

# **CONTOUR BASED 3D BIOLOGICAL IMAGE RECONSTRUCTION AND PARTIAL RETRIEVAL**

by

YONG LI

Under the Direction of Saeid Belkasim

## **ABSTRACT**

Image segmentation is one of the most difficult tasks in image processing. Segmentation algorithms are generally based on searching a region where pixels share similar gray level intensity and satisfy a set of defined criteria. However, the segmented region cannot be used directly for partial image retrieval. In this dissertation, a Contour Based Image Structure (CBIS) model is introduced. In this model, images are divided into several objects defined by their bounding contours. The bounding contour structure allows individual object extraction, and partial object matching and retrieval from a standard CBIS image structure.

The CBIS model allows the representation of 3D objects by their bounding contours which is suitable for parallel implementation particularly when extracting contour features and matching them for 3D images require heavy computations. This computational burden becomes worse for images with high resolution and large contour density. In this essence we designed two parallel algorithms; Contour Parallelization Algorithm (CPA) and Partial Retrieval Parallelization Algorithm (PRPA). Both algorithms have considerably improved the performance of CBIS for both contour shape matching as well as partial image retrieval.

To improve the effectiveness of CBIS in segmenting images with inhomogeneous backgrounds we used the phase congruency invariant features of Fourier transform components to highlight boundaries of objects prior to extracting their contours. The contour matching process has also been improved by constructing a fuzzy contour matching system that allows unbiased matching decisions.

Further improvements have been achieved through the use of a contour tailored Fourier descriptor to make translation and rotation invariance. It is proved to be suitable for general contour shape matching where translation, rotation, and scaling invariance are required.

For those images which are hard to be classified by object contours such as bacterial images, we define a multi-level cosine transform to extract their texture features for image classification. The low frequency Discrete Cosine Transform coefficients and Zenike moments derived from images are trained by Support Vector Machine (SVM) to generate multiple classifiers.

## INDEX WORDS

Image Processing, Content Based Image Retrieval, Image Segmentation, Image Shape Matching, XML Image Structure, 3D Reconstruction, 3D Partial Retrieval, Fourier Transform, Phase Congruency, Multi-level Cosine Transform, Fuzzy Logic, Genetic Algorithm.

**CONTOUR BASED 3D BIOLOGICAL IMAGE RECONSTRUCTION AND  
PARTIAL RETRIEVAL**

by

YONG LI

A Dissertation Submitted in Partial Fulfillment of Requirements for the Degree of

Doctor of Philosophy

In the College of Arts and Sciences

Georgia State University

2007

Copyright by  
Yong Li  
2007

**CONTOUR BASED 3D BIOLOGICAL IMAGE RECONSTRUCTION AND  
PARTIAL RETRIEVAL**

by

YONG LI

Major Professor: Saeid Belkasim  
Committee: Yi Pan  
Rajshekhar Sunderraman  
Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies  
College of Arts and Sciences  
Georgia State University  
December 2007

## **Acknowledgments**

I would like to thank my advisor, Dr. Saeid Belkasim, for his valuable and generous guidance and endless support throughout my Ph.D study and during the process of my dissertation. I am also extremely grateful to the members of my committee, Dr. Yi Pan, Dr. Rajshekhar Sunderraman, and Dr. Yichuan Zhao for their well-appreciated support and assistance during my graduate study. The dissertation would not have been possible without their helps.

Last, but not least, I would like to thank my family for their strong, patient and persistent encouragement, understanding and supporting me in my educational pursuits.

# TABLE OF CONTENTS

<b>CHAPTER 1 INTRODUCTION .....</b>	<b>1</b>
<b>CHAPTER 2 CONTOUR BASED IMAGE STRUCTURE (CBIS) AND XML IMAGE STRUCTURE.....</b>	<b>6</b>
2.1 BACKGROUND .....	6
2.2 SEPARATING STAGE .....	8
2.3 GROUPING STAGE .....	12
2.4 CONTOUR XML STRUCTURE .....	15
2.5 3D RECONSTRUCTION AND PARTIAL RETRIEVAL .....	17
<b>CHAPTER 3 PARALLEL IMPLEMENTATION .....</b>	<b>23</b>
3.1 LAYER PARALLELIZATION ALGORITHM (LPA) .....	23
3.2 CONTOUR PARALLELIZATION ALGORITHM (CPA) .....	25
3.3 PARTIAL RETRIEVAL PARALLELIZATION ALGORITHM (PRPA) .....	26
3.4 EXPERIMENTAL RESULTS ON PARALLEL IMPLEMENTATION .....	27
<b>CHAPTER 4 SEGMENTATION USING PHASE CONGRUENCY .....</b>	<b>33</b>
4.1 THE PROBLEM OF OPTIMAL THRESHOLD METHOD .....	34
4.2 PHASE CONGRUENCY FOR EDGE DETECTION .....	39
4.3 SEGMENTATION USING PHASE CONGRUENCY .....	45
<b>CHAPTER 5 FUZZY CONTOUR MATCHING.....</b>	<b>53</b>
5.1 THE CONSIDERATIONS OF THE FLS INPUTS .....	54
5.2 FUZZY LOGIC SYSTEM FOR CONTOUR MATCHING .....	57
5.2.1 MEMBERSHIP FUNCTIONS .....	58
5.2.2 FUZZY RULES .....	61
5.2.3 FUZZY INFERENCE AND DEFUZZIFICATION .....	62
5.2.4 TUNING THE MEMBERSHIP FUNCTIONS BY GENETIC ALGORITHMS (GA) .....	63
5.3 USING FCMS TO BUILD CONTOUR STRUCTURE FROM IMAGE STACK .....	64
<b>CHAPTER 6 INVARIANT IMAGE FEATURE EXTRACTION FROM FREQUENCY DOMAIN .....</b>	<b>68</b>
6.1 SHAPE SIGNATURE AND COMPLEX CONTOUR VECTOR .....	68
6.2 CONTOUR FEATURES EXTRACTED BY FOURIER DESCRIPTOR .....	69
6.3 DISCRETE COSINE TRANSFORM (DCT) .....	70
6.4 ODD AND EVEN COSINE TRANSFORM FOR IMAGE FEATURE EXTRACTION .....	71
<b>CHAPTER 7 TEXTURE IMAGE CLASSIFICATION USING MULTI-LEVEL COSINE TRANSFORM.....</b>	<b>74</b>
7.1 TEXTURE FEATURE EXTRACTION AND MULTI-LEVEL DISCRETE COSINE TRANSFORM .....	78
7.1.1 DCT COEFFICIENTS AS IMAGE TEXTURE FEATURES .....	78
7.1.2 IMAGE FEATURE FROM ZENIKE MOMENTS .....	81
7.2 IMAGE FEATURE TRAINING AND CLASSIFICATION USING SVM .....	82

7.2.1 BINARY CLASSIFICATION .....	82
7.2.2 MULTI-CLASS CLASSIFICATION.....	83
7.3 EXPERIMENTAL ANALYSIS .....	84
<b>CHAPTER 8 CONCLUSIONS AND FUTURE WORK .....</b>	<b>88</b>
8.1 CONCLUSIONS .....	88
8.2 FUTURE WORK .....	94
<b>BIBLIOGRAPHY .....</b>	<b>97</b>



## LIST OF FIGURES

Figure 2.1 (a) Confocal microscopic image slice of crayfish neuron (b) After applying optimal threshold (c) Image contours .....	10
Figure 2.2 (a) Original image with noise (b) Image after applying contour segmentation (c) Removing noise using minimum contour size threshold .....	11
Figure 2.3 (a) Original confocal microscopic image slices of crayfish neuron (b) Enhanced contour images generated from the xml structure .....	13
Figure 2.4 Size verification of the two successive contours .....	15
Figure 2.5 (a) Example of 3 adjacent slices (b) Contour data structure for (a) .....	16
Figure 2.6 Contour objects in the xml structure .....	17
Figure 2.7 3D model of a crayfish neuron confocal image stack .....	20
Figure 2.8 3D components of a crayfish neuron branch .....	22
Figure 3.1 Layer parallelization algorithm (LPA) .....	24
Figure 3.2 Contour parallelization algorithm (CPA) .....	25
Figure 3.3 Partial retrieval parallelization algorithm (PRPA) .....	26
Figure 3.4 Speedups of the LPA and CPA .....	32
Figure 3.5 Parallel partial image retrieval speedups .....	32
Figure 4.1 (a) Confocal microscopic image slice of crayfish neuron (b) After applying optimal thresholding (c) After applying decreased threshold .....	35
Figure 4.2 (a) Saturn (b) Bacteria .....	40
Figure 4.3 (a) Saturn (phase) + Bacteria (magnitude) (b) Bacteria (phase) + Saturn (magnitude) .....	42
Figure 4.4 (a) Phase congruency image (b) Sobel edge detection image .....	44
Figure 4.5 (a) Original image (b) Phase congruency image (c) After applying edge filter .....	48
Figure 4.6 Block diagram of the algorithm .....	50
Figure 4.7 (a) Contour based image segmentation using phase congruency for edge detection (b) Contour based image segmentation using Sobel method for edge detection .....	51
Figure 5.1 The structure of a FLS .....	58
Figure 5.2 The fuzzy membership functions (a) Non-overlapping ratio (b) Difference of lighting intensity (c) Difference of object orientation (d) The output .....	59

<i>Figure 5.3 Contour based XML image structure built by FCMS.....</i>	<i>65</i>
<i>Figure 5.4 The Algorithm to list all the 3D subcomponents in the xml image structure ..</i>	<i>67</i>
<i>Figure 7.1 Bacteria Images (a) Bacillus (b) Bartonella henselae (c) Bordetella pertussis</i> <i>(d) Staphylococcus.....</i>	<i>75</i>
<i>Figure 7.2 Edge detection images of bacteria (a) Bacillus (b) Bartonella henselae (c)</i> <i>Bordetella pertussis (d) Staphylococcus .....</i>	<i>76</i>
<i>Figure 7.3 (a) Square image, dark and gray evenly divided (b) Stripe image, stripe size: 4</i> <i>by 64.....</i>	<i>79</i>
<i>Figure 7.4 Three 4 by 4 image blocks.....</i>	<i>80</i>
<i>Figure 7.5 Experimental texture classes .....</i>	<i>85</i>

## LIST OF TABLES

<i>Table 2.1 New storage size for an image stack using contour data structure. Original image stack consists of 20 crayfish neuron confocal microscope (2048 x 2048)....</i>	<i>16</i>
<i>Table 3.1 Speedup of LPA implementation original image stack consists of 20 crayfish neuron confocal microscope (2048 × 2048) .....</i>	<i>28</i>
<i>Table 3.2 Speedup of CPA implementation. Original image stack consists of 20 crayfish neuron confocal microscope (2048 × 2048) .....</i>	<i>29</i>
<i>Table 3.3 Speedup of CPA implementation for multiple 3D component retrieval .....</i>	<i>30</i>
<i>Table 4.1 Edge filter mask for object boundary feature .....</i>	<i>46</i>
<i>Table 7.1 SVM testing accuracies (%) of 9 classifiers using one-versus-rest method for the textures with different resolutions .....</i>	<i>86</i>

## **LIST OF ABBREVIATIONS**

Contour Based Image Structure	CBIS
Contour Parallelization Algorithm	CPA
Discrete Cosine Transforms	DCT
Fuzzy Contour Matching System	FCMS
Fuzzy Logic System	FLS
Genetic Algorithm	GA
Layer Parallelization Algorithm	LPA
Multi-level Discrete Cosine Transform	MDCT
Partial Retrieval Parallelization Algorithm	PRPA
Support Vector Machine	SVM
Zenike Moments	ZM

## Chapter 1 Introduction

Image segmentation, shape matching, partial retrieval, and image classification have been widely used in biological research as well as in medical treatment (Sarti *et al.*, 2000). For example, in neuroscience, 3D reconstruction of neuronal structures facilitates visualization of the anatomical relationships of neurons and their patterns of dendritic and axonal contact within nervous tissues. It also provides anatomical data for construction of electrical and biochemical circuit models of neuron functions that are used in computer simulations running on such platforms as NEURON and GENESIS (Bowen and Beeman, 1998). Generally this process involves the following steps: Image enhancement, segmentation, registration, and volume or surface rendering. Image enhancement is used to improve the image quality such as de-noising, sharpening, or smoothing the image. Segmentation is used to decompose an image into several parts of an object based on certain criteria. Shape matching is used in image registration to group the similar segmented objects on different images. Volume or surface rendering is applied to create 3D data sets from 2D image slices. Many 3D reconstruction software packages have been developed in recent years, including NEUROLUCIDA (Microbrightfield, Inc.), a semi 3D reconstruction package for neuron anatomical analysis, and 3D-Doctor (AbleSoftware Corp), a vector based architecture for 3D modeling. These products are suitable for users to interact with the created 3D models by locating the coordinates of the cursor on the screen. These approaches have very limited capabilities of automatic partial 3D component retrieval and analysis since raw images cannot be easily converted into standard image structure. We provide a contour based image structure (CBIS) which is suitable for 2D and 3D image partial retrieval. In CBIS, an image is divided into objects

which are described by their boundaries and spatial features and saved as nodes in an xml structure. In the xml structure, each node corresponds to a segmented object in the original image and is composed of several elements reflecting the object spatial features such as the coordinates of its contour, object centroid, different degrees of the moments, object principal direction, and so on. In this way, the xml structure not only records the basic shape of the object but also many other information. CBIS makes it possible for 2D partial image retrieval based on object properties in a single slice and for 3D component retrieval based on linking similar objects between all the adjacent image slices in an image stack.

Segmentation is one of the critical parts in CBIS because it determines how object contours reflect the original image. Generally the segmentation algorithms are based on searching the area where pixel intensity value has a sharp change, or partitioning an image into regions according to a set of defined criteria. Since in CBIS, we mainly focus on the boundary of an object, we enhance the object boundaries before applying a threshold for the segmentation process. We introduce a new segmentation method which uses phase congruency of the Fourier transform coefficients. It has shown that the phase congruency is insensitive to those images with uneven background illumination.

For 3D reconstruction and 3D partial retrieval, we need to bond the object contours together when they appear to belong to the same 3D component. Contour shape matching is used in this stage. Shape matching is a basic task in image registration. There are many methods have been studied including template matching, string matching, shape-specific point matching, principal axis matching, dynamic programming, mutually-best matching, chamfer matching, graph matching, relaxation, elastic matching, and etc (Wu and Wang,

1999; Veltkamp and Hagedoorn, 1999). To make an unbiased matching decision, we should use as many object features as possible as long as the computation load is reasonable. We create a Fuzzy Contour Matching System (FCMS) to combine object features such as object average intensity, overlapping ratio of two objects, and principal direction of an object. In biological images, similar tissues have similar intensity values. Two objects having close intensity imply a matching. We use overlapping ratio as a simplified correlation function since the neighbor slices are very close. We use predominate axes as the object principal orientation based on the fact that, in our case, the entire neuron branch stretches to a particular direction.

Obtaining object features and matching related contours on adjacent slices need to perform heavy computations. This computing burden is even severer when images have high contour density. Thus, solving these problems on a high-performance computing system is essential to combat both excessive amounts of time and memory constraints existing on a single processor system. CBIS makes the distribution of the contour matching tasks among multiple processors much simple. We have designed Layer Parallelization Algorithm (LPA), Contour Parallelization Algorithm (CPA), and Partial Retrieval Parallelization Algorithm (PRPA). Those algorithms all reach reasonable speedups.

In CBIS, we make the assumption the corresponding contours are very close and their shapes change gradually because the distance between two neighbor slices in the same image stack is very small. Base on this assumption, contour translation and rotation invariance is not critical to the matching decision. We also discuss how to extend our FCMS to take into the consideration of object contour translation and rotation invariance

features. Since image objects are represented by their contours, pixels inside the contour can be treated as redundant information. We use Fourier descriptor to transform a 2D image contour into one complex vector to simplify the shape matching task. From the Fourier descriptor we can easily extract invariant object features and use them as the inputs of FCMS.

CBIS is proved to be suitable for 3D reconstruction and partial retrieval for those image stacks where object contours are possible to be extracted, for example, the crayfish neuron image stacks used in our experiments. It is worth to note that not all 2D images are suitable for this approach. For example, some bacterial images have certain pattern periodically repeated in the image which makes those images more like texture images. In this case, it is not possible to break an image into objects which are represented by their contours because there are too many edges in the image. To analyze those images, it is applicable to use the whole image as the work unit. A better approach rather than CBIS should be created. Since in texture images, certain patterns are repeatedly distributed in the image, image features derived from frequency domain will be used. It is well known that low-frequency coefficients of the Discrete Cosine Transforms (DCTs) preserve the most important image features. We define a Multi-level DCT (MDCT) which can be used to extract image features based on different level of image resolution. We have used MDCT coefficients for the texture image classification. Given a texture image, the texture feature vectors generated from MDCT coefficients and Zernike moments are trained and classified by Support Vector Machines (SVMs) to build multiple classifiers. Different classifiers are combined to distinguish images of one group from the others and thus make it possible for disease diagnosis. It has shown that texture images are easier to



be classified if the image features are derived from MDCT coefficients. Since MDCT is standard DCT applied on the same image with different resolutions, it is easy to be implemented. Our study shows that low resolution texture images could benefit not only the classification speed but also classification accuracy.

## **Chapter 2 Contour Based Image Structure (CBIS) and XML Image Structure**

### **2.1 Background**

The methods of 3D reconstruction from multiple images can be grouped into two categories. The first one focuses on the object surface only. Given a fixed scene, multiple images are taken by the camera with different perspectives. The camera and images are calibrated to obtain the coordinates of the pixels on the surface. Those pixels define the 3D surface of the object. In this approach, only very few images are needed to cover the 360 degree views of the object surface. The type of methods only maintains the pixels located on the surface and thus no extra information about the materials inside the surface can be told. Since there is no redundant information not related to the surface, this approach generally reaches a very good 3D surface quality in terms of high resolution.

The second method is based on taking the cross-section images of a 3D object. Those parallel image slices are piled up to make an image stack. Image stacks can be MRI, CT, and confocal microscopic images. Unlike the first approach which focuses on the object surface only, image stacks provide more information about the object, not only about the surface but also about the tissues inside the surface. The object surface of a tissue is normally represented by those pixels located on the boundaries of the 2D image slices. If we only concern about reconstructing the object surface, this may not be a good approach comparing with the first one we mentioned since too many redundant information inside object surface will be useless and the derived 3D surface may have a less resolution. For example, if the resolution of the 2D images used in the first method is 1000 by 1000

pixels, to reach the exact same resolution of the 3D surface derived by the first method, a total number of 1000 image slices should be provided in the image stack used by the second method. This is normally unapproachable. The benefits by using image stacks are reflected to many aspects. It provides the whole volume data of the 3D objects. It makes it possible for users to look inside the 3D objects. With the 3D volume data, it is possible to generate a new cross-section image from arbitrary angles. Medical image analysis based on MRI and CT has been widely adopted and 2D image slices have been defined by standard format such as DICOM.

In our study, we define a new contour based image structure. The purposes of developing this new image structure are:

- 1) Converting a raw image data set into a standard xml structure.
- 2) Making it convenient for image analysis and partial retrieval.

Mapping image stacks into CBIS involves two stages: separating stage and grouping stage. Separating stage refers to the process of segmentation on a 2D image slice. In this stage, an optimal threshold is first chosen and applied to binarize the 2D image. Then an 8-connective tracer is used to outline the object contours. Grouping stage refers to the process of linking the objects on adjacent image slices which are considered belonging to the same 3D component in the whole image stack. In this stage, shape matching is applied to the contours on adjacent slices. Given two contours on adjacent slices, the matching decision is determined by a Fuzzy Contour Matching model. We also define a size verification factor which reflects the contour depth continuity.

The results of the above stages are recorded in an xml file. In the xml file, each node corresponds to a segmented object in the 2D image. Object features are represented as

elements of the node, including *layer*, *centroid*, *link*, and etc. Layer element indicates the slice on which the object is located. Centroid element indicates the position of the object in the slice. Link element indicates the matching object on its neighbor images. These three elements determine the overall topological tree structure of the image stack.

## 2.2 Separating Stage

Image segmentation methods are used to decompose an image into several parts or objects. These methods may be categorized into statistical classification, region growing, and boundary methods (Drebin *et al.*, 1988; Boskovitz and Guterman, 2002; Carlbom *et al.*, 1994). Several methods can be effectively used to detect edges (Frei and Chen, 1977; Canny, 1986). The main problem with these methods is the lack of continuity of edges, which requires post-processing to link the broken edges. The linking algorithms may introduce unnecessary ambiguity as well as linking noisy data. In CBIS, we automatically segment the object by their boundaries (Belkasim *et al.*, 2004). The optimum automatic thresholding procedure is combined with edge detection to produce continuously connected object border and leads fully segmented image.

Given an image with and a threshold  $\mathbf{T}$ , we define:

$$NPC(T) = \sum_{i=1}^{NC(T)} n_i(T) \quad (2.1)$$

where  $NC(T)$  is the number of contours obtained with  $\mathbf{T}$ . and  $n_i(T)$  is the number of pixels in the  $i$ th contour. Then the optimum threshold  $t$  is the one which maximizes the following formula:

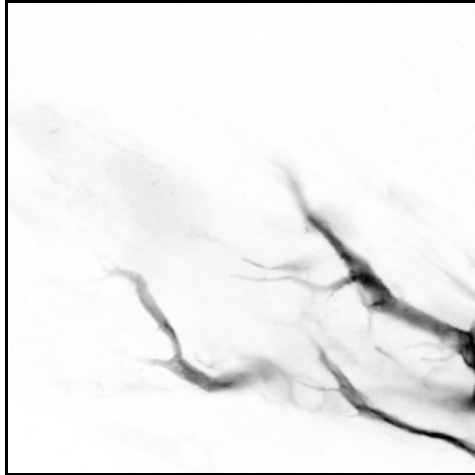
$$PMT(t) = \frac{NPC(t)}{NC(t)} \quad (2.2)$$

From the above formulas we know that the optimum threshold is trying to balance the total number of pixels on the object contours and the average length of the contour. The optimum threshold makes the maximum value of the average contour length.

