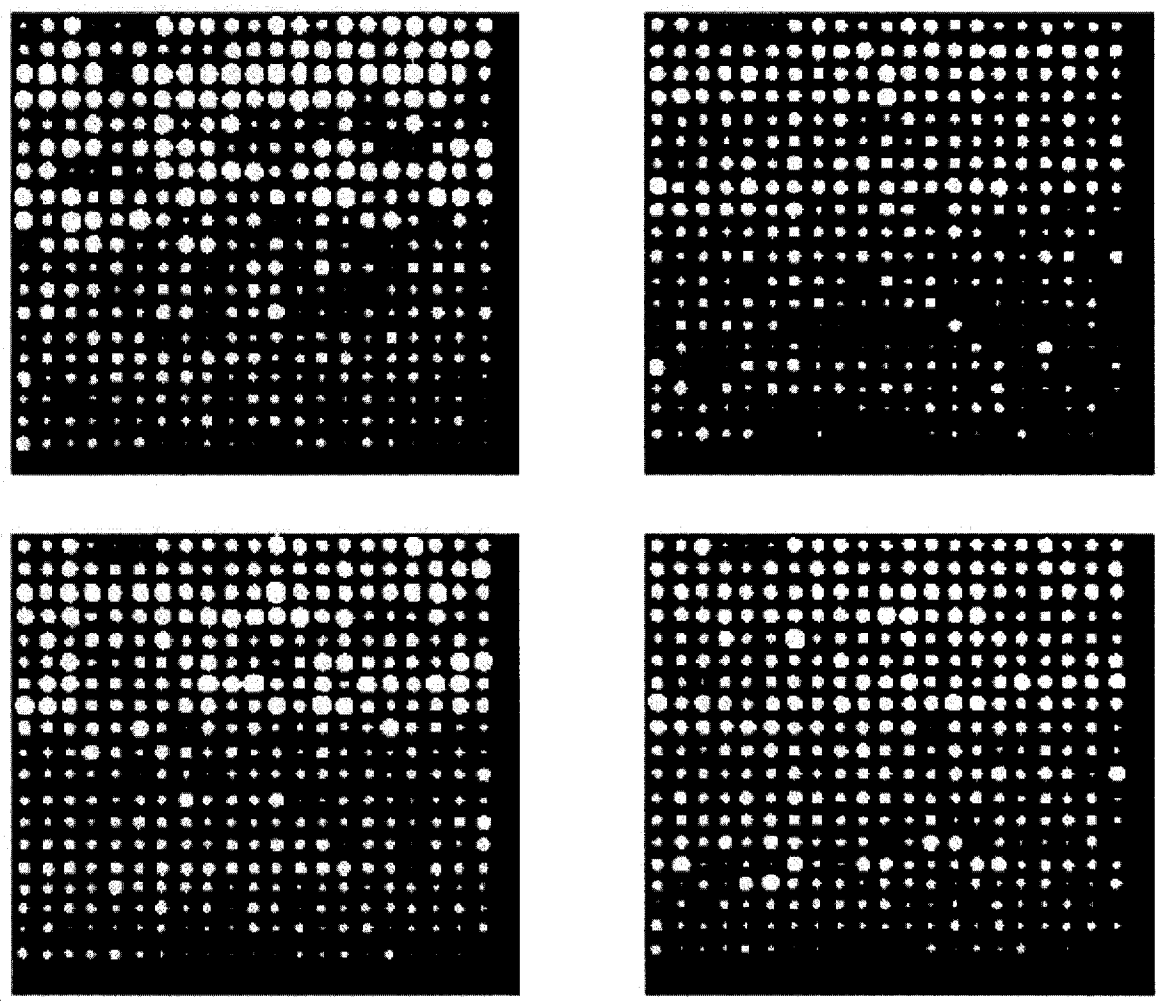












Appendix B. Source Code

1. MLE Algorithm Implementation

```
function label=omniMLE(ds,k);

  [featuresize,datasize]=size(ds);
  %guess an initial mu and set sigma=1
  [x,idx]=sort(ds(1,:));
  mu=zeros(featuresize,k);
  sigma=zeros([k,featuresize,featuresize]);

  % init method 1
  step=floor(datasize/k);
  for i=1:k
    for j=1:step
      mu(:,i)=mu(:,i)+ds(:,idx((i-1)*step+j));
    end
    mu(:,i)=mu(:,i)/step;
  end

  for i=1:k
    sig=zeros(featuresize,featuresize);
    for j=1:step
      fv=ds(:,idx((i-1)*step+j));
      sig=sig+(fv-mu(:,i))*(fv-mu(:,i))';
    end
    sigma(i,:,:)=sig/step;
  end


  changes=0;
  muold=zeros(featuresize,k);
  for i=1:k
    p(i)=1/k;
    changes=changes+abs(mu(1,i)-muold(1,i));
  end;

  rounds=1;
  while ( changes>1 )
    rounds
    rounds=rounds+1;
    muold=mu;

    for i=1:datasize
      sum=0;
```

```
    for j=1:k
      b(:,:)=sigma(j,:,:);
      s(j)=det(b)^(-1/2)*exp(-1/2*(ds(:,i)-mu(:,j))'*(b^(-1))*(ds(:,i)-mu(:,j)))*p(j);
      sum=sum+s(j);
    end
    for j=1:k
      pc(i,j)=s(j)/sum;
    end
  end

  for j=1:k
    p(j)=0;
    for i=1:datasize
      p(j)=p(j)+pc(i,j);
    end
    p(j)=p(j)/datasize;
  end

  for j=1:k
    s1=zeros(featuresize,1);s2=0;
    for i=1:datasize
      s1=s1+pc(i,j)*ds(:,i);
      s2=s2+pc(i,j);
    end
    mu(:,j)=(s1/s2);
  end

  s1=zeros(featuresize,featuresize);s3=0;
  for j=1:k
    b(:,:)=sigma(j,:,:);
    for i=1:datasize
      s1=s1+pc(i,j)*(ds(:,i)-mu(:,j))*(ds(:,i)-mu(:,j))';
      s3=s3+pc(i,j);
    end
    sigma(j,:,:)=s1/s3;
  end

  changes=0;
  for i=1:k
    changes=changes+abs(mu(1,i)-muold(1,i));
  end;
end

label=zeros(datasize,1);
for i=1:datasize
  [max,idx]=max(pc(i,:));
  label(i)=idx;
end
```

2. The *k*-Means Algorithm Implementation

```
function g=omnikMeans(ds, k);
% use ins to initial first, change to radius.

