A 3-D Visualization System for Serial Microscope Images

by

Jianping Li
B.S., Sichuan University, China, 1984
M.S., Sichuan University, China, 1987

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

DOCTOR OF PHILOSOPHY

in the Department of
Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

Dr. P. Agathoklis, Supervisor (Dept. of Electrical and Computer Engineering)

Dr. M. A. Stuchly, Departmental Member (Dept. of Electrical and Computer Engineering)

Dr. F. K. Li, Departmental Member (Dept. of Electrical and Computer Engineering)

Dr. P. Fisher, Outside Member (School of Health Information Science)

Dr. R. K. Ward, External Examiner (Dept. of Electrical Engineering, University of British Columbia)

Supervisor: Dr. Pan Agathoklis

ABSTRACT

A three-dimensional (3-D) visualization system for serial microscope images is developed with special reference to its application in microscopy and cell biology. The 3-D visualization system involves three process stages, namely data acquisition, volume data modeling and object rendering.

The data acquisition part deals with collecting serial microscope images and is carried out by optical sectioning which records serial microscope images from the top to the bottom of a specimen. A new algorithm is proposed to computationally remove out-of-focus information from each recorded image of a specimen. This algorithm processes serial images independently and thus avoids computationally expensive 3-D convolution and 3-D Fourier transforms. Further, an extensive study of imaging properties of defocused microscopes is carried out in the thesis. The defocused point spread functions and optical transfer functions of transmitted light microscopes have been analyzed. An extensive comparison of the two approaches of obtaining these functions, namely direct measurements and theoretical calculations, is conducted. An interesting observation is made that the results of these two approaches correspond well with each other only for low magnification and low numerical aperture objective lenses.

Modeling of serial microscope images is carried out by an isosurface modeling algorithm. This algorithm is based on the marching-cube algorithm with two modifications proposed in the thesis. One modification is to detect and prevent the redundancy existing in the original marching-cube algorithm. The other modification is to use the middle-point algorithm to avoid linear interpolation to locate the vertex of a polygon. Results show that the modified marching-cube algorithm proposed in the thesis significantly reduces the number of polygons generated and thus greatly increases the computation efficiency for surface generation and object rendering.

Object rendering, which is to generate a realistic image, is carried out by using a C library SImple Polygon Processor (SIPP). A graphic user-interface is designed and imple-

mented to facilitate the modeling and rendering processes. It offers various user-friendly functions, such as rotation, zooming and cutting, for close examination of the objects under study and can be used for visualizing various volume data. The interactive system together with the data acquisition algorithm form a 3-D visualization system for serial microscope images.

The results of visualization show that the 3-D visualization system developed in this thesis realistically and efficiently reconstructs objects of interest from serial microscope images as well as from various volume data such as Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) medical images.

Examiners:

Dr. P. Agathoklis, Supervisor (Dept. of Electrical and Computer Engineering)

Dr. M. A. Stuchly, Departmental Member (Dept. of Electrical and Computer Engineering)

Dr. K. F. Li, Department Member (Dept. of Electrical and Computer Engineering)

Dr. P. Fisher, Outside Member (School of Health Information Science)

Dr. R. K. Ward, External Examiner (Dept. of Electrical Engineering, University of British Columbia)

Table of Contents

List of Tables

List of Figures

Acknowledgments

I owe thanks to many people who have made this thesis possible.

First, I would like to thank my supervisor, Dr. Pan Agathoklis of the Department of Electrical and Computer Engineering of the University of Victoria, for his constant inspiration, academic supervision, his support, understanding and patience during the entire process of this dissertation.

I would like to thank my external examiner Dr. Rabab K. Ward from the University of British Columbia and my supervisory committee members, Dr. Paul Fisher, Dr. Maria Stuchly and Dr. Kin Li for spending their precious summertime to read and correct my thesis and for their academic advice. I also would like to thank my previous supervisory committee members, Dr. Ged McLean and Dr. Fayez EL-Guibaly, who could not come to my defence because they were on study leave, for their academic advice in the early stage of this thesis.

I would like to thank Mr. Garry Jensen, Dr. Fred Peet and Dr. Tara Shahota of Pacific Forestry Canada in Victoria for their constant support and the knowledge of cellular biology and microscopy that they provided.

I would like to thank Mr. Tom Gore, Dept. of Biology of University of Victoria for his generosity by letting me access the department microscope facilities and giving me the opportunity to the demonstrations of volume visualization packages from vari-

ous companies.

I would like to thank Mr. Charles Card for his kindness and generous help with proofreading the first draft of my thesis, which was hardest to read, at his busiest time. I would like to thank many friends who have proof read my thesis. They are Dr. O'Grady, Mrs. O'Grady, Mr. Inderpreet Singh, Mr. Srikanth Subramanian, Mr. Graham Cooke and Dr. Ping Xue.

I would like to thank many friends and fellow graduate students, especially those in Micronet, for their encouragement, useful academic discussions and suggestions during the course of this thesis.

The financial support from my supervisor, from Micronet, and from the BC Science Council are also gratefully acknowledged.

Finally, I would like to thank my family. Without their moral and physical support, I cannot imagine how I could have ever finished this dissertation. Many thanks are owed to my parents who always sacrifice themselves for me to accomplish my dream. Thanks to my husband for his support, encouragement and understanding. Thanks to my daughter who unconsciously sacrificed. Thanks to my brothers, sister-in-laws, nephews and for their support and encouragement. Thanks also go to my father-in-law and mother-in-law who always pay great attention to education. It was unfortunately that my mother-in-law was not able to wait till my thesis completion.

After all, without all these people, this thesis would not be possible!

Dedications

To my father and mother

To my husband and daughter

Chapter 1

Introduction

1.1 Three-dimensional Visualization

Three-dimensional (3-D) visualization is a method for extracting meaningful information from volume data and of using techniques of computer graphics to create a realistic 3-D like image of objects of interest. This method provides mechanics for peering into the structure of objects to understand their complexity and dynamics. It is an effective way to gain insight into complex structural details of objects and of the spatial relationships among them when actual viewing of the object is impossible. Three-dimensional visualizations are very useful in many applications, such as medicine [20] [28] [59] [83] [85], geoscience [44], astrophysics, chemistry [58], microscopy [53], mechanical engineering [39], non-destructive testing and many other scientific and engineering areas [69] [74].

Medical applications are well-known examples of 3-D visualization. Images from Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) have greatly increased the information available to the radiologist. Diseases can be diagnosed by interpolating the serial cross-sections of CT or MRI images, with a consequent improvement in diagnosing performance. However, for a proper treatment of the patient, the radiologist needs to describe the diagnosis to the others. Verbal descriptions of the findings may be difficult and may complicate the problem. Three-dimensional visualization generates realistic 2-D images of the 3-D objects of interest. It can offer the radiologist not only a more objective means of diagnosing diseases, but also an

interface to communicate with physicians.

Application of 3-D visualization is also important in microscopy and cell biology [53]. Plant and insect tissues are complex organizations of different cell types arranged in a three-dimensional array. Visualizing this structure is a key component in understanding functional relationships among cells and tissues. Actual visualization of cells and tissues is limited by the *depth-of-field* of a microscope. Anything outside the depth-of-field is blurred, which leads to the infeasibility of viewing the 3-D structure. The thicker the specimen is, the greater is the blur and the more difficult it is to view the 3-D information. One common practice, used in the past and even today to visualize 3-D structures, is to physically section a thick specimen into very thin slices or use an expensive confocal laser scan microscope to obtain thin sections, and then to mentally interpolate serial microscope images of these slices. This mental interpolation is very subjective and often very difficult due to the complexity of the structure. In contrast, 3-D visualization offers possibilities of reconstructing the 3-D structures and helps biologists to gain insights into the objects through 3-D-like images. The 3-D visualization also offers features which are difficult to accomplish in actual visualization. Examples of these features are (i) transformation of the reconstructed structures to expose different aspects of the objects and (ii) cuts through the objects virtually at any desirable angles to reveal the internal structures of the object.

Three-dimensional visualization systems, especially economical ones that use only conventional microscopes and moderately equipped computers, for serial microscope images are still at a primitive stage. There are two main obstacles: the lack of effective and efficient techniques for collecting serial microscope images of a thick specimen without destruction or distortion; and the lack of efficient algorithms to reconstruct the objects from the massive amount of data collected. Three-dimensional visualization in microscopy is important and has wide potential for applications; thus it has been a rapidly growing area. The aim of this thesis within this evolving field is to develop a 3-D visualization system for serial microscope images obtained from trans-

mitted light microscopes. This thesis focuses on data acquisition for collecting serial microscope images and modeling of the large amount of data involved in the visualization. A close examination of imaging properties of a microscope and existing approaches leads to the proposal of an optical sectioning algorithm (the partial-minimization-and-constraint-iterative algorithm) and two modifications of the marching-cube algorithm (the redundancy removal algorithm and the middle-point algorithm). The visualization system developed is efficient and gives satisfactory results for the applications studied.

1.2 A 3-D Visualization System

This section presents a general visualization system independent of its application area. Fig. 1.1 is an illustration of a general process flow of a 3-D visualization system, which usually can be divided into three main processes: *data acquisition, data modeling and representation*, and *rendering*. A 3-D visualization system first collects volume data, then models the volume data and outputs a proper representation of objects, and sends the represented object to the rendering process to create a 3-D realistic look image.

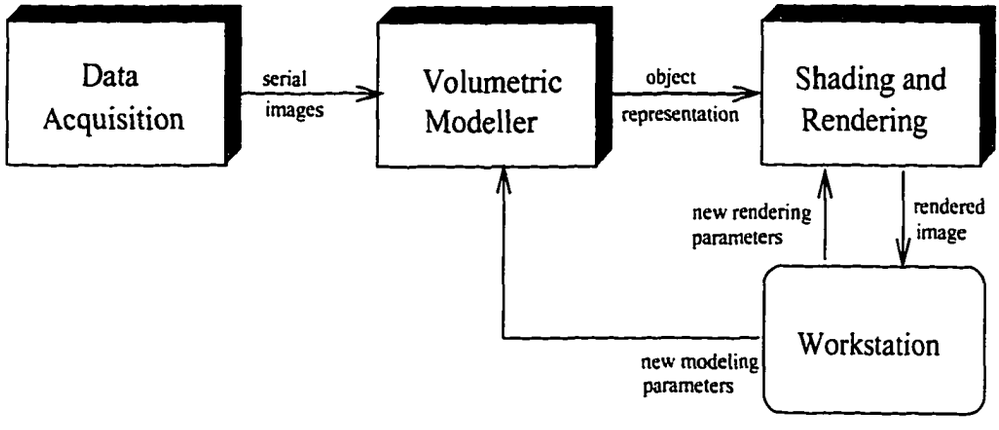


Figure 1.1 The process stages of a 3-D visualization system.

Data Acquisition: Data acquisition of a 3-D visualization system is the process of obtaining volume data which is defined on a 3-D lattice consisting of identically sized, tightly packed parallelepipeds as illustrated in Fig. 1.2(a). One or more values are assigned to each grid point of volume data. Each parallelepiped is called a volume element or *voxel*. Common values assigned to grid points in volume data are related to intensity, density, pressure, temperature, electrostatic charge, velocity etc.

A volume dataset can be collected by voxelizing a geometric description of objects [25], or can be acquired by stacking sequential images as illustrated in Fig. 1.2 (b), or by other means. Volume datasets can be treated similarly as a 3-D array despite the fact that they are acquired from different resources.

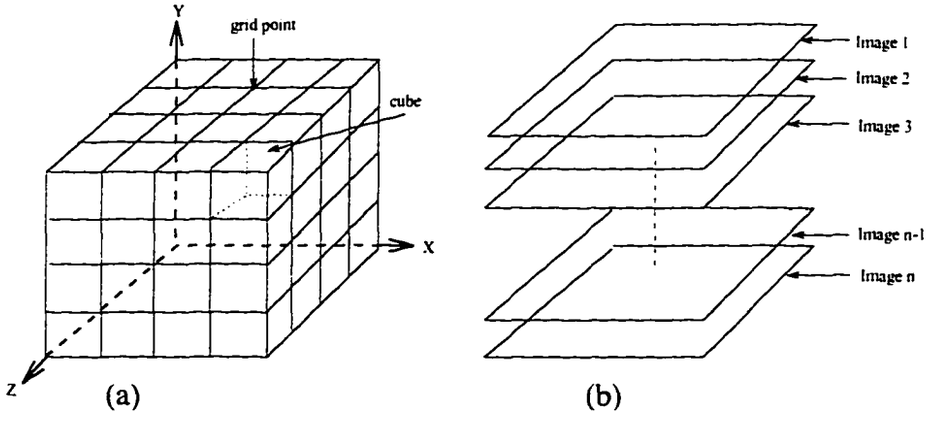


Figure 1.2 A volume data lattice (a) and its formation by stacking sequential images (b).

Volume Modeling: The purpose of volume modeling is to extract features relevant to the reconstruction of objects of interest and to produce a representation of features extracted from volume data. One modeling example is to detect object surfaces and to use polygon meshes to fit object surfaces [52]. One characteristics of volume visualization is that massive amounts of data are usually involved. Volume data often consist of

hundreds and thousands of megabytes and even gigabytes. Massive data of this kind mean a tremendous demand for computer resources. Therefore, efficiency is always an issue in volume modeling.

Rendering: Rendering is the process of producing and displaying realistic images of objects of interest from volume modeling after giving properties of object surfaces and the lighting environment surrounding them [25]. The rendering process includes shading surfaces, determining the visibility of surfaces and displaying them.

A simple example of illustrating the process of a 3-D visualization system is as follows. Data acquisition is accomplished, in this example, by voxelizing the saddle function, given in Eqn. 1.1, into a 49 x 49 x 49 array.

$$f(x,y,z) = \begin{cases} 1, & \text{if } \frac{x^2}{a^2} - \frac{y^2}{b^2} - 2z = 0 \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

where a and b are user-defined constants. The resulting volume data in serial image form, with every third image in the volume being shown, are presented in Fig. 1.3. Each image is a 2-D array of the saddle function at the x-y plane with a certain z value. From the sequence images in Fig. 1.3 alone, the shape of the object is relatively difficult to visualize.

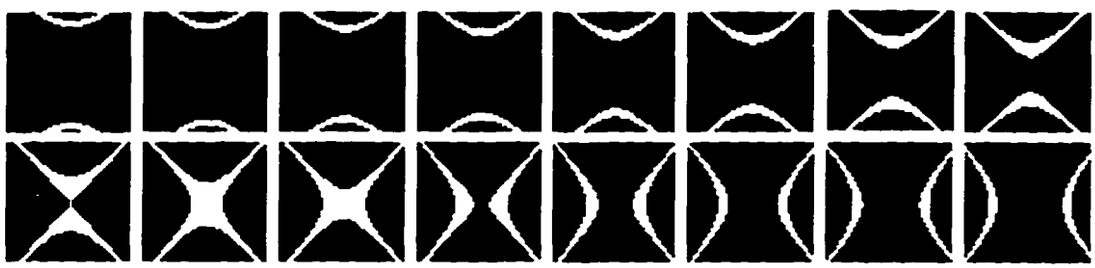


Figure 1.3 The sequential images, i.e., volume data, obtained by voxelizing a saddle function.

In this example, thresholding is used as part of volume data modeling to extract the surface of the object. Sample points whose values are above the threshold value are assumed on or inside the object surface, and those whose values are below the threshold value are outside the surface. Polygon meshes are used to connect all those points which are assumed to be *on* or *inside* the surface. The represented surface is the wire frame of the polygons generated.

The polygon meshes are passed to the rendering process for shading and display. The rendered object surfaces are presented in Fig. 1.4 which results from four different viewing positions.

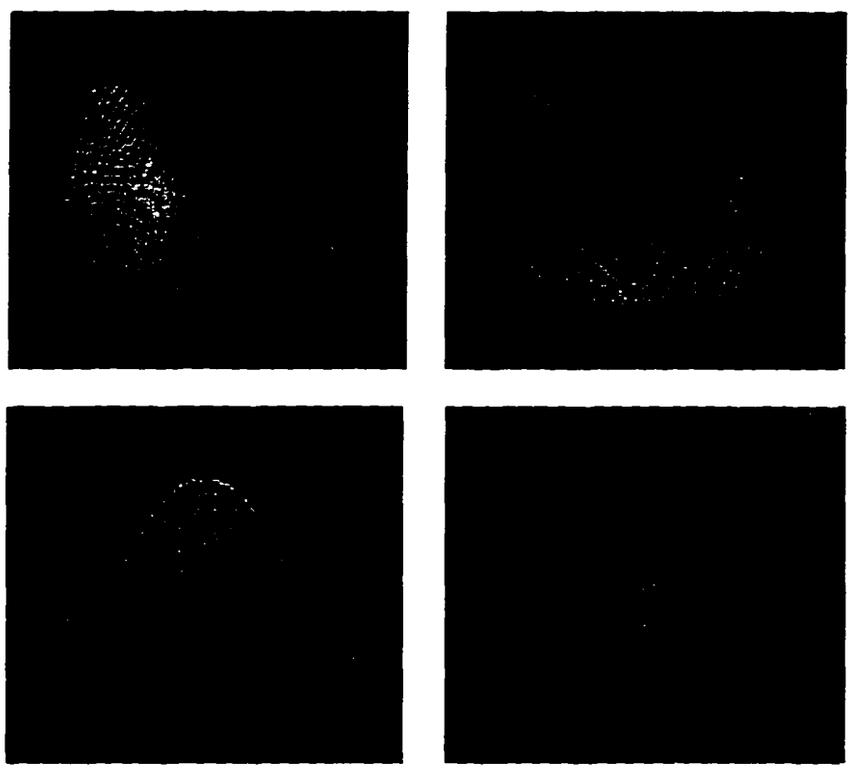


Figure 1.4 The rendering results of the saddle function represented by polygon meshes. Four different viewing positions were used to give different aspects of the object.