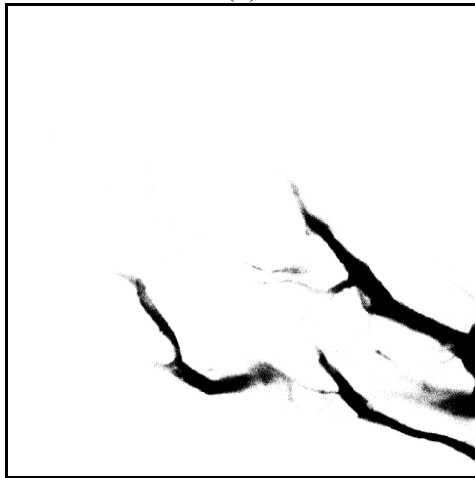
Linking boundary pixels using the optimum automatic threshold can avoid the ambiguity associated with gray level edge detection methods. Object contours are detected using binary images. An 8-connective path template is used to link contour pixels.

The border tracking algorithm records coordinates of boundary pixels in clockwise direction and terminates when it returns to the starting point. The algorithm also calculates the contour parameters such as contour length, inner area, and object centroid. Each contour will be stored as a node in an xml file and all parameters are linked to their corresponding nodes. The algorithm is implemented to track the boundaries of crayfish neuron slices which are shown in Figure 2.1.

We also define a threshold of contour size to remove the small contours, which may result from the noise. For example, in Figure 2.2b, many tiny contours have been generated by the noise pixels in the image of Figure 2.2a. Figure 2.2c shows the result after removing the noisy contours based on contour length threshold. This threshold is proven to be very effective in removing noise. Removing noise is also critical to surface rendering.



(a)

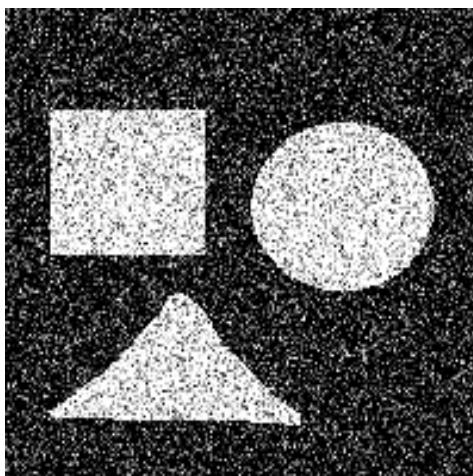


(b)

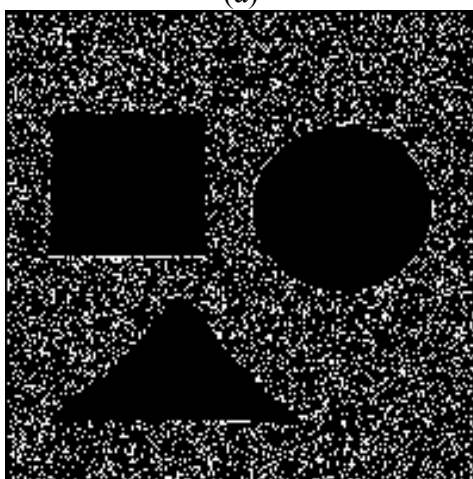


(c)

Figure 2.1 (a) Confocal microscopic image slice of crayfish neuron (b) After applying optimal threshold (c) Image contours



(a)



(b)



(c)

Figure 2.2 (a) Original image with noise (b) Image after applying contour segmentation  
(c) Removing noise using minimum contour size threshold

Optimum threshold method is applicable to images where the intensities and backgrounds of the objects are different from each other. In some cases, when images have inhomogeneous backgrounds, it is impossible to use a single threshold to separate the objects from their backgrounds. To solve the problem, we design a phase congruency method to highlight the object boundaries before the segmentation process. Phase congruency method will be discussed in the later chapter.

### **2.3 Grouping Stage**

In the segmentation stage, image objects in each slice are segmented. The separated objects in the same image slice are considered belonging to different 3D components. Figure 2.3 shows the contour objects derived from two crayfish neuron confocal image slices. To implement the 3D component retrieval from the whole image stack, all the contour objects in the different slices which belong to the same 3D component have to be grouped together. For example, the two objects pointed out in Figure 2.3b should be linked together in the 3D structure.

Contour matching is applied on two adjacent slices. This step is used to link contours on adjacent slices. A link will be created between any two matched contours in our data structure. Contours on adjacent slices are linked through the leading pixels.

In most practical cases we can make additional assumptions based on the fact that adjacent slices are very close and the contour shape changes gradually and continuously. Since all parameters of contours have been extracted in the segmentation step, we will be able to use those parameters directly in the matching task. It is hard to decide which contour feature is the dominating parameter to link the objects. We define fuzzy rules to help us make the matching decision by using multiple contour features. The fuzzy system



takes object intensity, overlapped area of two contours, and principal axis into the consideration. The fuzzy contour matching system will be discussed in the later chapter.

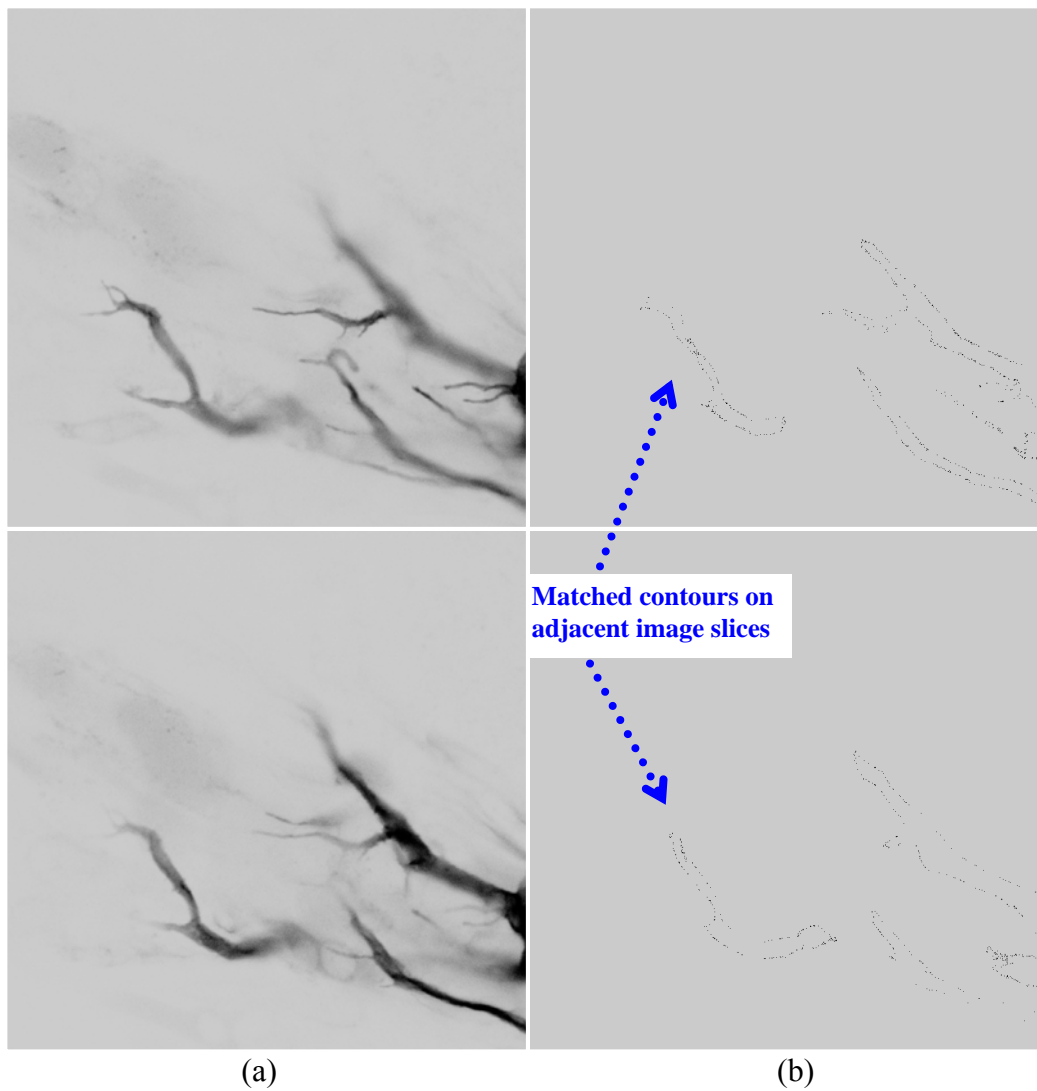


Figure 2.3 (a) Original confocal microscopic image slices of crayfish neuron (b) Enhanced contour images generated from the xml structure

The extracted object contours are connected to each other to form a contour tree structure. This process is divided into three main steps:

- 1) Extract all object contours within each 2D slice.
- 2) Label contours using their leading pixels by applying left-right, top-down scanning.
- 3) Repeat 1 and 2 for all slices.

The tree structure consists of two sub trees:

- 1) x-y-slice tree
- 2) z object tree

The x-y-slice tree is constructed based on linking all objects on the same slice as in (Belkasim *et al.*, 2004). The z-tree is based on matching and linking contours that belong to the same object on each slice.

Size verification of the two successive contours is used to verify the contour depth continuity. We define a size continuity factor  $\lambda$ . This factor depends on z-direction resolution. A high resolution on z-direction tells us same object shapes change slowly between two slices which implies a small value of  $\lambda$ . A discontinuous contour or sudden reduction in contour size implies large value of  $\lambda$ . This process requires the analysis of each three successive contours. For example, the contours in Figure 2.4 show a large value of  $\lambda$  between slice 1 and 2, which is greater than  $\lambda$  between slice 1 and 3. This contradiction can be used to claim that the data in slice 2 are either corrupted or missing. We may substitute the contours on slice 2 with the contours on its following slice which is slice 3.

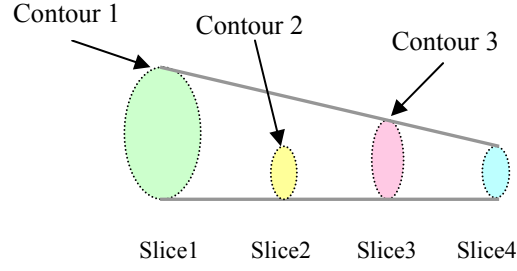


Figure 2.4 Size verification of the two successive contours

## 2.4 Contour XML Structure

After separating and grouping stage, raw image data will be able to be represented by a contour XML structure. This can be illustrated by Figure 2.5. In Figure 2.5, serial images are aligned in X, Y and Z axis, where Z-axis represents stack direction. On the X-Y plane, a 2D tree structure represents the object contours of the same image slice. For example, in Figure 2.5a, each 2D image slice is described by its four object contours. Based on the 2D structure, the related contours on neighboring slices are linked to form the 3D structure which is shown in Figure 2.5b.

For each image stack, we use an xml file to represent the whole volume data set. The benefit of using this data structure is that the volume data set is not represented as pixels, but contours instead. The new data format will be used not only to identically reconstruct the segmented images but also provide additional features such as contour sizes, moments, and contour relationships between two adjacent slices. Moreover, the required storage space for the image stack has been significantly decreased. Table 2.1 shows the storage sizes of contour-based xml files. We can see, the larger thresholds we choose, the smaller storage size we need.

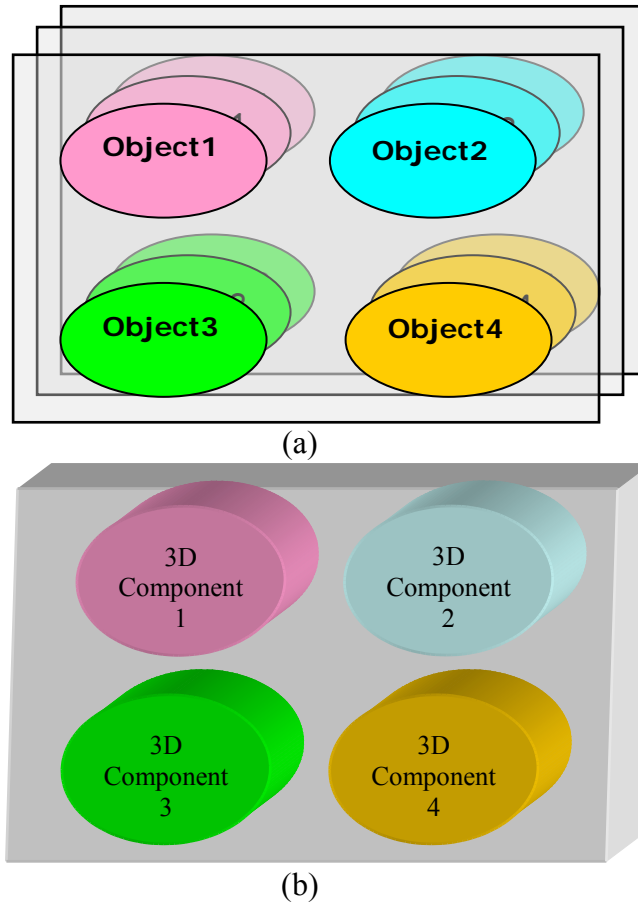


Figure 2.5 (a) Example of 3 adjacent slices (b) Contour data structure for (a)

Table 2.1 New storage size for an image stack using contour data structure. Original image stack consists of 20 crayfish neuron confocal microscope (2048 x 2048)

<b>Thresholding Values</b>	<b>Without Contour Size Filter</b>	<b>With Contour Size Filter</b>	
		<b>Filter 500</b>	<b>Filter 1000</b>
20	109	6.00	5.30
40	36	3.22	2.86
60	17	1.95	1.58
80	10	1.31	1.08

For each contour node in the xml structure, we use the fuzzy contour matching system to locate the object contours on the lower layer slice. The structure is updated by adding a *Link* element in the contour node if a matched contour is found. After this process, the related contours are then grouped to form a solid three dimensional object. A tree structure model is formed where each node in the tree is a representative of a segmented object and each edge in the tree is a representative of contour matching between two objects on adjacent slices. Figure 2.6 shows how a contour object is represented in the xml structure:

```

...
<Object>
  <Image_Name>01\lgaff049a01002.tif</Image_Name>
  <Size>2048 2048</Size>
  <Layer>3</Layer>
  <Intensity>74</Intensity>
  <Link>(1516 2020)</Link>
  <Centroid>1504 2032</Centroid>
  <Area>1762</Area>
  <Location>(1466 2033)</Location>
  <Contour_Length>546</Contour_Length>
  <Contour>
    (1466,2033)(1466,2034).....
  </Contour>
</Object>
...

```

Figure 2.6 Contour objects in the xml structure

## 2.5 3D Reconstruction and Partial Retrieval

In the xml structure, each object corresponds to a contour. Contour element records the boundary pixels in the raw image slice which determine the contour shape. It will be used for the reconstruction of the solid contour object and partial 2D image analysis. Contour features are represented by the object elements of *Contour\_Length*, *Area*,

*Moment*, *Centroid*, and etc. They are calculated once and can be easily extracted and repeatedly used. More features of contours can be added to the structure conveniently by simply defining more elements in the structure. The elements of *Centroid*, *Link*, and *Layer* in the xml structure determine the overall topology of the 3D image structure.

Based on the xml contour structure, a tree-like structure has been successfully built. In this structure, contours are defined by their centroid coordinates and depths in the tree. Matched contours on adjacent layers are linked. There are isolated nodes in this structure. In most cases, isolated nodes have small contour sizes and are caused by image noise or a lower threshold at the image preprocessing stage. To solve this problem, we apply local optimum thresholding in the isolated contour areas. A high local threshold is helpful to remove the unwanted noisy contours and a lower local threshold value may be helpful to recover broken edges between separated contours which could belong to a same object. After obtaining the xml tree structure, we use a new contour length filter, which is larger than the filter used in the separating stage, to remove the isolated small size contours which may be generated by noises.

From the xml contour structure, we can reconstruct the solid objects in 2D image slices by first drawing the object boundary according to each contour element and then filling the pixels inside contour boundary using a constant grey level or their individual grey level in the original image slices. By applying this process to all the contours which have the same layer values, we obtain all the segmented objects in the same image slice.

There are two basic approaches for 3D reconstruction from an image stack: surface rendering and volume rendering. Volume rendering assumes that data are available as 3D grids. In surface rendering, volumetric data are converted into geometric primitives first

and these primitives are then rendered for display (Meyers and Skinner, 1992; Drebin *et al.*, 1988). It is shown if a set of closed contours determines the surfaces of the objects to be reconstructed, then a surfaced-based approach will be preferred (Meyers and Skinner, 1992). We use isosurface-rendering technique to create the 3D objects from the contour structure.

We applied the contour based tree structure and the fuzzy contour matching system on the image stacks which consist of crayfish neuron confocal microscope images. Figure 2.7a represents an end branch of a crayfish neuron and Figure 2.7b represents the middle part of the same crayfish neuron. Each slice is saved as 2048 by 2048 pixels with TIFF format. The distance between two consecutive slices is  $2.1\mu m$ .

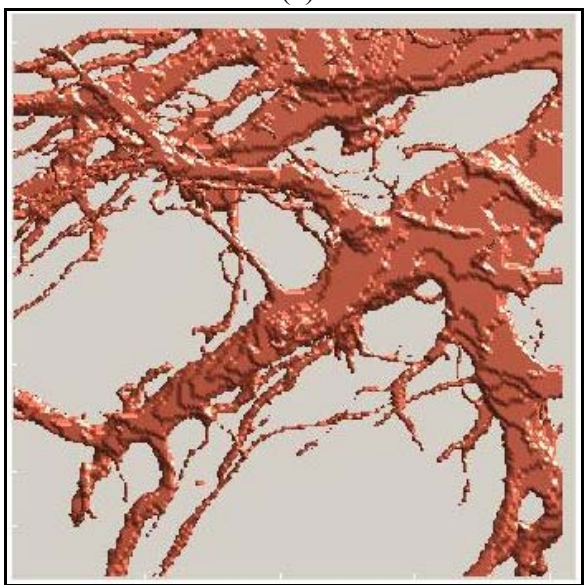
From Figure 2.7 we can see this 3D model successfully outlines the main spatial structure of the crayfish neuron. To testify the accuracy of the results, we reconstruct the image slices from the xml structure and compare them with the original image slices. We notice that there are very few neuron branches missed in the xml structure and all of the missed branches are very small objects. These errors are brought from two resources.

- 1) In some cases, the boundaries of the objects are hard to be outlined since their intensities are very close to the surrounding area and the optimal thresholding algorithm was not able to use a fixed threshold to distinguish the pixels on the boundaries from the pixels located in the background.
- 2) Since the microscopic images are obtained by virtually cutting the 3D object slice by slice, it is possible that some portions of the 3D components are isolated from the main object when they are displayed in the 2D images. When such isolated objects are small enough, they may be eliminated as noises by the contour length

filter. 3D reconstruction based on contour xml structure is much easier than the one based on the raw image stack since the whole image stack is represented by a single xml file.



(a)



(b)

Figure 2.7 3D model of a crayfish neuron confocal image stack



In biological research, instead of requiring the whole 3D model, researchers may be interested in the local 3D structures of a specimen. For example, in Figure 2.3 we know the two highlighted contours belong to the same neuron branch. It may be useful for certain applications to retrieve the 3D branch in which the 2D contours reside.

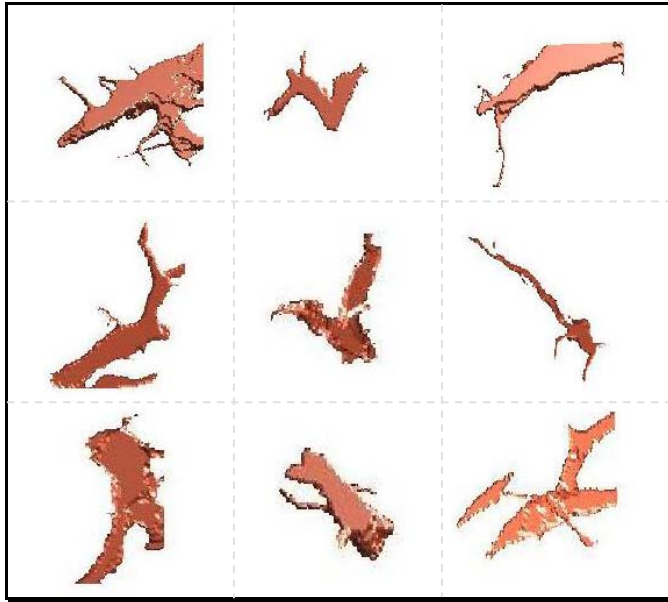
To fulfill the 3D partial retrieval, we use the xml tree structure described above for 3D content based component retrieval. Since we use contours as basic units to represent the 3D volumetric data, the corresponding 3D object can be divided into several components, each of which is made of a group of connected contours. Given an arbitrary pixel in a 2D image, we can easily identify its corresponding contour. By applying a contour depth-first search in the tree structure, we can again easily find the 3D subcomponent in which the contour resides. Our experimental data has shown that 3D component retrieval from the contour xml structure is extremely faster than retrieval from the original image slices.

Using the tree structure, image querying schema can be extended by defining various searching rules in the xml contour structure.