  ins=reshape(ds(1,:),121,1);
  rad=reshape(ds(2,:),121,1);
  [sorted,idx]=sort(rad);
  [n,datasize]=size(ds);

  %initialize
  Number=zeros(k,1);
  means=zeros(k,1);
  Label=zeros(datasize,1);
  Label=Label+1;

  chunk=floor(datasize/k);
  for c=1:k
    for i=1:chunk
      Label(idx(chunk*(c-1)+i))=c;
    end
  end

  for f=1:n
    for c=1:k
      submeans(f,c)=0;
    end
  end

  for i=1:datasize
    Number(Label(i))=Number(Label(i))+1;
    for f=1:n
      submeans(f,Label(i))=submeans(f,Label(i))+ds(f,i);
    end
  end
  for i=1:n
    for c=1:c
      submeans(i,c)=submeans(i,c)/Number(c);
    end
  end

  % loop
  bChange=999;
  while ( bChange>1 )
    bChange=0;
    for i=1:datasize
      for j=1:k
        distance(j)=0;
        for f=1:n
          distance(j)=distance(j)+(ds(f,i)-submeans(f,j))^2;
```

```
        end
        distance(j)=sqrt(distance(j));
      end
      [mini,new]=min(distance);
      if ( Label(i)~=new )
        old=Label(i);
        Label(i)=new;
        for f=1:n
          submeans(f,old)=(submeans(f,old)*Number(old)-ds(f,i))/(Number(old)-1);
          submeans(f,new)=(submeans(f,new)*Number(new)+ds(f,i))/(Number(new)+1);
        end
        Number(old)=Number(old)-1;
        Number(new)=Number(new)+1;
        bChange=bChange+1;
      end
    end
  end
  g=Label;
```

3. The Fuzzy *k*-Means Algorithm Implementation

```
function g=fkMeans(D, k);
  b=1.25;
  ins=reshape(D,121,1);
  [sorted,idx]=sort(ins);
  [datasize,x]=size(ins);

  %initialize
  Number=zeros(k,1);
  means=zeros(k,1);
  M=zeros(k,datasize);

  chunk=floor(datasize/k);
  for c=1:k
    for i=1:chunk
      M(c,idx(chunk*(c-1)+i))=1;
    end
  end

  for i=1:datasize
    for c=1:k
      if M(c,i)==1
        means(c)=means(c)+ins(i);
        Number(c)=Number(c)+1;
      end
    end
  end
```

```
for c=1:c
  means(c)=means(c)/Number(c);
end

% loop
bChange=999;
while ( bChange>0.2 )
  bChange=0;
  for j=1:datasize
    sum=0;
    for i=1:k
      d(i)=(ins(j)-means(i))^2;
      item(i)=(1/d(i))^(1/(b-1));
      sum=sum+item(i);
    end
    for i=1:k
      M(i,j)=item(i)/sum;
    end
  end

  for i=1:k
    sum1=0;
    sum2=0;
    for j=1:datasize
      sum1=sum1+M(i,j)^b*ins(j);
      sum2=sum2+M(i,j)^b;
    end
    newMean=sum1/sum2;
    bChange=bChange+abs(newMean-means(i));
    means(i)=sum1/sum2;
  end
end

%return label with the largest probability
for i=1:datasize
  for j=1:k
    b(j)=M(j,i);
  end
  [a,idx]=max(b);
  label(i)=idx;
end
g=label;
```

4. The SKMIS Implementation

```
clear all;
```

```
%the parameters of the image
grid_rows_in_array = 4;
grid_cols_in_array = 4;

rows_in_grid=19;
cols_in_grid=21;

spot_number=grid_rows_in_array*grid_cols_in_array*rows_in_grid*cols_in_grid;

width=11;
height=11;

half_w=(width-1)/2;
half_h=(height-1)/2;


%load the preprocessed spot center coordinates
load  D:\Lily\WindsorStudy\RA\apo-data\images\try2\ac.1.center;
%read in the image file
img=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1G.tif.marray','tif');
ima=double(img);
imr=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1R.tif.marray','tif');
imb=double(imr);
im1=ima+imb;
start_x=ac(1,2)-half_w;
start_y=ac(1,1)-half_h;
imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
imnew2=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
subgrid=im1(round(start_y):round(start_y)+size_imnew(1),round(start_x):round(start_x+size_im
new(2)));
[M,N]=size(subgrid);
edge=zeros(M,N);

%get the subgrid of the images
start_x=ac(1,2)-half_w;
start_y=ac(1,1)-half_h;
imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
suba=ima(round(start_y):round(start_y)+size_imnew(1)-
1,round(start_x):round(start_x+size_imnew(2)-1));
subb=imb(round(start_y):round(start_y)+size_imnew(1)-
1,round(start_x):round(start_x+size_imnew(2)-1));

countfg=zeros(19*21,1);
countbg=zeros(19*21,1);
sumfgg=zeros(19*21,1);
sumbgg=zeros(19*21,1);
sumfgr=zeros(19*21,1);
sumbgr=zeros(19*21,1);
```

```
for spot=1:21*19
  spot
  %get the spot grid
  spa2=suba(round(ac(spot,1)-half_w-start_y+1):...
       round(ac(spot,1)+half_w-start_y+1),...
       round(ac(spot,2)-half_h-start_x+1):...
       round(ac(spot,2)+half_h-start_x+1));
  spb2=subb(round(ac(spot,1)-half_w-start_y+1):...