The top left one in the figure is the result of viewing from a diagonal position and the remaining three are from the far side along one of the three main axes respectively. These rendered images are more intuitive in terms of understanding the shapes of the saddle function. In this example, one mechanism of a 3-D visualization system is presented namely that it can reveal different aspects of the object by defining different viewing positions.

1.3 Visualizing Serial Microscope Images and Contributions of the Thesis

This thesis aims at developing a 3-D visualization system for serial microscope images, with special reference to its application in microscopy and cell biology. Data acquisition of collecting serial microscope images comes as a challenge at the very beginning step. One conventional way to obtain serial microscope images is to image physically sectioned specimens [10] [96]. Obviously, this brutal method is not suitable for live cells in a specimen. Furthermore, errors such as curling and compression may occur because the sliced specimens are very thin. The most critical disadvantage of this method is that the registration among the slices will be missing after physical slicing. Due to the very small scales of specimens, it is almost impossible to induce marks for alignment before slicing the specimens. An erroneously aligned slice will affect the final reconstruction results tremendously. An alternative data acquisition method involves the use of a Confocal Laser Scan Microscope (CLSM) [53] [82]. A CLSM is a specialized microscope which uses a very small pinhole to block out-of-focus light from a specimen and only collects the information from a focal plane. The focal plane is moved from the top to the bottom of a specimen, thus a CLSM produces serial microscope images of a thick specimen without losing their registration. However, a CLSM is very expensive and is usually only used in large laboratories. This greatly limits the application range of a visualization system. Another disadvantage is that a CLSM uses

a laser as a light source, which in most cases is so strong that it harms and even destroys the specimen, especially live cells.

Thus, much recent research has been focusing on optical sectioning to collect serial microscope images. Optical sectioning records serial microscope images by moving the object stage of a microscope so that the focal plane is moved through a thick specimen. Each recorded image contains in-focus information from the focal plane and out-of-focus information from the remainder of the specimen. Much of the out-of-focus information can be removed computationally by optical sectioning algorithms *after* images are captured [2] [12] [13] [22] [42]. The principle of optical sectioning is similar to that of a CLSM except that the out-of-focus information is removed mathematically rather than mechanically or optically by a small pinhole. Using optical sectioning for data acquisition is much less expensive than using a CLSM microscope. Further, it has the advantage of being not harmful to living cells since it uses conventional light instead of a laser beam. Existing optical sectioning algorithms are either not effective for most microscope images or they are computationally so expensive that specialized computers are required. In this thesis, a new optical sectioning algorithm, the *partial-minimization-and- constrained-iterative algorithm (PMCI)*, is developed to accomplish the data acquisition process. The principle of the algorithm is to minimize the errors between the observed images and the estimated images by estimating the object intensity functions on the focal plane. The object intensity functions are obtained when the minimization is reached. The PMCI algorithm offers several advantages. First it processes images independently, no 3-D convolution and 3-D Fourier transform are required resulting in tremendous reduction on memory requirements and computation time, and makes it possible to be implemented on a moderately equipped computer. The algorithm also offers the flexibility in choice of a number of planes above and below the focal plane involved in the minimization. Thus the trade-off between computation complexity and approximation of the algorithm can be controlled by the user. The algorithm decomposes the 3-D problem, which is common in optical

sectioning algorithms [2] [12] [22] [42], into a series of 2-D problems and thus available 2-D algorithms can be used. The proposal of the PMCI algorithm is based on the close examination, conducted in the thesis, of the defocused imaging properties of a microscope.

The defocused imaging properties are usually described by their defocused point spread functions (PSF) and optical transfer functions (OTF) [43] [81]. The focus and defocused PSF or OTF are also used to determine the quality of an optical system. They are very important functions for optical design, testing and evaluations. Two approaches are commonly used to obtain the defocused PSF and OTF. One is through mathematical models [13] [43] [81] and the other one is through direct measurement [3] [80]. A properly measured PSF reflects the conditions of the microscope used and thus is often preferred [2] [12]. However, measuring the PSF of a transmitted light microscope is relatively difficult due to problems in finding a proper point light source. Even though such a point light source is available, there are several technical issues, such as the low signal-to-noise ratio due to very small size of the point light source. As a result, theoretical approaches are preferred in some cases [22] [42]. However, a proper comparison of the two approaches has not been adequately studied in the literature. This motivated an extensive study of the two approaches besides examining the imaging properties of a microscope. Results of defocused PSFs and OTFs from both approaches show that a defocused transmitted light microscope tends to discriminate high frequency information and thus introduce haze into an image. The maximum intensity of a defocused PSF drops rapidly with increasing defocus, especially for high numerical aperture lenses. This indicates that a defocused object contributes less to a recorded image with increase of defocus. This property leads the PMCI algorithm to use the recorded image with a focal plane at level i in a thick specimen to estimate object intensity function on the focal plane. The comparison between the two approaches shows that the defocused PSF and OTF from experimental approach agree with those from theoretical approach for low numerical aperture lenses. However, for

objective lenses with high numerical aperture, these two approaches produce different results in terms of their diffraction patterns and sizes as will be discussed later. The theoretical approach produces more diffraction rings and larger diffraction patterns, which implies more haze in final images.

The second step of a 3-D visualization system is modeling the volume data collected. Ad hoc modeling approaches in the early stage of volume visualization usually involve manual or automatic boundary extraction of objects of interest in each image [1] [10] [96]. These extracted boundaries are stacked and connected using polygon meshes. One problem associated with this approach is that manual boundary extraction involves a considerable amount of human effort and techniques for automatic boundary extraction are not mature enough. Another problem with this approach is the so-called branch problem: if there is more than one boundary extracted from a slice, users will have to interactively intervene in the process so as to overcome the ambiguity regarding which boundary to connect. This branch problem is particularly undesirable for an automatic visualization system. More advanced volume modeling algorithms, such as those called volume rendering modeling, are effective for visualization of serial microscope image reconstruction in that they reconstruct the objects of interest automatically [20] [49] [76] [87]. However they are not necessarily efficient for serial microscope images because they usually require specially equipped computers due to their intensive computation and large memory space requirements. The ultimate goal for visualizing microscope images, especially cell images, is to visualize object shapes and their spatial relationships. Object shapes can be well represented by their surfaces which usually possess only a small number of voxels in volume data. In view of these characteristics, a high resolution surface modeling algorithm, the marching-cube algorithm, is chosen. The algorithm uses a threshold value to detect object surfaces and uses polygon meshes to patch the isosurface detected. This algorithm does not require excessive computer memory as in volume rendering modeling.

Two efficiency improvements based on the marching-cube modeling are pro-

posed in this thesis. One is the redundancy removal algorithm whose purpose is to detect and prevent the generation of redundant polygons by the original algorithm. The redundancy problem, as discovered in this thesis, occurs when a sample point is considered to be *on* the surface. The other improvement is called the middle-point algorithm, which is to avoid linear interpolation for polygon vertices for each polygon generated. Results show that these two improvements significantly reduce the number of polygons generated and thus increase the efficiency in terms of computing time and memory.

One of the advantages of using surface modeling is that it can use existing software and hardware for polygonal rendering in computer graphics. In this thesis, a C library polygon rendering package - SIPP - is used to accomplish the rendering process.

A user friendly interface is established to integrate the modeling and rendering processes. The user-friendly system allows a new user to learn basic commands faster and to start doing productive work without going through complicated start-up procedures. The reason for not linking the data acquisition process into the user interface is that the integrated modeling and rendering can be used as a general volume data visualization system without the knowledge of the data field, as shown in Chapter 8. On the other hand, the data acquisition part can be used to deconvolve any microscope images for image quality enhancement rather than for visualization purposes.

The hardware of a 3-D visualization system can be simple, consisting of only a device for data acquisition plus one computer for data processing, modeling and rendering. Fig. 1.5 shows the hardware used in the 3-D visualization system in this thesis. The microscope is equipped with a CCD camera and digitizer (frame grabber), and the computer is used for optical sectioning, modeling and rendering.

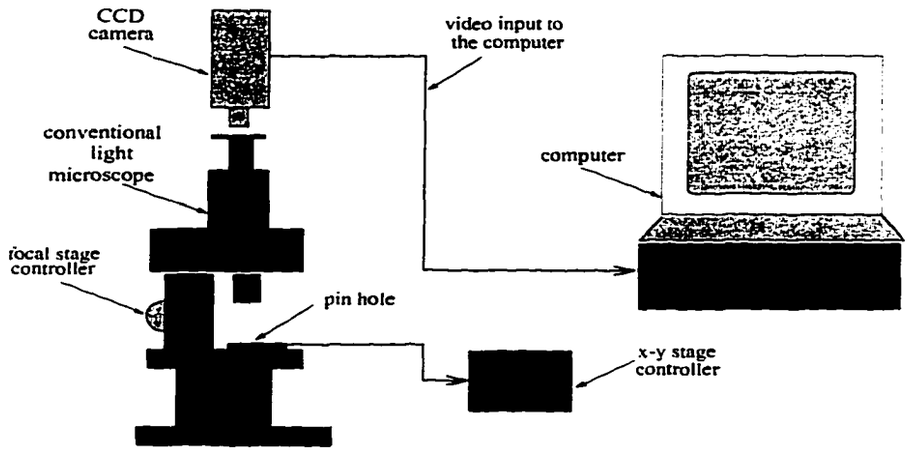


Figure 1.5 The hardware of the visualization system used in this research.

In summary, the 3-D visualization system developed in this thesis uses the proposed PMCI optical sectioning algorithm for data acquisition. An efficient surface modeling algorithm, the modified marching-cube algorithm, is used for modeling. Rendering is accomplished by the SIPP C library. The contributions of the thesis can be summarized as the following four aspects; First, a new optical sectioning algorithm is developed which has the advantages over existing ones in terms of significant reduction in computational cost and an additional freedom to control the algorithm. Second, the thesis intensively studies the defocused imaging properties of a microscope with a proper comparison between theoretical and experimental results. Third, it proposes two efficiency enhancement algorithms for the existing modeling, which substantially reduce a number of polygons generated, thus resulting in a significant increase of the computing efficiency. Finally, the complete system developed itself is a contribution. As an innovation, it applies visualization techniques to the micro-object area and offers a better understanding of some central properties of visualization of serial microscope images.

1.4 Organization of the Thesis

The chapters of the thesis are organized according to the process stages of the visualization system, namely data acquisition, modeling and rendering. Chapter 1, as presented here, is a brief introduction of a general visualization system and the system for serial microscope images being established in this thesis. Chapters 2 and 3 discuss the data acquisition process. Chapter 2 concentrates on the defocused imaging properties of a microscope. The theoretical and experimental approaches for obtaining defocused PSF and OTF are presented, with a detailed comparison of these two approaches. Chapter 3 concentrates on the algorithms of optical sectioning. The proposed PMCI algorithm is presented which is based on the results from Chapter 2. In Chapters 4 and 5, discussions correspond to the modeling process. In Chapter 4, various volume modeling techniques are discussed in terms of their suitability for visualizing serial microscope images. The proposed efficiency improvements of an isosurface algorithm are presented in Chapter 5. Chapter 6 corresponds to the rendering process. The general steps of rendering 3-D polygon meshes to realistic images are briefly presented. The design and implementation of the user-interface for the visualization system are presented in Chapter 7. The visualization results from various sources, such as medical images, fibers and cells are presented in Chapter 8. The final chapter, Chapter 9, summarizes the thesis and discusses possible future work. A graphical plot of the organization of the thesis is shown in Fig. 1.6 where the upper rows and the bottom squares show the chapters and the processing stages respectively. Their correspondences are indicated by the dashed arrows. The solid arrows indicate the flow in the thesis and the system.

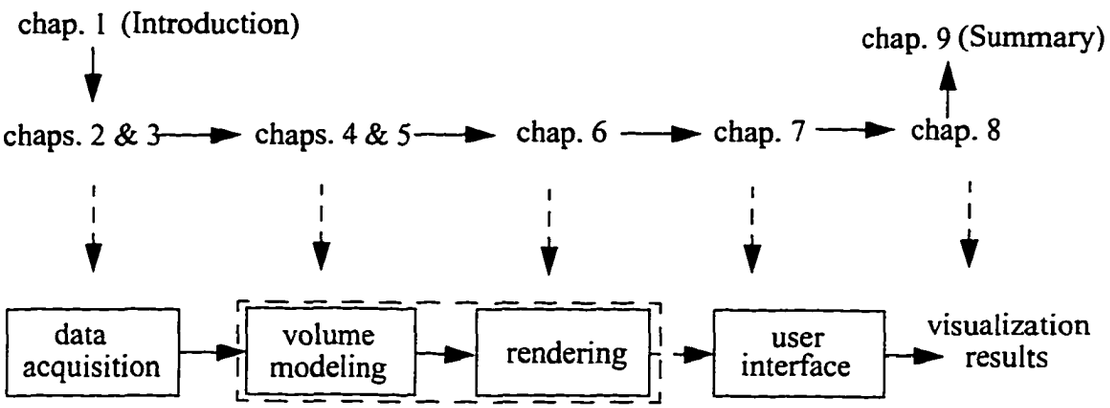


Figure 1.6 The correspondence of the thesis chapters with the process stages of the visualization system.

Chapter 2

Volume Data Acquisition I --- Defocus Imaging Properties of a Microscope

Defocused imaging properties of a light microscope provide essential information in optical sectioning for volume data acquisition which is the very first step of the visualization system. These properties determine the contributions of a 3-D object whose components are at various amount of defocus, to a recorded 2-D image. Thus, it is possible to use algorithms of optical sectioning to remove out-of-focus components and to obtain the object information at the focal plane from the recorded image.

Imaging properties of a microscope are usually determined by its point spread functions and optical transfer functions. A point spread function (PSF) is the response of an optical system to a point light source. An optical transfer function (OTF) is the Fourier transform of a PSF. The defocused PSF and OTF refer to responses of an optical system with a small amount of defocus. The focused PSF and OTF are special cases in which the defocus amount is equal to zero. Since any object can be considered as a weighted function of a large number of point light sources, the specification of the PSF or OTF of an optical system can determine the image of any object.

There are two main approaches in obtaining the defocused PSFs or OTFs. One is *direct measurement* which measures the response of an optical system to a very small

point light source at proper amount of defocus. The measured PSFs and OTFs are more faithful in reflecting the imaging conditions of a microscope. For most direct measurements, a fluorescent microscope with a fluorescent spherical bead as a point light source has been used [3] [40] [68] [79] [80]. Much less work has been done on transmitted light microscopes, which will be studied in this thesis. The other approach for obtaining PSFs and OTFs is by the determination of *mathematical models* from diffraction optics [13] [43] [79] [81]. A defocused PSF and OTF obtained from mathematical models are desirable when a point light source is not available. A defocused PSF and OTF obtained from mathematical models also avoid the noise problem and avoid the consequences arising from poor microscope adjustment and from a geometrically distorted point light source. The question of how well the mathematical models reflect the true PSF or OTF has not been adequately studied in the literature, only [79] referred to a comparison of the mathematical models with experimental approach but it considered only two examples and made no reference to the numerical aperture. It is thus necessary to evaluate the mathematical models based on comparisons of the results from mathematical models and from measurements.

In this thesis, both direct measurement and mathematical models are used to determine defocused imaging properties of a transmitted light microscope. Furthermore, a comprehensive comparison of these two approaches is conducted. From both approaches, results indicate that the defocus functions tends to discriminate against the high frequencies. Thus defocusing produces haze over images, which is consistent with common observations. The intensity of a defocused PSF drops dramatically with increasing defocus and makes a smaller contribution to the focal-plane image. Three criteria are used for evaluation of the mathematical models: namely, visual comparison in the space domain, comparison in the frequency domain and correlation coefficient computation in both the space and the frequency domains. It is observed that the results from direct measurements and mathematical models match very well for objective lenses of low numerical aperture. However, for objective lenses with high numerical apertures,

this is not the case. Possible factors for the disagreement, such as the magnification and numerical aperture of an objective lens, are discussed and the results indicate that numerical aperture is the main cause of the difference.

In Section 2.1, the mathematical models and their numerical calculation are discussed. In Section 2.2, the arrangement for direct measurement used in this thesis will be presented. The three evaluation criteria for the comparison of the two approaches are presented in the Section 2.3. Results from objective lenses used are presented in Section 2.4 and the differences for large numerical aperture lenses are discussed in Section 2.5.

2.1 Theoretical Approach - Mathematical Models

2.1.1 Mathematical models of in-focus and defocused PSF and OTF

Consider a simplified microscope as illustrated in Fig. 2.1. The distance between the objective lens and the image plane d_i is fixed by a manufacturer of a microscope. According to optical principles [9], the distance from focal plane to the objective lens d_f is

$$d_f = \frac{f d_i}{d_i - f} = \frac{M+1}{M} f$$

where f denotes the focal length of the optical system and M is the magnification of the system given as the quotient of d_i by d_f. Thus moving the object stage up or down about the focal plane corresponds to negative and positive defocus.