Figure 2.8 shows the result of the 3D partial retrieval by querying the xml tree structure based on a single contour and all those contours which can be reached from the start contour. Figure 2.8a and Figure 2.8b are the sub-3D components extracted from the 3D objects shown in Figure2.7a and Figure2.7b respectively.



(a)



(b)

Figure 2.8 3D components of a crayfish neuron branch

## Chapter 3 Parallel Implementation

Even though the original image data have been simplified using the xml structure, getting the contour features and matching related contours on adjacent slices still need to perform heavy computations. For example, getting the contour centroid of an object needs to calculate the moments up to the second order. This amounts to  $\Theta(M \times N)$  time complexity, where  $M \times N$  is the size of the 2D image. In our experiment, we need to handle a huge three-dimensional array with the size of  $(2048 \times 2048 \times 20)$ , which consumes a large amount of memory space. Even for matching a very small contour to its neighbor slices, we still need to create a template of  $(M \times N)$  pixels to use it in the matching process. This computing burden is even worse for images with high contour density. Thus, solving these problems on a high-performance computing system is essential to combat both excessive amounts of time and memory constraints existing on a single processor system (Pan *et al.*, 2000; Quinn, 2004).

### 3.1 Layer Parallelization Algorithm (LPA)

Since the whole image stack volume dataset has been stored in the contour based xml structure, individual processor can interact with the xml files efficiently without image loading operations and preprocessing. The contour structure makes the distribution of the contour matching tasks among multiple processors much simple. Figure 3.1 shows the layer parallelization algorithm using MPI.

1. All processors load the xml structure.
2. Each processor scans the whole structure and obtains its tasks (image slices) based on its rank and the layer numbers of the slices.
3. Each processor calculates the contour features and passes the calculation results to the root processor (rank 0).
4. Once receiving the result messages from all the other processors, the root processor updates the structure.
5. The root processor sends acknowledgement messages to all the non-root processors.
6. All processors load the updated structure.
7. Each processor scans the new updated structure and obtains its tasks (image slices) based on its rank and the layers number of the slices.
8. Each processor applies appropriated matching techniques to its assigned slices and records matched contours on its lower adjacent layer if found.
9. The non-root processors send all the contour matching information to the root processor.
10. The root processor updates the structure.

Figure 3.1 Layer parallelization algorithm (LPA)

In LPA, we minimize the communication between the root processor and the non-root processors. The tasks of contour feature calculation and contour matching are obtained by individual processors instead of being distributed by the root processor. The results from individual processors are stored in one large string and passed only once to the root processor. To avoid unbalanced task loading which may drag down the whole program performance, computing tasks should be distributed evenly. In this algorithm, image slices are chunked by the number of the processors. That is, the processors extract the contours from the xml structure according to their ranks and the layer values of the slices. Therefore, contours on the same slice will be handled by the same processor.

### 3.2 Contour Parallelization Algorithm (CPA)

Another approach based on contour index is described in Figure 3.2.

1. All processors load the xml image structure.
2. Each processor scans the whole xml image structure and obtains its tasks (contours) based on its rank and the contour index.
3. Each processor calculates the contour features and passes the calculation results to the root processor (rank 0).
4. Once receiving the result messages from all the other processors, the root processor updates the xml image structure.
5. The root processor sends acknowledgement messages to all the non-root processors.
6. All processors load the updated xml image structure.
7. Each processor scans the new updated xml image structure and obtains its tasks (contours) based on its rank and the contour index.
8. Each processor applies appropriated matching techniques to its assigned slices and records matched contours on its lower adjacent layer if found.
9. The non-root processors send all the contour matching information to the root processor.
10. The root processor updates the xml image structure.

Figure 3.2 Contour parallelization algorithm (CPA)

CPA is similar to LPA except that an object contour is used as the basic task element. By using the object contour as the basic element, job tasks can be simply distributed sequentially and iteratively among the processors. The whole contour list is chunked by the number of the processors. In this approach, contours are extracted from the xml structure according to the processors' ranks and the contour sequence numbers in the

structure. The experimental results, which will be discussed in 3.4, show the CPA has a better overall performance than LPA.

### 3.3 Partial Retrieval Parallelization Algorithm (PRPA)

3D image retrieval could be the bottleneck of computing performance when large amounts of queries involve 3D image analysis. The contour data structure is convenient for parallelizing image component retrieval since in the xml structure, each contour has a link element indicating the matched contour on its neighboring slice, and a layer element indicating its slice level. Given a retrieval task, a processor thus can travel the xml structure efficiently through *Layer* and *Link* elements to form the 3D component without communicating with other processors. Distributing retrieval tasks among multiple processors is straightforward based on the contour xml structure. Figure 3.3 shows the partial contour parallel algorithm (PRPA) using MPI.

1. All processors load xml image structure.
2. The root processor sends messages to the non-root processors for retrieval tasks.
3. Each processor fulfills its retrieval tasks and creates a sub- xml image structure for each retrieval task.
4. The non-root processors send acknowledgement messages to the root processor.
5. The root processor updates the structure after receiving acknowledgement messages from the other processors.

Figure 3.3 Partial retrieval parallelization algorithm (PRPA)

Since a retrieval result is a sub-component of the whole xml structure, the processors can save the retrieval result in a small xml file. To minimize the message passing between the root and non-root processors, the name of the newly created xml file is automatically determined by the query information. For example, if we want to find the 3D component in which a particular pixel resides, we may use the information of the pixel coordinate and the slice layer as the name of the result xml file. In this way, the root processor has the knowledge of where to locate the retrieval results at the time it distributes retrieval tasks. Thus, the retrieval results need not to be passed from the non-root processors to the root processor. Our experimental results show a substantial speed up for this implementation.

### **3.4 Experimental Results on Parallel Implementation**

To investigate the validity of our assumptions on the contour parallel implementations and to verify the gain in processing speed as well as storage efficiency we used several 3-D image data slices. All the experiments are run on a 24-processor hypercube-based shared-memory NUMA machine from Silicon Graphics Inc (Origin, 2000).

We apply layer parallelization algorithm (LPA) on the image stack. In this experiment, the entire image slice is used as the working unit of each job task. Table 3.1 shows the speedup for contour matching process using multiple processors.

Table 3.1 Speedup of LPA implementation original image stack consists of 20 crayfish neuron confocal microscope ( $2048 \times 2048$ )

<i># of Pros</i>	<i>Speedup</i>	<i># of Pros</i>	<i>Speedup</i>
1	1.00	13	5.63
2	1.76	14	5.80
3	2.78	15	5.71
4	2.72	16	5.74
5	2.94	17	5.81
6	3.79	18	5.84
7	4.16	19	5.84
8	4.48	20	5.92
9	4.89	21	5.89
10	4.75	22	5.91
11	4.79	23	5.86
12	5.37	24	5.89

This table shows a considerable speedup when the number of processors is less than 8. But beyond 8, it shows less significant speedup. By studying the original image stacks, we found that contour density varies on different layers. Contour density is high for the slices in the middle of the stack. Images on the top or bottom layers may have very few or no contours at all. Thus, using the entire image slice as the working unit may cause an unbalanced loading. This shortcoming is even more obvious when the number of processors is more than the number of image slices. In this case, some processors will be idle. From Table 3.1, we can find that the speedups have little difference when the processor number is larger than 16. This can be explained by studying the xml structure, in which only 15 out of the total 20 image slices include image contours. This implies 9 processors are idle during the process.



We also apply contour parallelization algorithm (CPA) on the same image stack. In this experiment, the object contour is used as the working unit of each job task. Table 3.2 summarizes the speedup for the same task using the LPA algorithm.

Table 3.2 Speedup of CPA implementation. Original image stack consists of 20 crayfish neuron confocal microscope ( $2048 \times 2048$ )

<b><i># of Pros</i></b>	<b><i>Speedup</i></b>	<b><i># of Pros</i></b>	<b><i>Speedup</i></b>
1	1.00	13	7.00
2	1.85	14	7.51
3	2.34	15	7.06
4	3.36	16	8.36
5	3.38	17	7.09
6	4.39	18	8.80
7	4.32	19	8.38
8	5.97	20	8.95
9	6.12	21	7.25
10	5.18	22	9.40
11	6.07	23	9.24
12	6.32	24	8.35

To test the partial retrieval parallel algorithm, we apply the algorithm on four sets of 3D component retrieval tasks. For each retrieval task, we randomly pick up a pixel from an image slice. The program first locates the contour object which encloses that pixel, and then generates the 3D component which contains the contour. Table 3.3 summarizes the speedup of the retrieval tasks with the size of 30, 60, 90, and 120. Figure 3.5 compares the corresponding speedups of the four retrieval sets.

Table 3.3 Speedup of CPA implementation for multiple 3D component retrieval

<b># of Pros</b>	<b>30 Components</b>	<b>60 Components</b>	<b>90 Components</b>	<b>120 Components</b>
1	1.00	1.00	1.00	1.00
2	1.75	1.87	1.92	1.93
3	2.37	2.67	2.75	2.77
4	2.94	3.44	3.65	3.64
5	3.34	4.00	4.29	4.34
6	3.71	4.55	4.92	5.02
7	4.05	5.11	5.83	5.90
8	4.29	5.51	6.46	6.46
9	4.56	6.00	6.87	7.09
10	4.82	6.40	7.28	7.48
11	4.93	7.00	8.10	8.42
12	5.28	7.17	8.32	8.74
13	5.32	7.50	8.95	9.40
14	5.53	7.81	9.76	9.85
15	5.68	8.06	9.65	10.01
16	5.72	8.14	10.25	11.08
17	5.70	8.39	10.56	11.25
18	5.76	8.76	10.92	11.28
19	5.93	8.80	11.49	12.06
20	6.28	9.39	11.48	12.14
21	6.29	9.45	12.44	12.64
22	6.31	9.45	11.97	13.34
23	6.31	9.55	12.61	13.25
24	6.23	9.73	12.27	13.31

The results show the code is scalable up to 24 processors for the large set of 3D component retrieval tasks. We can see that the overall speedup increases as the amount of retrieval task increases. Note that in this program, there are only two short messages passing between the root processor and non-root processors during the whole process. The communication cost is very low. The time for the root processor to update xml structure is also very short comparing with a single 3D retrieval task. Thus the loading

balance is the key for the overall performance. We know that the number of the enclosing contours of a 3D component is determined by the length of a path in the contour structure tree. The retrieval task for the pixels located on a long path will cost more time than the task for the pixels located on a short path. It may happen that some processors are assigned more pixels on the long-path while other processors are assigned more pixels on the short-path. This will cause unbalanced loading. The chance of unbalanced loading is minimized if the set of retrieval tasks is large. Thus, the average speedup will be optimal for a large amount of retrieval task. This is reflected in Figure 3.5. The parallel retrieval process with the size of 120 tasks has the best speedup while the speedup for 30 tasks is the worst.

Images produced from biological specimens do not lend themselves easily to direct parallelization due to objects or parts of objects having widely varying sizes and brightness, high noise, and background staining. Our experimental results indicate that, the contour structure is suitable for parallel implementation for both 3D reconstruction and partial 3D image retrieval. Besides its simplicity to be parallelized, the image contour data structure can also be used to minimize the noise and object overlapping problems existing in many biological images.

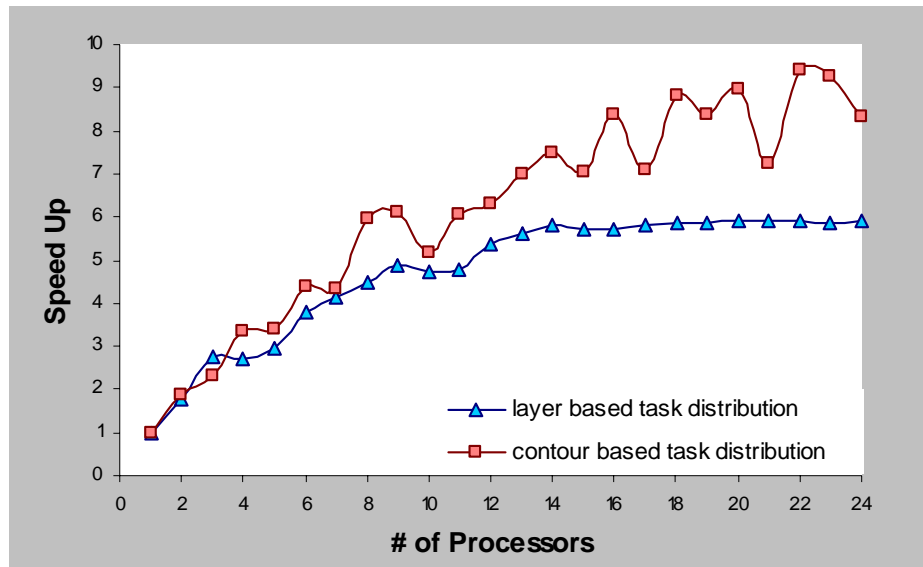


Figure 3.4 Speedups of the LPA and CPA

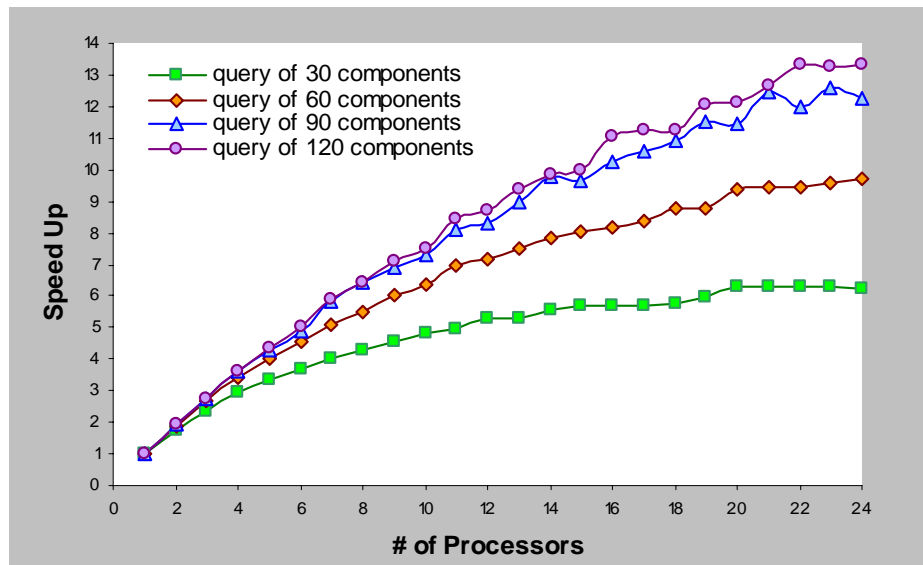


Figure 3.5 Parallel partial image retrieval speedups

## Chapter 4 Segmentation Using Phase Congruency

In CBIS, optimal thresholding is applied to obtain the binary image. It has been shown, in many cases, segmented images from a fixed global threshold either include unwanted noise or miss valuable information. This problem is more apparent when the objects are located in an uneven illumination background (Chan *et al.*, 1998). For example, due to the unbalanced illuminations, high pass threshold may filter out the objects which are located in the dark background areas. If we decrease the threshold, more noises may be included in the segmented images. To solve this problem, we introduce a new technique based on the phase congruency of the Fourier transform components. This method is based on the fact that high phase congruency corresponds to the image features such as edges and corners. Since phase congruency is a dimensionless quantity that is invariant to changes in image brightness or contrast (Kovesi, 1999), it can be used to track the object boundaries in the images with an uneven illumination. Given an image, we first generate its 2D phase congruency matrix which represents the phase congruency for all the pixels and then highlight the pixels which have high phase congruency. After that, we apply optimal thresholding to the enhanced image. In this way, a high-pass threshold filter will be able to detect the object boundaries which have been highlighted in the dim background. To remove the noise, we design two filters, one based on the length of the object contour and the other based on comparing the intensity of the pixels in the phase congruency image with the average intensity of its neighborhood pixels in the original image.

#### 4.1 The Problem of Optimal Threshold Method

As mentioned in Chapter 2, we obtain binary data from the image by using optimum automatic thresholding methods as described in (Belkasim *et al.*, 2004) and then identify the object contour of the binary image with a border follower algorithm that uses an 8-connectivity path template to link contour pixels. This method is capable of finding the main objects from background. However, in some cases, this approach fails to detect the detailed image features of an object which is located in the inhomogeneous background. This problem can be shown in the following example. We apply the optimum shareholding algorithm to track the object boundaries of a crayfish neuron filled with a fluorescent tracer as displayed in Figure 4.1a. Segmented objects are represented by object boundary contours in Figure 4.1b. Even we can always find a threshold for a given image according to the algorithm, we still need to verify if this threshold is capable of extracting the object contours from a 2D image. The following questions need to be addressed:

- 1) Can the optimal threshold separate objects efficiently with all the objects selected and without noise included?
- 2) Are the object contours connected without any broken points?

The answers to the questions are critical to the CBIS. If too much noisy pixels included in the binarized image, then fault contours will be created and saved in the xml structure. Those contours will affect the accuracy of 3D reconstruction process and partial retrieval. The noisy pixels may also link the object contours which are not belong to the same 3D subcomponent. One the other hand, improper threshold can break an

object into several pieces. This problem can be illustrated by observing Figure 4.1b and Figure 4.1c.

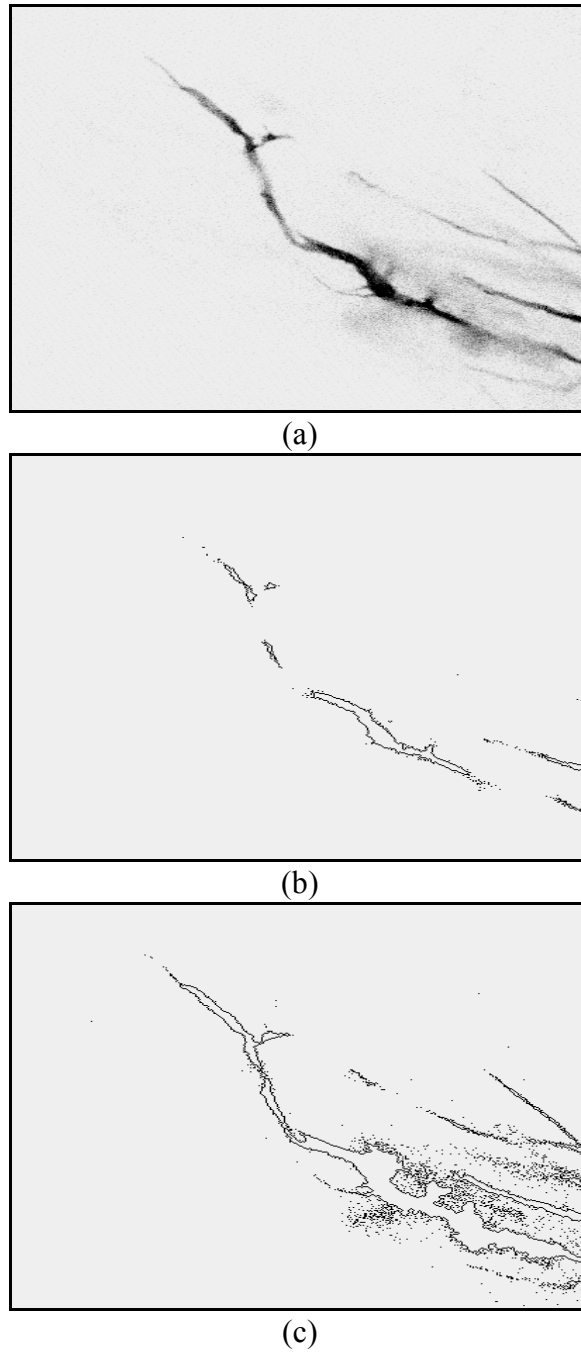


Figure 4.1 (a) Confocal microscopic image slice of crayfish neuron (b) After applying optimal thresholding (c) After applying decreased threshold

From Figure 4.1b we can see the object details in Figure 4.1a have been filtered out because the overall brightness in that area is close to the background. To include the missed information, we have to decrease the threshold value. Figure 4.1c shows the segmented image with a low threshold. It is clearly observed that more noise has been included in Figure 4.1c when the threshold is adjusted to be able to detect the missed objects in Figure 4.1b. This is inevitable when a global threshold is applied and the absolute intensity of the missing object is close to the intensity of the noise or the background. No matter what threshold we choose, the binarized image will either miss out some valuable information or bring unnecessary noise. If an image contains a strong illumination gradient, no global threshold can successfully segment the raw image directly.

Many approaches have been studied to deal with this situation. One approach is adaptive thresholding which separate desirable foreground image objects from the background based on the difference in pixel intensities of each region. It divides the image into several sub images and applies different optimal thresholds for each sub-area. But how to subdivide the image efficiently still remains a challenging problem.

In the segmentation process above, objects are traced along their boundaries, in which we are interested. This implies: for an image with uneven illumination, if we can distinguish an object's boundary from its local surrounding area, we will be able to first highlight all the boundaries of the objects which are located in various intensity backgrounds and then apply the optimal thresholding method for image segmentation. That is, if the object boundaries can be enhanced, our optimal thresholding program will not be sensitive to the threshold value and the object contours will be connective with



less broken points. Object edges are one of the main image features and can be detected by many image feature extraction methods (Canny, 1996). Image segmentation using object contour provides us the opportunities of utilizing the well-studied image feature detection methods for segmentation.