       round(ac(spot,1)+half_w-start_y+1),...
       round(ac(spot,2)-half_h-start_x+1):...
       round(ac(spot,2)+half_h-start_x+1));

  %square root of the intensity
  spa=sqrt(spa2);
  spb=sqrt(spb2);

  %maximum value of the two channels
  for i=1:11
    for j=1:11
      ims(i,j)=max(spa(i,j),spb(i,j));
    end
  end
  ins=reshape(ims,121,1);
  [sorted,idx]=sort(ins);

  %initialize
  NumberFore=0;
  NumberBack=0;
  MinPixel=sorted(1);
  MaxPixel=sorted(121);
  MeanFore=0;
  MeanBack=0;
  for i=1:121
    if abs(ins(i)-MinPixel)>abs(ins(i)-MaxPixel)
      label(i)='f';
      NumberFore=NumberFore+1;
      MeanFore=MeanFore+ins(i);
    else
      label(i)='b';
      NumberBack=NumberBack+1;
      MeanBack=MeanBack+ins(i);
    end
  end
  MeanFore=MeanFore/NumberFore;
  MeanBack=MeanBack/NumberBack;

  %loop
  bChange=999;
  while ( bChange>4 )
    bChange=0;
```

```
    for i=1:121
      if ( label(i)=='b' & NumberBack * abs(ins(i) - MeanBack) / (NumberBack-1) > NumberFore
* abs(ins(i) - MeanFore) / (NumberFore+1) )
        label(i)='f';
        MeanFore=(MeanFore*NumberFore+ins(i))/(NumberFore+1);
        MeanBack=(MeanBack*NumberBack-ins(i))/(NumberBack-1);
        NumberFore=NumberFore+1;
        NumberBack=NumberBack-1;
        bChange=bChange+1;
      elseif ( label(i)=='f' & NumberFore * abs(ins(i) - MeanFore) / (NumberFore-1) >
NumberBack * abs(ins(i) - MeanBack) / (NumberBack+1))
        label(i)='b';
        MeanFore=(MeanFore*NumberFore-ins(i))/(NumberFore-1);
        MeanBack=(MeanBack*NumberBack+ins(i))/(NumberBack+1);
        NumberFore=NumberFore-1;
        NumberBack=NumberBack+1;
        bChange=bChange+1;
      end
    end
  end

  row=ceil(spot/cols_in_grid);
  col=rem(spot,cols_in_grid);
  if col==0
    col=cols_in_grid;
  end

  %show the binary labeled image
  for i=1:11
    for j=1:11
      if ( label((j-1)*11+i)=='f' )
        im3(i,j)=255;
      else
        im3(i,j)=0;
      end
    end
  end

  %find the largest continuouse region, set the area as foreground
  %im4=findArea(im3);
  im4=im3;
  for i=1:11
    for j=1:11
      if (im4(i,j)==255)
        edge((row-1)*11+i,(col-1)*11+j)=255;
        countfg(spot)=countfg(spot)+1;
        sumfgg(spot)=sumfgg(spot)+spa2(i,j);
        sumfgr(spot)=sumfgr(spot)+spb2(i,j);
      else
        edge((row-1)*11+i,(col-1)*11+j)=0;
        sumbgg(spot)=sumbgg(spot)+spa2(i,j);
```

```
            sumbgr(spot)=sumbgr(spot)+spb2(i,j);
          end
        end
      end

      countfg(spot);
      if countfg(spot)==0
        meanfgg(spot)=0;
        meanfgr(spot)=0;
        meanbgg(spot)=sumbgg(spot)/121;
        meanbgr(spot)=sumbgr(spot)/121;
      elseif countfg(spot)==121
        meanfgg(spot)=sumfgg(spot)/countfg(spot);
        meanfgr(spot)=sumfgr(spot)/countfg(spot);
        meanbgg(spot)=0;
        meanbgr(spot)=0;
      else
        meanfgg(spot)=sumfgg(spot)/countfg(spot);
        meanfgr(spot)=sumfgr(spot)/countfg(spot);
        meanbgg(spot)=sumbgg(spot)/(121-countfg(spot));
        meanbgr(spot)=sumbgr(spot)/(121-countfg(spot));
      end

%     figure;imshow(uint8(ims*4));
%     figure;imshow(uint8(im3));
end
means=zeros(19*21,4);
means(:,1)=meanfgg';
means(:,2)=meanfgr';
means(:,3)=meanbgg';
means(:,4)=meanbgr';

save('intensity-wu-nn.txt','means','-ascii');
figure;imshow(uint8(edge));

% getedge2() return the edges which are the background pixels
edge2=getedge2(edge)*255;

%newsubgrid(1:M,1:N,1)=edge+edge2;
%newsubgrid(1:M,1:N,2)=edge-edge2;
%newsubgrid(1:M,1:N,3)=edge-edge2;
%figure,imshow(uint8(newsubgrid));

sum(countfg)
sum(meanfgg)
sum(meanfgr)
sum(meanbgg)
sum(meanbgr)
```

5. The OKMIS Implementation

```
clear all;

%parameters
grid_rows_in_array = 4
grid_cols_in_array = 4

rows_in_grid=19
cols_in_grid=21

spot_number=grid_rows_in_array*grid_cols_in_array*rows_in_grid*cols_in_grid

width=11;
height=11;

half_w=(width-1)/2;
half_h=(height-1)/2;

%read the image and spot centers from the file
load D:\Lily\WindsorStudy\RA\apo-data\images\try2\ac.1.center;

ima=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1G.tif.marray','tif');
ima=double(ima);
imb=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1R.tif.marray','tif');
imb=double(imb);
im1=ima+imb;

subgrid=0;
start_x0=ac(subgrid*rows_in_grid*cols_in_grid+1,2)-half_w;
start_y0=ac(subgrid*rows_in_grid*cols_in_grid+1,1)-half_h;
imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
suba=ima(round(start_y0):round(start_y0)+size_imnew(1)-
1,round(start_x0):round(start_x0+size_imnew(2)-1));
subb=imb(round(start_y0):round(start_y0)+size_imnew(1)-
1,round(start_x0):round(start_x0+size_imnew(2)-1));
subgrid=im1(round(start_y0):round(start_y0)+size_imnew(1),round(start_x0):round(start_x0+siz
e_imnew(2)));
subgrid2=sqrt(suba)+sqrt(subb);
figure;imshow(uint8(subgrid2));

countfg=zeros(19*21,1);
countbg=zeros(19*21,1);
sumfgg=zeros(19*21,1);
sumbgg=zeros(19*21,1);
sumfgr=zeros(19*21,1);
sumbgr=zeros(19*21,1);

for spot=1:21*19
```

```
spot
start_x1=ac(spot,2)-half_w;
start_y1=ac(spot,1)-half_h;
im2=im1(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-half_h):
round( ac(spot,2) +half_h));
spa2=suba(round(ac(spot,1)-half_w-start_y0+1):...