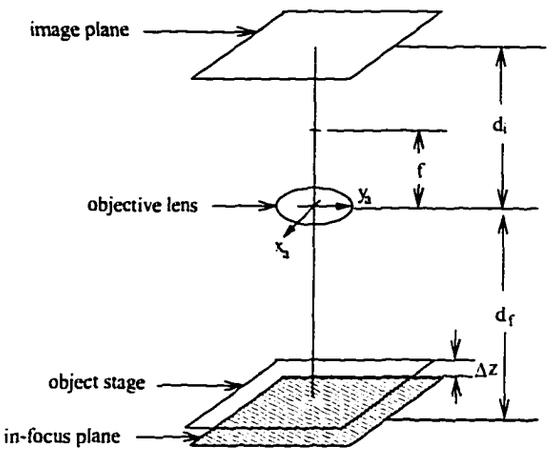


Figure 2.1 A simplified microscope system for illustration of defocus.

Consider a light source emitting incoherent light, i.e., all the point light sources vary in phase independently. For a focused optical system under incoherent illumination, diffraction optics theory implies that the system is linear in intensity, and the incoherent PSF is the power spectrum of the pupil function of the system [32] [43]. If $p(x_a, y_a)$ is the pupil function which is the spatial distribution of transmittance of the system aperture, then the PSF of the system is:

$$h(x, y) = \left| F\{ p(\lambda d_i x_a, \lambda d_i y_a) \} \right|^2 \tag{2.1}$$

where F represents Fourier transform, x_a and y_a are the coordinators on the pupil plane and λ is the wavelength of the light source. From the auto-correlation theorem, the normalized incoherent OTF is the normalized auto-correlation function of the pupil function [35], given by:

$$H(u, v) = \frac{R_p(u, v)}{R_p(0, 0)} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(\lambda d_i x, \lambda d_i y)\, p(\lambda d_i x - u, \lambda d_i y - v)\, dxdy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p^2(\lambda d_i x, \lambda d_i y)\, dxdy} \quad (2.2)$$

For a circular pupil aperture with diameter d, the focus PSF and OTF are [13]:

$$h(r) = \left[2\frac{J_1[\pi(r/r_0)]}{\pi(r/r_0)}\right]^2 \quad (2.3)$$

$$H(q) = \frac{1}{\pi}[2\beta - \sin(2\beta)]^{1} \quad (2.4)$$

where

$$r = \sqrt{x^2 + y^2},$$

$$q = \sqrt{u^2 + v^2},$$

and (x, y) are the spatial coordinates of PSF and (u, v) are the frequency coordinates of OTF. Further,

$$r_0 = \frac{\lambda d_i}{d},$$

$$f_c = \frac{d}{\lambda d_i} = \frac{1}{r_0},$$

$$3 = \cos^{-1}(q/f_c).$$

---

1. Eqn. 2.4 is given in Castleman [13] p. 360 and Agard [2]. Numerical evaluation of the integrals in Eqn. 2.2 indicates good correspondence between Eqn. 2.4 and Eqn. 2.2. The equation given by Castleman [13] for in-focus OTF on p. 263 does not lead to similar good numerical correspondence with Eqn. 2.2.

Since the transmitted light microscope uses incoherent illumination, all the PSFs and OTFs discussed in this paper will refer to those which use incoherent illumination.

2.1.2 Approximations

For a defocused optical system, the pupil function changes. For the same circular pupil aperture, the function now is [43]:

$$p(r) = \Pi\left(\frac{r}{d}\right) e^{j4kw\left(r^2/d^2\right)} \tag{2.5}$$

where $k = 2\pi/\lambda$, and *w*, the maximum defocus path length error, is given by:

$$w = -d_f - \Delta z\cos\alpha + \left(d_f^2 + 2d_f\Delta z + \Delta z^2\cos^2\alpha\right)^{1/2} \tag{2.6}$$

Δz is the defocus amount and $sin(\alpha)$ is equal to the numerical aperture *NA* divided by the refractive index η of the media between the specimen and the objective lens [48]. Substituting Eqn. 2.6 into Eqn. 2.1 and Eqn. 2.2, the defocused PSF and OTF can be obtained respectively. However, this is not straightforward and approximations have been proposed in the literature.

Hopkins shows that the defocused OTF of the system with a circular aperture given by Eqn. 2.5 can be approximated by [43]

$$\begin{aligned} H(u, v) = {} & \frac{4}{\pi a}\cos\left(\frac{ab}{2}\right) x \\ & \left\{\beta J_1(a) + \sum_{n=1}^{\infty}(-1)^{n+1}\frac{\sin(2n\beta)}{2n}\left[\, J_{2n-1}(a) - J_{2n+1}(a)\,\right]\right\} \\ & \frac{4}{\pi a}\sin\left(\frac{ab}{2}\right)\sum_{n=0}^{\infty}(-1)^n\frac{\sin(2n+1)\beta}{2n+1}\left[J_{2n}(a) - J_{2n+2}(a)\right] \end{aligned} \tag{2.7}$$

in which

$$f_c = \frac{1}{r_0} = \frac{d}{\lambda d_i} = \frac{2\eta \sin\alpha}{\lambda} = \frac{2NA}{\lambda} \tag{2.8}$$

$a = 4\pi wb$

$b = \frac{2q}{f_c}$

where J_n is the nth order Bessel function of the first kind. As can be seen from Eqn. 2.7, Hopkins' approximation involves a series of Bessel functions, which tend to converge slowly.

Stokseth approximates Hopkins' approximation of Eqn. 2.7 by the following form [81]:

$$H(u,v) = \left(1 - 0.69b + 0.0076b^2 + 0.043b^3\right) jinc\left[\frac{8\pi w}{\lambda}\left(1 - \frac{q}{f_c}\right)\frac{q}{f_c}\right] \tag{2.9}$$

$$jinc(x) = 2\frac{J_1(x)}{x}$$

Stokseth's approach is modified by Castleman with the intention to improve accuracy at small amount of defocus ($w \leq 5\lambda$) by substituting the in-focus OTF for the polynomial of Stokseth's approach [13].

$$H(u,v) = \frac{1}{\pi}(2\beta - \sin 2\beta) jinc\left[\frac{8\pi w}{\lambda}\left(1 - \frac{q}{f_c}\right)\frac{q}{f_c}\right] \tag{2.10}$$

The *jinc* function is equal to unity when its argument is equal to zero, and it is equal to zero when its argument is negative. The defocused OTF function *H(u,v)* has non-zero values only when $q \leq f_c$. The defocused OTF is thus a band-limited function with band cut-off frequency at f_c. The cut-off frequency is determined by the numerical aperture of the system and the wavelength used, according to Eqn. 2.8.

2.1.3 Numerical calculation

The defocused OTF approximations given in Eqn. 2.7, Eqn. 2.9 and Eqn. 2.10 are continuous functions in the frequency domain. Obtaining the defocused PSF for the corresponding lenses using Eqn. 2.1 is not straightforward. The approach used in this thesis is to discretize the OTF in the frequency domain and obtain the PSF using the discrete inverse Fourier transform. This approach can be implemented on a computer easily using existing software packages and the results are good approximations of the corresponding continuous functions provided that sampling theorem is satisfied. Consider sampling the OTF in the frequency domain in more detail. Since a 2-D defocused OTF is symmetric in coordinates u and v, the discussion will be based on the 1-D case. The two parameters needed for discretization are the sampling rate Δu in the frequency domain and the width of truncation window L_u which is the range of the OTF to be sampled (see Fig. 2.2). From sampling theory, the sampling rate Δx in the space domain and the range L_x (width of truncation window) of the resulting PSF will be related to Δu and L_u as in Table 2.1 (also see Fig. 2.2) [67]. In the table, L_u, L_x are the widths of truncation window, Δu and Δx are the sampling rate in the frequency and the space domains respectively, and N is the number of sample points. The table implies that when L_u and Δu (or N) are chosen, Δx and L_x follow immediately. The resulting PSF in the space domain will be a good approximation of the continuous PSF if the Nyquist conditions in both the space and frequency domains are satisfied which imply the following two conditions:

a. The sampling rate Δu in the frequency domain is sufficiently small so that the corresponding L_x covers the significant part of the PSF (the part of the PSF where it is not equal to zero, see Fig. 2.2).

b. L_u is equal to or larger than the bandwidth of the OTF function, i.e., $L_u \geq 2f_c$, so that the sampling rate in the space domain is less than the inverse of bandwidth of the function, i.e., $\Delta x \leq \frac{1}{2f_c}$.

Table 2.1. Summary of sampling rates and widths of truncation windows.

Parameter	Relations
Number of sample points	$N = \frac{L_x}{\Delta x} = \frac{L_u}{\Delta u}$
Space sampling rate	$\Delta x = \frac{L_x}{N} = \frac{1}{L_u}$
Frequency sampling rate	$\Delta u = \frac{L_u}{N} = \frac{1}{L_x}$
Width of truncation window (space)	$L_x = \Delta x \cdot N = \frac{1}{\Delta u}$
Width of truncation window (maximum computed frequency)	$L_u = \Delta u \cdot N = \frac{1}{\Delta x}$

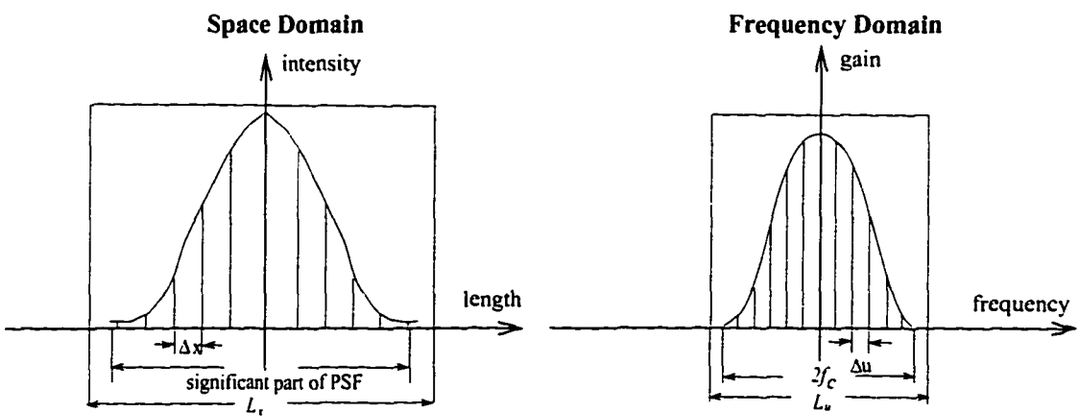
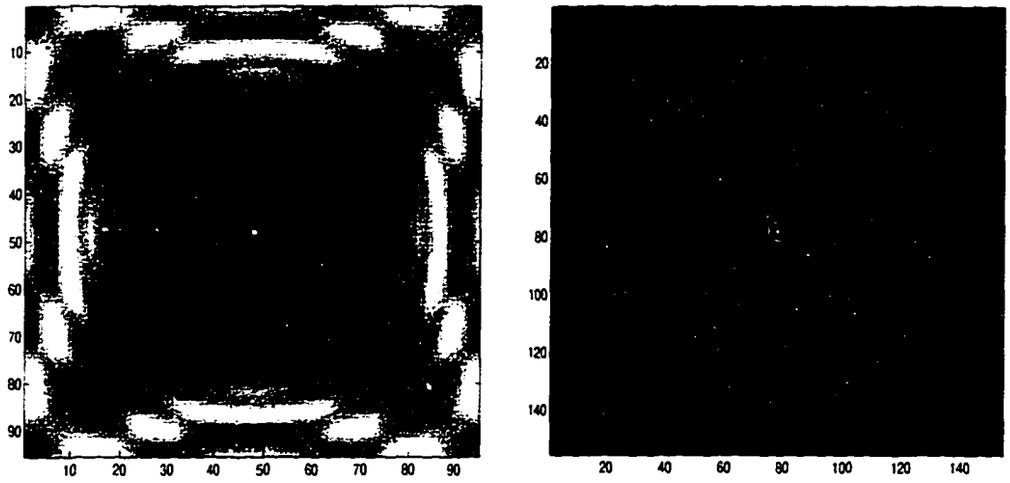


Figure 2.2 Relations between the space and frequency sampling rate and the widths of truncation windows in the space and the frequency domains.

If the condition (a) is not satisfied, then aliasing in the space domain will be observed because the resulting periodic PSF will create overlapping. To illustrate the aliasing, consider an example of discretizing the OTF for a 40x magnification objective lens with a 0.95 numerical aperture. The bandwidth L_u is chosen to be equal to $2f_c$. In Fig. 2.3, two defocused PSFs are presented which are obtained using different sampling rates Δu (related to N) in the frequency domain. In Fig. 2.3 (a), the number of sampling points is 95 and aliasing is obvious, while in Fig. 2.3 (b), the number of sampling points is 155 and no aliasing is present.



(a) (b)

Figure 2.3 Two examples of defocused PSFs obtained from the inverse Fourier transform of the defocused OTF using different sampling rates in the frequency domain. The axes correspond to number of pixels of defocused PSFs.

If condition (b) is not satisfied which means that $L_u = L_s < 2f_c$ as in Fig. 2.4, the periodic OTF will overlap and aliasing in the frequency domain will occur. However, if

the amount of overlapping is relatively small, this will not create severe aliasing. This is often the case with a defocused OTF where the gain is large only at low frequencies, as illustrated in Fig. 2.5, and overlapping high frequencies do not create noticeable aliasing.

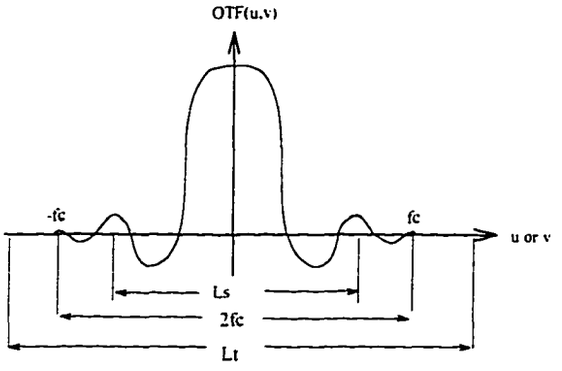


Figure 2.4 Relationship of the width of the truncation window and cutoff frequency of an OTF function.

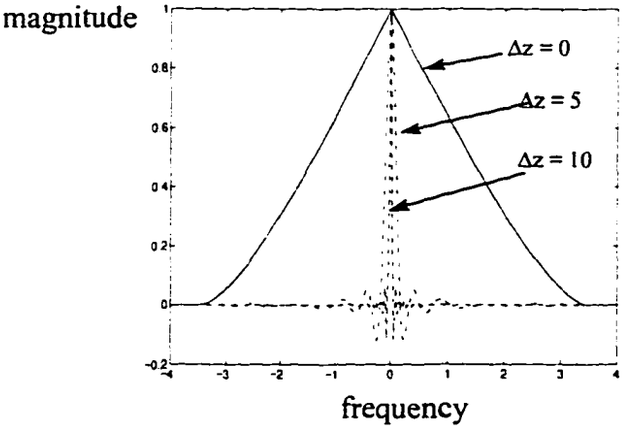


Figure 2.5 Defocused OTF results of the 40x, 0.95NA lens at various amounts of defocus. Gains for high frequencies decrease rapidly with the increase in defocus.

2.2 Experimental Approach - Direct Measurement

The principle of direct measurement is based on the fact that an image of a point light source is the impulse response of a microscope and therefore the image of a point light source at small amounts of defocus is the defocused PSF of the microscope. Thus, digitizing and recording this image will yield the discrete form of the defocused PSF. The experimental defocused OTF can be obtained by taking the Fourier transform of the defocused PSF obtained.

The basic configuration used for the measurement is illustrated in Fig. 1.5 of Chapter 1. It involves a transmitted light microscope, a pinhole, a stage controller, a CCD camera digitizer and a computer. Two microscope systems are used for measurement so that system setup errors if any can be detected. One system, system 1, which is exactly the same as illustrated in Fig. 1.5 of Chapter 1, is at the Pacific Forestry Centre in Victoria. The other system, system 2, is at the Department of Biology, University of Victoria. The two systems are fundamentally identical except that system 2 doesn't have a *x-y* stage controller. System 1 uses a Zeiss universal microscope and a SONY XC-77ce CCD video camera as a digitizer. On the microscope there is a monochromator in the light path which allows users to choose a desired wavelength. The focal stage controller is used to control the movement of the focal plane (the z-direction) with a resolution of 2μm. The *x-y* stage controller allows users to move the stage along *x-y* direction with a resolution of 0.5μm. System 2 also uses a Zeiss universal light microscope and a SONY CCD camera. Each division on the focal stage controller is 1μm. Users can choose a filter of desired wavelength to attach to the light source. Both systems use the light passing through the 1μm diameter pinhole as an object. The pinhole is in a thin (0.0025mm) stainless steel foil which is further mounted on a thin metal disc to prevent distortion, as illustrated in Fig. 2.6. Both microscopes are aligned so that all optical elements of the microscope are on the same optical axis before measurements are made [33]. The microscope is adjusted for Köhler illumination. For each objective lens, the aperture dia-

phragm is adjusted to ensure each objective is fully illuminated. The illumination is adjusted according to Zeiss's instructions.