Most of the edge detection methods are based on the changes of image spatial features. Object boundaries often cause sharp changes in brightness: a light object lies on a dim background, or a dark object lies on a light background. The boundaries can be estimated by taking derivatives of the image. For a discrete image, derivatives are naturally approximated by finite differences. For example, we might estimate a partial derivative of:

$$\frac{\partial f}{\partial x} = \lim_{\alpha \rightarrow 0} \frac{f(x + \alpha, y) - f(x, y)}{\alpha} \quad (4.1)$$

as a symmetric difference:

$$\frac{\partial f}{\partial x} \approx h_{i+1,j} - h_{i-1,j} \quad (4.2)$$

which is same as a convolution with the kernel of:

$$\sigma = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{Bmatrix} \quad (4.3)$$

Based on the above definition, Laplacian operator can be derived as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (4.4)$$

where

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \quad (4.5)$$

and

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y). \quad (4.6)$$

Thus,  $\nabla^2 f$  is equals to:

$$f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \quad (4.7)$$

which can be implemented by the mask of:

$$\sigma = \begin{Bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{Bmatrix} \quad (4.8)$$

By taking the diagonal direction, the mask can be rewritten as:

$$\sigma = \begin{Bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{Bmatrix} \quad (4.9)$$

Similarly, we can also use first derivatives for edge enhancement: the gradient operator. In image processing, first derivatives are implemented using the magnitude of the gradient. For function  $f(x, y)$ , the gradient of the function at  $(x, y)$  is the vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (4.10)$$

The magnitude of the vector is defined as:

$$\left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \quad (4.11)$$

which can be simplified as:  $|G_x| + |G_y|$ . Sobel edge detection kernel includes:

$$\begin{Bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{Bmatrix} \quad (4.12)$$

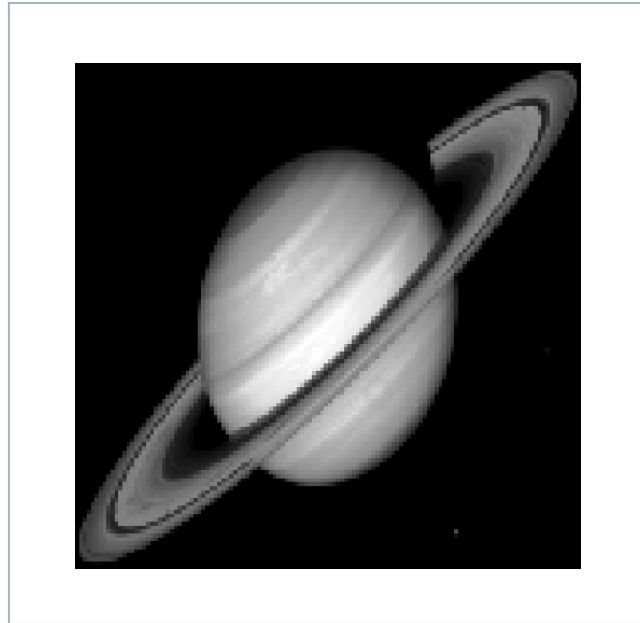
which reflects the derivative on y direction, and

$$\begin{Bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{Bmatrix} \quad (4.13)$$

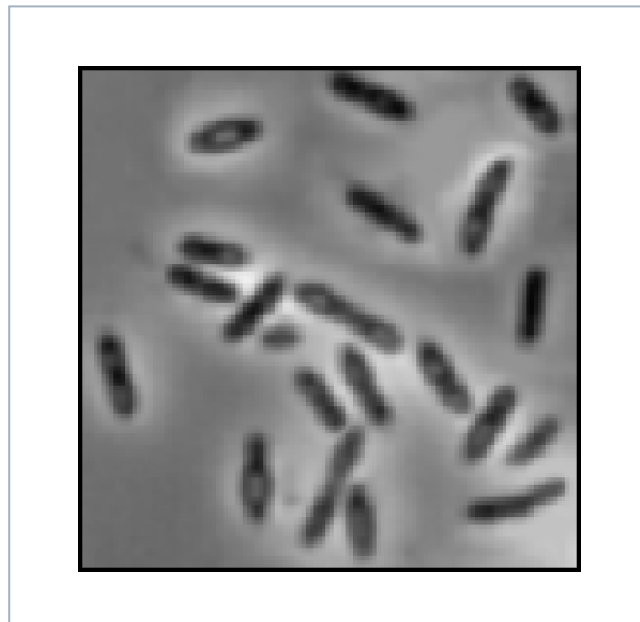
reflects the derivative on x direction. We use the Sobel operator to enhance the object boundaries before applying the optimum thresholding.

## 4.2 Phase Congruency for Edge Detection

Like Sobel operator, most of the edge detection techniques are based on gradient operators. These methods are sensitive to variations in image illumination, blurring, and magnification (Kovesi, 1999). Instead of processing image spatially, phase congruency method is capable of extracting image features using the phase and amplitude of the individual frequency components in an image. Phase congruency method is based on the phase congruency of the Fourier transform coefficients. Fourier transform components can be represented by its magnitude and phase. Phase carries out more image information. It can be seen in the following example. There are two images with the same size: Saturn and Bacteria shown in Figure 4.2.



(a)



(b)

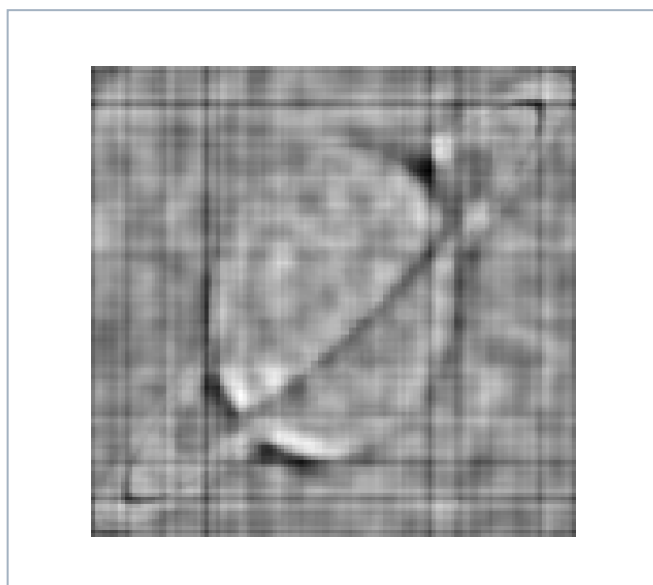
Figure 4.2 (a) Saturn (b) Bacteria

We take Fourier Transform for the both images. Then exchange magnitude information between the two images. And we then do the inverse Fourier Transform. The

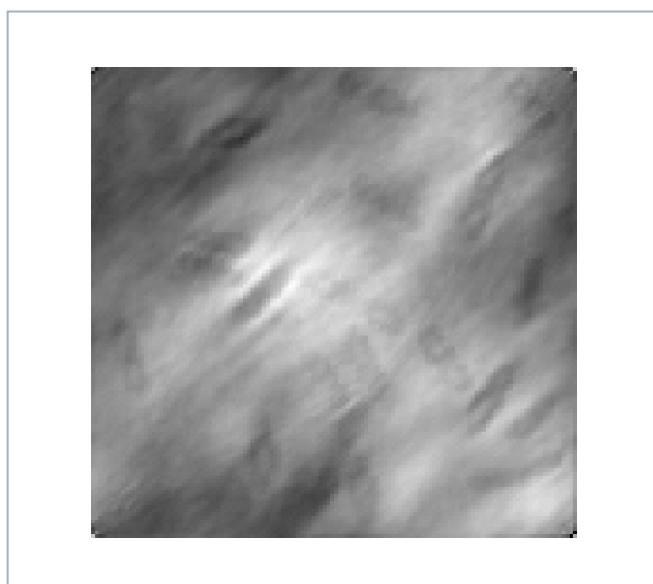
two newly created images are displayed in Figure 4.3. Figure 4.3a is created based on the phase information from Saturn and magnitude information from Bacteria. Figure 4.3b has the phase information from Bacteria and magnitude information from Saturn. It is easy to find out that the phase information brings the basic shape of the original image. But the magnitude information doesn't make any obvious contribution to the newly created images.

A local energy model for feature perception has been introduced in (Morrone *et al.*, 1986; Morrone and Owens, 1987). This frequency-based model is capable of performing calculations using the phase and amplitude of the individual frequency components in a signal. Based on this model, image features can be perceived by locating the points where the Fourier components of the image are maximal in phase.

The based idea of phase congruency method can be explained by looking at a one dimension signal. Suppose we have a signal which is represented by  $n$  discrete samples  $\{S_0, S_1, \dots, S_{n-1}\}$ . Applying Fourier transform to  $\{S_0, S_1, \dots, S_{n-1}\}$  we have the Fourier coefficients  $\{T_0, T_1, \dots, T_{n-1}\}$ .  $\{T_0, T_1, \dots, T_{n-1}\}$  can also be represented as  $\{R_0/\theta_0, R_1/\theta_1 \dots R_{n-1}/\theta_{n-1}\}$ , where  $R$  is the magnitude and  $\theta$  is the phase. According to inverse Fourier transform, when we need to reconstruct a sample say  $S_x$ , we need use  $\{R_0/\theta_0, R_1/\theta_1 \dots R_{n-1}/\theta_{n-1}\}$  to build another array of complex number  $\{R_0/\theta_{0-x}, R_1/\theta_{1-x} \dots R_{n-1}/\theta_{(n-1)-x}\}$  by changing the phase of each coefficient accordingly.  $S_x$  equals the sum of the elements in the array. The idea of the phase congruency is to measure how closely  $\theta_{0-x}, \theta_{1-x} \dots \theta_{(n-1)-x}$  are related to each other. If  $\theta_{0-x}, \theta_{1-x} \dots \theta_{(n-1)-x}$  are close enough, we can predict that  $S_x$  is a significant point. In 2D image, it can be a pixel on the edge or on the corner.



(a)



(b)

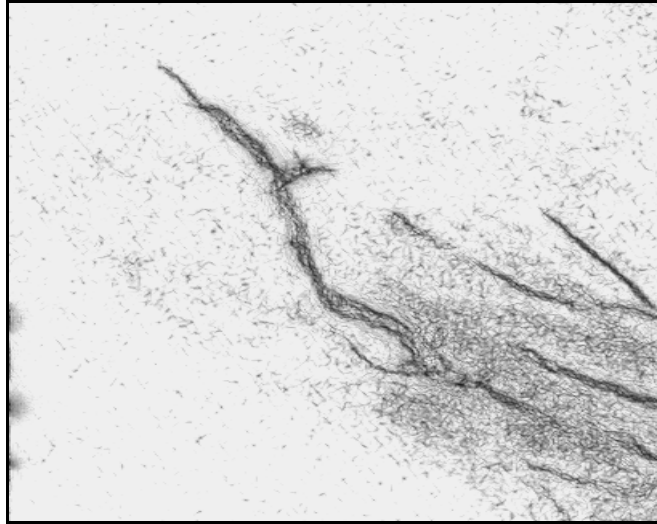
Figure 4.3 (a) Saturn (phase) + Bacteria (magnitude) (b) Bacteria (phase) + Saturn (magnitude)

The phase congruency function for 1D signal is defined as follows:

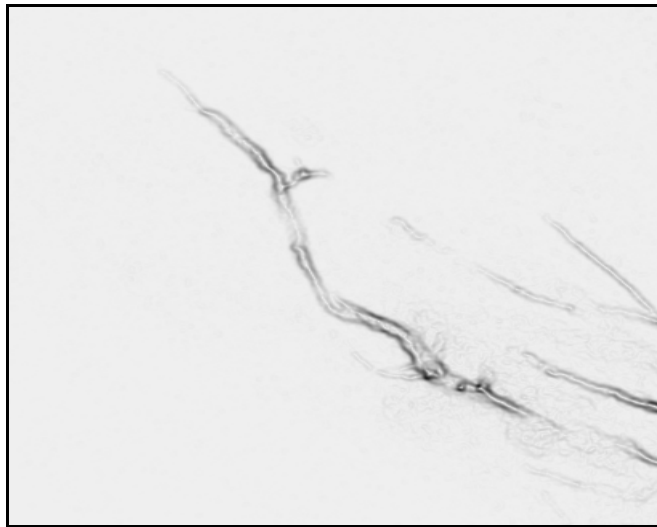
$$PC(x) = \frac{\sum_n A_n (\cos(\phi(x) - \bar{\phi}(x)))}{\sum_n A_n(x)} \quad (4.14)$$

where  $x$  represents the location of the signal,  $A_n$  represents the amplitude of the  $n$ th Fourier component,  $\phi(x)$  represents the local phase of the Fourier component at  $x$ , and  $\bar{\phi}(x)$  is the angle which maximizes the function. When  $PC(x)$  is equal to 1, the phase terms are all equal and the highest phase congruency occurs. A high phase congruency at  $x$  implies a significant feature at  $x$ ,  $PC(x)$  takes on a value between 0 and 1.

Phase congruency can be calculated via log Gabor wavelets. For a 2D image, by applying one dimensional analysis over several orientations and combining the results, a phase congruency image can be derived (Kovesi, 1999). Component value in the new phase congruency image represents the feature significance of corresponding pixels in original image. Figure 4.4a is the phase congruency image obtained from Figure 4.1a. It is easy to see that the boundaries of the neuron branches are all displayed prominently regardless of the variation of object intensities in the original image. Figure 4.4b is the result of applying Sobel edge detection method on the Figure 4.1a. Both phase congruency and Sobel methods successfully outline the object edges. Notice that the vague parts pointed in Figure 4.1a can be detected on Figure 4.4a and Figure 4.4b.



(a)



(b)

Figure 4.4 (a) Phase congruency image (b) Sobel edge detection image

Comparing Figure 4.4a with Figure 4.4b, even though phase congruency method creates more additional noises, the pointed area in Figure 4.1a is more apparent and distinguishable from the background. These object edges can be used to enhance the objects in the homogenous background.



### 4.3 Segmentation Using Phase Congruency

Since phase congruency is a dimensionless quantity that is invariant to image illuminations, we can use the phase congruency image to compensate object contours in the uneven illumination background which cannot be detected by the fixed threshold. The main idea is to overlap the phase congruency image with the original image and then apply optimal thresholding on the new created image. We modify the optimal threshold algorithms as follows:

- 1) Remove noise using contour length filter.
- 2) Smooth image by the average filter.
- 3) Obtain the phase congruency image.
- 4) Apply the edge filter on the phase congruency image.
- 5) Smooth the phase congruency image and apply the contour length filter.
- 6) Combine the phase congruency image with the original image.
- 7) Apply the optimal thresholding on the combined image.

Since the filters used in this algorithm depend on the object intensity of the processed image, to simplify our discussion, we assume that segmented objects have higher intensity than those of its local surrounding area. Segmentation for images with dark objects and bright backgrounds can be easily adjusted using the similar approach.

Like gradient operator, phase congruency function is sensitive to noise, especially for isolated noise with small size. To remove those noises, in step 1 of the algorithm, we first binarize the image with a low threshold and then extract all the objects by tracing their boundaries. We apply a contour length filter (LF) to remove the noise. That is, we remove the object which has a very small contour length from the image. In step 2, the

average filter is also helpful to remove the noise. In step 3, phase congruency image is obtained by applying log Gabor wavelets on the de-noised image.

Pixels which have high values in the phase congruency image correspond to high significance of the image features. In this segmentation process, we only focus on image features that represent the edges of image objects. Because the image objects are brighter than their surrounding background as we assumed, in most cases, we can conclude that a pixel on the object boundary has a higher intensity than the average intensity of its 8-neighborhood. Based on this assumption, in step 4, we design an edge filter to generate the true/false table to check whether the pixel in the phase congruency image represents the object boundary feature. Table 4.1 is the edge filter mask.

Table 4.1 Edge filter mask for object boundary feature

$F(-1,-1) = -1/8$	$F(-1,0) = -1/8$	$F(-1,1) = -1/8$
$F(0,-1) = -1/8$	$F(0,0) = 1$	$F(0,1) = -1/8$
$F(1,-1) = -1/8$	$F(1,0) = -1/8$	$F(1,1) = -1/8$

The edge filter works as follows. For an image  $g$  and its phase congruency image  $p$ , we first obtain the boundary true/false image  $b$  as:

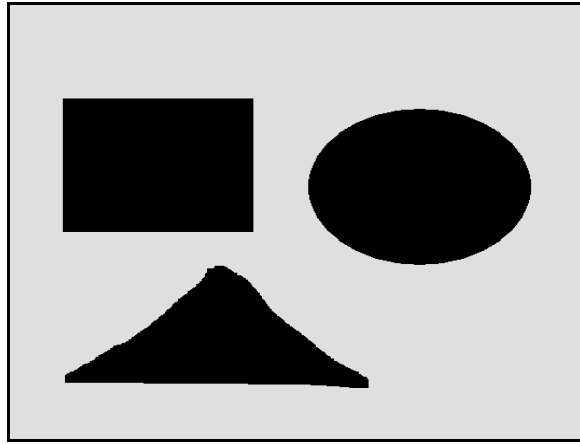
$$b(x, y) = \sum_{a=-1}^1 \sum_{b=-1}^1 g(x+a, y+b) f(a, b) \quad (4.15)$$

where  $f(a,b)$  is from Table 4.1. We then use image  $b$  to remove the components in phase congruency image  $p$  which are unrelated to object boundary feature by applying:

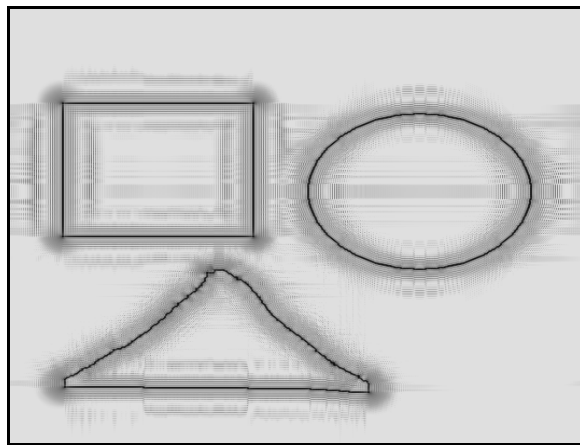
$$p(x, y) = 0 \quad \text{if} \quad b(x, y) \leq 0 \quad (4.16)$$

Phase congruency method will highlight all the pixels which have significant features in the images including those pixels not located on the object boundaries. Using edge filter, we can easily find out which pixel are the candidate pixel locating on the boundaries. Combing the phase congruency image and the candidate pixels, those highlighted pixels which do not represent object edge or corner will be able to remove from the phase congruency image. To see how the edge filters work, we create a simple image where object are easily distinguished by human eyes. Phase congruency method is first applied to the image and then edge filter is used to remove noisy pixels from the created phase congruency image. Figure 4.5 shows how this filter works.

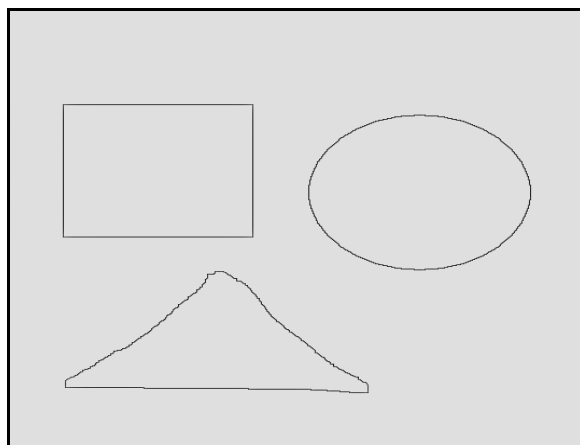
We can see that Figure 4.5b blurs around the object boundaries. This is because the phase congruency image highlights not only the pixels which are significant for the object edges but also others. Using edge filter, non-edge related features can be removed efficiently.



(a)



(b)



(c)

Figure 4.5 (a) Original image (b) Phase congruency image (c) After applying edge filter

In step 6, we combine the phase congruency image and original image by combining their intensities:

$$p(x,y)=k*p(x,y)+(1-k)*g(x,y) \quad (4.17)$$

where  $k \in [0,1]$ . We use a higher value of  $k$  when the image is highly inhomogeneous. In this case, the phase congruency image weights more in the final image since more compensation should be applied to the pixels located on object boundaries. On the other hand, if the original image can be directly segmented using a global threshold, the phase congruency image will weight less in the final image.

In the last step, optimal thresholding is applied to the combined image where the object boundaries have been enhanced. Figure 4.6 is the block diagram of the whole process.

We apply the algorithm on a confocal microscope crayfish neuron image slice which is displayed in Figure 4.1a. Figure 4.7a is the segmented image extracted from this process. The value of contour length filter is set to 50 and the factor of overlapping between the phase congruency image and the original image is set to 0.5, which averages the intensities of the pixels on the both images. To compare the phase congruency method with other edge detection methods in spatial domain, we repeat the same process on Figure 4.1a by using Sobel operator to highlight the object boundaries. The length filter and the overlap factor keep the same. Figure 4.7b shows the segmentation result by applying Sobel operator.