      round(ac(spot,1)+half_w-start_y0+1),...
      round(ac(spot,2)-half_h-start_x0+1):...
      round(ac(spot,2)+half_h-start_x0+1));
spb2=subb(round(ac(spot,1)-half_w-start_y0+1):...
      round(ac(spot,1)+half_w-start_y0+1),...
      round(ac(spot,2)-half_h-start_x0+1):...
      round(ac(spot,2)+half_h-start_x0+1));

%square root of the intensity
spa=sqrt(spa2);
spb=sqrt(spb2);

%maximum value of the two channels
for i=1:11
  for j=1:11
    ims(i,j)=spa(i,j)+spb(i,j);
  end
end
start_x=ac(1,2)-half_w;
start_y=ac(1,1)-half_h;

a=round(ac(spot,1)-half_w);
b=round(ac(spot,2)-half_h);

%x1 intensity of the pixel
x1=ims;

% mu
Ti=0;
mu=[0;0];
for i=1:11
  for j=1:11
    Ti=im2(i,j)+Ti;
    mu=+mu+[i;j]*im2(i,j);
  end
end
mu=mu/(Ti);

% x2 radius
x2=zeros(width,height);
for i=1:11
  for j=1:11
    x2(i,j)=((i-mu(1))^2+(j-mu(2))^2);
  end
```

```
end

%x3 mean of the surrounding pixels within <radius> range
radius=1;
x3=zeros(width,height);
for ii=1:width
  for jj=1:height
    k = 0; s=0;
    for i = 1:width
      for j = 1:height
        if abs(ii-i)+abs(jj-j) <= radius
          k = k+1;
          s = s+ims(i,j);
        end
      end
    end
    x3(ii,jj)=s/k;
  end
end

y1=reshape(x1,1,width*height);
y2=reshape(x2,1,width*height);
y3=reshape(x3,1,width*height);

% scaling
[sorted1,idx1]=sort(y1);
[sorted2,idx2]=sort(y2);
scale=(sorted2(121)-sorted2(1))/(sorted1(121)-sorted1(1))*0.9;
y1=y1*scale;

spot_dataset(1,1:width*height)=y1;
spot_dataset(2,1:width*height)=y2;
%spot_dataset(3,1:width*height)=y3;

% omnikMeans: dataset, k
label=omnikMeans(spot_dataset, 3);
for i=1:11
  for j=1:11
    if ( label((j-1)*11+i)==1 )
      im3(i,j)=255;
      disp_im3(i,j)=0;
    elseif ( label((j-1)*11+i)==2 )
      im3(i,j)=0;
      disp_im3(i,j)=255;
    else
      im3(i,j)=0;
      disp_im3(i,j)=127;
    end
  end
end
%figure;imshow(uint8(im3));
```

```
%find the largest continuouse region, set the area as foreground
%im4=findArea(im3);
%or not.
im4=im3;
row=ceil(spot/cols_in_grid);
col=rem(spot,cols_in_grid);
if col==0
   col=cols_in_grid;
end

%display the result
f=1;b=1;e=1;
for i=1:11
   for j=1:11
      if ( im4(i,j)==255 )
         y1f(f)=y1((j-1)*11+i);
         y2f(f)=y2((j-1)*11+i);
         edge((row-1)*11+i,(col-1)*11+j)=255;
         im3(i,j)=65535;
         f=f+1;
         countfg(spot)=countfg(spot)+1;
         sumfgg(spot)=sumfgg(spot)+spa2(i,j);
         sumfgr(spot)=sumfgr(spot)+spb2(i,j);
      elseif ( label((j-1)*11+i)==2 | label((j-1)*11+i)==3 )
         y1e(e)=y1((j-1)*11+i);
         y2e(e)=y2((j-1)*11+i);
         edge((row-1)*11+i,(col-1)*11+j)=0;
         im3(i,j)=0;
         b=b+1;
         countbg(spot)=countbg(spot)+1;
         sumbgg(spot)=sumbgg(spot)+spa2(i,j);
         sumbgr(spot)=sumbgr(spot)+spb2(i,j);
      end
   end
end

countfg(spot);
meanfgg(spot)=sumfgg(spot)/countfg(spot);
meanfgr(spot)=sumfgr(spot)/countfg(spot);
meanbgg(spot)=sumbgg(spot)/countbg(spot);
meanbgr(spot)=sumbgr(spot)/countbg(spot);

start_x=start_x1-start_x0;
start_y=start_y1-start_y0;
subgrid(round(start_y+1):round(start_y+11),round(start_x+1):round(start_x+11))=im3;
end

%figure; imshow(uint16(subgrid*10));
figure;imshow(uint8(edge));
```

```
means=zeros(19*21,4);
means(:,1)=meanfgg';
means(:,2)=meanfgr';
means(:,3)=meanbgg';
means(:,4)=meanbgr';

save('intensity3-okmis-nn.txt','means','-ascii');
sum(countfg)
sum(meanfgg)
sum(meanfgr)
sum(meanbgg)
sum(meanbgr)
```

6. The LCR Implementation

```
function g=findArea(im3)
k=1;
%initial the Mark metrix
Mark=zeros(11,11);
for i=1:11
  for j=1:11
    %if the pixel is foreground, and it is not marked, mark it with k
    %paint is recursive
    if ( im3(i,j)==255 & Mark(i,j)==0 )
      Mark=paint(im3,Mark,i,j,k);
      k=k+1;
    end
  end
end

%find the largest area
labels=zeros(k,1);
for i=1:11
  for j=1:11
    if Mark(i,j)~=0
      labels(Mark(i,j))=labels(Mark(i,j))+1;
    end
  end
end
[a,b]=max(labels);

%set the area = 255, other = 0
for i=1:11
  for j=1:11
    if Mark(i,j)==b
      Mark(i,j)=255;
    else
      Mark(i,j)=0;
```

```
    end
  end
end

g=Mark;
```

7. The Paint Implementation

```
function g=paint(im,Mark,x,y,k)

%paint the surrounding 4 pixels if they are not marked