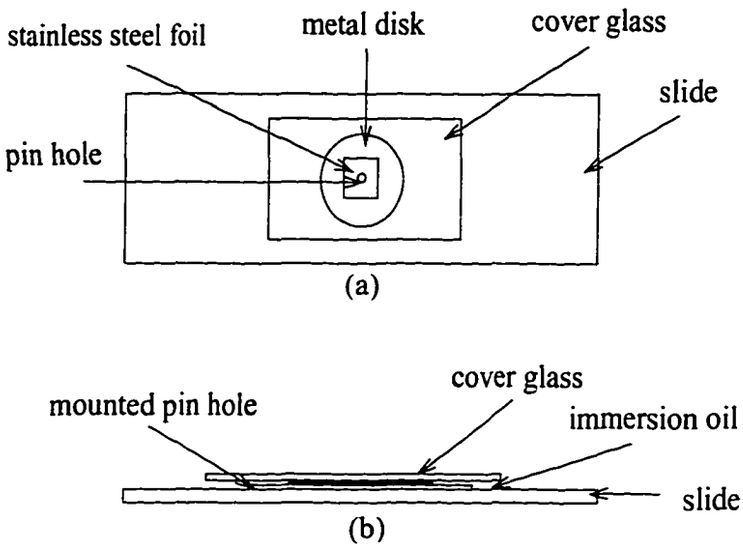


Figure 2.6 The mounted pinhole. Top view (a) and side view (b).

Digitizing an image is a process of sampling its continuous distribution by the CCD camera mounted on top of the image plane of the microscope. The sampling rate mapped to the object plane of the microscope is the *space sampling rate* Δx. The space sampling rate is calculated by dividing the size of the digitizer by the magnification of the objective lens of the microscope. However, it is more precise to measure the space sampling rate directly from experiments. For system 1, the space sampling rate is obtained by measuring the distance, at the object plane, that results from moving 512 pixels. A reference point on a specimen is located and placed on the left edge of the 512 frame on the monitor, and the *x-y* stage reading is set to zero. The reference point is then moved horizontally to the right edge of the 512 pixel frame with the joystick of the *x-y* controller. Each reading on the *x-y* stage controller is 0.5 μm. Thus the reading on the *x*-

y stage controller multiplied by 0.5μm is the actual distance of moving 512 pixels. The distance divided by 512 is the space sampling rate. In short, the measurement of the space sampling rate for system 1 is accomplished by fixing the number of pixels (512 in this case) and measuring the total size of 512 pixels on the object plane. For system 2, the measurement is made in the opposite way. The distance on the object plane is fixed by a micrometer, and the number of pixels within that fixed distance is measured. The micrometer consists of 100 bars with a total size of 0.01 mm. Thus the distance between two consecutive bars is 0.1μm. The measurement for the number of pixels is accomplished by using an image acquisition program called OPTIMAS[1] which allows a user to draw a line from one point to another on the monitor and measure the pixel difference between these two points. For a line drawn between two bars of the micrometer, the length of a line is the total number of bars within the line multiplied by 0.1μm. Thus the space sampling rate is the division of the length of the line by the measured number of pixels within the line. The measurements of the space sampling rate of the two systems are illustrated in Fig. 2.7.

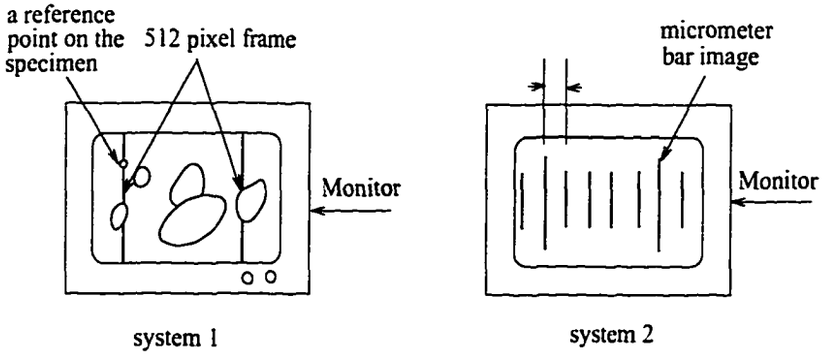


Figure 2.7 The measurements of the space sampling rate for the two systems. System 1 measures the distance of moving 512 pixels. System 2 measures the number of pixels within a line between two bars of a micrometer.

---

1. OPTIMAS is a commercial software product of Optimas Co.

The space sampling rate (resolution of the digitizer) is related to the resolving power of a digitizer as it records the details of objects. If the resolution of the digitizer is low, some details in images may be lost. An example is illustrated in Fig. 2.8. For the right digitizer cell, as also shown in Fig. 2.8 (b), there are two lines. The cell integrates the intensities of the two lines, and outputs only one intensity after digitizing. This corresponds to a loss of high frequency components of the image. If the spectrum of the image is as illustrated in Fig. 2.8(c), then a low resolution will introducing aliasing, and the spectrum of the digitized image will be as illustrated in Fig. 2.8(d).

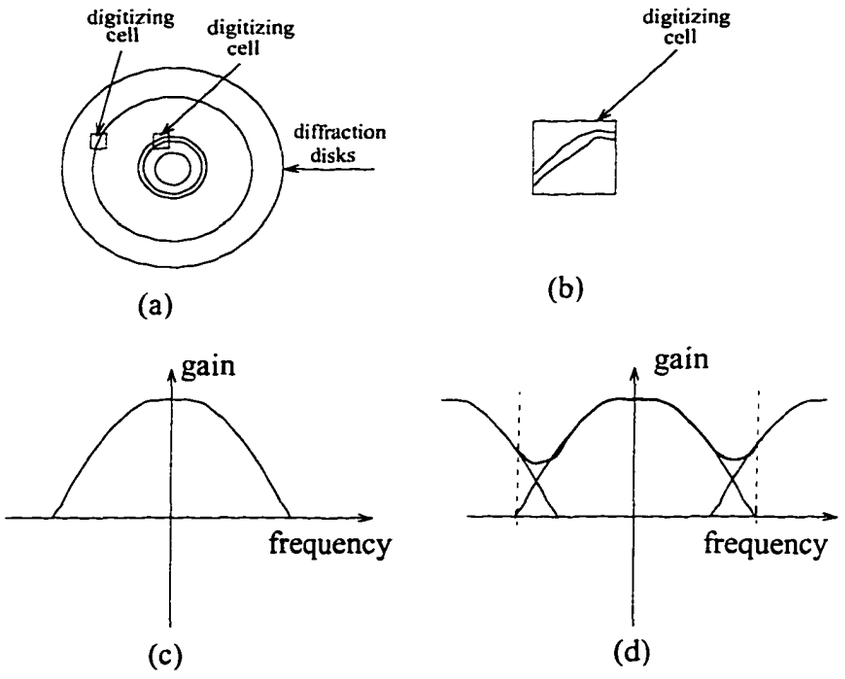


Figure 2.8 (a) An example of diffraction rings with two sample digitizing cells. (b) If the size of the digitizing cell is too large, the right cell will integrate the two intensities of the two lines and output one intensity. (c) The spectrum of an undigitized diffraction rings. (d) The spectrum of an image after digitizing at a low rate.

The space sampling rate is also used to determine whether the light source, the pin-hole, can be considered as a point light source to determine whether the digitized image is a truly defocused PSF. A light source can be considered as a point light source only if its size is smaller than the space sampling rate. Strictly speaking, only a point light source can be used to measure the defocused PSF. However, a light source can be modelled as an integration of point light sources located at different positions as illustrated in Fig. 2.9. Therefore, a recorded image of a light source is the superimposed impulse response of these point light sources. Suppose *i(x, y)* is the response of a microscope to a light source *o(x, y)*; *p(x, y)* and *P(u, v)* are the PSF and OTF of the system. Then according to linear system theory,

$$i(x, y) = o(x, y) \otimes p(x, y)$$

$$I(u, v) = O(u, v) P(u, v)$$

where $\otimes$ denotes convolution, and *I(u, v)* and *O(u, v)* are the Fourier transform of *i(x, y)* and *o(x, y)* respectively. *o(x, y)* can be written as:

$$o(x, y) = \sum_{(n\Delta x)^2 + (m\Delta y)^2 \le r^2} \delta(x - n\Delta x, y - m\Delta y) \qquad (2.11)$$

where *r* is the radius of the light source. *i(x, y)* is the measured response and *I(u, v)* can be obtained by taking the Fourier transform of the measured *i(x, y)*. If $o(x, y) = \delta(x, y)$, the light source is a point light source and the measured *i(x, y)* is the PSF; otherwise, it is the superimposed PSF.

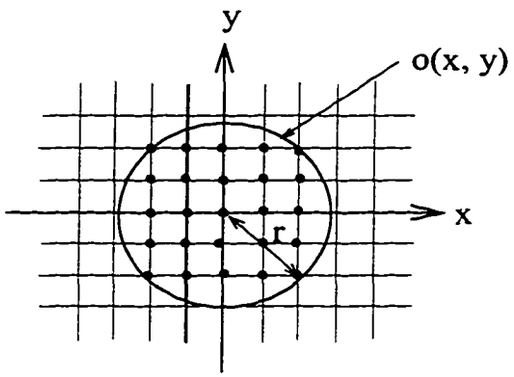


Figure 2.9 The decomposition of a light source into point light sources.

2.3 Comparison of the Two Approaches

In order to compare the results of mathematical models and direct measurement, it is necessary to unify the units used in both approaches. The space sampling rate Δx is measured in experiments and the number of pixels of the digitized image N is determined by the number of elements of the digitizer. The units L_u, Δu (as illustrated in Fig. 2.2) of the experimental OTF are determined, as in Table 2.1. Theoretically, if these parameters are used for discretizing the mathematical models of defocused OTF, the units used in discrete theoretical results will be same as in the experiments. However the number of sampling elements on a digitizer is usually large and diffraction patterns often cover only a small portion of it. To digitize and store an image which has the size of the digitizer, and to use this size to compute the theoretical defocused OTF and PSF is very expensive computationally. The number of sample points N thus is chosen for mathematical models for purpose of comparison in the following way: increasing the number until the theoretical diffraction pattern is not changing further; or, the N sufficiently covers the experimental PSF, whichever is larger. This number is used in computing the theoretical OTF and PSF as well as experimental OTF. The space sampling rate Δx from experi-

ments is used to determine the frequency band-width L_u for the mathematical models. Up to this point, L_u, L_x, Δx and Δu used in both approaches are unified.

In the cases where the pinhole is not sufficiently small to be considered as a point light source, the experimental results obtained are actually the results of the superimposed impulse responses from the light source as discussed in Section 2.2. The same superimposed effects should be applied to the theoretical defocused PSF so that the two approaches are identical in terms of the light source. The space sampling rate of the experiments is used in convolving the light source with the theoretical defocused PSF to accomplish the effects.

Three criteria have been used to evaluate how close the computed image is with the measured one: visual comparisons in the space domain, comparison in the frequency domain and correlation coefficient computation in both the space and frequency domains. Visual comparisons involve comparing the two images with respect to the number of diffraction rings, their distributions and sizes. Comparison in the frequency domain is carried out using the 1-D radial slices of the 2-D Fourier transforms of the images, due to the fact that these images are circularly symmetric. These 1-D radial slices of the frequency components of the computer and measured images are matched after they are brought to the same scale. Further, the correlation coefficients for the computed and measured images in the space and frequency domains are computed. The correlation coefficient is the normalized cross-correlation of two functions. Coefficients close to one indicate a good match between the two functions while coefficients close to zero indicate no matching at all.

2.4 Results

The objective lenses used for measurements are listed in Table 2.2. In the table, the parameters of these lenses are listed in terms of their numerical apertures (NA), magnifications and the measured space sampling rate. The wavelength used for system 1 is 0.550 μm and 0.544 μm for system 2. The pinhole size is 1.0μm in diameter. All the axes shown on the defocused PSFs correspond to the number of pixels unless otherwise indicated.

Table 2.2. The Objective lenses used for the measurements

lens type	magnification	numerical aperture	measured sampling rate on system 1	measured sampling rate on system 2
Planachromat	16	0.35	0.58	/
Planachromat	25	0.45	0.37	/
Planachromat	25	0.65	/	0.38
Planachromat	40	0.65	/	0.24
Planapochromat	40	0.95	0.21	0.24
Planapochromat	63	0.80	0.137	0.157

2.4.1 Results for analysis of imaging properties

In Fig. 2.10, the two mesh plots of figures are shown from the experimental results for the focused pinhole of system 1. Fig. 2.10 (a) is from the image of the 16x, 0.35NA objective lens, and (b) is from the 25x, 0.45NA lens. Since the space sampling rate for the 16x objective lens is 0.58 μm, the size of the pinhole used is smaller than the

space sampling rate. Thus the function plotted in Fig. 2.10 (a) is the actual focus PSF of the microscope with the 16x objective lens. For Fig. 2.10 (b), because the space sampling rate for the 25x lens is 0.37 μm, it is smaller than the pinhole size. Thus the obtained image is actually the integration of several point light sources; i.e. a 3x3 light array in this case as illustrated in Fig. 2.11. Note the distribution of intensities in both mesh plots are fairly symmetric. The results indicate that there are no noticeable spherical aberrations for the microscope.

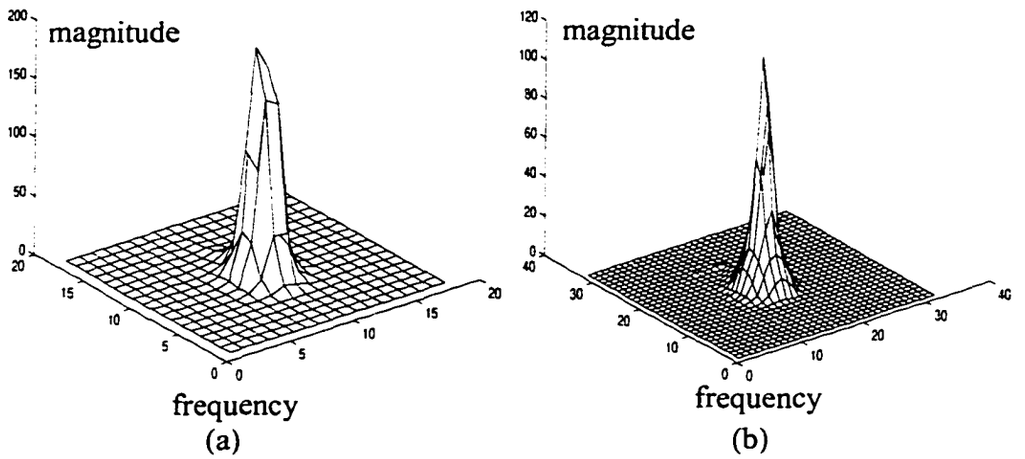


Figure 2.10 Mesh plots of images of a focused pinhole light source of a microscope with two objective lenses. (a) The result of the 16x with 0.35 NA lens. (b) The result of the 25x with 0.45 NA lens. The wavelength of the illumination light is 0.55 μm.

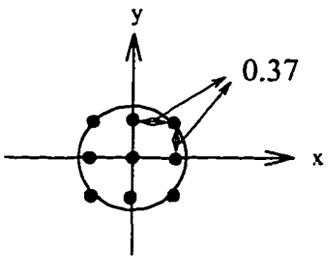


Figure 2.11 The 3x3 light array decomposed from the pinhole for the 25x, 0.45NA objective lens.

Shown in Fig. 2.12 are the normalized results of one dimensional OTF from direct measurement of the 16x, 0.35NA lens with various amount of defocus. Shown in Fig. 2.13 are the normalized results of one dimensional OTF from the mathematical model of the 40x, 0.95 NA lens with various amount of defocus. Defocused OTFs are circularly symmetric and the results shown in Fig. 2.12 and Fig. 2.13 rotating about the Z axis will yield the two-dimensional OTF.

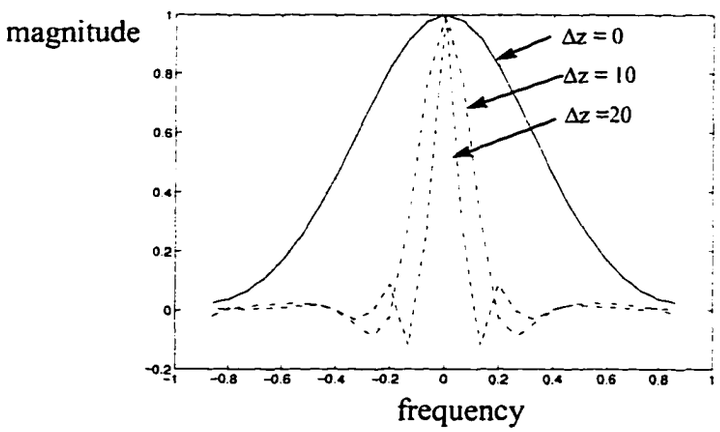


Figure 2.12 Defocused OTF results from experiments of the 16x, 0.35NA lens at various amount of defocus. f_c = 1.2727.

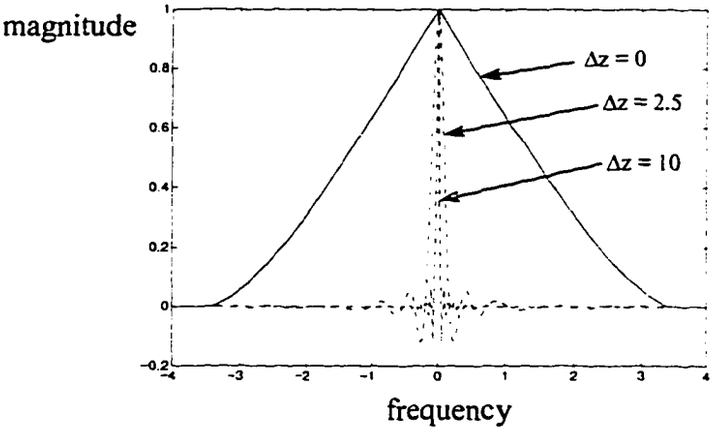


Figure 2.13 Defocused OTF results from theoretical approaches of the 40x, 0.95NA lens at various amount of defocus. $f_c = 3.4545$.