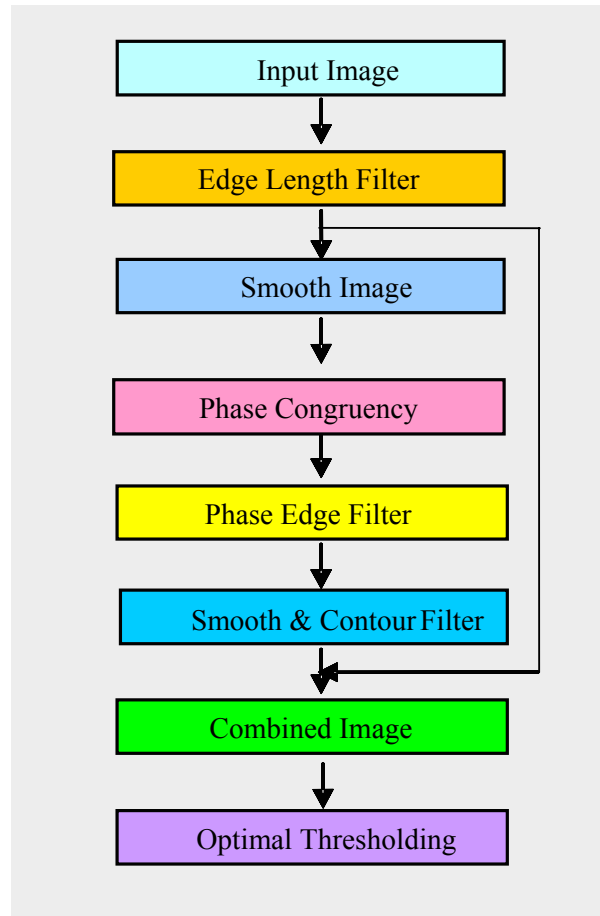
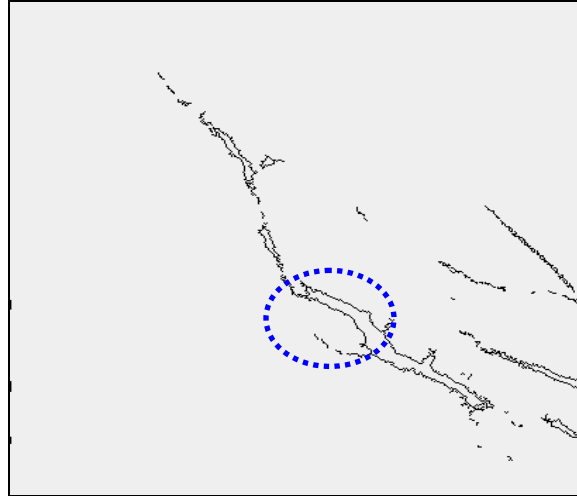
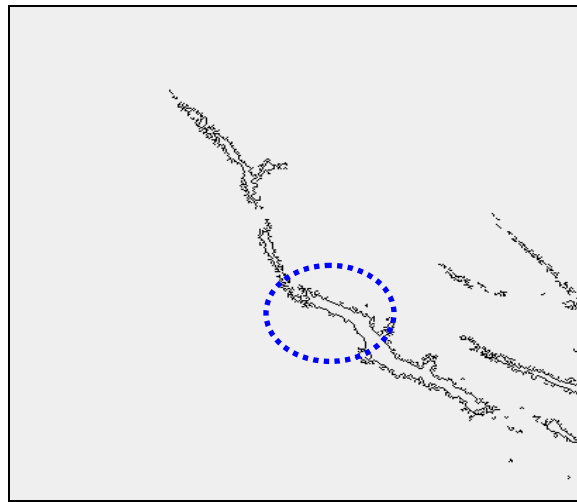


Figure 4.6 Block diagram of the algorithm

The both segmented images bring more detailed information than Figure 4.1b where optimal threshold is directly applied to the original image. This proves that by compensating the intensities of those boundaries pixels, global threshold can be used to images with inhomogeneous background.



(a)



(b)

Figure 4.7 (a) Contour based image segmentation using phase congruency for edge detection (b) Contour based image segmentation using Sobel method for edge detection

By looking at the circled area in Figure 4.7a and Figure 4.7b, we find that more detailed information is extracted in Figure 4.7a. This result suggests that phase congruency method is a better approach in which case the object boundaries are highlighted.

In CBIS, we use phase congruency method combined with optimum threshold and 8-connective tracer to outline the object contour. Separating step is applied to each slice in the image stack. After segmentation, two images are converted into the xml data structure. At this stage, the xml data structure does not include information about how object contours are related to each other.

Each object contour is saved as a node in the xml data structure. Not only the coordinates of the pixels on the contour are recorded in the node but also other contour features such as centroid, moments, average intensity. Contour features will be used in the grouping stage to link the related objects together. In the grouping state, the matching decision is made by the contour feature of the node saved in the xml structure. The grouping stage will interact with the xml image structure directly instead of with the raw image data set. After a match has been found, the xml image structure will be modified by adding a link between two object contours.



## Chapter 5 Fuzzy Contour Matching

In this section, we discuss how to link the contours on the adjacent image slices which belong to the same 3D subcomponent. This is the grouping stage to build the CBIS. In this stage, for each contour on the 2D image, we need find out if there is a corresponding contour on image slice next to it. Two related contours should have similar spatial features. For example, the centroid of the object should be close and the contour shape should be consistent.

2D object recognition and matching are widely used and many methods have been proposed, including template matching, string matching, shape-specific point matching, principal axis matching, dynamic programming, mutually-best matching, chamfer matching, graph matching, relaxation, elastic matching, and etc (Wu and Wang, 1999; Veltkamp *et al.*, 1999).

Many factors could affect the decision of whether two neighbor object contours on adjoining slices belong to the same 3D component. In many cases, we can not predict the dominating factor. In this section, a Fuzzy Contour Matching System (FCMS) is proposed to help us solve contour matching problem. Fuzzy logic has been credited with handling imprecision and uncertainties since it was introduced (Zadeh, 1965). A FLS will be constructed to determine which objects in the different image slices belong to the same 3D component or not. In other words, whether two object contours match each other. FCMS consists of four parts: fuzzy inputs, membership function for inputs and output, fuzzy rules, and fuzzy inference and defuzzification.

## 5.1 The Considerations of the FLS Inputs

Given two object contours, the FLS takes the consideration of the following aspects as the system inputs: the spatial relation between two objects, the lighting intensities of the objects, and the orientation relation of the objects.

1) *Non-overlapping Ratio*: Given two image object, we can treat them as two functions, say  $f(x,y)$  and  $h(x,y)$ . Correlation defines a way of combining two functions and can be used to find the matching between the two functions. The correlation of the two discrete functions  $f(x,y)$  and  $h(x,y)$  is defined as:

$$f(x,y) \circ h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n) h(x+m, y+n) \quad (5.1)$$

where  $f^*$  denotes the complex conjugate of  $f$ . Since we are dealing with image which can be represented by a real function,  $f^*$  is equal to  $f$ . Thus the simplest form of correlation between  $f(x,y)$  and  $h(x,y)$  can be derived:

$$f(x,y) \circ h(x,y) = \sum_s \sum_t f(s,t) h(x+s, y+t) \quad (5.2)$$

For  $x = 0, 1, 2, \dots, M-1, y = 0, 1, 2, \dots, N-1$ , and the summation is taken over the image region where  $f$  and  $h$  overlap. This formula has the disadvantages of being sensitive to the changes in the amplitude for  $f$  and  $h$  and the high computation requirements since the similar computation should be applied on each pixel location  $(x,y)$ .

In this study, we also assume that each pair of contours belonging to the same 3D component in two adjacent slices should have the similar spatial parameters. This essential assumption is made based on the fact that adjacent slices are very close and adjacent contours of the same object will differ by very few pixel points. Experimental

measurements taken from this project indicate that the distance between adjacent slices is around  $2.1\mu m$ . In most cases this assumption is valid particularly when the contour shape changes gradually and continuously. This implies that we can simplify the correlation function further by assuming matched images on the adjacent slices should be overlapped perfectly. To simplify the fuzzy system model, instead of using the correlation function, we use the ratio of the non-overlapped area to measure the spatial relation of two objects. Given two contours from adjacent image slices, the non-overlapping ratio denotes the percentage of pixels in one object contour but not in another one when two image slices are overlapped. More formally, given two contour objects A and B, we define the coordinate sets  $P_a$  as  $\{(X_i, Y_i) \mid (X_i, Y_i) \text{ is one of the coordinates of the pixels in contour A}\}$  and  $P_b$  as  $\{(X_j, Y_j) \mid (X_j, Y_j) \text{ is one of the coordinates of the pixel in contour B}\}$ . Then the non-overlapping ratio of A is  $1 - (|P_a \cap P_b| / |P_a|)$  and the non-overlapping ratio of B is  $1 - (|P_a \cap P_b| / |P_b|)$ , where  $| \quad |$  define the size of a set.

Thus, we can assume that two contours which have a high non-overlapping ratio do not likely belong to the same 3D component. We use the smaller object contour to calculate the non-overlapping ratio of two objects based on the fact that the non-overlapping ratio derived from the larger object contour always has a larger value than the one derived from the small contour. When a 3D component branches or shrinks at the position where two image slices are taken, the higher non-overlapping ratio from the large object doesn't reflect the spatial relation between two contours and is prone to separate the two contour objects. The non-overlapping ratio is the first input of the fuzzy system.

2) *Lighting Intensities*: In biological specimen images, lighting intensities usually differ among different tissues. For example, the intensity value of bone's area is different from intensity value of skin's area. The intensity histograms are commonly used to distinguish the objects which do not belong to the same tissue. For each object contour, we calculate the average intensity of all the pixels enclosed in the object contour as the whole object intensity. The intensity difference of two objects is the second input in the fuzzy system.

3) *Orientation of Objects*: Furthermore, we also assume that two contours likely have the similar orientation on adjacent slices if they belong to the same 3D component. This assumption is reasonable since, in our case, the entire neuron branches stretch to a particular direction. The information about the object orientation can be derived by using the second order central moments to construct a covariance matrix. Covariance matrix is defined as:

$$\text{cov} [I(x, y)] = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \quad (5.3)$$

where

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y) \quad (5.4)$$

$\bar{x}$  and  $\bar{y}$  are the centroid of the object. Since we only consider the orientation of the object based on their shapes, we can simplify the formula by setting the intensity value of all the pixels inside the contour to a constant, for example, setting  $I(x, y)$  in formula 5.4 always to 1 regardless of the value of  $x$  and  $y$ .

The eigenvectors of this matrix constitute the predominate axes of the object, and the orientation can thus be extracted from the angle of the eigenvector associated with the largest eigenvalue. This angle is given by the following formula:

$$\theta = \frac{1}{2} \tan^{-1} \left( \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right) \quad (5.5)$$

The orientation difference of two objects is set as the third input of the fuzzy system.

## 5.2 Fuzzy Logic System for Contour Matching

FLSs have been successfully used to handle the imprecision and uncertainties in real-world applications. Figure 5.1 shows the basic structure of a FLS. It usually contains four components: fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier. Fuzzifier maps crisp inputs into fuzzy sets. Fuzzy rule base is used to represent the fuzzy relationships between input and output fuzzy sets. Fuzzy rules are expressed in IF-THEN statements. The inference engine combines the fired fuzzy rules and maps crisp inputs into fuzzy output sets. The defuzzifier is used to convert output fuzzy sets into crisp outputs.

Two popular fuzzy logic models are widely used to construct a FLS: Mamdani fuzzy model (Mamdani, 1974) and Takagi-Sugeno-Kang (TSK) model (Takagi and Sugeno, 1985). The difference between the two models is the consequence part of an IF-THEN fuzzy rule. The Mamdani model represents the consequence of a fuzzy rule using linguistic variables and fuzzy sets, while the TSK model describes the consequence part by a function of FLS inputs. We apply Mamdani model to construct our fuzzy contour matching system. The detailed design of the fuzzy contour matching model is presented as follows.

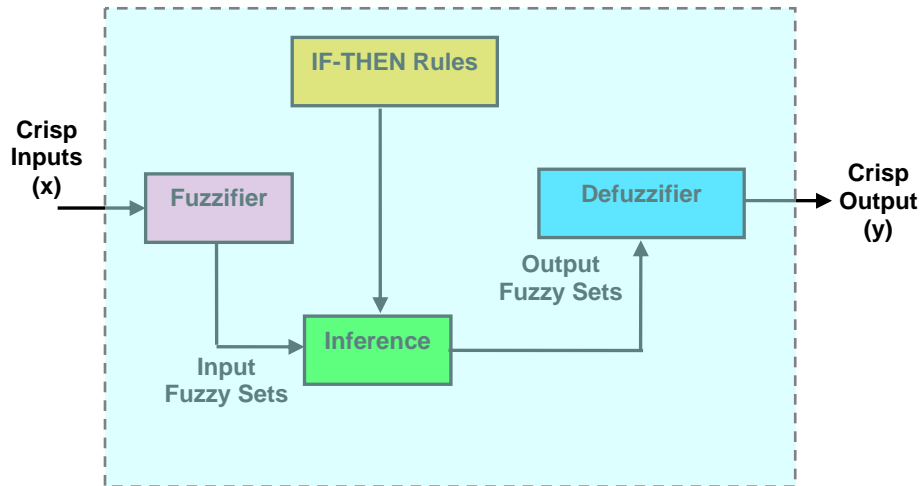
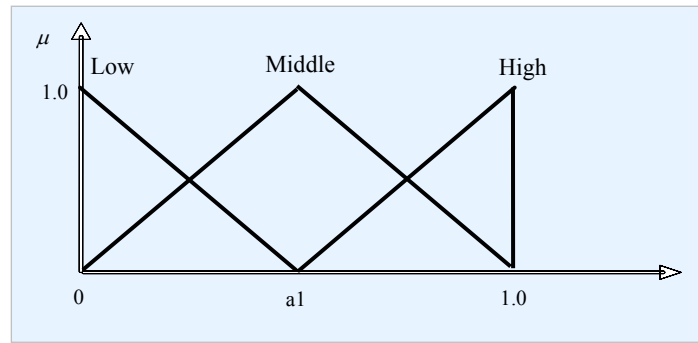


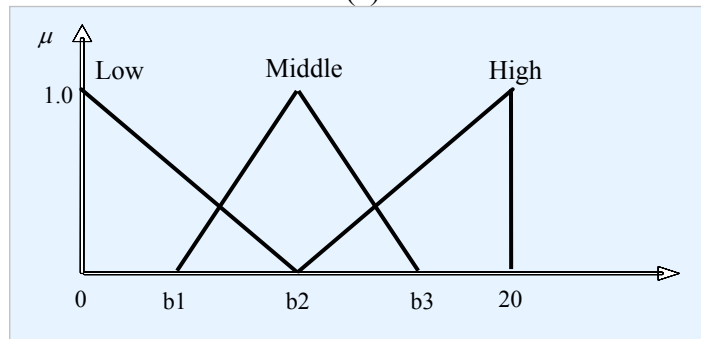
Figure 5.1 The structure of a FLS

### 5.2.1 Membership Functions

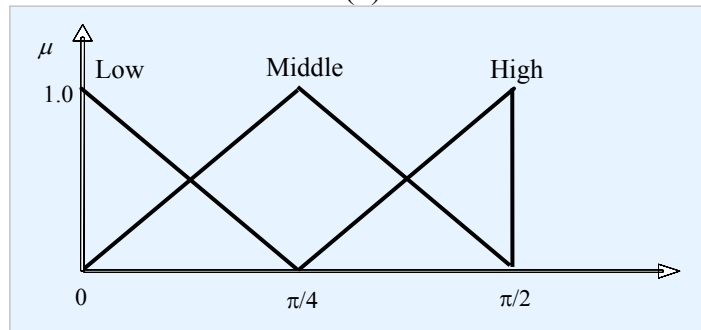
Figure 5.2 shows the membership functions of the fuzzy system's inputs and the output. All the three inputs are represented by three fuzzy sets: low, middle and high, and the output is represented by five fuzzy sets: low, LM, middle, MH, and high. Figure 5.2a shows the membership functions of the non-overlapping ratio of the two objects. The value of  $a_1$  is tunable. For example, a lower value of  $a_1$  should be set when two adjacent slices are very close since in this case, the object shape is supposed to be changed slightly from one slice to another.



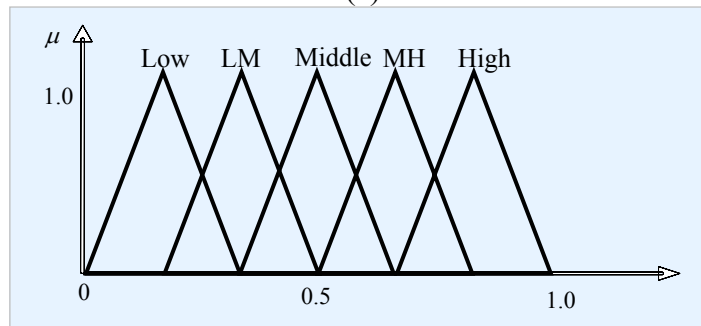
(a)



(b)



(c)



(d)

Figure 5.2 The fuzzy membership functions (a) Non-overlapping ratio (b) Difference of lighting intensity (c) Difference of object orientation (d) The output

Figure 5.2b shows the membership functions of the intensity difference between two object contours. Here we use the gray value of pixels in the contour objects. The linguistic variable of intensity difference can be extended into three linguistic variables if the color value of the contour objects is used by simply calculating the difference of the color value of red, green or blue element respectively. We use 20 as the threshold of the intensity difference which indicates that two similar tissues usually have an average intensity difference less than 20. In case that the intensity difference of two objects is larger than 20, the membership value of “High” fuzzy set is set to 1.  $b_1$ ,  $b_2$  and  $b_3$  are adjustable as well. For example, for an image slice with inhomogeneous background, there should have more overlapped intensities among the background areas and objects. The value of  $b_2$  should be set to a larger value in this case and the distance between  $b_1$  and  $b_3$  should be larger.

Figure 5.2c shows the membership functions of orientation difference between two object contours. We use  $\pi/2$  as the threshold which indicates that the difference of the orientation between the related contours should be always less than  $\pi/2$ . Similar to non-overlap function, the value of  $c$  can be adjusted based on the density of image stack. For image stack with high density, the adjacent images are very close. In this case a value less than  $\pi/4$  will be more suitable since there is slim chance that principal axis of the related contour may stretch to different direction.

Figure 5.2d is the output function. The output is in the range  $[0, 1]$  and it reflects the matching degree of two object contours: the larger the value of the output, the bigger chance of the two objects contours matching and belonging to the same 3D component.



The parameter  $\{a_1, b_1, b_2, b_3, c, d\}$  in the four fuzzy functions above are tunable. It can be optimized by using a genetic algorithm which discussed in the later section.

### 5.2.2 Fuzzy Rules

Since the system has three inputs and each input has three possibilities respectively, there are  $3 \times 3 = 27$  fuzzy rules in total. Based on the three inputs and one output, we define the  $i$ th ( $i = 1 \dots 27$ ) fuzzy rule as follows:

IF  $a_1$  is  $A_1^i$  and  $a_2$  is  $A_2^i$  and  $a_3$  is  $A_3^i$ , THEN  $g_i$  is  $G_i$  ( $i = 1 \dots 27$ ).

where  $a_1$ ,  $a_2$  and  $a_3$  denote three input values for non-overlapping ratio, intensity difference and orientation difference respectively,  $g_i$  denotes the output;  $A_1^i$ ,  $A_2^i$ , and  $A_3^i$  ( $i = 1 \dots 27$ ) denote the three input fuzzy sets for the  $i$ th rule which are in  $\{\text{Low, Middle, High}\}$ , and  $G_i$  ( $i = 1 \dots 27$ ) denotes the output fuzzy set of the consequence of the  $i$ th rule which is in  $\{\text{Low, LM, Middle, MH, High}\}$ .

When we set the output fuzzy sets for the consequences of the fuzzy rules, we consider the image object characteristics and combine the information of non-overlapping ratio, difference of intensity, and difference of orientation. For example, if all the three inputs are high, the output fuzzy set of the consequence of that rule is low, indicating the two object contours are less likely to match each other. On the other hand, if all the three factors are low, the output fuzzy set of the consequence of that rule is high, indicating that the two object contours are more likely to match each other. Based on the same consideration, the output fuzzy sets of the consequences for other rules can be determined.

### 5.2.3 Fuzzy Inference and Defuzzification

The output of the fuzzy contour matching system is calculated by aggregating individual rule contributions as follows:

$$y = \sum_{i=1}^{27} \beta_i g_i / \sum_{i=1}^{27} \beta_i \quad (5.6)$$

where  $g_i$  is the output value of the  $i$ th rule ( $i = 1 \dots 27$ ). The output value of the  $i$ th fuzzy rule is determined by the center of gravity of the isosceles triangle, which represents the output fuzzy set of the consequence part of the  $i$ th rule.  $\beta_i$  is the firing strength of the  $i$ th rule. It is defined by product  $t$ -norm:

$$\beta_i = \prod_{j=1}^3 \mu_{A_j^i}(a_j) \quad (5.7)$$

where  $\mu_{A_j^i}(a_j)$ ,  $j=1 \dots 3$ , is the membership grade of input in the fuzzy set  $A_j^i$ .

If the output value is greater than or equal to 0.5, we consider the two contours match. Otherwise, they don't match.

For each contour node in the xml structure, we use the fuzzy contour matching system to locate the object contours on the lower layer slice adjoining to it. The structure is updated by adding a *Link* element in the contour node if a matched contour is found. After this process, related contours are then grouped to form a solid three dimensional object. A tree structure model is formed where each node in the tree is a representative of a segmented object and each edge in the tree is a representative of contour matching between two objects on adjacent slices.

The contour fuzzy matching model is suitable to handle imprecise and uncertain situations and make it possible for unbiased contour matching decision based on various image features. Instead of using three contour object properties as the system inputs, the

fuzzy contour matching system can be extended by adding more contour object characteristics which are helpful for the contour matching decision.

#### **5.2.4 Tuning the Membership Functions by Genetic Algorithms (GA)**

The fuzzy membership functions determine the matching decision of how two object contours relate one another and it is the crucial part in the FCMS. That implies, how to choose the values of  $\{a_1, b_1, b_2, b_3, c, d\}$  in Figure 5.2 is very important. We use Genetic Algorithms to tune the fuzzy membership functions. GAs are optimization algorithms which are inspired by natural evolution. The basic idea is to maintain a population of chromosomes over time through a process of variation and competition (Goldberg, 1989). The optimization process of GAs can be described as follows. First, an initial population of chromosomes is generated within the ranges of genes randomly. Each chromosome may contain many genes. Next, individuals or chromosomes in the first population or generation are selected based on their fitness to reproduce the next generation by performing a number of genetic operations upon the selected chromosomes. The common genetic operators include: crossover, which creates new chromosomes from parts of parents; and mutation, which introduces variation into the population by randomly changing selected genes of chromosomes. The process of selection, recombination and mutation is repeated iteratively, generation after generation, until either the required fitness is met or the user-defined number of iterations is reached. The best chromosome in the final population contains the optimal or approximate optimal solution to the problem (Chen *et al.*, 2005).