if im(x,y)==255
  Mark(x,y)=k;
  if (y>1 &im(x,y-1)==255 & Mark(x,y-1)==0)
    Mark=paint(im,Mark,x,y-1,k);
  end
  if (y<11 &im(x,y+1)==255 & Mark(x,y+1)==0)
    Mark=paint(im,Mark,x,y+1,k);
  end
  if (x>1 &im(x-1,y)==255 & Mark(x-1,y)==0)
    Mark=paint(im,Mark,x-1,y,k);
  end
  if (x<11 &im(x+1,y)==255 & Mark(x+1,y)==0)
    Mark=paint(im,Mark,x+1,y,k);
  end
end
g=Mark;
```

8. The Adaptive Circle (Laplacian) Implementation

```
clear all;

grid_rows_in_array = 4
grid_cols_in_array = 4

rows_in_grid=19
cols_in_grid=21

spot_number=grid_rows_in_array*grid_cols_in_array*rows_in_grid*cols_in_grid;

width=11;
height=11;
```

```
half_w=(width-1)/2;
half_h=(height-1)/2;

img=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1G.tif.marray','tif');
imr=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1R.tif.marray','tif');
load  D:\Lily\WindsorStudy\RA\apo-data\images\try2\ac.1.center
img=double(img);
imr=double(imr);
size_im=size(imr);
im1=img+imr;

grid=0;

ac(grid*rows_in_grid*cols_in_grid+1,1);
ac(grid*rows_in_grid*cols_in_grid+1,2);
start_x=ac(grid*rows_in_grid*cols_in_grid+1,2)-half_w;
start_y=ac(grid*rows_in_grid*cols_in_grid+1,1)-half_h;
imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
imnew2=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
subgrid=im1(round(start_y):round(start_y)+size_imnew(1),round(start_x):round(start_x+size_im
new(2)));
[a,b]=size(subgrid);
imnew2=zeros(a,b);

%smooth the picture
k=[1/9,1/9,1/9;1/9,1/9,1/9;1/9,1/9,1/9]
im2=imfilter(double(subgrid),k,'replicate');
im2=imfilter(double(im2),k,'replicate');

%apply laplacian of gaucian
k2=fspecial('log',[5,5],0.5);

im2=imfilter(double(im2),k2,'replicate');

%find zero-crossings
%may have double-edge problem
[M,N]=size(subgrid);
edge=zeros(M,N);
edge2=zeros(M,N);
subgrid2=sqrt(subgrid);
subgridnew=zeros(M,N);

for i=1:M
   for j=1:N
      count=0;
      if i-1>0 & im2(i,j)*im2(i-1,j)<0
         count =count+1;
      end
      if i+1<=M & im2(i,j)*im2(i+1,j)<0
```

```
        count =count+1;
      end
      if j+1 <= N & im2(i,j)*im2(i,j+1)<0
        count =count+1;
      end
      if j-1>0 & im2(i,j)*im2(i,j-1)<0
        count =count+1;
      end
      if j-1>0 & i-1>0 & im2(i,j)*im2(i-1,j-1)<0
        count =count+1;
      end
      if j+1 <= N & i-1>0 & im2(i,j)*im2(i-1,j+1)<0
        count =count+1;
      end
      if i+1<=M & j-1>0 & im2(i,j)*im2(i+1,j-1)<0
        count =count+1;
      end
      if j+1 <= N & i+1<=M & im2(i,j)*im2(i+1,j+1)<0
        count =count+1;
      end
      if im2(i,j)==0
        count = count+1;
      end
      %
      if count >= 4
        edge(i,j)=1;
      else
        edge(i,j)=0;
      end
    end
end

figure;imshow(uint16(im2*10));
figure;imshow(uint8(edge*256));

edge2=zeros(a,b);
for spot=1:21*19
  spot;
  radius=0;n=0;
  clear r;
  for i=1:width
    for j=1:height
      co_y=round(ac(spot,1)-start_y)-half_h+j;
      co_x=round(ac(spot,2)-start_x)-half_w+i;
      if (edge(co_y,co_x)==1)
        n=n+1;
        r(n)=sqrt((i-5.5)^2+(j-5.5)^2);
      end
    end
  end
  %radius=mean(r)
```

```
   radius=median(r);

   int1=im1(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-
half_h):round(ac(spot,2)+half_h));
   Ti=0;
   mu=[0;0];
   for i=1:11
      for j=1:11
         Ti=int1(i,j)+Ti;
         mu=mu+[i;j]*int1(i,j);
      end
   end
   mu=mu/(Ti);

   starty=round(ac(spot,1)-start_y-half_w);
   startx=round(ac(spot,2)-start_x-half_h);
   for i=1:width
      for j=1:height
         dd=sqrt((i-mu(1))^2+(j-mu(2))^2);
         co_y=starty+i;
         co_x=startx+j;
         if abs(dd-radius)<0.5
            edge2(co_y,co_x)=1;
         else
            edge2(co_y,co_x)=0;
         end

         if dd-radius < 0
            subgridnew(co_y,co_x)=255;
         else
            subgridnew(co_y,co_x)=0;
         end

%          edge2((spot-1)*width+i,(spot-1)*height+j)=edge2(i,j)*65536;
         edge2(co_y,co_x)=edge2(co_y,co_x)*65536;
      end
   end
end

imnew2=subgrid;

newsubgrid_display(1:a,1:b,1)=subgrid*4+edge2;
newsubgrid_display(1:a,1:b,2)=(subgrid-edge2)*4;
newsubgrid_display(1:a,1:b,3)=(subgrid-edge2)*4;
figure,imshow(uint16(newsubgrid_display));


% the figure below is for comparison with other method.