Results in Fig. 2.12 and Fig. 2.13 indicate that with the increase of defocus, the gain for high frequencies decreases dramatically and the frequency range with high gain becomes narrower. For high numerical aperture and high power lenses, this phenomenon is even more obvious. The discrimination of the defocused OTF against high frequencies will result in introducing haze and blur into images. This is consistent with the common observation that an out-of-focus image is blurred, which corresponds to the loss of high frequencies in the frequency domain.

In Fig. 2.14, defocused PSFs are presented from direct measurement of the 16x, 0.35 NA lens with different amounts of defocus. In Fig. 2.15, defocused PSFs are presented from the mathematical model of the 40x, 0.95NA lens with different amounts of defocus. As can been seen from the figures, a defocused image of a point light source tends to spread in space. Thus, a defocused image of an object which can be considered as a weighted sum of point light sources is the integration of these spread diffraction patterns.This integration results in a hazy and blurred image. These results are consistent with those from OTF plots.

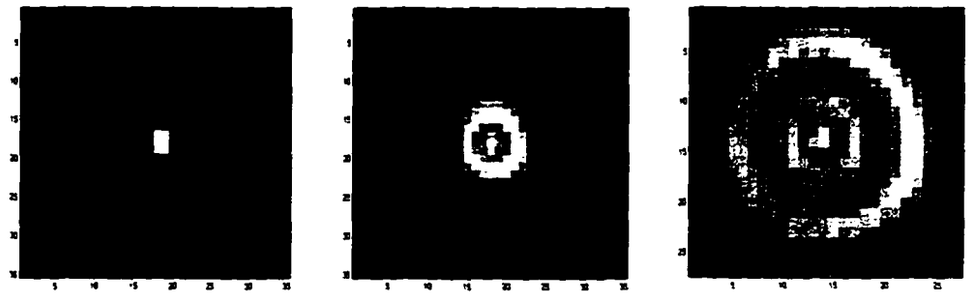


Figure 2.14 Defocused PSFs of the 16x, 0.35NA lens from direct measurement. From left to right, the amount of defocus corresponds to 0 (focus), 5 μm and 10μm defocus. f_c = 1.2727.

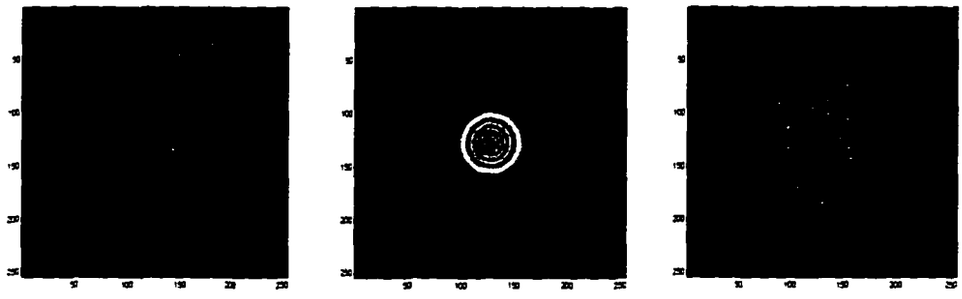


Figure 2.15 Defocused PSFs of the 40x, 0.95NA lens from the mathematical model. From left to right, the amount of defocus corresponds to 0 (focus), 5 μm and 10μm defocus. f_c = 3.4545.

Listed in Table 2.3 are the relative maximum intensities of defocused PSFs to those of focused PSFs with various amount of defocus and for various objective lenses. As can be seen from the table, the maximum intensity drops dramatically for high numerical aperture lenses with a small increase of the amount of defocus. For low numerical aperture lenses, the decrease of the maximum intensity is slower. For the 16x, 0.35NA lens, however, the maximum intensity of the defocused PSF of 10μm defocus is only about 18.6% of that of the in-focus PSF. This imaging property indicates that a

defocused object contributes less (depending on the magnification and numerical aperture of the lens used) to the image with increasing defocus. This imaging property is very useful in optical sectioning which is discussed in the next chapter.

Table 2.3. Relative maximum intensity of defocused PSF to that of the in-focus PSF.

Amount of defocus	16x 0.35NA	25x 0.45NA	40x 0.95NA	63x 0.80NA
2.5 μm	81.2%	45.2%	6.6%	10.9%
5.0 μm	44.3%	3.2%	3.2%	5.4%
10.0 μm	18.6%	1.6%	1.6%	2.4%

Before the comparison of the results from mathematical models and from direct measurement, it is worthwhile to explore the difference between Castleman's approximation and Stokseth's approximation. In Figure. 2.16 and Fig. 2.17, results from both approximations are presented. The two approximations use the same parameters. The results from these two approximations are quite similar in terms of the diffraction patterns, number of rings and the size of the rings. The only noticeable difference is that Stokseth's approach produces relatively sharper images than those of Castleman. Results presented in Fig. 2.18 and Fig. 2.19 are the defocused OTFs from the two approaches. They are very close.

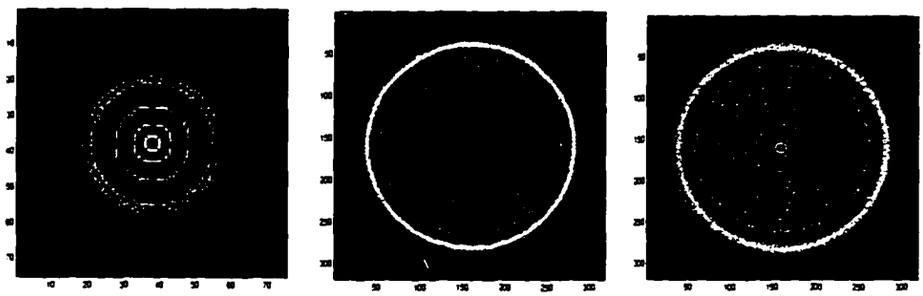


Figure 2.16 Stokseth's defocused PSF at defocus 20μm. Left: 25x 0.45NA; middle: 40x, 0.95NA; right: 63x, 0.80NA.

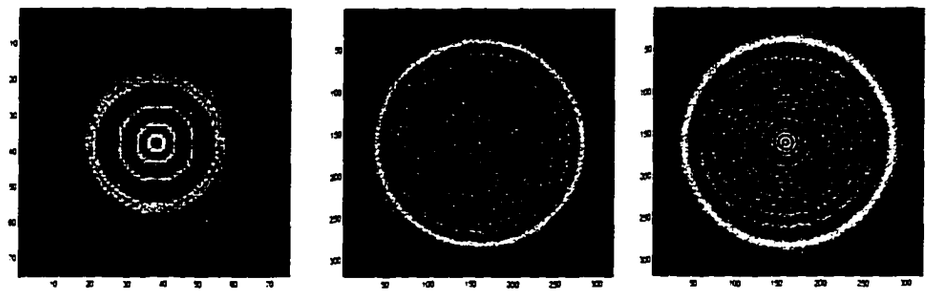


Figure 2.17 Castleman's defocused PSF at defocus 20μm. left: 25x 0.45NA; middle: 40x 0.95NA; right: 63x 0.80NA.

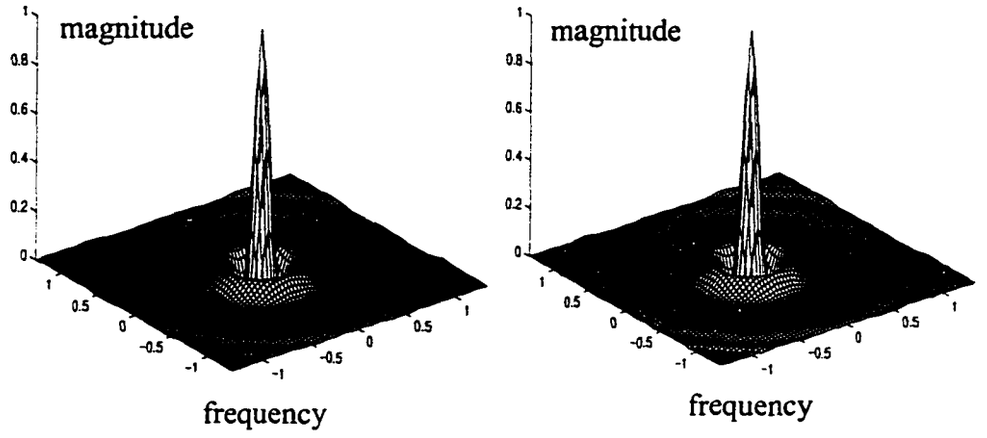


Figure 2.18 OTF mesh of the 25x 0.45NA lens at 10μm defocus Left: Castleman; Right: Stokseth. $f_c = 2.3435$

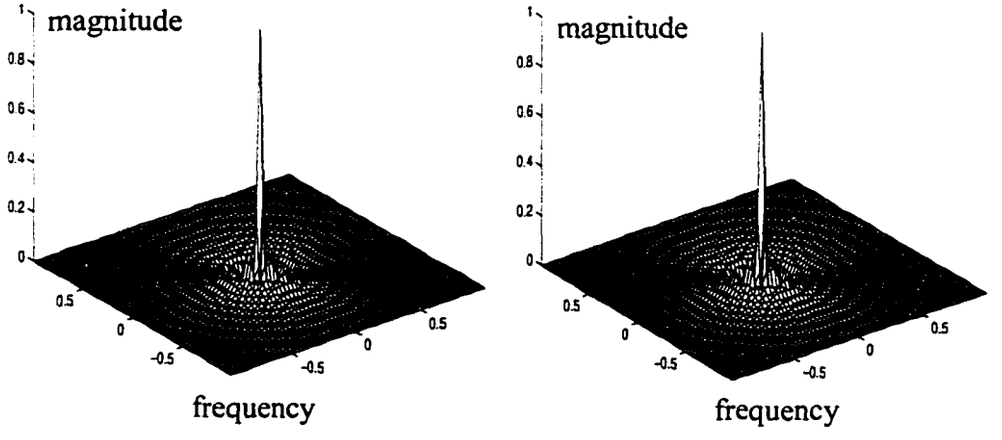


Figure 2.19 OTF mesh of the 40x 0.95NA lens at 10 μm defocus. left: Castleman; right: Stokseth. $f_c = 3.4545$.

Hopkins' approach of Eqn. 2.7 is also used to calculate defocused PSFs and OTFs. The results are comparable with those from Castleman's approximation in terms

of number and size of diffraction rings. Therefore, in this thesis, all the results presented are from the simple form of Castleman's approach.

2.4.2 Results of a comparison of the two approaches

As mentioned early, the comparison of direct measurement and mathematical models is accomplished by three means: visual comparison in the space domain, comparison in the frequency domain and correlation coefficient computation in both the space and frequency domains. In the following, comparison results are presented for representative objective lenses used. All the images from mathematical models in the comparison have been convolved with a proper amount of point light sources.

A. 16x, 0.35 NA lens: The resulting space sampling rate is 0.58 μm. In this case the pinhole is small enough so that the digitized image is the true PSF. The defocused PSF for the 20μm of defocus is digitized and also computed from the OTF given by Eqn. 2.10 using the inverse discrete Fourier transform. Visual comparisons (see Fig. 2.20) indicate that the two images are very close- same size, same number of diffraction rings and same intensity distribution. The resemblance can also be observed in Fig. 2.21 which is the 1-D OTFs from the two approaches. The correlation coefficients for these functions, given in Table 2.4, range from 0.9451 to 0.9544 indicating good matches.

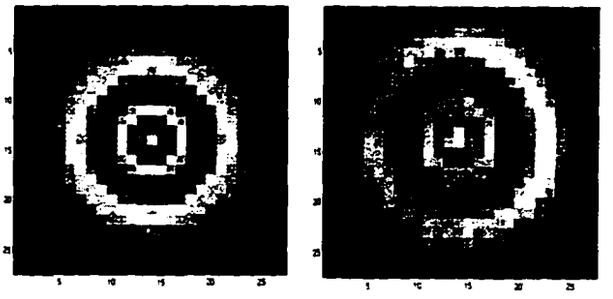


Figure 2.20 Defocused PSFs for the 16x, 0.35NA lens at a defocus amount of 20. Left: theoretical result. Right: experimental result. Both images have size of 37x37.

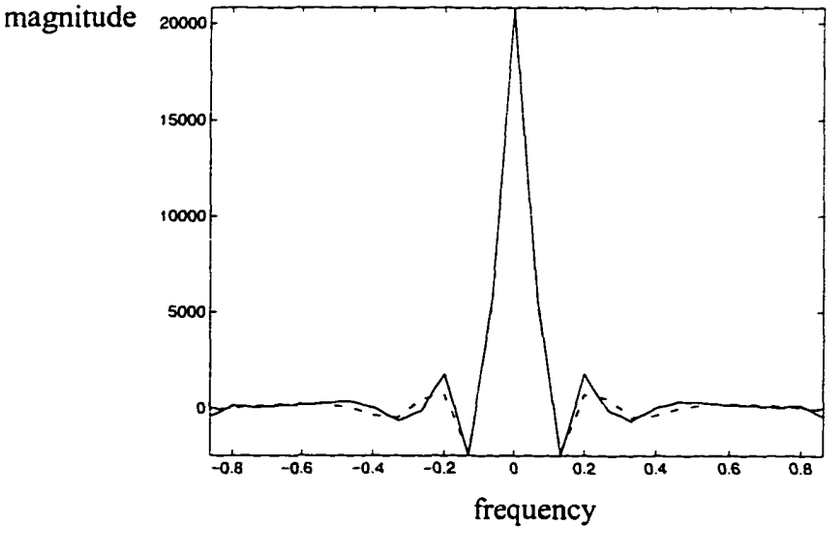


Figure 2.21 Center cross section of the defocused OTFs of the 16x, 0.35NA lens at a defocus amount of 20μm. The solid line is the experimental OTF, the dashed one is the theoretical one. f_c=1.2727.

B. 25x 0.45NA lens: The resulting space sampling rate is 0.37 μm. In this case, the pin hole is a 3x3 array of point light sources. The defocused PSFs for 10 and 20μm of defocus are computed from the corresponding defocused OTF given by Eqn. 2.10 using the inverse discrete Fourier transform. The computed images result from the convolution of the defocused PSF with the array of point light sources. The computed image and a corresponding measured image for a 10μm defocus are presented in Fig. 2.22 and images for a 20μm defocus are presented in Fig. 2.23. Visual comparisons from Fig. 11 as well as Fig. 2.23 indicate that images from the mathematical model and measurements are very close. The comparison in the frequency domain, as presented in Fig. 2.24, provides the same indication. The correlation coefficients in both the space and frequency domains ranging from 0.9262 to 0.9735, as given in Table 2.4, also indicate that the two approaches lead to similar images.

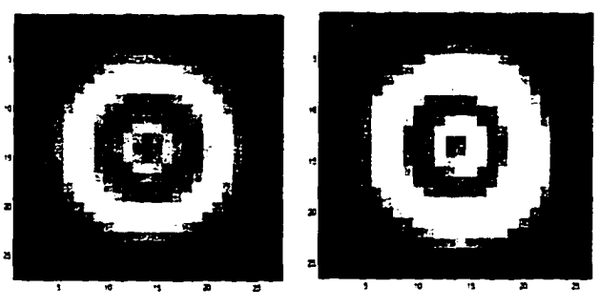


Figure 2.22 Defocused PSFs for 25x, 0.45NA lens at a defocus amount of 10 μm. Left: theoretical result. Right: experimental result. Both images are 26x26.

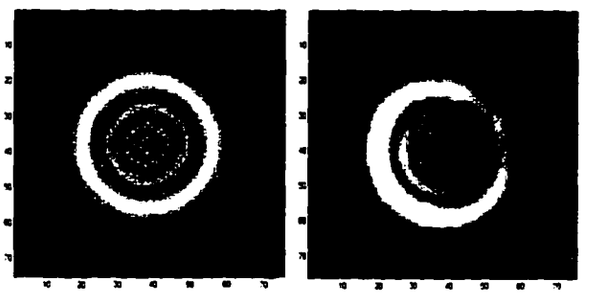


Figure 2.23 Defocused PSFs for 25x lens at a defocus amount of 20 μm. Left: theoretical result. Right: experimental result. Image size is 75x75.

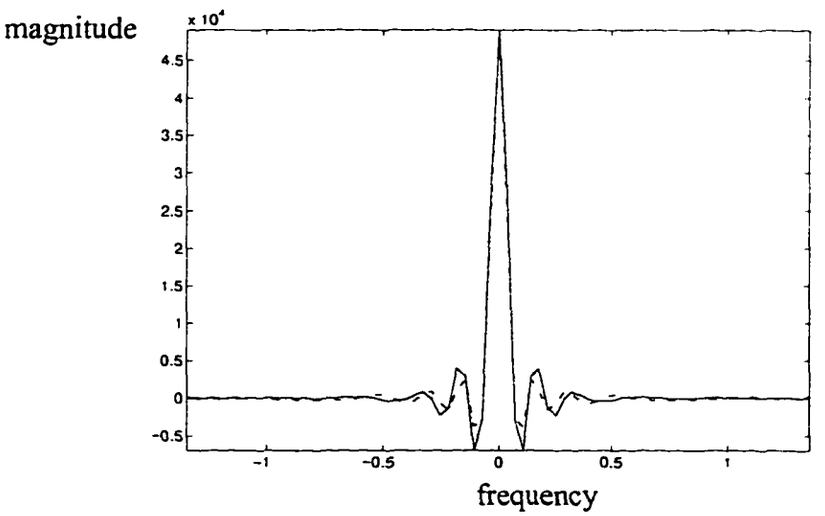


Figure 2.24 Center cross section of defocused OTFs of 25x, 0.45NA lens at a defocus amount of 20 μm. The solid line represents the experimental OTF; the dashed line represents the theoretical OTF. $f_c = 1.6364$.