From Figure 5.2, we can see that, in our case, a chromosome includes 6 genes to represent the 9 fuzzy input sets and 5 fuzzy output sets. Given the ranges of the values for

elements in  $\{a1, b1, b2, b3, c, d\}$ , the genetic tuning process can be summarized as follows:

*Fitness of the GA:* Before the tuning process, we first select several image stacks, automatically segment the images, and match the contours on adjacent slices manually. The similarity of the match decisions derived from this step and from the FCMS is used for the fitness of the GA. The accuracy of an individual FCMS is defined as:

$$1 - (w_{link} + M_{link}) / T_{link} \quad (5.8)$$

where  $w_{link}$ ,  $M_{link}$ , and  $T_{link}$  represent the number of wrong links, missed links derived from FCMS, and the number of correct links respectively.

*Selection and Elitism:* We use standard proportional selection scheme (also referred to as roulette-wheel selection). A chromosome, which contains 6 genes, is selected for the next generation based on its fitness or its matching accuracy. The chromosome containing the best fuzzy sets is guaranteed to survive over the generation.

*Crossover Operation:* We use one-point crossover. The fuzzy sets of two parents are crossed and recombined to generate two offspring chromosomes of fuzzy sets. The crossover point is randomly chosen among 6 genes of parent chromosomes.

*Mutation Operator:* Random uniform mutation is used. A fuzzy set is selected randomly and replaced by a random value within its range.

### 5.3 Using FCMS to Build Contour Structure from Image Stack

FCMS is tested on an image stack which includes 19 image slices of crayfish microscopic 3D images. This image stack is first converted into an xml file by using the optimum segmentation methods described in Chapter 4. Then the FCMS testify each

object contour with all the contours on the next layer and try to find a link between two slices.

In the initial experiment, we set the parameters  $\{a1, b1, b2, b3, c, d\}$  of the fuzzy membership functions as  $\{0.5, 5, 10, 20, \pi/4, 0.5\}$ . The parameters work fine in the most cases. To testify the accuracy of FCMS, we confirm the matching decisions by removing the incorrect links of unrelated contours. We also add the links between object contours which are not detected by the FCMS. This confirmation process is carried out manually. Figure 5.3 displays a confirmed contour matching structure of an image stack.

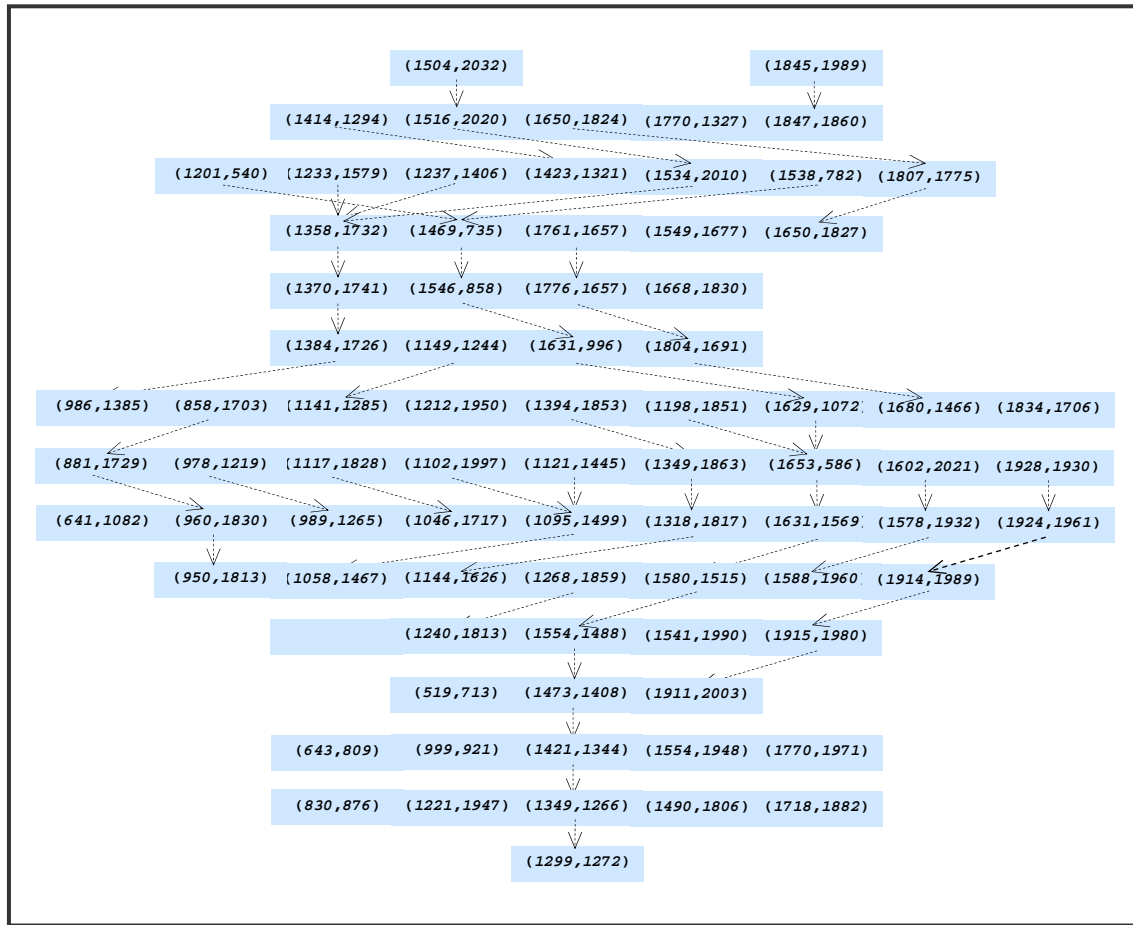


Figure 5.3 Contour based XML image structure built by FCMS

This confirmed matching decision is used for optimal membership functions' tuning. Given an image stack with the matching information confirmed, we can always expect to find better fuzzy membership functions by using GAs. The tuning process is time-consuming, considering that for each chromosome, we need to run its corresponding FCMS to test its accuracy.

In Figure 5.3, the contour xml structure has only 15 layers which are less than the number of the slices in the original image stack. This is because no significant objects have been detected in the slices on the top and bottom of the image stack. To avoid duplicate contour matching computation, the contour links are always created from the top slice to the bottom one. By default, the created xml structure forms a directed acyclic graph (DAG) as shown in Figure 5.3. We know that one of major purposes to create contour xml structure is for 3D image partial retrieval. When applying 3D partial retrieval, it is not necessarily to restrict the searching algorithm to follow the constraints in DAG. In most cases, we can just treat the link at dual direction. The following algorithm shows how to extract all the 3D sub-components.

*Initialization:*

*unhandle\_nodes\_list = {all the nodes in xml tree structure}*

*3D\_subcomponets\_list = {};*

*for (all node X in Unhandle\_nodes)*

*if (X is an isolated node)*

*new\_3D\_subcomponet = {X}*

*3D\_subcomponets\_list->push(new\_3D\_subcomponet)*

*unhandle\_nodes->remove(X)*

*while (unhandle\_nodes->isNotEmpty)*

*new\_3D\_subcomponet = {unhandle\_nodes->pop}*

*while ( exist Y link new\_3D\_subcomponet AND Y in unhandle\_nodes)*

*new\_3D\_subcomponet-> push (Y)*

*unhandle\_nodes->remove(X)*

*3D\_subcomponets\_list->push(new\_3D\_subcomponet)*

*return 3D\_subcomponets\_list*

Figure 5.4 The algorithm to list all the 3D subcomponents in the xml image structure

## **Chapter 6 Invariant Image Feature Extraction from Frequency Domain**

In CBIS, fuzzy contour matching model is valid because adjacent images are close and the matching task can be simplified without taking the consideration of object rotation or scaling. In the more general cases, two objects may have similar shapes with different principle axes and scales. This is also true for image stacks with low density. In such case, the matching algorithm should be rotation and scale invariant. To solve this problem, the fuzzy contour matching system should include fuzzy inputs which are invariant to rotation, translation and scaling. We discuss how to use image features derived from frequency domain to achieve rotation, translation, and scale invariance.

Furthermore, the shape matching algorithm is not applicable to the images which can not be described by object contours, for example, some bacterial images which look like texture. We define a multiple-level cosine transform for image feature extraction in texture images.

### **6.1 Shape Signature and Complex Contour Vector**

Generally, shape representation and description methods can be categorized into contour-based and region-based depending on whether the shape features are extracted from contour only or are extracted from the whole shape region (Zhang and Lu, 2004).

In CBIS, the contour is the basic unit in the xml image structure. The shape of the contour is determined by the pixels on the contour. The coordinates of the pixels are recorded in the xml image structure. Base those coordinates, an object contour shape can be described by a dimensional function which is called shape signature. Shape signature functions include centroidal profile, complex coordinates, centroid distance, tangent



angle, cumulative angle, and so on (Davies, 1997). Given an object contour, even though we can easily reconstruct the solid object based on their boundary pixels, those pixels inside the contour will not make significant contribution to the shape matching decision since they are completely determined by the boundary. We can use complex coordinates to convert 2D object boundary into a 1D vector. The idea is, given an object contour with  $L$  pixels in the x-y plane, we find their relative location to the object centroid. That is, given the contour coordinate sequence  $\{(x_0, y_0), (x_1, y_1), \dots, (x_{L-1}, y_{L-1})\}$ , we will convert it to a complex vector  $V$  as  $\{(x_0 - x_c) + i(y_0 - y_c), (x_1 - x_c) + i(y_1 - y_c), \dots, (x_{L-1} - x_c) + i(y_{L-1} - y_c)\}$ . Here,  $(x_c, y_c)$  is the centroid of the contour.

## 6.2 Contour Features Extracted by Fourier Descriptor

To obtain the contour features from frequency domain, we can apply discrete Fourier transform on the complex vector  $V$ , the Fourier transform coefficients are:

$$F(u) = \sum_{k=0}^{L-1} L(k) e^{-j2\pi ku} \quad (6.1)$$

It is not suitable to use the derived coefficients directly in the shape analysis since the features are not rotation and scale invariant. For rotation invariance, we can discard the phase information and use the magnitude only. For example, we can use the distance from each pixel to centroid to represent the contour. Scale invariance can be achieved by dividing the DC value  $F(0)$  for each coefficient. It is well known that high-frequency components account for the fine detail and low frequency components account for the global shape. This implies that, given two shapes, we can make shape matching decision based on their low frequency components.

A Generic Fourier Descriptor was introduced in (Zhang and Lu, 2002) where a modified polar FT was proposed by treating the polar image in polar space as a normal 2D image in Cartesian space. Given an image  $f(x,y)$ , the modified polar FT is defined as:

$$PF(\rho, \phi) = \sum_r \sum_i f(r, \theta_i) \exp[j2\pi(\frac{r}{R}\rho + \frac{2\pi i}{T}\phi)] \quad (6.2)$$

where  $r$  denotes the distance between the pixel and the centroid.  $\theta_i = j(2\pi/T)$ .  $R$  and  $T$  are the radial and angular resolutions. Since the transform is based on the center of the image, it is translation invariant. The rotation and scaling invariance can be achieved by the following normalization:

$$\left\{ \frac{|PF(0,0)|}{area}, \frac{|PF(0,1)|}{|PF(0,0)|}, \dots, \frac{|PF(0,n)|}{|PF(0,0)|}, \dots, \frac{|PF(m,0)|}{|PF(0,0)|}, \dots, \frac{|PF(m,n)|}{|PF(0,0)|} \right\} \quad (6.3)$$

where area is the area of the boundary circled,  $m$  is the maximum number of the radial frequencies, and  $n$  is the maximum number of angular frequencies.

Now we the Generic Fourier Descriptor, we can extend the FCMS to include contour features which are rotation, translation and scaling invariant. This can be done by adding contour features derived from generic Fourier Descriptor to the fuzzy inputs and modify the fuzzy membership function accordingly.

### 6.3 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform is similar to the Fourier Transform. It transforms a signal from spatial or time domain to the frequency domain. The DCT is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)K\right] \quad k = 0, \dots, N-1 \quad (6.4)$$

The corresponding Multidimensional DCT is defined as:

$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1} x_{n_2} \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)K_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)K_2\right] \quad (6.5)$$

The advantage of using DCT is its efficiency especially for image compression. For example, JPEG images and MPGE videos are based on DCT compression techniques (Wallace, 1991). The process to compress an image using DCT can be summarized in the following steps:

- 1) Image is divided into 8 by 8 pixels block
- 2) Apply DCT on each sub-block
- 3) Based on the same size 8 by 8 quantization matrix, a quantizer rounds off the DCT coefficient which makes many high frequency coefficients equal to 0
- 4) Compress the “lossy” coefficients to an output .jpg file

DCT compression proves that most of the image information which is sensitive to human eyes is reserved in the low frequency components. This also implies that low frequency coefficients are good features for image classification.

#### 6.4 Odd and Even Cosine Transform For Image Feature Extraction

We propose a new transform which is modified version of cosine transform and is suitable for multi-resolution analysis. The basic idea is, given a complex vector  $V$  which describes the boundary shape of a object, we define different levels of cosine transform of  $V$ . Suppose  $V$  is the vector with  $2^m$  element, we define even and odd cosine transform for each level  $L$  as:

$$T'_l(k) = \sum_{i=0}^{N/2^l-1} \left( \sum_{j=2^l*i}^{2^l*(i+1)-1} V(j) \right) \cos\left[ \frac{\pi * 2^l}{N} \left( j + \frac{1}{2} \right) k \right] \quad (6.6)$$

$$T_l''(k) = \sum_{i=0}^{n/2^l-1} \left( \sum_{j=2^l*i}^{2^l*(i+1/2)-1} V(j) - \sum_{j=2^l*(i+1/2)}^{2^l*(i+1)-1} V(j) \right) \cos\left[\frac{\pi*2^l}{N}(j+\frac{1}{2})k\right] \quad (6.7)$$

where  $T_l'(k)$  represents the even cosine transform and  $T_l''(k)$  represents the odd cosine transform.  $k \in [0, \frac{N}{2^l}]$  for any level  $L$ , where  $L \in [0, m]$  and  $m = \log_2 N$ . When  $L=0$ ,  $T_0'(k)$  is the standard DCT-II.

$T_l'(k)$  is the cosine transform for the scaled signal with the scaled factor of  $2^l$ .  $T_l''(k)$  is the cosine transform of the difference of the neighbor elements for the scaled signal with the scaled factor of  $2^l$ . We define the inverse even and odd cosine transforms as:

$$V_l'(k) = \frac{1}{2} T_l'(0) + \sum_{i=1}^{N/2^l-1} T_l'(i) \cos\left[\frac{\pi*2^l}{N}(k+\frac{1}{2})\right] \quad (6.8)$$

$$V_l''(k) = \frac{1}{2} T_l''(0) + \sum_{i=1}^{N/2^l-1} T_l''(i) \cos\left[\frac{\pi*2^l}{N}(k+\frac{1}{2})\right] \quad (6.9)$$

From the above formulas we can see, given the level  $L$  and even and odd cosine transform  $T'(L)$  and  $T''(L)$ , we can easily reconstruct the  $V'(L-1)$ . This can be implemented by first calculating the  $V'(L)$  and  $V''(L)$  from  $T'(L)$  and  $T''(L)$ . Then we can construct  $V(L-1)$  by its odd elements and even elements. The elements with even indexes in  $V(L-1)$  equal  $[V'(L) + V''(L)]/2$  and the elements with odd index in  $V(L-1)$  equal  $[V'(L) - V''(L)]/2$ .

Furthermore, the original signal can be reconstructed from  $V'(1)$  and  $V''(1)$ .  $V'(1)$  can be reconstructed from  $V'(2)$  and  $V''(1)$  and so on. It turns out that we can identify the original signal using the vector  $V = \{V'(k), V''(k), V''(k-1), V''(k-2), \dots, V''(1)\}$ . Similarly,

the original signal can be reconstructed by its multi-level cosine transform vectors  $\{T''(k), T''(k-1), T''(k-2), \dots, T''(1)\}$ .

It is worthy noticing that the total number of the elements in the multi-level cosine transform vectors is equal to the number of the elements in original signal. The multi-level cosine transform vectors provide:

- 1) Multi-resolution factors of the signal.
- 2) Signal features from the frequency domain.

Generic Fourier Descriptor makes it possible for FCMS to handle contour matching problem when rotation, translation and scaling invariance are required. This is important for general shape matching. Odd and Even Discrete Cosine Transform provides us options to handle images where object contours are hard to be extracted. For texture images where a certain type of pattern is repeatedly distributed in the image, we can divide the image into blocks and apply DCT to each block. By studying the similarity and difference of low frequency DCT components among the blocks, we can classify different texture groups.

## **Chapter 7 Texture Image Classification Using Multi-level Cosine Transform**

In 2D image retrieval, image features such as texture, shape, spatial layout, and color are used to specify queries. In the previous chapter, images are described by the contour shapes. Contour analysis may be adequate for untextured images but not for texture images. This is because texture images normally generate meaningless tangled web of contours (Malik *et al.*, 2001). For those images such as many bacterial images, extracting texture features other than contours will be a better approach to describe the images. Figure 7.1 shows several examples of bacterial images. These images are very similar to texture images. In Figure 7.1, Bacillus image is similar to straw texture and other bacteria are all look like certain textures. Those images are hard to be segmented by object contours. Figure 7.2 shows the edge detection images generated from Figure 7.1. We can find that it is very hard for CBIS to efficiently convert those images into a standard xml image structure since there are too many contours in the image and there are no apparent features which can be used to describe most of the contours.

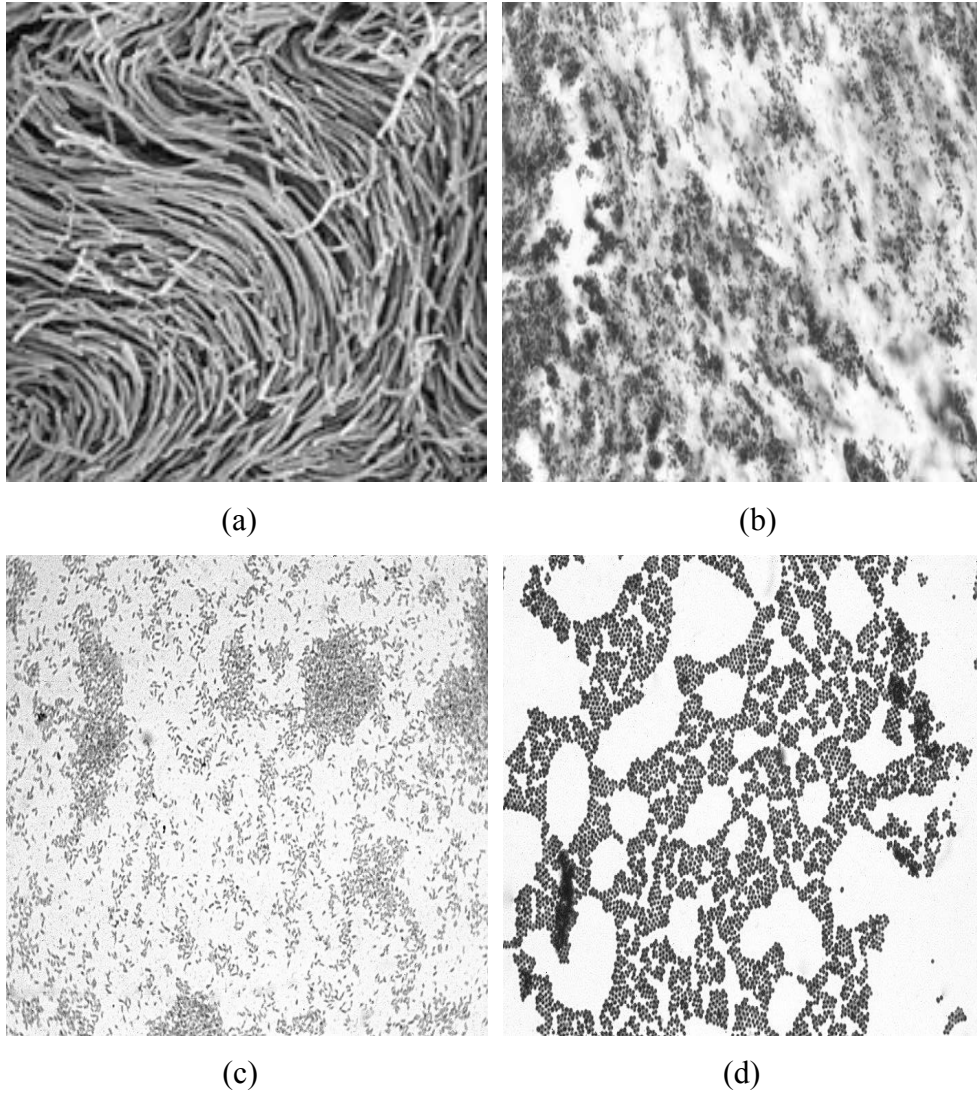


Figure 7.1 Bacteria Images (a) *Bacillus* (b) *Bartonella henselae* (c) *Bordetella pertussis* (d) *Staphylococcus*