% the figure above shows better result than the figure below.
% getedge2() return the edges which are the background pixels
edge2=getedge2(subgridnew)*255;
```

```
newsubgrid(1:M,1:N,1)=subgrid2+edge2;
newsubgrid(1:M,1:N,2)=subgrid2-edge2;
newsubgrid(1:M,1:N,3)=subgrid2-edge2;
figure,imshow(uint8(newsubgrid));
```

9. The Adaptive Circle (Threshold) Implementation

```
clear all;

grid_rows_in_array = 4;
grid_cols_in_array = 4;
rows_in_grid=19;
cols_in_grid=21;
width=11;
height=11;
spot_number=grid_rows_in_array*grid_cols_in_array*rows_in_grid*cols_in_grid;
grid=0;

half_w=(width-1)/2;
half_h=(height-1)/2;

load D:\Lily\WindsorStudy\RA\apo-data\images\try2\ac.1.center
imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
imnew2=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
start_x0=ac(grid*rows_in_grid*cols_in_grid+1,2)-half_w;
start_y0=ac(grid*rows_in_grid*cols_in_grid+1,1)-half_h;

ima=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1G.tif.marray','tif');
ima=double(ima);
imb=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1R.tif.marray','tif');
imb=double(imb);
im1=ima+imb;

im1=ima+imb;
subgrid=im1(round(start_y0):round(start_y0)+size_imnew(1),round(start_x0):round(start_x0+siz
e_imnew(2)));
subgridnew=subgrid;
subgrid2=subgrid;
countfg=zeros(19*21,1);
valid_count_fg=zeros(19*21,1);
countbg=zeros(19*21,1);
sumfgg=zeros(19*21,1);
sumbgg=zeros(19*21,1);
sumfgr=zeros(19*21,1);
sumbgr=zeros(19*21,1);
```

```
meanfgg=zeros(1,19*21);
meanfgr=zeros(1,19*21);
meanbgg=zeros(1,19*21);
meanbgr=zeros(1,19*21);
weak = 0;

[M,N]=size(subgrid);
edge2=zeros(M,N);

for spot=1:21*19

  spot
  im2=im1(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-
half_h):round(ac(spot,2)+half_h));
  img=ima(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-
half_h):round(ac(spot,2)+half_h));
  start_x=ac(spot,2)-half_w;
  start_y=ac(spot,1)-half_h;
  start_x=round(ac(spot,2)-half_w-start_x0);
  start_y=round(ac(spot,1)-half_h-start_y0);

  d1=im2;

  %calculate mu, the center mean
  Ti=0;
  mu=[0;0];
  for i=1:11
    for j=1:11
      Ti=d1(i,j)+Ti;
      mu=mu+[i;j]*d1(i,j);
    end
  end
  mu=mu/(Ti);

  %figure;plot(y1,y2,'--rs');
  %figure;plot(round(y1),round(y2),'--rs');

  d2=reshape(im2,121,1);
  [sortedi,idx]=sort(d2);
  y(1)=im2(1,1);
  y(2)=im2(1,11);
  y(3)=im2(11,1);
  y(4)=im2(11,11);
  y(5)=im2(6,1);
  y(6)=im2(6,11);
  y(7)=im2(1,6);
  y(8)=im2(11,6);

  flag=1;
  xy(1:8)=y;
  MWT=0;
```

```
while (MWT<90)
  if flag+7>size(sortedi)
    threhd=65535;
    break;
  end
  x(1:8)=sortedi(flag:flag+7);
  xy(9:16)=x;
  [xy2,idx]=sort(xy);
  MWT=0;
  for i=1:16
    if idx(i)>8
      MWT=MWT+i;
    end
  end
  flag=flag+1;
end
threhd=min(x);

for i=1:11
  for j=1:11
    radius2(i,j)=sqrt((i-mu(1,1))^2+(j-mu(2,1))^2);
  end
end

radius=reshape(radius2,121,1);
[sortedr,idxr]=sort(radius);
ins = sort(im2);

%improvement.