C. 40x, 0.95NA lens: The resulting space sampling rate is 0.21 μm. In this case, the pinhole is decomposed into a 5x5 array of point light sources where all the elements have the value one except the four corners which have the value zero. The computed defocused PSF is convolved with the light array and is compared with the image from the experiment. The images for 10 and 20μm defocus are presented in Fig. 2.25 and Fig. 2.26 respectively. Visual comparisons (Fig. 2.25 as well as Fig. 2.26) indicate that for this medium magnification and high numerical aperture lens, the results from the two approaches have great differences. The computed images have more diffraction rings and the overall sizes of the rings are larger. In the 10μm defocus case, the size of computed diffraction rings is, 155x155, over three times larger that the experimental pattern which is 47x47. The results shown in Fig. 2.27 are from system 2 using the same objective lens as used in system 1. Fig. 2.27 indicates the same disagreement that the size of the diffraction pattern from the mathematical model is 115x115, over three times more than the experimental 37x37. Because the comparison results from the two systems giving the same disagreement, the possibility that the disagreement comes from system error is eliminated. And not surprisingly, their corresponding OTFs are different as shown in Fig. 2.28. The correlation coefficients in both domains range from 0.2715 to 0.3775, as in Table 2.4, which are low and indicate no matching.

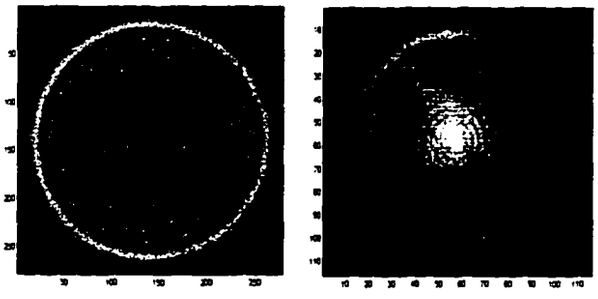


Figure 2.25 Defocused PSFs from system 1 for the 40x, 0.95NA lens at defocus amount of 20 μm. Left: theoretical result; size:279x279. Right: experimental result; size:116x116.

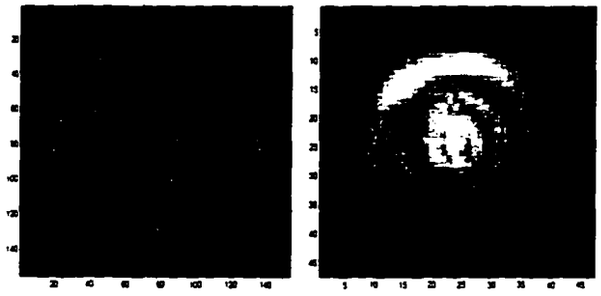


Figure 2.26 Defocused PSFs from system 1 for the 40x, 0.95NA lens at defocus amount of 10 μm. Left: theoretical result; size: 155 x155. Right: experimental result; size: 47x47.

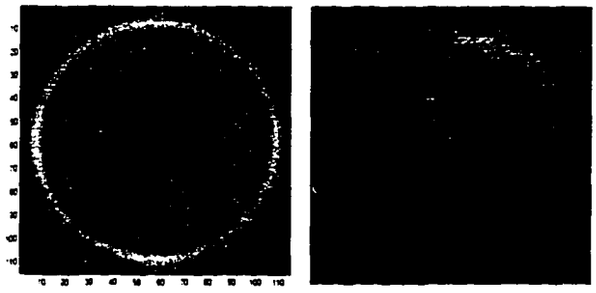


Figure 2.27 Defocused PSF from system 2 for the 40x, 0.95NA lens at a defocus amount of 10 μm. Left: theoretical result; size: 115 x115. Right: experimental result; size: 37x37.

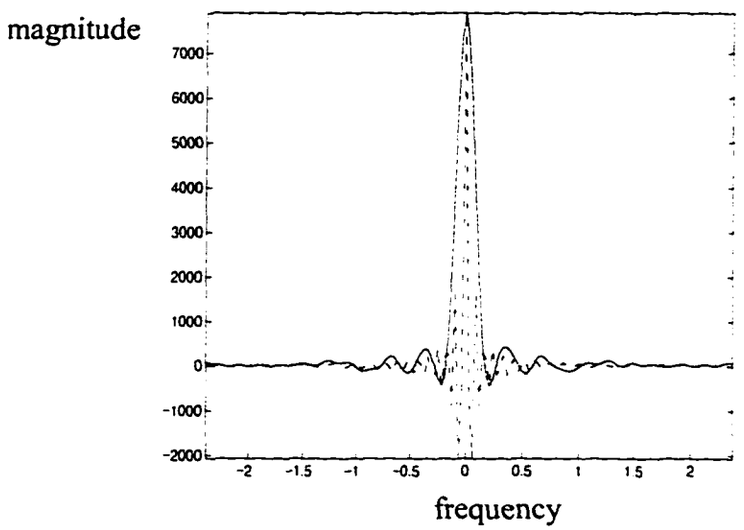


Figure 2.28 Center cross-section of defocused OTF of 40x, 0.95 lens at a defocus amount of 10μm. The solid line represents the experimental OTF; the dashed line represents the computed OTF.

D. 60x, 0.80NA lens: The resulting space sampling rate 0.137 μm. In this case, the pin-hole is decomposed into a 7x7 point light array where all the elements have the value one except the four corners which have the value zero. The computed defocused PSF is convolved with the light array and is compared with the image from the experiment. The defocused image for 10μm is presented in Fig. 2.29. Visual comparisons (Fig. 2.29) indicate that for this moderately high magnification and high numerical aperture lens, the results from the two approaches differ greatly. The image from the mathematical model has more diffraction rings and the overall size of the rings is larger. For the 10μm defocus case, the size of the computed diffraction rings is 155x155 while that of the experimental pattern is 83x83. The comparison in the frequency domain (see Fig. 2.30) also shows differences in the two approaches. The correlation coefficients in both

domains range from 0.4464 to 0.4642, as in Table 2.4, which are low indicating little matching.

However, at the same amount of defocus, the size of the computed diffraction pattern from the 63x, 0.80NA objective lens is about two times larger than that from experiments, while the difference for the 40x, 0.95NA lens is three times. The frequency components from both approaches for the 63x, 0.80NA objective lens, as in Fig. 2.30, are also closer than that of the 40x, 0.95NA objective lens, as in Fig. 2.28. The correlation coefficients of the 63x, 0.63NA lens are also higher that those of the 40x, 0.95NA lens. This suggests that the numerical aperture of an objective lens plays a more important role than its magnification for the mismatching of results of mathematical models and measurements. Section 2.5 provides more discussion on this point.

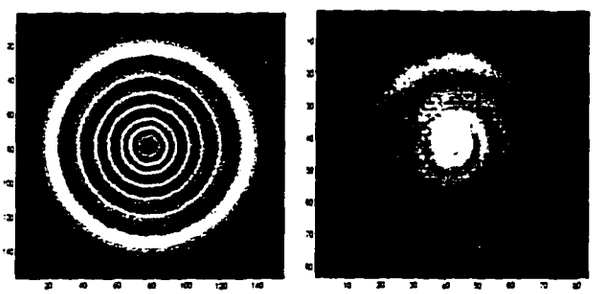


Figure 2.29 Defocused PSFs for the 63x, 0.80NA lens at a defocus amount of 10μm. Left: theoretical result; size:155x155. Right: experimental result; size:83x83.

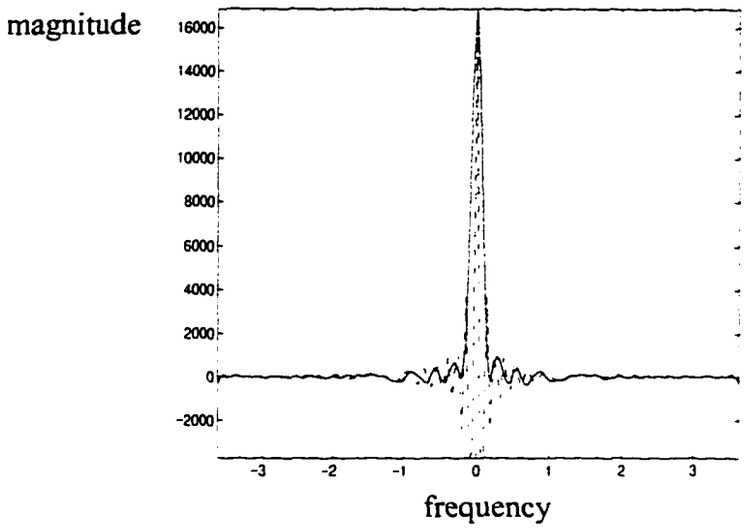


Figure 2.30 Center cross section of defocus OTF of 63x, 0.80 lens at a defocus amount of 10 μm. The solid line represents the experimental OTF; the dashed line represents the theoretical one.

Table 2.4. The correlation coefficients of defocused images of the pinhole, in the space and the frequency domain at various amount of defocus.

Objective lenses	Space Domain		Frequency Domain	
	10μm	20μm	10μm	20μm
16x, 0.35NA	0.9487	0.9451	0.9510	0.9544
25x, 0.45NA	0.9420	0.9262	0.9735	0.9710
40x, 0.95NA	0.2715	0.3572	0.2832	0.3775
63x, 0.80NA	0.4464	/	0.4642	/

2.5 Discussion

As has been seen from the last section, for objective lenses with a low magnification and numerical aperture, the images of the pinhole computed from the mathematical model and measured from experiments are fairly close to each other. For higher numerical aperture and magnification lenses, the results from these two approaches have great differences in terms of their number and size of diffraction rings. The effects of the two possible causes, namely magnification and numerical aperture of an objective lens, are simulated. The effects of the magnification on the diffraction patterns are simulated by varying magnifications and their corresponding space sampling rates while the numerical aperture is fixed at 0.95. The four magnifications, namely 16x, 25x, 40x and 63x, of the objective lenses used in the experiments are used for simulation so that their corresponding space sampling rates are known. The simulation results are presented in Fig. 2.31, where the units of the x- and y- axis are micrometers in all the results. The lengths of the x- and y-axis are deliberately fixed for all the plots in the figure so that the difference, if any, in the width of the diffraction rings can be easily observed. From Fig. 2.31, it can be seen that increasing magnification brings out more diffraction rings but the overall size of the diffraction pattern remains the same. The increased number of diffraction rings for high magnification, i.e., small space sampling rate is due to increasing resolving power so that the system is able to distinguish finer details. Because the change in magnification does not increase the size of the diffraction rings, it is unlikely that the cause of disagreement of the two approaches is related to the magnification and the space sampling rate.

Another piece of evidence suggesting that the magnification does not affect the size of diffraction rate is from another 40x objective lens which has a numerical aperture of 0.65. The results of defocused PSFs of the lens for a defocus amount of 10μm from the two approaches are shown in Fig. 2.32. While the size of the diffraction pattern for

the mathematical model one is 54x54 and that for the experimental one is 41x41, they are much closer than those shown in 2.27. All the parameters used for the results in Fig. 2.32 are the same as those in Fig. 2.27 except that the numerical aperture is different. Thus the numerical aperture probably is the factor causing disagreement.

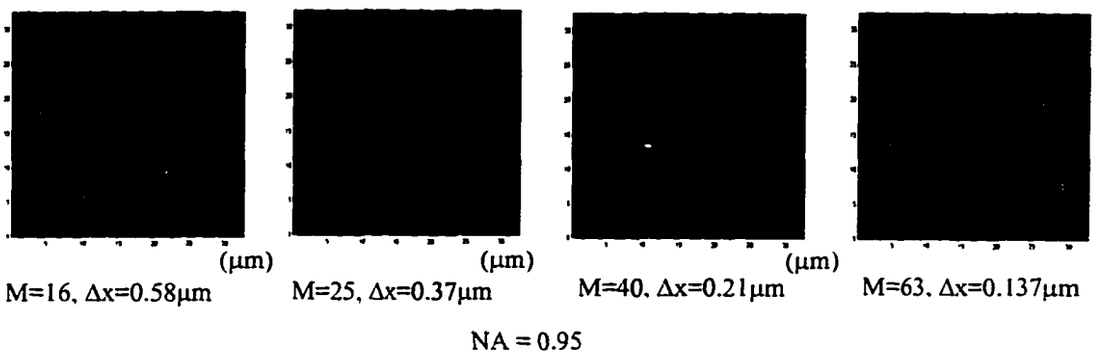


Figure 2.31 Simulation of effects of the space sampling rate to defocused PSF of the 0.95NA objective lens at 10μm. The units on *x* and *y axis* are micrometers.

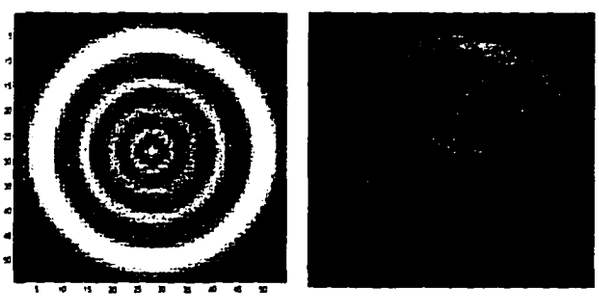


Figure 2.32 Defocused PSF for the 40x, 0.65NA lens at the defocus amount of 10 μm. Left: theoretical result; size: 54 x 54. Right: experimental result; size: 41x41.

The effects of the numerical aperture on the diffraction patterns are simulated by varying the numerical aperture while the magnification is fixed. Four numerical aperture values, namely 0.65, 0.75, 0.85 and 0.95 are used for simulation for the 40x magnification lens. The simulation results are presented in Fig. 2.33. Again, the length of the x- and y- axis for all plots is fixed and the units of the two axes are micrometers. From the figures, it can be seen that the number as well as the size of the diffraction rings increases with the numerical aperture. The size of diffraction rings for 0.65NA is about half of 0.95NA for the 40x lens. This indicates that a wrong value of numerical aperture can cause the disagreement of the two approaches in terms of size and number of diffraction rings. Possible causes of a wrong value for the numerical aperture could be that the real numerical aperture of a microscope or the effective numerical aperture of the microscope system arrangement is smaller than that indicated on the objective lens, or that the mathematical model does not take the value of the numerical aperture into consideration correctly.

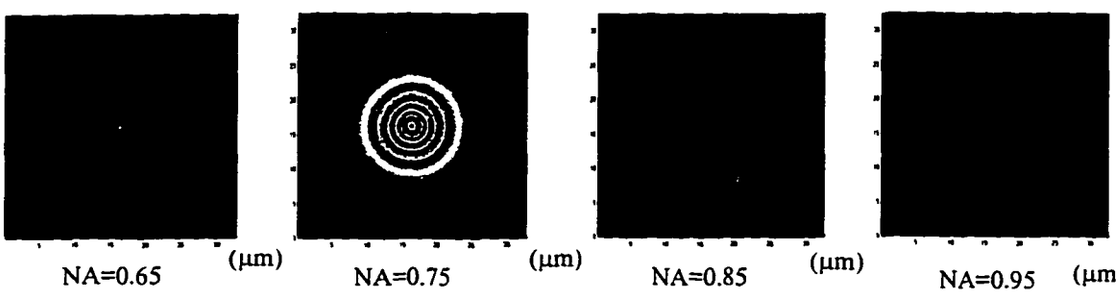


Figure 2.33 Simulation of effects of numerical aperture on the defocused PSF of the 40x objective lens at 10μm. The units of the x and y axis are micrometers.

2.6 Summary and Conclusion

Both direct measurement and mathematical models are used for obtaining defocused PSFs and OTFs. Results of the defocused PSFs and OTFs from both approaches show that a defocused transmitted light microscope tends to discriminate against high frequency information and thus introduce haze to an image. The maximum intensity of a defocused PSF drops with increasing defocus. For high numerical aperture objective lenses, the maximum intensity drops dramatically with a small amount of defocus. The comparison between the two approaches shows that the defocused PSF and OTF from the experimental approach agree with those from the theoretical approach for low numerical aperture lenses. However, for objective lenses with high numerical aperture, these two approaches produce different results in terms of their diffraction patterns and sizes. The theoretical approach produces more diffraction rings and a larger diffraction pattern, and thus introduces more haze to final images. Simulations indicate that a wrong value of numerical aperture can cause the disagreement between the two approaches in terms of size and number of diffraction rings. A wrong value for the numerical aperture may be related to causes such as the real numerical aperture of a microscope or the effective numerical aperture of the microscope system arrangement is smaller than that indicated on the objective lens, or the mathematical model not taking the value of the numerical aperture into consideration correctly. These results suggest that defocused PSFs and OTFs from direct measurement are more appropriate for applications involving high numerical objective lenses; whereas defocused PSFs and OTFs from both direct measurement and mathematical models are appropriate in applications involving low numerical aperture objective lenses.