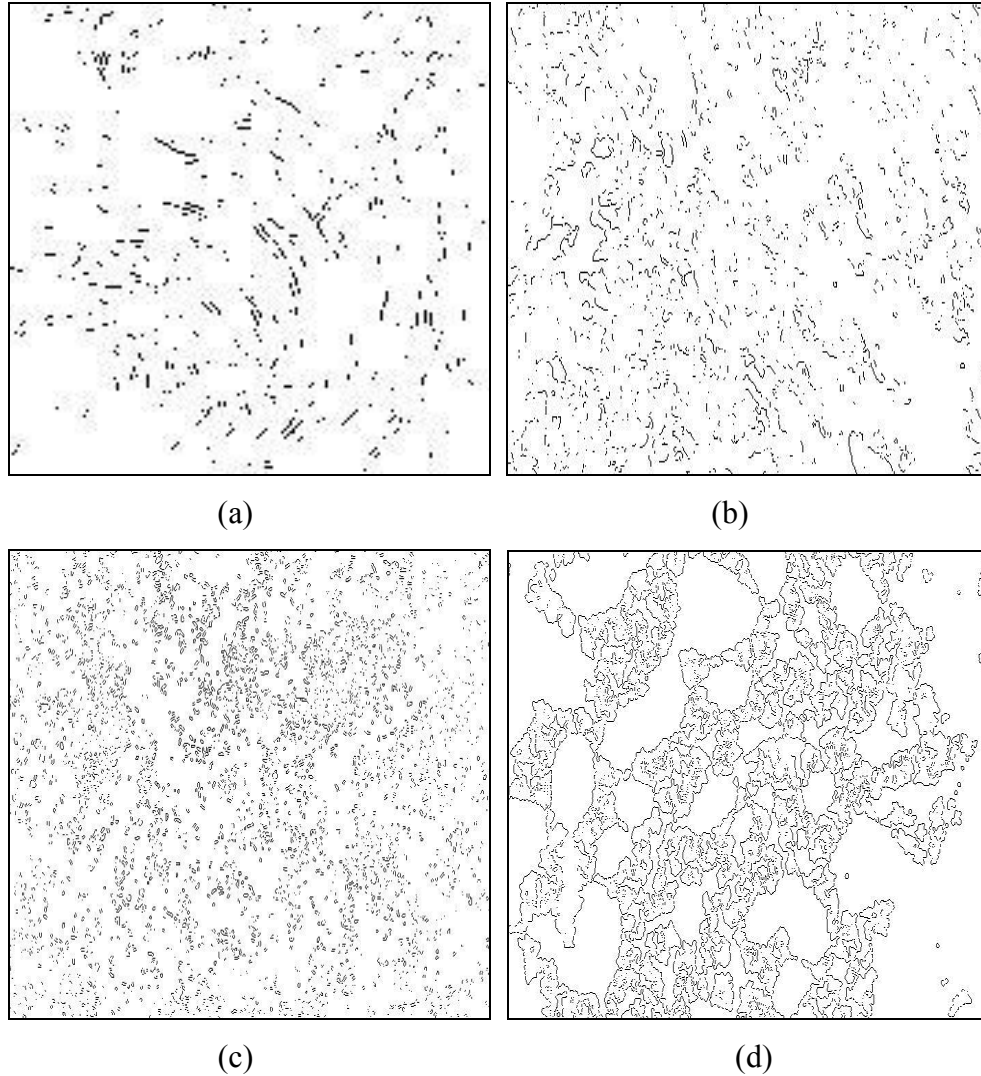


Figure 7.2 Edge detection images of bacteria (a) *Bacillus* (b) *Bartonella henselae* (c) *Bordetella pertussis* (d) *Staphylococcus*

Texture contains important information about the structural arrangement of surfaces and their relationships to the surrounding environments (Rui *et al.*, 1999). Texture is the most important visual cue in identifying a variety of materials. Texture analysis has been used for disease diagnosis and medical image analysis (Ji *et al.*, 2000, Kim and Park



1999, Ginneken *et al.*, 2002). It can also be used in remote sensing to obtain the boundary map separating the differently textured regions and in image compression where synthesis texture may replace backgrounds in natural scenes thus leading to a dramatic bit saving (Li *et al.*, 2006). A variety of techniques, ranging from statistical methods to multi-resolution filtering have been developed for texture analysis. Two-dimensional Gabor Filter as one of the multiresolution filtering techniques is proved to be very useful in texture analysis and is widely adopted in the literature (Manjunath and Ma, 1996; Grigorescu *et al.*, 2002; Weldon *et al.* 1996; Hamamoto *et al.*, 1998). It has been shown that image retrieval using Gabor Filter outperforms other methods using various wavelet transforms, including orthogonal and bi-orthogonal wavelet transforms, and tree-structured wavelet transform. However, the computational cost of using the above wavelet transforms is expensive for image retrieval. Normally, the retrieval mechanisms make similarity measure by contrasting the features of the query image and the features from the images in the structure. An efficient as well as simple feature extraction scheme is obligatory for real-time image retrieval.

It is well known that low-frequency coefficients of the Discrete Cosine Transforms (DCTs) preserve the most important image features. In our study, we use Multi-level DCT (MDCT) coefficients to create texture feature vectors. MDCT is the Even-Discrete Cosine Transform we introduced in Chapter 6. We also calculate the Zernike moments as image texture features. The texture feature vector is the combination of MDCT coefficients and Zernike moments. Support Vector Machines (SVMs), which have demonstrated powerful and promising generalization abilities in image processing and

other classification applications, are used to train the feature vectors to obtain multiple classifiers.

## **7.1 Texture Feature Extraction and Multi-level Discrete Cosine Transform**

The discrete cosine transform (DCT) is often used in signal and image processing, especially for lossy data compression. The excellent energy compaction property of the DCT is the main reason for its popularity. This property enables most of the signal information to be concentrated in a few low-frequency components of the DCT (Chen and Pratt, 1984; Min *et al.*, 1998). Smith and Chang (Smith and Chang, 1994) compared several subband-energy features which can be used for texture classification. Huang (Huang 2005) introduced extracting texture features directly from the DCT coefficients in the DCT-code image.

### **7.1.1 DCT Coefficients as Image Texture Features**

In a texture image, for example the bacterial images in Figure 7.1, we can see that a certain pattern is repeatedly distributed. Thus, we can break down the whole image into several blocks. The process of image classification using DCT coefficients can be summarized as follows: An input image is partitioned into sub-blocks with the size of  $N \times N$ . DCT is performed on each block. There are total  $N^2$  coefficients within each block and the variance and mean values of each coefficient among the blocks are calculated to generate  $2N^2$  features. The  $2N^2$  feature vectors generated by the training images are mapped into a reduced space using Fisher Discriminant Analysis, which works by finding the eigenvectors of scatter matrices. A subset of the resulting eigenvectors that accounts for the largest total variation is used to assign the new texture images to the nearest classes.

The multi-resolution DCTs presented here are derived in the hope that they can represent the image texture features from its low resolution such that the processing time will be significantly reduced. For example, the total computing cost of MDCT1 and MDCT2 is about 1/4 and 1/16 of that by using the standard DCT when applying to an image.

Secondly, in some cases, MDCT may detect the difference of the image texture features resided in two images while the standard DCT fails to do so. For example, suppose we want to distinguish the following two images shown in Figure 7.3, both with the size of 64 by 64:

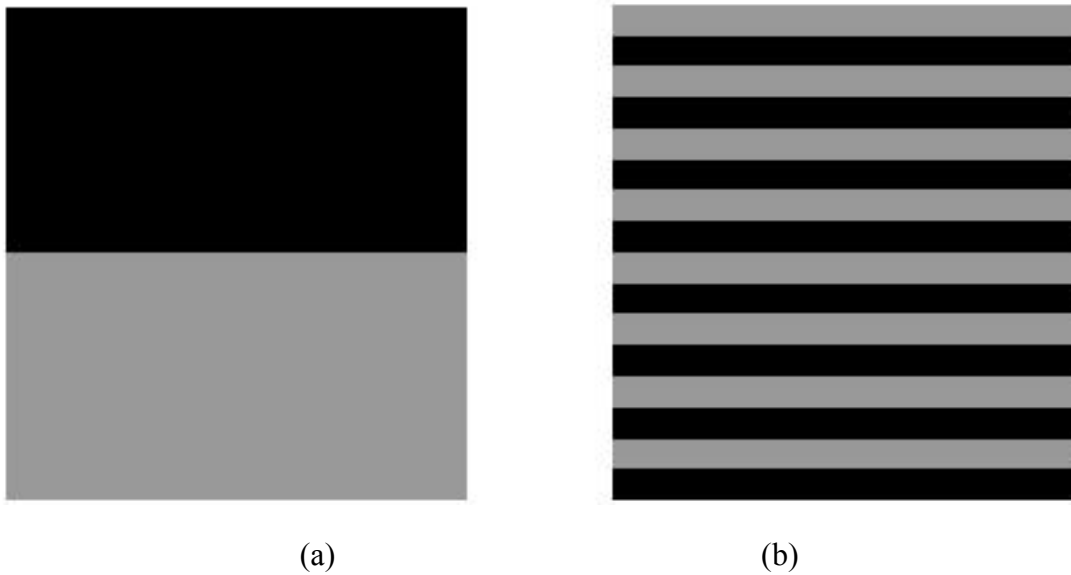


Figure 7.3 (a) Square image, dark and gray evenly divided (b) Stripe image, stripe size: 4 by 64

We divide both images into 4 by 4 blocks as mentioned in (Smith and Chang, 1994). As a result, both images have two kinds of 4 by 4 blocks: pure dark and pure gray as shown in Figure 7.4a and 7.4b. In this case, we do not have to go through calculating the

DCT coefficients before we can tell that the feature vectors extracted from the two images are identical, since both contain the same 128 black blocks and 128 gray blocks. Alternatively, we can use MDCT2 to fetch the feature vectors. As we discussed above, implementing MDCT2 is same as implementing standard DCT-II on the low resolution images with the compression ratio of 2. The low resolution images are similar to the ones in Figure 7.3 with the exceptions: both images have a reduced size of 32 by 32. Stripe size in Figure 7.3b decreases to 2 by 32. Since we still use 4 by 4 sub-bands, we can see, Figure 7.3a is divided into dark/gray blocks and Figure 7.3b is divided into blocks of half gray and half dark as shown in Figure 7.4c. When we apply DCT to the blocks in Figure 7.4a, 7.4b and 7.4c, we can expect all zero values of the AC coefficients in Figure 7.4a and 7b since ‘no-change’ occurs in these two blocks. There must be non-zero AC coefficients in the block in Figure 7.4c. Thus, the feature vectors extracted from Figure 7.3a and 7.3b by using MDCT2 will be different and be able to be used to distinguish one image from another.

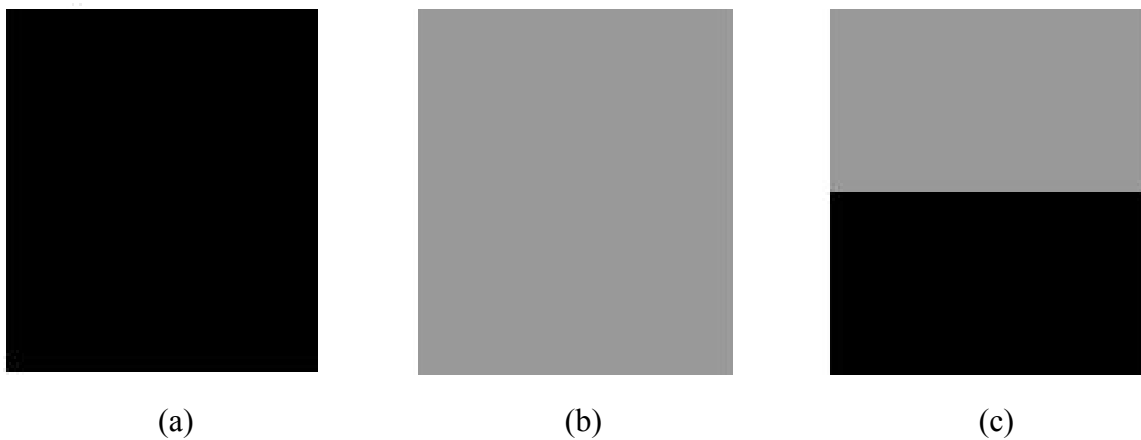


Figure 7.4 Three 4 by 4 image blocks

### 7.1.2 Image Feature from Zenike Moments

We also use Zenike Moments (ZM) to extract image features in this approach. Since the DCT coefficients are derived from individual blocks of the image. We may somehow lose the image features from a global perspective. Zernike moments are taken from the whole image data set and have many desirable properties, such as rotation invariance, robustness to noise, and expression efficiency. The complex ZM is derived from Zernike polynomials which are a set of complex, orthogonal polynomials defined over the interior of a unit circle  $x^2 + y^2 = 1$ .

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp(jm\theta) \quad (7.1)$$

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s! (\frac{n-|m|}{2}-s)! (\frac{n-|m|}{2}-s)!} \rho^{n-2s} \quad (7.2)$$

where  $n$  is a non-negative integer,  $m$  is an integer such that  $n - |m|$  is even and  $|m| \leq n$ ,

$\rho = \sqrt{x^2 + y^2}$ , and  $\theta = \tan^{-1} \frac{y}{x}$ . Projecting the image function onto the basis set,

the Zernike moments of order  $n$  with repetition  $m$  is given by:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x, y) V_{nm}(x, y), \quad x^2 + y^2 \leq 1 \quad (7.3)$$

It has been shown that the ZM on a rotated image has the same magnitudes.  $|A_{nm}|$  can be used as a rotation invariant feature of the image. It has also been shown in (Fu *et al.*, 2006) that a ZM order of 4 or 6 is suitable for image feature description. In this study 9 ZM moments are included in the feature vectors.

## 7.2 Image Feature Training and Classification Using SVM

SVMs are used to train the image feature vectors to classify different texture images. Given a testing image, SVMs will predict the texture classes based on the training texture feature vectors. This process involves binary classification and multi-class classification.

### 7.2.1 Binary Classification

Assume there is a training data set  $S: \{(x_i, y_i)\}_{i=1}^N$ , where each input  $x_i \in \mathbb{R}^m$  and output  $y_i \in \{\pm 1\}$ . The goal of SVMs is to find an optimal hyperplane  $w \cdot z + b = 0$  in a *feature space*, which can be transformed from the input vector space  $\mathbf{x}$  by the mapping of  $z = \phi(x)$ , and to separate the training data into two classes with the maximum margin in the feature space. Here,  $w = \sum_{i=1}^N \alpha_i y_i z_i$ , where  $\alpha_i$  is a set of Lagrange multipliers to the following dual problem (Vapnik, 1995):

$$\begin{aligned} \text{Maximize: } W(\alpha) &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j (z_i \cdot z_j) \\ \text{Subject to: } C &\geq \alpha_i \geq 0, \quad \sum_{i=1}^N \alpha_i y_i = 0. \end{aligned} \quad (7.4)$$

where  $C$  is a user-defined regularization parameter, determining the tradeoff between maximizing margin and minimizing the number of training examples misclassified. It is useful to handle non-separable problems and outliers.

The examples  $\mathbf{x}_i$  with  $\alpha_i > 0$  are called support vectors. They are the most informative training data examples lying close to the decision boundary. If support vectors are removed, the separating hyperplane would be changed.

The *kernel trick* of SVMs allows us to substitute the dot product of data points in (7.4) with just a *kernel function*:

$$K(x_i \cdot x_j) = z_i \cdot z_j \quad (7.5)$$

The decision function is made by computing:

$$f(x) = \text{sign}(w \cdot z + b) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i K(x_i \cdot x) + b\right) \quad (7.6)$$

Several kernel functions have been used widely and successfully, such as, *polynomial kernel* with degree  $d$ ,

$$K(x_i, x_j) = (1 + x_i \cdot x_j)^d \quad (7.7)$$

*Gaussian RBF kernel* with parameter  $\sigma$ ,

$$K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2 / (2\sigma^2)\right) \quad (7.8)$$

and *sigmoid kernel* with parameter  $\theta$ ,

$$K(x_i, x_j) = \tanh(x_i \cdot x_j - \theta) \quad (7.9)$$

### 7.2.2 Multi-Class Classification

SVMs are designed for binary classification.  $k$ -class classification problems can be solved using a  $k$ -class SVM which constructs a decision function by considering all  $k$  classes altogether.  $k$ -class classification problems can also be reduced to a collection of binary classification problems through several strategies, among which one-versus-rest strategy (Weston and Watkins, 1998) and pairwise strategy (Hastie and Tibshirani, 1996) are often used.

The one-versus-rest method constructs  $k$  binary classifiers, one for each class. The  $n$ th classifier constructs a decision boundary between class  $n$  and the  $k - 1$  rest classes. A testing example is classified to the class for which the distance from the margin in the positive direction is maximal.

The pairwise strategy creates classifiers, one for each pair of classes. A majority voting is applied to make a decision for a testing example.

The comparison study in (Weston and Watkins, 1998) shows the methods above to solve multi-class classification problems produce roughly similar accuracy.

### 7.3 Experimental Analysis

A total number 360 texture images are used in the experiment. Those texture images are cataloged into 9 texture classes. A sample image of each group is shown in Figure 7.5. All images are in grayscale JPEG format, each containing 640 by 480 pixels. Detailed information about the image can be found in (Lazebnik *et al.*, 2005). For each image, a feature vector is created which include 41 texture features. Among those features, 32 are derived from DCT coefficients and 9 from the ZMs. For each texture class, 35 samples are used in the SVM training process and 5 samples are used as testing data.

The image data are classified using  $SVM^{light}$  (Joachims, 1999). We choose RBF kernel with the parameter  $\sigma$  of 0.001. The regularization parameter  $C$  is set to 1. We use one-versus-rest method to classify 9 textures, since it constructs much less classifiers than pairwise method (9 vs. 36 classifiers for each level of MDCT) and still achieves nice performance.



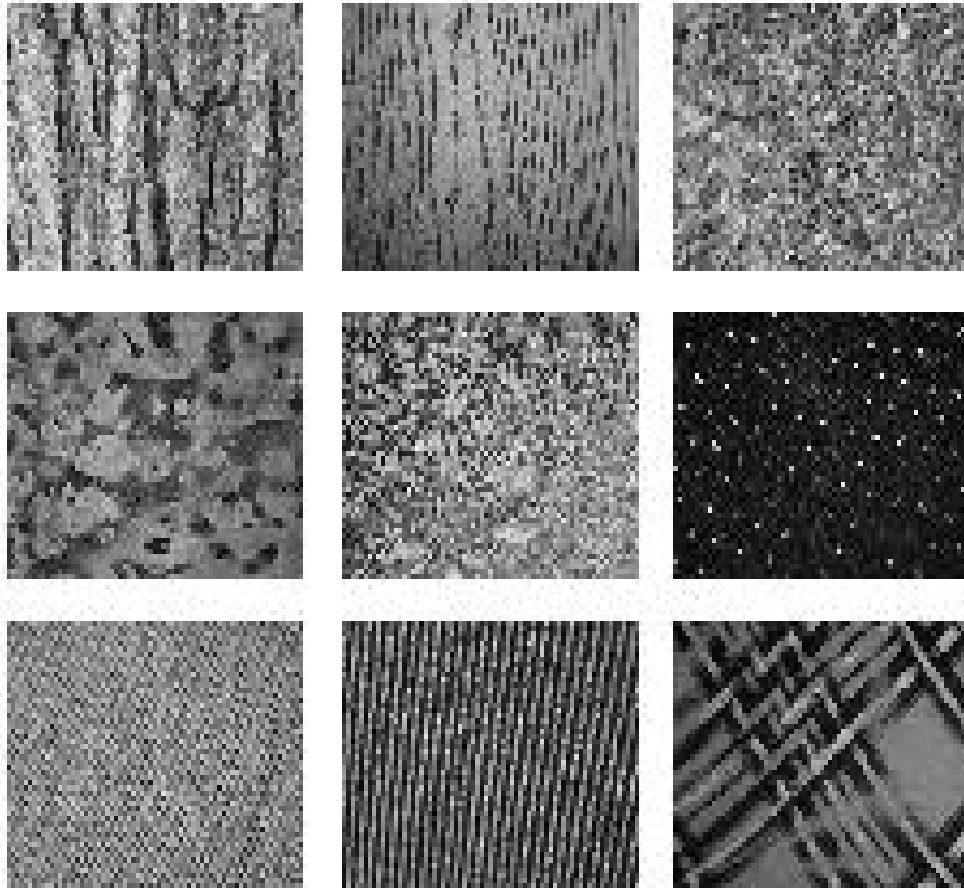


Figure 7.5 Experimental texture classes

Table 7.1 shows the testing accuracies of classifiers for the image data with the different level of resolutions. For each level of resolution, MDCTs is applied and 9 classifiers are created. The last column (with resolution of 1) reflects the standard DCT-II while the second column (with resolution of .125) reflects the MDCT2. In fact, to calculate the MDCTs, we can first create a low resolution image and then apply the standard DCT on it. The last row shows the testing accuracies by combining the decisions from 9 classifiers.

Table 7.1 SVM testing accuracies (%) of 9 classifiers using one-versus-rest method for the textures with different resolutions

<b>Classifier #</b>	<b>.125</b>	<b>.25</b>	<b>.5</b>	<b>.75</b>	<b>1</b>
1	95.6	95.6	97.8	97.8	100
2	91.1	95.6	97.8	97.8	97.8
3	97.8	100	95.6	95.6	95.6
4	100	100	100	100	100
5	95.6	95.6	97.8	97.8	97.8
6	100	100	100	100	100
7	91.1	91.1	100	100	91.1
8	88.9	91.1	95.6	95.6	95.6
9	100	93.3	93.3	93.3	93.3
<b>Combined</b>	<b>93.3</b>	<b>91.1</b>	<b>86.7</b>	<b>82.2</b>	<b>82.2</b>

From Table 7.1 we can see that there are no apparent differences among the average binary classification accuracies when the different image resolutions are applied. That is because the classifiers have the similar performance when used to answer whether an image belongs to a certain group. Given an image, to answer the question of which group it belongs to, we use the one-versus-rest method as discussed previously. The combined accuracies are shown in the last row. From the combined result, we found an interesting phenomenon: multi-class classification of texture images with low resolution achieves better classification performance. This is because the pixel intensity of the texture images with high resolution changes so slowly that the AC coefficients are all the zeros in most of blocks. In this case, the image features will be mainly reflected by its DC component

and the mean and standard deviation of the AC coefficients will be too ‘flat’ to distinguish themselves from the other texture classes.