rings=zeros(121,1);
countOn = 0;
countOff = 0;
i=1;
stop='n';
while (stop=='n')
  countOn = 0;
  countOff = 0;
  for j=i:i+2
    if ( j<=121)
      a = d2(idxr(j));
      if a> threhd
        countOn=countOn+1;
      else
        countOff=countOff+1;
      end
    end
  end
  if i>=121
    stop='a';
  elseif ( countOn<countOff )
    stop='y';
```

```
  else
    i=i+3;
  end
end

if stop=='y'
  count=i;
else
  count=121;
end
radiusFg=sortedr(count);

if (count >= 114)
  weak = weak+1;
end

%pixels within the radius are foreground
starty=round(ac(spot,1)-start_y0-half_w);
startx=round(ac(spot,2)-start_x0-half_h);
dispSpot=zeros(11,11);
for i=1:11
  for j=1:11
    if ( radius2(i,j)<=radiusFg )
      dispSpot(i,j)=255;
      countfg(spot)=countfg(spot)+1;
      sumfgg(spot)=sumfgg(spot)+img(i,j);
      sumfgr(spot)=sumfgr(spot)+img(i,j);
    else
      sumbgg(spot)=sumbgg(spot)+img(i,j);
      sumbgr(spot)=sumbgr(spot)+img(i,j);
    end
  end
end
%figure;imshow(uint8(dispSpot));

if count < 114
  valid_count_fg(spot) = countfg(spot);
  if countfg(spot)==0
    meanfgg(spot)=0;
    meanfgr(spot)=0;
    meanbgg(spot)=sumbgg(spot)/121;
    meanbgr(spot)=sumbgr(spot)/121;
  elseif countfg(spot)==121
    meanfgg(spot)=sumfgg(spot)/countfg(spot);
    meanfgr(spot)=sumfgr(spot)/countfg(spot);
    meanbgg(spot)=0;
    meanbgr(spot)=0;
  else
    meanfgg(spot)=sumfgg(spot)/countfg(spot);
    meanfgr(spot)=sumfgr(spot)/countfg(spot);
    meanbgg(spot)=sumbgg(spot)/(121-countfg(spot));
```

```
        meanbgr(spot)=sumbgr(spot)/(121-countfg(spot));
      end
    end
    row=ceil(spot/cols_in_grid);
    col=rem(spot,cols_in_grid);
    if col==0
      col=cols_in_grid;
    end
    subgridnew(start_y+1:start_y+11,start_x+1:start_x+11)=dispSpot;

end
means=zeros(19*21,4);
means(:,1)=meanfgg';
means(:,2)=meanfgr';
means(:,3)=meanbgg';
means(:,4)=meanbgr';
means(:,5)=valid_count_fg;

save('intensity-adpcir.txt','means','-ascii');

% getedge2() return the edges which are the background pixels
edge2=getedge2(subgridnew)*65535;

newsubgrid(1:M,1:N,1)=subgrid2*3+edge2;
newsubgrid(1:M,1:N,2)=(subgrid2-edge2)*3;
newsubgrid(1:M,1:N,3)=(subgrid2-edge2)*3;
% figure; imshow(uint16(subgrid*8));

% figure; imshow(uint8(subgridnew));
figure,imshow(uint16(newsubgrid));
sum(countfg)/(399-weak)
sum(meanfgg)/(399-weak)
sum(meanfgr)/(399-weak)
sum(meanbgg)/(399-weak)
sum(meanbgr)/(399-weak)
weak
sum(valid_count_fg)/(399-weak)
```

10. The Adaptive Ellipse Method Implementation

```
clear all;

%parameter of the image files
grid_rows_in_array = 4;
grid_cols_in_array = 4;
rows_in_grid=19;
cols_in_grid=21;
```

```
width=11;
height=11;
spot_number=grid_rows_in_array*grid_cols_in_array*rows_in_grid*cols_in_grid;
grid=0;

half_w=(width-1)/2;
half_h=(height-1)/2;

%read from the image files and parameter file
load D:\Lily\WindsorStudy\RA\apo-data\images\try2\ac.1.center
ima=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1G.tif.marray','tif');
ima=double(ima);
imb=imread('D:\Lily\WindsorStudy\RA\apo-data\images\1230c1R.tif.marray','tif');
imb=double(imb);
im1=ima+imb;

imnew=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
imnew2=zeros(round(rows_in_grid*height),round(cols_in_grid*width));
size_imnew=size(imnew);
start_x0=ac(grid*rows_in_grid*cols_in_grid+1,2)-half_w;
start_y0=ac(grid*rows_in_grid*cols_in_grid+1,1)-half_h;

% using uint16 seems generating better result than using uint8
% ima=sqrt(ima);
% imb=sqrt(imb);
im1=ima+imb;
subgrid=im1(round(start_y0):round(start_y0)+size_imnew(1),round(start_x0):round(start_x0+siz
e_imnew(2)));
[a,b]=size(subgrid);
subgridnew=zeros(a,b);
subgrid2=subgrid;

countfg=zeros(19*21,1);
valid_countfg=zeros(19*21,1);
countbg=zeros(19*21,1);
sumfgg=zeros(19*21,1);
sumbgg=zeros(19*21,1);
sumfgr=zeros(19*21,1);
sumbgr=zeros(19*21,1);
meanfgg=zeros(1,19*21);
meanfgr=zeros(1,19*21);
meanbgg=zeros(1,19*21);
meanbgr=zeros(1,19*21);
weak = 0;

[M,N]=size(subgrid);
edge2=zeros(M,N);

for spot=1:21*19

  spot
```

```
  im2=im1(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-
half_h):round(ac(spot,2)+half_h));
  img=ima(round(ac(spot,1)-half_w):round(ac(spot,1)+half_w),round(ac(spot,2)-
half_h):round(ac(spot,2)+half_h));
  im3=sqrt(im2);
  start_x=ac(spot,2)-half_w;
  start_y=ac(spot,1)-half_h;
  start_x=round(ac(spot,2)-half_w-start_x0);
  start_y=round(ac(spot,1)-half_h-start_y0);

  d1=im2;

  %calculate mu, the center mean
  Ti=0;
  mu=[0;0];
  for i=1:11
    for j=1:11
      Ti=d1(i,j)+Ti;
      mu=mu+[i;j]*d1(i,j);
    end
  end
  mu=mu/(Ti);

  %calculate sigma, the covariance matrix
  sigma=zeros(2,2);
  for i=1:11
    for j=1:11
      %ellipse -> round
      x11=[i;j]-mu;
      sigma=sigma+x11*x11'*d1(i,j);
    end
  end
  sigma=sigma/(Ti);

  [U,S,V]=svd(sigma);
  %equal
  S;
  V'*sigma*V;
  V*sigma*V';

  %equal
  eye(2);
  S^(-1/2)*S*S^(-1/2);

  %transform
  for i=1:11
    for j=1:11
      a=S^(-1/2)*V'*[i;j];
      y1(i,j)=a(1,1);
      y2(i,j)=a(2,1);
    end
```

```
end

%varify, the result of method 1 should be same as that of method 2
%calculate mu, the center mean in transformed space
%method 1
Ti=0;
mu2=[0;0];
for i=1:11
  for j=1:11
    Ti=d1(i,j)+Ti;
    mu2=mu2+[y1(i,j);y2(i,j)]*d1(i,j);
  end
end
mu2=mu2/(Ti);
%method 2
mu2=S^(-1/2)*V'*mu;

%varify, the result of method 1 should be same as that of method 2, and should be identity matrix
%calculate sigma, the covariance matrix
%method 1
sigma2=zeros(2,2);
for i=1:11
  for j=1:11
    %ellipse -> round
    x11=[y1(i,j);y2(i,j)]-mu2;
    sigma2=sigma2+x11*x11'*d1(i,j);
  end
end
sigma2=sigma2/(Ti);
%method 2
sigma2=S^(-1/2)*V'*sigma;

%figure;plot(y1,y2,'--rs');
%figure;plot(round(y1),round(y2),'--rs');

d2=reshape(im2,121,1);
[sortedi,idx]=sort(d2);

%improvement from
%y(1:8)=d2(1:8);
y(1)=im2(1,1);
y(2)=im2(1,11);
y(3)=im2(11,1);
y(4)=im2(11,11);
y(5)=im2(6,1);
y(6)=im2(6,11);
y(7)=im2(1,6);
y(8)=im2(11,6);

%calculate the threshold using Mann-Whitney test, set mwt = 90.