Chapter 3

Volume Data Acquisition II — Optical Sectioning

A microscope image of a thick specimen contains object information from the in-focus plane as well as that from out-of-focus planes. The out-of-focus information can be removed mathematically after an image is acquired, which leads to a clean image containing only information on the focal plane. Serial microscope images of the specimen are obtained by moving the objective stage of the microscope so that the in-focus plane moves from the top to the bottom of the specimen at a constant interval as mentioned in Chapter 1. This method of collecting serial microscope images and mathematically removing out-of-focus information from images obtained is called optical sectioning. Algorithms for optical sectioning vary in terms of the mathematical method used to remove out-of-focus information.

The defocus information of a thick specimen from out-of-focus planes can be considered as a distortion of the image of the object on the focal plane. As will be seen in Section 3.1, this distortion is the result of a 3-D convolution of a 3-D object, i.e., the thick specimen, with the 3-D transfer function of the microscope. The 3-D transfer function of the microscope is represented by the stack of sequential 2-D defocus PSFs such as the ones obtained in Chapter 2. The actual image acquisition corresponds to the result of the 3-D convolution at a plane (the image plane). The 3-D convolution resem-

bles the model of image distortion in 2-D image processing where an observed image is a result of 2-D convolution of the original image with a 2-D distortion transfer function. In two-dimensional image restoration, the original image is recovered from the observed image provided that the distortion transfer function is known. Optical sectioning can be regarded as an extension of 2-D image restoration to 3-D where the 3-D distortion transfer function is known and the objective is to recover the 3-D original object from the 3-D observed *image*. The 3-D observed image is the stack of serial 2-D images obtained by moving the object stage of a microscope so that the focal plane moves from the top to the bottom of the 3-D object at a constant interval. The 2-D image at level j within the 3-D observed image is obtained by placing the focal plane at level j within the 3-D object. The 2-D image contains in-focus information of the object at level j on the focal-plane plus the out-of-focus information from the remaining of the object. Thus, the 3-D observed image contains serial 2-D images with the focal plane placed from the top to the bottom of the object. Optical sectioning is to recover the 3-D original object which is the in-focus information from the serial 2-D images, i.e., the 3-D observed image.

Previous studies extend 2-D image restoration algorithms to 3-D deconvolution for 3-D image restoration [13] [41] [42] [53] [94]. However, since 3-D deconvolution is involved which is very computationally intensive, most of the algorithms require specialized computers. Several optical sectioning algorithms use rough approximations to reduce the computational effort, but they are not effective for most images [2] [13]. In this thesis, a new algorithm, the partial-minimization-and-constrained-iterative algorithm, is proposed, which represents a compromise between the 3-D deconvolution and the rough approximation approaches. Results show that this algorithm is effective at removing out-of-focus information. The algorithm processes serial images independently, thus computationally expensive operations such as 3-D convolution and 3-D Fourier transform are eliminated. It allows the user to choose the degree of approximation and thus the computation complexity. The algorithm is also very attractive for par-

allel implementation because all the images can be processed independently. Another advantage of the algorithm is that it breaks a 3-D problem down into series of 2-D problems, and thus it can utilize many existing 2-D computing resources such as 2-D FFT.

This chapter is organized as follows. In Section 3.1, models for image formation of a thick specimen are presented. The principle of optical sectioning by which the models of image formation are obtained is discussed. In Section 3.2, a brief overview of existing algorithms for optical sectioning is given. The proposed partial-minimization-and-constrained-iterative algorithm is discussed in Section 3.3, and the results of optical sectioning of this algorithm are presented in Section 3.4.

3.1 Models of Image Formation and Optical Sectioning

A highly simplified microscope consisting of an objective lens and a specimen with thickness T are illustrated in Fig. 3.1. The bottom of the specimen is placed on the object stage of the microscope. Two coordinate systems, the system of the microscope (x', y', z') and that of the specimen (x, y, z), are also depicted in the figure. The origin of the microscope coordinate system is at the image plane of the microscope with the z-axis coinciding with the optical axis of the microscope. The origin of the specimen coordinate is at the top of the specimen with the z-axis along the z-axis of the microscope. If d_i, d_f and f represent the image-to-lens distance, focal-plane-to-lens distance and focal length respectively, as illustrated in Fig. 3.1, these three parameters have the following relation according to optics theory [9]:

$$\frac{1}{d_i} + \frac{1}{d_f} = \frac{1}{f} \tag{3.1}$$

Because the d_i and f of a microscope are fixed by the manufacturer, the focal-plane-to-lens distance d_f is therefore fixed by Eqn. 3.1. The *object stage* of the microscope is

moved up or down so that the focal plane is placed at the desired level within the specimen.

Let $f(x, y, z)$ denote the 3-D distribution of optical density within the specimen, and $g'_{zf}(x', y')$ denote the 2-D image resulting from the specimen with the focal plane located at level $z = z_f$. If a relation between $f(x, y, z)$ and $g'_{zf}(x', y')$ can be established, then it is possible to obtain $f(x, y, z)$ from the resulting images. To establish such a relation, it is necessary to unify the coordinate systems of $f(x, y, z)$ and $g'_{zf}(x', y')$. The function $g'_{zf}(x', y')$ can be projected from the image plane to the focal plane, counteracting the magnification and a 180° rotation. This projection places the image back into the coordinate system of the specimen because the focal plane is within the specimen. Let $g(x, y, z_f)$ be the projection of the image $g'_{zf}(x', y')$ from the image plane to the focal plane $z = z_f$. Thus a point (x', y') on the image plane is mapped to a point at (x, y, z_f) on the focal plane in object coordinates.

The model for image formation is a relation between a 3-D object and a 3-D transfer function leading to a 3-D image. The model for image formation is developed in the following way: consider the contribution of a thin slice of the 3-D object to the 2-D image with the focal plane placed at $z = z_f$; sum the contributions of all the thin slices of the 3-D object to obtain the observed 2-D image; place the focal plane at a different z level to obtain another 2-D image, repeat this process to obtain a series of 2-D images, i.e., the 3-D observed image.

Consider a very thin slice of a 3-D object $f(x, y, z_1)$ at $z = z_1$, which means the slice is out-of-focus by the amount of $(z_1 - z_f)$ (see Fig. 3.1). Its contribution to the image $g(x, y, z_f)$, according to linear theory, is the 2-D convolution of $f(x, y, z_1)$ with the defocus transfer function $h(x, y, z_1 - z_f)$ [67], as in Eqn. 3.2.

$$g_1(x, y, z_f) = f(x, y, z_1) \otimes h(x, y, z_1 - z_f) \tag{3.2}$$

In the equation, $\otimes$ represents the 2-D convolution operation with respect to x and y, and $h(x, y, z)$ stands for the defocused PSF.

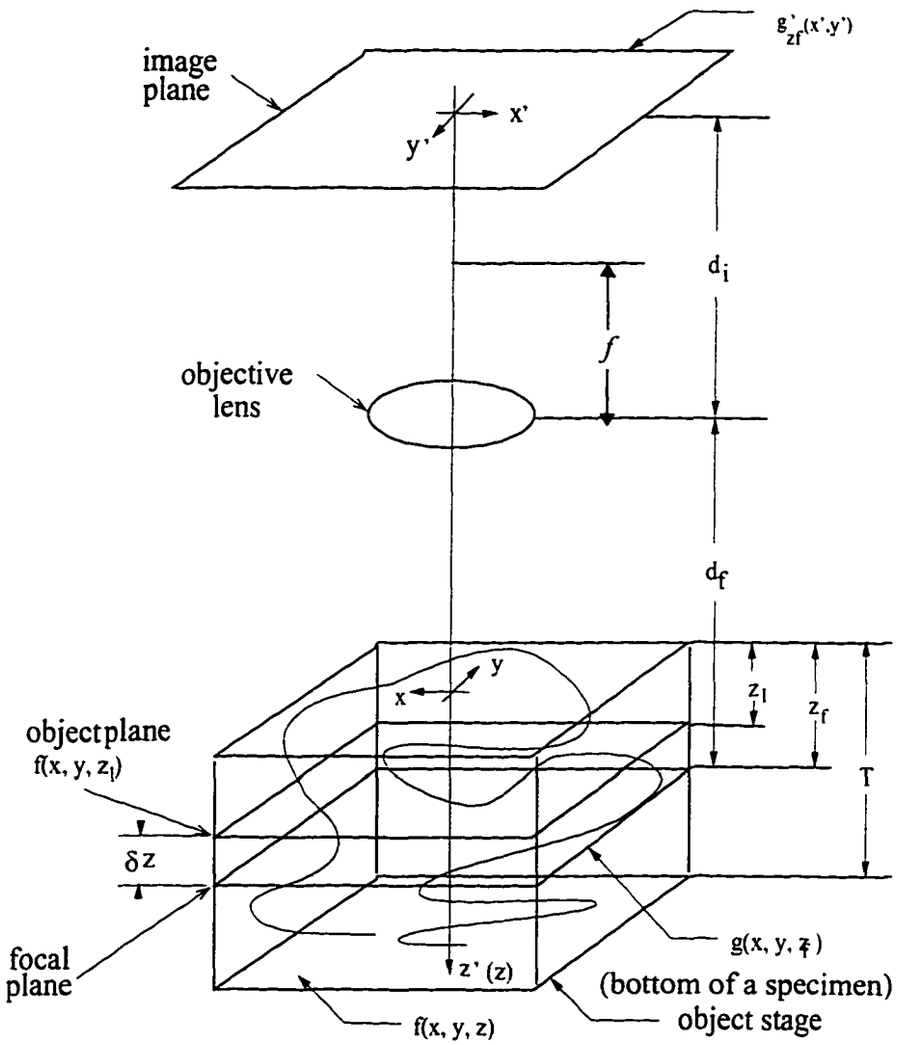


Figure 3.1 A diagram of a highly simplified microscope with a specimen with thickness T.

The 3-D object can be modeled as a stack of (N+1) thin slices located at small inter-

vals Δz along the z axis. A 2-D image with the focal plane placed at $z = z_f$ is the sum of the contributions of all these thin slices. Therefore the 2-D image with the focal plane at $z = z_f$ of the 3-D object is given by (see Fig. 3.1):

$$g(x, y, z_f) = \sum_{j=0}^{N} f(x, y, j\Delta z) \otimes h(x, y, j\Delta z - z_f) \tag{3.3}$$

This equation states that the image of a thick specimen with the focal plane at z_f is the sum of the *(N+1)* specimen planes blurred by the corresponding out-of-focus PSF.

Serial 2-D images (3-D observed image) can be obtained by moving the object stage of the microscope in the same increment Δz so that the focal plane is placed from the top to the bottom of a 3-D object (specimen). This movement corresponds to change the value of z_f from zero to T (where T=NΔz):

$$z_f = i\Delta z \qquad 0 \le i \le N.$$

Thus the *i*th 2-D image is obtained as

$$g(x, y, i\Delta z) = \sum_{j=0}^{N} f(x, y, j\Delta z) \otimes h(x, y, j\Delta z - i\Delta z) \tag{3.4}$$

The notation of the above equation can be simplified if *x, y* are dropped and the constant Δz is implicitly included in the function *h*, as:

$$g_i = \sum_{j=0}^{N} f_j \otimes h_{j-i} \tag{3.5}$$

After substituting *j* with *j-i* where *i*Δz represents the distance between the focal plane and object plane, i.e., the defocus amount, Eqn. 3.4 then can be written as:

$$g_i = \sum_{j=-i}^{N-i} f_{i+j} \otimes h_j \tag{3.6}$$

The above equation can then be rewritten as:

$$g_i = f_i \otimes h_0 + \sum_{j=-i}^{-1} f_{i+j} \otimes h_j + \sum_{j=1}^{N-i} f_{i+j} \otimes h_j \tag{3.7}$$

The above equation states that an observed image with the focal plane at z = $i\Delta z$ is the sum of the true density of the object f at plane i convolved with the in-focus PSF h_0 and its adjacent specimen planes f_{i+j}, above and below the focal plane, blurred by the out-of-focus PSF h_j.

As discussed in Chapter 2, the maximum intensity of the defocused PSF decreases with increasing defocus.This indicates that object slices farther from the focal plane contribute less to the image than those closer to the focal plane. Often, the contributions from farther slices can be ignored. For this reason, only M planes above and below (where $(2M+1) < N$) the focal plane are considered to attain reasonable approximation and Eqn. 3.7 can be rewritten as:

$$g_i = f_i \otimes h_0 + \sum_{j=-M}^{-1} f_{i+j} \otimes h_j + \sum_{j=1}^{M} f_{i+j} \otimes h_j \tag{3.8}$$

It is possible to use Eqn. 3.6 or Eqn. 3.7 or 3.8 to obtain f_i if $(N+1)$ images g_i $(i = 1$ *to* $N)$ taken with the focal plane placed at different levels within the 3-D object are known. This simultaneous equation model is one of the two models of image formation used to remove out-of-focus information from images of the thick specimen which will be discussed in Section 3.2.1.

Another model of image formation of a microscope is developed by Castleman based on Eqn. 3.3 [17]. If the interval Δz approaches zero, N approaches infinity, and the summation becomes an integral:

$$g(x, y, z_f) = \int_o^T f(x, y, z) \otimes h(x, y, z_f - z)\, dz$$

Upon writing out the 2-D convolution and letting $f(x, y, z)$ be zero outside the range of

$0 \le z \le T$, the above equation becomes:

$$g(x, y, z_f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (f(x_n, y_n, z)\, h(x-x_n, y-y_n, z_f-z))\, dx_n dy_n dz \qquad (3.9)$$

Eqn. 3.9 is a 3-D convolution, a general model for image formation of a microscope with a thick specimen. A series of images of $g(x, y, z_f = i\Delta z)$ $(1<i<N)$ taken at different focal plane levels z_f can be stacked as 3-D images as $g(x, y, z)$. The function $f(x, y, z)$ can be recovered by 3-D deconvolution, which will be discussed in Section 3.2.2.

3.2 Algorithms for Optical Sectioning

3.2.1 Algorithms based on simultaneous equations

3.2.1.1 Simultaneous linear equations

In the frequency domain, Eqn. 3.6 becomes:

$$G_i = \sum_{j=0}^{N} F_j H_{i-j} = \sum_{j=-i}^{N-i} F_{i+j} H_j \qquad (3.10)$$

in which G, F and H are the corresponding 2-D Fourier Transform of g, f and h. Since $(N+1)$ of G_j is known, Eqn. 3.10 represents a set of $(N+1)$ simultaneous linear equations of $(N+1)$ unknowns F_j's. These $(N+1)$ equations, if solved, lead to the functions F and thus to the intensity functions f. Even if a solution exists, however, the computational complexity of solving these equations is tremendous and in most cases a solution is unpractical because the number $(N+1)$ of sections in a thick specimen is usually large.

3.2.1.2 Nearest neighbor algorithms

Castleman *et. al.*proposes an approximation method, the nearest-neighbor algorithm, to obtain a good approximation of the intensity functions at a reasonable computational cost [13] [17]. The nearest-neighbor algorithm is based on the following three assumptions:

The first assumption is that the specimen plane f_i can be approximated by a high-pass-filtered version of the image g_j, that is

$$f_j \approx g_j \otimes k_0$$

The ground for this assumption is that the defocus PSF tends to discriminate high-frequencies and to allow low frequencies to pass, as discussed in the preceding chapter. Thus the image spectrum G_i contains the specimen spectrum F_i plus excessive low-frequency information from adjacent planes. Excessive low-frequencies from defocus planes are removed using a highpass filter applied to the observed g_i. The second assumption is that the effects of in-focus PSF can be ignored, that is $f_i \otimes h_o \approx f_i$. The third assumption is that only a small number of M planes adjacent to the focal plane contribute to g_i.

After rearranging Eqn. 3.7 into the following form

$$f_i \otimes h_0 = g_i - \sum_{j=-i}^{-1} f_{i+j} \otimes h_j - \sum_{j=0}^{N-i} f_{i+j} \otimes h_j \tag{3.11}$$

and substituting the mathematical form of the assumptions, the above equation can be written as:

$$f_i \approx g_i - \sum_{j=1}^{M} (g_{j-i} \otimes h_{-j} + g_{i+j} \otimes h_j) \otimes k_0 \tag{3.12}$$

where M < N. This equation suggests that the specimen at plane j can be approximately

restored by removing the defocused information, i.e., by subtracting 2M images of adjacent planes that have been convolved with the appropriate defocused PSF and a high pass filter k_0. The high-pass filter and the number M of adjacent planes must be selected to give reasonable results. Often, M is set to be equal to 1 which means that only three planes, the in-focus plane, one plane above, and one below the focal plane, are involved in the deconvolution.

Agard et. al. suggests a different approximation as [2] [3] [4]:

$$f_i \approx c_2 \left[g_i - c_1 \sum_{j=1}^{M} (g_{j-i} \otimes h_{-j} + g_{i+j} \otimes h_j) \right] \tag{3.13}$$

where constants c_1 and c_2 are used to balance out the relative contributions of the focal plane and the adjacent planes. For large spacing between sections, it was found that $c_1=0.45$ and $c_2=0.9$ are the best values for restoration results.

The first and last assumptions used in the nearest-neighbor algorithms are relatively accurate for a large spacing between sections but not for a small spacing. Thus nearest-neighbor algorithms are effective and efficient only for optical sectioning with large spacing.