The experimental results suggest that texture image features extracting from their low resolution images by MDCT can be used to achieve both higher classification accuracy and less computing cost. The result shows that low frequency DCT components combined with Zenike moments are sufficient to extract texture features for image classification. Since diseases can be detected by taking bacterial pictures from certain tissues which are infected, this method provides a new solution for disease diagnosis via imaging analysis. The combined accuracy in Table 7.1 suggests that DCTs from low resolution image will not only decrease the computation complexity but also increase the accuracy performance. This implies that our method can be used in a fast disease diagnosis system where a large number of texture images are to be classified.

## Chapter 8 Conclusions and Future Work

### 8.1 Conclusions

In this thesis, we have successfully built a contour based image structure for 3D reconstruction and partial retrieval. This structure is suitable for those images which can be described by the object contours. CBIS converts a raw image data set into the standard xml data structure. In the xml data structure, object contour is the basic unit to represent a 2D image. A corresponding contour node is generated in the xml structure for each object contour. The contour node also includes contour features such as contour length, moments, centroid, and so on. These features are one time calculated and can be easily retrieved from the xml structure. CBIS is different from raw image data set because this structure tells the relationship among pixels. Since object contour is the basic unit in CBIS, if two pixels reside in the same contour, these two pixels belong to the same object in a 2D image. If two pixels belong to two contours which are located on different image slices, we say these two pixels belong to the same 3D object as long as there is a path in the xml structure between them. Given an image stack, we can easily carry out the 3D segmentation by first converting the image stack into the xml image structure and then following the segmentation algorithms introduced in Chapter 2. To perform 3D partial retrieval from a specific pixel, we can first initialize the 3D component as the original contour node which includes this pixel, and then add all other nodes to the 3D component which have paths to the original node.

The first stage to build the xml image structure from an image stack is to break a 2D image into several objects. We use an optimum threshold method to binarize the 2D image and use an 8-connective tracer to outline the contour of the object. The optimum

threshold is to balance the total number of pixels located on the object contours and the average length of the contour. The 8-connective tracer simply travels along the object contour clockwise, starting from the top-left point of the object and stopping when it reaches the starting point. This segmentation method is suitable when the intensity differences between objects and backgrounds are apparent. When an image has an inhomogeneous background, the optimum threshold method will not be able to segment the image efficiently since no such global threshold exists. To solve this problem, we introduced a phase congruency method to highlight the object contours before applying optimum thresholded method. Phase congruency method is able to tell if a pixel is a significant one in the image which represents a point on an edge or in a corner. This method is based on measuring the phase similarity of the local Fourier Transform coefficient. The local Fourier Transform coefficients are those complex numbers to be summed in the inverse Fourier Transform to reconstruct an original pixel. By calculating all the phase congruency values of each pixel in the image, we obtain a phase congruency image based on the phase congruency values of the pixel. In phase congruency image, those pixels which have high intensity are considered to be the pixels located on an edge or in a corner. This method is insensitive to an image with an inhomogeneous background. This implies that object contours can always be highlighted regardless of the illumination condition of the original image. We overlap the phase congruency image with the original image to enhance the pixels locating on the object contour which can not be detected directly by an optimum threshold. We also design an edge filter to remove the noise created in the phase congruency image. The noise refers to those pixels which have been highlighted by the phase congruency method but are actually not

supposed to belong to any object edges. The edge filter is designed from the observation that a pixel on an edge will usually have different intensity comparing with the average intensity of its neighbors. This method is reasonable because for those pixels locating on an edge, their neighbor pixels fall into two categories: the background or the object. The average intensity of the background and object should always be different from the intensity of the object. Instead of the using the phase congruency method to highlight the object boundaries, we also tried the edge detection methods in spatial domain. We conclude that phase congruency method is a better choice in the CBIS.

The second stage to build CBIS is the grouping stage. In this stage, for each contour, we need to find out all its corresponding objects (if any) on its next layer image. To measure the similarity of two object contours, we build a fuzzy contour matching system to make an unbiased matching decision. FCMS takes into the consideration of various contour features including the average intensity of two contours, the overlapping ratio of two contours, and the principal axis of the contours. For each input, there are 3 linguistic variables (high, middle, low) are defined and a corresponding fuzzy membership function is used to map a crispy value into three linguistic variables. Since the system has three inputs and each input has three possibilities, there are 27 fuzzy rules in total. For output, there are 5 linguistic variables (high, high-middle, middle, low-middle low) which are used to map all the 27 fuzzy rules. Given a set of fuzzy input, all the applicable fuzzy rules will be fired and aggregated to the fuzzy output to make the matching decision. In FCMS, fuzzy membership functions are critical. The membership functions are tunable based on the density of the image stack and the illumination condition of the images. A genetic algorithm (GA) has been used to tune the fuzzy membership functions. In this

tuning algorithm, a chromosome includes 6 genes which reflect the overlapping ratio, difference of intensity, principal axis, and fuzzy output function setting. The fitness of the GA is measured by the number of wrong matching decisions, missed matching decisions and the number of matching decisions which are supposed to be made. A chromosome is selected for the next generation based on its fitness or its matching accuracy. The chromosome containing the best fuzzy sets is guaranteed to survive over the generation. The crossover point is randomly chosen among 6 genes of parent chromosomes. The mutation operator is performed by randomly selecting a fuzzy set and replacing it with a random value within its range. GA is suitable to refine the fuzzy member function automatically as long as a confirmed xml image structure can be used to define the fitness.

CBIS is also suitable for parallel image processing. We defined two parallel algorithms for creating the xml image structure and for 3D partial retrieval. Both algorithms are implemented using MPI. Given an object contour, we need to calculate the contour features and find out its corresponding contours on its neighbor slice. This process requires heavy computation and parallel implementation is desirable. In Contour Parallelization Algorithm (CPA), each processor can obtain its task by its rank and the contour index. Root processor collects the computation results from non-root processors and notifies the non-root processors about the processes of the entire task. There are only three messages sent out between each individual processor and root processor. In the Partial Retrieval Parallelization Algorithms (PRPA), root processor sends a message to assign the retrieval tasks to all individual non-root processors. Since a retrieval result is a sub-component of the whole xml structure, the processors can save the retrieval result as

a small xml file. The name of newly created xml file is determined by the query task which is also known by the root processor. Non-root processors only need to send an acknowledge message to the root messages when their query tasks have been finished. The root processor will be able to allocate the query results upon receive the acknowledgement message. In this algorithm, only two messages are passed between a non-root processor and root processor.

Both CPA and PRPA limit the messages between the non-root processor and the root processor. These two algorithms are suitable to be implemented using MPI where communication between processors is a major concern. The experimental results show reasonable speedups when the computation loads are large enough thus the chance of unbalanced task distribution has been minimized.

CBIS is designed for 3D reconstruction and partial retrieval from 2D image stacks. In most of the cases, adjacent image slices in an image stack are very close and our FCMS makes reasonable assumptions accordingly to simplify the contour matching task. One of the important assumptions is: if two object contours on the adjacent slices belong to the same 3D sub-component, they should have similar spatial features. Based on this assumption, fuzzy membership functions enforce the non-overlapping ratio and the difference of the principal axis less than certain values. The fuzzy matching system has been tested on several confocal crayfish neuron image stacks where the distance between two neighbor slices is about  $2.1\mu m$ .

FCMS is proved to be suitable in our test. For a general shape matching case, for example if the image slices are not close enough or two object contours come from arbitrary images, the shape matching algorithms need to take the consideration of



extracting contour features which are translation, rotation, and scaling invariant. In CBIS, since contour is the based unit in the xml image structure, we can convert a 2D object contour into a one-dimension complex vector. Given the contour coordinate sequence, it is converted into a complex vector by calculating the relative location of each pixel to the centroid of the contour. A Generic Fourier Descriptor can be used to extract invariant features from the contour. Since the transform is based on the center of the image, it is translation invariant. The rotation and scaling invariance can be achieved by dividing the DC coefficient with the area of the contour circled and dividing all the AC coefficients by DC.

CBIS is applicable for the crayfish neuron image stacks where the 2D images are untextured images which can be described by object contours. But for texture images such as many bacterial images, it is improper to describe the 2D images by contours. We defined a multi-level discrete cosine transform (MDCT) for texture image analysis. The DCT coefficient features combined with Zenike moments are used as texture feature vector. Texture feature vectors are trained by SVM to generate the classifier. MDCT can be implemented by applying standard DCT on the image with different level of resolutions. Given a texture image, we first divide the image into several sub blocks and apply the MDCT to each block. The mean and variance of the same frequency coefficient from all the blocks are calculated and added to the feature vector. If the sub-block size is small, all the coefficients can be used. Otherwise, only the low frequencies will be used. The reason we divide the image into sub-blocks is because in a texture image, certain patterns are repeatedly distributed in the image. Since image is blocked, some image features from a global perspective may be lost. Zenike moments are also used as image

features for classification. Zernike moments are taken from the whole image data set and have many desirable properties, such as rotation invariance, robustness to noise, and expression efficiency. After having the feature vector, One-versus-rest method is used to construct nine binary classifiers. That is, for each texture image class A, we created two image groups: one group contains images from class A and another group contains images from the other 8 classes. Those two image groups are trained by SVM to generate class A's classifier. The experimental results suggest that texture features extracted from its low resolution images by MDCT can be used to achieve higher classification accuracy with less computing cost. This implies that our method can be used in fast disease diagnosis system where a large number of texture images are to be handled.

## **8.2 Future Work**

This dissertation starts from a specific image processing problem of 3D reconstruction and analysis from image stack and is extended to several general pattern recognition problems including invariant shape matching and texture image classification. It is reasonable to predict that CBIS, FCMS, and the parallel algorithms also work for other image data sets such as MRI and CT data which are similar to the confocal neuron image stacks used in our research. More image data sets will be used to testify the robust of the system. Although the experiments demonstrate a promising performance, more research should be carried out to improve the proposed methods.

In FCMS, more object features could be included in the fuzzy inputs. New fuzzy input candidates can be the invariant features such as moments, and normalized Fourier Transform coefficients. CBIS can also be used for 3D shape matching and registrations which are the practical applications in plastic and dental surgery. Some preliminary work

has been carried out. Given two 3D surfaces, we have successfully converted the 3D surface registration problem into a non-linear least square problem by following the Interactive Closest Point method (Besl and McKay, 1992; Chen and Medioni, 1999; Zhang, 1994). To register one 3D surface to another, we need to find out the best transformation matrix:

$$\begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma & d_x \\ \sin\alpha\cos\beta & \sin\alpha\cos\beta\sin\gamma-\cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma & d_y \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (8.1)$$

In other words, we need to optimize the six parameters in the matrix:  $\gamma, \beta, \alpha, d_x, d_y$ , and  $d_z$  which represent the rotation and translation. We use the Levenberg–Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) in the optimization process. One constrain in ICP method is that it assumes that each point on one surface has a corresponding point on the other surface to be registered. The algorithm may bring a fault matching decision when this assumption is not valid. One possible solution is to select significant areas on both 3D surfaces and use the selected areas instead of the whole 3D surfaces to enforce the assumption of ICP validation. Furthermore, selected areas account for a very small portion of the whole surface and only limited vertexes will be used in the optimization process thus can reduce the computation burden. To automatically select the area of interest on both surfaces, we need to find out the significant regions on both surfaces.

In our previous research, the xml image structure makes it possible for 3D partial retrieval; MDCT and Zenike moments have been proved to be efficient for image feature extraction; FCMS/SVM gives us the model to combine multiple criteria for shape

matching. In the future research, automatic area selection in 3D surface registration can be carried out in the following steps:

- 1) 3D surfaces are segmented in the CBIS.
- 2) Invariant surface features are calculated for each 3D subcomponent.
- 3) Like 2D contour shape matching in FCMS, pairs of 3D subcomponents from the two surfaces are tested by SVM/FLS for their similarities.
- 4) The similarities of each pair of subcomponent are quantified. The selected areas on both surfaces are pairs of the subcomponents which have high similarities.

In the future, we will focus on how to extract invariant 3D features by comparing different methods. We will also design new matching/classification system for 3D surface classification.

## Bibliography

- Belkasim, S.O., Ghazal, A., and Basir, O. (1999). Correlated phase thresholding. *IEEE-EURASIP Workshop on Nonlinear Signal Processing (NSIP99)*.
- Belkasim, S.O., Ghazal, A., and Basir, O. (2000). Edge enhanced optimum automatic thresholding. *Proceedings of the 2000 international Computer Symposium*, Taiwan, 78-86.
- Belkasim, S.O., Hong, X., and Badir, O. (2004). Content based image retrieval using discrete wavelet transform. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(1), 19- 32.
- Belkasim, S.O., Li, Y., Dogdu, E, Hong, X., and Li, Z. (2004). Contented-based image retrieval in biological databases. *Int. Conf. on Computational Intelligence*, 512-515.
- Besl, P.J., and McKay, N.D. (1992). A method for registration of 3D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14 (2), 239-256.
- Boggress, A., and Narcowich, F.J. (2001). *A First Course in Wavelets with Fourier Analysis*, Prentice Hall.
- Boskovitz, V., and Guterman, H. (2002). An adaptive neuro-fuzzy system for automatic image segmentation and edge detection. *IEEE Transactions on Fuzzy Systems*, 10(2), 247 - 262.
- Bowen, J.M., Beeman, D. (1998). *The Book of genesis: Exploring Realistic Neural Models with the General Neural Simulation System*, New York: Telos, Springer-Verlag, Inc.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Patt. Anal Mach. Intell. (PAMI)*, 679-698.

- Carlbom, I., Terzopoulos, D., and Harris, K.M. (1994). Computer-assisted registration, segmentation, and 3D reconstruction from images of neuronal tissue sections. *IEEE Transactions on Medical Imaging*, 13(2), 351 – 362.
- Chan, F.H.Y, Lam, F.K., and Zhu, H. (1998). Adaptive thresholding by variational method. *IEEE Transactions on Image Processing*, 7(3), 468 – 473.
- Chen, W.H. and Pratt, W.K. (1984) Scene adoptive coder, *IEEE Transactions on Communications*, vol. 32, 225-232
- Chen, X., Harrison R., Zhang, Y. (2005). Genetic fuzzy fusion of SVM classifiers for biomedical data. *Proceedings of IEEE Congress on Evolutionary Computation (IEEE-CEC)*, 1, 654-659.
- Chen, Y., and Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing*, 10 (3), 145-155.
- Davies, E.R. (1997) *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press, New York, 171–191.
- Drebin, R.A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. *Comput. Graphics.*, 22(4), 65-74.
- Frei, W., and Chen, C.C. (1977). Fast boundary detection: A generalization and new algorithm. *IEEE Trans. Comput.* C-26(10), 988-988.
- Fu, X., Li, Y., Harrison, R., and Belkasim, S.O. (2006). Content-based image retrieval using Gabor-Zernike features. *The 18th International Conference on Pattern Recognition*, 417-420.

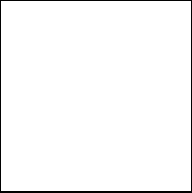
- Ginneken, B., Katsuragawa, S., Romeny, B., Doi, K., and Viergever, M. (2002) Automatic detection of abnormalities in chest radiographs using local texture analysis, *IEEE Transactions on medical imaging*, vol. 21, No. 2, 139-149
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc.
- Gonzalez, R.C, and Woods, R.E. (2002). *Digital Image Processing (2nd Edition)*, Prentice Hall.
- Grigorescu, S.E., Petkov, N. and Kruizinga, P. (2002). Comparison of texture features based on Gabor filters. *IEEE Transactions on Image Processing*, 11(10), 1160-1167.
- Hamamoto, Y. and Uchimura, S. (1998). A Gabor filter-based method for recognizing handwritten numbers. *Pattern Recognition*, 31(4), 395–400.
- Hastie, T. and Tibshirani, R. (1996). Classification by pairwise coupling. *Technical Report, Stanford University and University of Toronto*, 1996.
- Huang, Y.L. (2005). A fast method for textural analysis of DCT-based image. *Journal of Information Science and Engineering*, 21(1), 181-194.
- Ji Q., Engel, J. and Craine, E., (2000) Texture analysis for classification of cervix lesions, *IEEE Transactions on medical imaging*, vol. 19, No. 11, 1144-1149
- Joachims, T. (1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press. <http://svmlight.joachims.org>.
- Kim, J., and Park, H., (1999) Statistical textural features for detection of microcalcifications in digitized mammograms, *IEEE Transactions on medical imaging*, Vol. 18, No. 3. 231-238

- Kortgen, M., Park G.-J., Novotni, M., and Klein R. (2003). 3D shape matching with 3D shape contexts. *The 7th Central European Seminar on Computer Graphics*, Budmerice, Slovakia.
- Kovese, P. (1999). Image features from phase congruency. *Videre: Journal of Computer Vision Research*, 1(3).
- Lazebnik, S., Schmid, C., and Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8), 1265-1278.
- Levenberg, K. (1944). A method for the solution of certain problems in least squares. *Quart. Appl. Math.* 2, 164–168.
- Li, H., Liu, G., and Zhang, Z. (2006). A new texture generation method based on Pseudo-DCT Coefficients. *IEEE Transactions on Image Processing*, 15(5), 2006.
- Malik, J., Belongie S., Leung, T. and Shi, J. (2001) Conotur and texture analysis for image segmentation. *International journal of Computer Vision*, 43(1), 7-21
- Min C., Cho S., Lim, K.W., and Lee, H. (1998). New adaptive quantization method to reduce blocking effect. *IEEE Transactions on Consumer Electronics*, 44(3), 768-773.
- Mamdani, E.H. (1974). Application of fuzzy algorithms for control of simple dynamic plant. *IEEE Proceedings*, 121(12), 1585-1588.
- Manjunath, B.S. and Ma, W.Y. (1996). Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8), 837-42
- Marquardt, D. (1963). An algorithm for least squares estimation on nonlinear parameters. *J. Appl. Math*, 11, 431–41.



- Meyers, D. and Skinner, S. (1992). Surfaces from contours. *ACM Transactions on Graphics*, 11(3), 228-258.
- Mix, D.F., and Olejniczak, K.J. (2003). *Elements of Wavelets for Engineers and Scientists*, Wiley-Interscience.
- Morrone, M.C., and Owens, R.A. (1987). Feature detection from local energy. *Pattern Recognition Letters*, 6, 303–313.
- Morrone, M.C., Ross, J.R., Burr, D.C., and Owens, R.A. (1986). Mach bands are phase dependent. *Nature*, 324(6094), 250–253.
- Pan, Y., Li, Y., Li, J., Li, K., and Zheng, S.Q. (2002). Efficient parallel algorithms for distance maps of 2D binary images using an optical bus. *IEEE Transactions on System, Man, and Cybernetics – Part A*, 32(2), 228-236.
- Pan, Y., Ierotheou, C.S., and Hayat, M.M. (2000). Parallel gain-bandwidth characteristics calculations for thin avalanche photodiodes on an SGI origin 2000 supercomputer. *Concurrency and Computation: Practices an Experience*, 16, 1207-1225.
- Quinn, M. (2004). *Parallel programming in C with MPI and OpenMP*, McGraw-Hill.
- Rui, Y., Huang, T., and Chang, S. (1999). Image retrieval: current techniques, promising directions and open issues. *J. of Visual Communication and Image Representation*, 10(4), 39-62.
- Salvi, J., Matabosch, C., Fofi, D., and Forest, J. (2007). A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5), 578–596.

- Sarti, A. Ortiz de Solorzano, C. Lockett, S. Malladi, R. (2000). Geometric model for 3-D confocal image analysis. *IEEE Transactions on Biomedical Engineering*, 47(12), 1600 – 1609.
- Smith, J.R. and Chang, S.F. (1994) Transform feature for texture classification and discrimination in large image database. *IEEE Inter. Conf. on Image Processing*, 3, 407-411.
- Szeliski, R., and Lavalley, S. (1996). Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines. *International Journal of Computer Vision*, 18(2), 171 – 186.
- Takagi, T., and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on System, Man, Cybernetics*, 15(1), 116-132.
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.
- Veltkamp, R.C. and Hagedoorn, M. (1999). State of the art in shape matching. *Technical Report, UU-CS-1999-27*, Utrecht.
- Wallace, G.K. (1991). The JPEG still Picture compression standard. *Communications of the ACM*, 35.
- Weaver, H.J. (1992). *Applications of Discrete and Continuous Fourier Analysis*, Krieger Pub Co.
- Weldon, T., Higgins, W.E. and Dunn, D.F. (1996). Efficient Gabor-Filter design for texture segmentation. *Pattern Recognition*, 29(12), 2005–2016.

- 
- Wu, W-Y, and Wang, M.J. (1999). Two-dimensional object recognition through two-stage string matching. *IEEE Transactions of image processing*, 8(7).
  - Zadeh, L. (1965). Fuzzy set. *Information and Control*, 8, 338-353.
  - Zhang, D., and Lu, G., (2002). Generic Fourier descriptor for shape-based image retrieval. *IEEE International Conference on Multimedia and Expo*, 1, 425-428.
  - Zhang, D., and Lu, G., (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37, 1-19.
  - Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *Int. Journal of Computer Vision*, 13 (2), 119-152.