flag=1;
```

```
xy(1:8)=y;
MWT=0;
while (MWT<90)
  if flag+7>size(sortedi)
    threhd=65535;
    break;
  end
  x(1:8)=sortedi(flag:flag+7);
  xy(9:16)=x;
  [xy2,idx]=sort(xy);
  MWT=0;
  for i=1:16
    if idx(i)>8
      MWT=MWT+i;
    end
  end
  flag=flag+1;
end
threhd=min(x);

%calculate the distance from each spot to the center
for i=1:11
  for j=1:11
    radius2(i,j)=sqrt((y1(i,j)-mu2(1,1))^2+(y2(i,j)-mu2(2,1))^2);
  end
end

%sort the distance
radius=reshape(radius2,121,1);
[sortedr,idxr]=sort(radius);
ins = sort(d2);

%find the radius that define the foreground region
rings=zeros(121,1);
countOn = 0;
countOff = 0;
i=1;
stop='n';
while (stop=='n')
  countOn = 0;
  countOff = 0;
  for j=i:i+2
    if ( j<=121)
      a = d2(idxr(j));
      if a> threhd
        countOn=countOn+1;
      else
        countOff=countOff+1;
      end
    end
  end
```

```
  if i>=121
    stop='a';
  elseif ( countOn<countOff )
    stop='y';
  else
    i=i+3; % we can try i=i+1;
  end
end

if stop=='y'
  count=i;
else
  count=121;
end
radiusFg=sortedr(count);

if ( count >= 114 )
  weak = weak +1;
end

%pixels within the radius are foreground
starty=round(ac(spot,1)-start_y0-half_w);
startx=round(ac(spot,2)-start_x0-half_h);
dispSpot=zeros(11,11);
for i=1:11
  for j=1:11
    if ( radius2(i,j)<=radiusFg )
      dispSpot(i,j)=255;
      countfg(spot)=countfg(spot)+1;
      sumfgg(spot)=sumfgg(spot)+img(i,j);
      sumfgr(spot)=sumfgr(spot)+img(i,j);
    else
      sumbgg(spot)=sumbgg(spot)+img(i,j);
      sumbgr(spot)=sumbgr(spot)+img(i,j);
    end
  end
end
%figure;imshow(uint8(dispSpot));

%statistics
if countfg(spot)==0
  meanfgg(spot)=0;
  meanfgr(spot)=0;
  meanbgg(spot)=sumbgg(spot)/121;
  meanbgr(spot)=sumbgr(spot)/121;
elseif countfg(spot)==121
  meanfgg(spot)=sumfgg(spot)/countfg(spot);
  meanfgr(spot)=sumfgr(spot)/countfg(spot);
  meanbgg(spot)=0;
  meanbgr(spot)=0;
else
```

```
      meanfgg(spot)=sumfgg(spot)/countfg(spot);
      meanfgr(spot)=sumfgr(spot)/countfg(spot);
      meanbgg(spot)=sumbgg(spot)/(121-countfg(spot));
      meanbgr(spot)=sumbgr(spot)/(121-countfg(spot));
    end

end
  row=ceil(spot/cols_in_grid);
  col=rem(spot,cols_in_grid);
  if col==0
    col=cols_in_grid;
  end
  subgridnew(start_y+1:start_y+11,start_x+1:start_x+11)=dispSpot;

end
means=zeros(19*21,4);
means(:,1)=meanfgg';
means(:,2)=meanfgr';
means(:,3)=meanbgg';
means(:,4)=meanbgr';
means(:,5)=valid_countfg;
save('intensity-adpelp.txt','means','-ascii');

% getedge2() return the edges which are the background pixels
edge2=getedge2(subgridnew)*65535;
% newsubgrid(1:M,1:N,1)=subgrid2*3+edge2;
% newsubgrid(1:M,1:N,2)=(subgrid2-edge2)*3;
% newsubgrid(1:M,1:N,3)=(subgrid2-edge2)*3;
% figure; imshow(uint16(subgrid*4));
figure; imshow(uint8(subgridnew));

% figure,imshow(uint16(newsubgrid));
sum(countfg)/(399-weak)
sum(meanfgg)/(399-weak)
sum(meanfgr)/(399-weak)
sum(meanbgg)/(399-weak)
sum(meanbgr)/(399-weak)
sum(valid_countfg)/(399-weak)
```

VITA AUCTORIS

Li Qin was born in 1969 in Tianjin, China. She graduated from Tianjin No. 3 middle school in 1987. From there she went on to Beijing University of Posts and Telecommunications where she obtained a B.Eng. in Computer Communication in 1991. After working in the IT industry for more than 10 years, she went back to school for the Master's degree in Computer Science at the University of Windsor. She is expecting to graduate in Fall 2004.