3.2.2 Algorithms based on 3-D deconvolution

In 2-D image restoration, the model of image formation is the convolution of the transfer function with the original image plus additive noise as given by [31]:

$$g(x,y) = f(x,y) \otimes h(x,y) + n(x,y)$$

Two-dimensional image restoration is the recovery of $f(x, y)$ from $g(x, y)$ providing that the transfer function $h(x, y)$ is known. As described in Section 3.1, imaging a thick specimen is the 3-D convolution of the 3-D transfer function of the microscope with 3-D objects, which has the form identical to the 2-D model. Thus, 2-D image restoration algorithms, as can be found in [5] [8] [31] [47], are extended to the 3-D case to recover

3-D objects. The schema of extending 2-D to 3-D can be found in the literature of optical sectioning [13] [41] [42] [53] [94].

3.2.2.1 Inverse filtering

The frequency domain relation of Eqn. 3.9 is given by:

$$G(u, v, w) = F(u, v, w)\, H(u, v, w)$$

where u, v and w are frequency variables. Thus,

$$F(u, v, w) = G(u, v, w)\frac{1}{H(u, v, w)} = G(u, v, w)\, H'(u, v, w) \qquad (3.14)$$

Providing that $H'(u, v, w)$ exists, the intensity function $f(x, y, z)$ is the inverse Fourier transform of Eqn. 3.12. This method is called *inverse filtering* [13] [22].

The major problem with inverse filtering is the computational difficulty of the inverse of the 3-D OTF, $H(u, v, w)$, when $H(u, v, w)$ vanishes or becomes very small in any regions of interest in the uv plane and $H(u, v)$ cannot be simply inverted. Furthermore, if noise is present and additive to the image formation model, then the inverse of small values of the 3-D OTF multiplying the noise will dominate the restorted results.

3.2.2.2 Constrained-iterative approaches

A constrained-iterative schema iteratively evaluates an objective function to maximize or minimize it while imposing constraints on the estimated function. An example is the minimization of the objective function $\Phi(\hat{f}) = \|g - h \otimes \hat{f}\|^2$ [1] while constraining the estimated function $\hat{f}$ so that $\hat{f}$ must be positive. It is proven that the constrained-iterative schema is a means of regularizing image restoration[2] [47].

---

1. $\|.\|^2$ denotes the L^2-norm of a matrix unless otherwise stated.
2. Regularizing image restoration is to minimize a stabilized objective function.

Constrained-iterative approaches are used for image restoration for the following reasons. First, iterative computation of 3-D deconvolution can be easier than the direct evaluation and can be terminated before convergence. Secondly, the use of iterative schema allows additional constraints to be imposed on the estimated function. Thirdly, the iterative schema usually does not require inversion of h or division by H during calculations.

Most 3-D iterative approaches involve the non-negative constraint that the optical density for each pixel is non-negative because a negative value of a pixel does not have any physical meaning [13] [41] [42] [53] [94]. The commonly used initial estimation of the estimated function is the observed image. Given $(N+1)$ observed images g_i, the stack of a 3-D observed image g, 3-D iterative approaches are to find f_i (the 3-D intensity function f) such that the objective function $\Phi(\hat{f})$ is minimized. Algorithms for optical sectioning based on 3-D constrained-iterative schema can be summarized in the following form:

i) set an initial estimation $\hat{f}^0 = f_{initial}$. The imaging properties of a defocused microscope, as described in Chapter 2, are such that an image of a thick specimen mainly contains the in-focus information plus the out-of-focus information.Thus the initial value of the estimated function $\hat{f}$ can be set as the observed image g, that is $\hat{f}^0 = g$.

ii) set the *evaluation function* $\hat{f}^{k+1} = \hat{f}^k + \beta r^k$, where r^k is the *evaluation factor* usually depending on by the objective function chosen in the particular algorithm and β is a user-controlled parameter.

iii) impose the non-negative constraint on the estimated function so that if

$f^{k+1} < 0, f^{k+1} = 0$;

iv) continue the iteration until the estimated function converges or meets a preset

condition;

The main difference among various constrained-iterative schemes for 3-D optical sectioning is the choice of objective function and the choice of the evaluation factor r^k. The following are representative algorithms of constrained-iterative schemes used in 3-D optical sectioning.

VanCittert's constrained iterative algorithms:

A good result for the estimated function $\hat{f}$ convolved with the blur function h, i.e., $h \otimes \hat{f}$, should be approximately equal to the observed function g. This approach is the basis of the VanCittert's constrained-iterative algorithms. This type of algorithm minimizes the objective function $\Phi(\hat{f}) = \|g - h \otimes \hat{f}\|^2$ [47]. The evaluation factor r^k is set as the error of the observed function with the blurred estimated function, i.e.,

$$r_k = g - h \otimes \hat{f} \tag{3.15}$$

β can be chosen as a constant between zero to one, or as a function of the image coordinates as in Agard [2] so that $\beta^k = 1 - \left[\hat{g}^k - A\right]^2 / A^2$, where A is a constant set to the maximum value of $g^k/2$.

Constrained iterative Tikhonov-Miller regularized filters:

The well-known unconstrained Tikhonov-Miller filter of image restoration involves minimizing the following objective function [5]

$$\Phi(\hat{f}) = \|g - h \otimes \hat{f}\|^2 + \alpha \|C\hat{f}\|^2 \tag{3.16}$$

where C is the regularizing operation which usually is a Laplacian operator, and α is the regularization parameter chosen to equalize the equation. There are several ways to define the evaluation factor r^k in 2-D image restoration which can be used for the 3-D

constrained-iterative regularized Tikhonov-Miller filters. Representative methods are the steepest descent method and the conjugated gradients method.

In the steepest descent method, the evaluation factor r^k results from the largest decrease in the $\Phi(\hat{f})$ as [47]:

$$r^k = -\frac{1}{2}\nabla_{\hat{f}}\Phi(\hat{f}) = h_b^t g - \left(h_b^t h_b + \alpha C_b^t C_b\right)\hat{f}^k \tag{3.17}$$

The h_b and C_b are both block-circulant matrices of h and C [31]. x^t represents the transpose of matrix x. If α approaches zero, the Tikhonov-Miller filter becomes an iterative algorithm similar to VanCittert's iterative approach.

The conjugated gradient iterative method converges faster than the steepest descent method by setting an additional evaluation parameter p^k [11] [12] [89], which is used to obtain β^k and is related to the evaluation factor r^k as: $p^k = r^k + \gamma^k p^{k-1}$

in which $\gamma^k = \dfrac{\|r^k\|^2}{\|r^{k-1}\|^2}$

The evaluation parameter β is related to p^k as:

$$\beta^k = \frac{r^k p^k}{\|h_b p^k\|^2 + \alpha\|C_b p^k\|^2}$$

Stochastic approaches:

Stochastic approaches maximize the log-likelihood function of the probability density function (PDP) $p(f/g)$ with respect to f [41] [42] [94], that is, they maximize

$$L(f/g) = \ln(p(f/g))$$

The non-negative constraint is imposed by setting $f(x, y, z) = r^2(x, y, z)$. The maximization is implemented as steepest ascent which involves the partial derivative of the log-

likelihood function with respect to *r(x, y, z)*.

In short, algorithms for optical sectioning based on 3-D deconvolution are extension from 2-D deconvolution for image restoration. Deconvolution usually involves iterative approaches so that results can be terminated before convergence and additional constraints can be imposed. Algorithms differ from one another mainly in terms of the objective functions, and thus the evaluation factors used. However, 3-D deconvolution, is very computationally expensive and specialized computers are often required.

3.3 Partial-Minimization-and-Constrained-Iterative Algorithm

In this section, a new optical sectioning restoration algorithm, partial-minimization-and- constrained-iterative (PMCI) algorithm, is presented. This algorithm is based on the simultaneous equations of the image formation model and estimate the intensity function f_i on the focal plane by minimizing the errors between the observed images and the estimated images.

For *(N+1)*optical sectioning images, Eqn. 3.5 yields the simultaneous equations:

$$\begin{cases} g_0 = f_0 \otimes h_0 + f_1 \otimes h_1 + f_2 \otimes h_2 + \ldots + f_N \otimes h_N \\ g_1 = f_0 \otimes h_{-1} + f_1 \otimes h_0 + f_2 \otimes h_1 + \ldots + f_N \otimes h_{N-1} \\ g_2 = f_0 \otimes h_{-2} + f_1 \otimes h_{-1} + f_2 \otimes h_0 + \ldots + f_N \otimes h_{N-2} \\ \qquad\qquad \ldots\ldots \\ g_i = f_0 \otimes h_{-i} + f_1 \otimes h_{-(i+1)} + f_2 \otimes h_{-(i+2)} + \ldots + f_N \otimes h_{N-i} \\ \qquad\qquad \ldots\ldots \\ g_N = f_0 \otimes h_{-N} + f_1 \otimes h_{-(N-1)} + f_2 \otimes h_{-(N-1)} + \ldots + f_N \otimes h_0 \end{cases} \tag{3.18}$$

Let $\hat{f}_i$ (i=0 to N) be the 2-D estimated intensity functions of a 3-D object on the planes i (i= 0 to N). Thus, the estimated images $\hat{g}_i$ of the estimated functions $\hat{f}_i$ can be obtained through Eqn. 3.18. The errors between the observed images and the estimated images are given by $E_i = g_i - \hat{g}_i$ ($i=0$ *to* N). For example, the error for the second plane $i=1$ is:

$$E_1 = g_1 - \hat{g}_1$$
$$= (f_0 \otimes h_{-1} + f_1 \otimes h_0 + \ldots + f_N \otimes h_{N-1}) - \left(\hat{f}_0 \otimes h_{-1} + \hat{f}_1 \otimes h_0 + \ldots + \hat{f}_N \otimes h_{N-1}\right)$$

That is,

$$E_1 = g_1 - \hat{g}_1 = \left(f_0 - \hat{f}_0\right) \otimes h_{-1} + \left(f_1 - \hat{f}_1\right) \otimes h_0 + \ldots + \left(f_N - \hat{f}_N\right) \otimes h_{N-1} \qquad (3.19)$$

If all the differences of the original intensity functions and estimated intensity functions, i.e., $\left(f_i - \hat{f}_i\right)$ (i=0 to N) in Eqn. 3.19 are minimized (in some sense), then the error E_1 will also be minimized. In fact, if all $\left(f_i - \hat{f}_i\right)$ in the simultaneous equations are minimized, all the errors $E_i = g_i - \hat{g}_i$ (i=0 to N) are minimized. At this point, the estimated functions $\hat{f}_i$ are very close to the original intensity functions, and they can be considered as original intensity functions without blurred information from adjacent planes. However, minimizing all the $\left(f_i - \hat{f}_i\right)$ simultaneously is not a trivial task. It involves processing all the images simultaneously, which consumes large amounts of memory and requires many computations so that specialized computers may be required. Further, not all $\hat{f}_i$ approach the true function f_i at the same speed.

An alternative to simultaneous minimization is partial minimization. Partial minimization is similar to that of using the partial derivative of a multi-variable function. In

Eqn. 3.19, if all the factors $\left(f_i - \hat{f}_i\right)$ except the one on the focal plane are fixed, then the error E_1 is partially minimized when $\left(f_1 - \hat{f}_1\right)$ is minimized. Conversely, if the error $E_1 = g_1 - \hat{g}_1$ is minimized with respect to $\left(f_1 - \hat{f}_1\right)$ while other factors $\left(f_i - \hat{f}_i\right)$ are considered fixed, the estimated function $\hat{f}_1$ is an approximation of the original function f_1. The motivation for this approach is based on a property of microscopes, as presented in Chapter 2, which state that the maximum intensity of a defocused PSF drops rapidly with the increase of amount of defocus. It is thus assumed that the amount of out-of-focus information will be much less than the in-focus information. A good approximation of the in-focus f_i can be estimated from g_i using minimization for each of the equations in Eqn. 3.18. This implies that the following objective function is minimized for the (N+1) equations in Eqn. 3.18.

$$\Phi\left(\hat{f}_i\right) = \left\| g_i - \left(\hat{f}_i \otimes h_0 + \sum_{j=-i}^{-1} g_{i+j} \otimes h_j + \sum_{j=1}^{N-i} g_{i+j} \otimes h_j \right) \right\|^2 \tag{3.20}$$

where g_i is the 2-D observed image with focal plane at $i\Delta z$, $\hat{f}_i$ is the estimated 2-D intensity function of the object on the focal plane and $\|.\|$ denotes a L^2 norm. The proposed PMCI algorithm is to minimize the errors between the observed images (g_i) and the estimated images ($\hat{g}_i$) (i=0 to N) by estimating the intensity function on the focal plane. In other words, the PMCI algorithm is to partial minimize the objective function as in Eqn. 3.20. The minimization of $\Phi\left(\hat{f}_i\right)$ with respect to $\hat{f}_i$ (i=0 to N), is accomplished by applying 2-D VanCitter's constrained-iterative schema to ith equation of the simultaneous equations.

Assume that $(N+1)$ is the number of images observed, i.e., the thickness of the object is $N\Delta z$, M is the number of planes adjacent to the focal plane at one side where

($2M+1$) can be chosen either equal to or smaller than N, K is the user predefined number of iterations, ε is the error bound predetermined to terminate the iteration, and L_x and L_y are the size of the observed images on the x- and y-axis respectively. The code for the PMCI algorithm can be written as below:

procedure Partial-Minimization-and-Constrained-Iterative Algorithm

begin

for each image i **do**

$f_i^0 = g_i$ or $f_i^0 = g_i \otimes c_0$ {c_0 is a heuristic highpass filter}

end

for each image i **do**

for k:=1 to K **do**

$$g_i^k = f_i^k \otimes h_0 + \sum_{\substack{j=-i \\ (j\neq 0)}}^{N-i} f_{i+j}^0 \otimes h_j \tag{3.21}$$

or $$g_i^k = f_i^k \otimes h_0 + \sum_{\substack{j=-M \\ (j\neq 0)}}^{M} f_{i+j}^0 \otimes h_j \tag{3.22}$$

$$E_i^k = g_i - g_i^k \tag{3.23}$$

$$f_i^{k+1} = f_i^k + \beta E_i^k \tag{3.24}$$

if $f_i^{k+1}(x,y) < 0$ **then**

$f_i^{k+1}(x,y) = 0$

else if $f_i^{k+1}(x,y) > maximum(g_i)$ **then**

$$f_i^{k+1}(x,y) = maximum(g_i)$$

end {end of non-negative and up-bound constraint}

if $E_{an} = \frac{1}{L_x L_y}\left\|E_i^k\right\| < \varepsilon$ **then** (3.25)

end {terminate iteration if error-bound is reached}

else

continue {iteration}

end {terminate if desired number of iteration is reached}

end {finish estimation for all observed images}

end {finish the PMCI algorithm}

As discussed in Section 3.2.2, the initial estimation of the focal plane f_i can be set to its corresponding observed image g_i:

$$f_i^0 = g_i \tag{3.26}$$

An alternative to this initial estimation is the observed function filtered with a high-pass filter:

$$f_i^0 = g_i \otimes c_0 \tag{3.27}$$

where c_0 is some heuristically determined highpass filter. This alternative is based on the same ground as that of the first assumption in the nearest-neighbor algorithm. Defocus information tends to discriminate high frequencies, and thus the image spectrum G_i contains the specimen spectrum F_i plus the excessive low frequency information from adjacent planes. A high-pass filter thus removes a large amount of defocus information from the observed images and speeds up the convergence of iterations. Note that a high-pass filter is sensitive to noise; this alternative estimation should only be applied to images with high signal-to-noise ratios.

The 2-D VanCittert's iteration schema is applied to minimize the errors of the observed images and the estimated images. For each observed image, Eqn. 3.21 or Eqn. 3.22 is used to compute the estimated image. If Eqn. 3.21 is used, each iteration involves $(N+1)$x$(N+1)$ convolutions for a specimen with $(N+1)$ sections, which are quite computationally expensive. However, since the second summation term in Eqn. 3.21 is fixed for each iteration, only the convolution involving the intensity function of the in-focus plane needs to be updated. This saves a great number of computations as opposed to 3-D restoration algorithms. The error between the observed image and the estimated images is evaluated for each iteration. The error is used to make a new estimation of the function using Eqn. 3.24. The parameter β in Eqn. 3.24 controls the speed of convergence. A non-negative and upper-bound constraint is imposed on middle solutions of the estimated function f_i^k. The constraint trims the estimated function into the range of the observed image. An estimated function is considered close to the original function when its error between the observed image and the estimated image reaches a predefined averaged normal error bound (See Eqn. 3.25) or an acceptable iteration number. The PMCI algorithm terminates when all sections of the intensity functions f_i are estimated.

A great advantage of the PMCI algorithm is that it can process images independently and no 3-D convolution and 3-D Fourier transformation is involved. This tremendously reduces the computation expense, i.e., memory and computation time, so that the algorithm can be implemented on moderate sized computers. The PMCI algorithm also offers a flexibility for users to control the computation complexity and the degree of approximation. In many cases, the user can choose less computation complexity to achieve reasonable deconvolution results. One can choose M planes above and M planes below the focal plane in the simultaneous equations when $(2M+1)$ is less than N. When M is high, the approximation is better, but the computation complexity is high. On the other hand, when M is small, the approximation is not as good but the

computation complexity is low. For a thick specimen, N tends to be large. But M may be chosen to be 1, which corresponds to choosing one plane above and one plane below the focal plane, as in the nearest-neighbor algorithm. Because the PMCI algorithm can process images independently, it is very attractive for implementation with parallel processing. Parallel processing allows one to distribute the computation to several machines and thus significantly reduce the computation time. Finally, the PMCI algorithm breaks the 3-D problem into 2-D problems to avoid the 3-D convolution and 3-D Fourier transform, thus it can utilize the available functions in 2-D cases such as 2-D FFT.

3.4 Results of Optical Sectioning

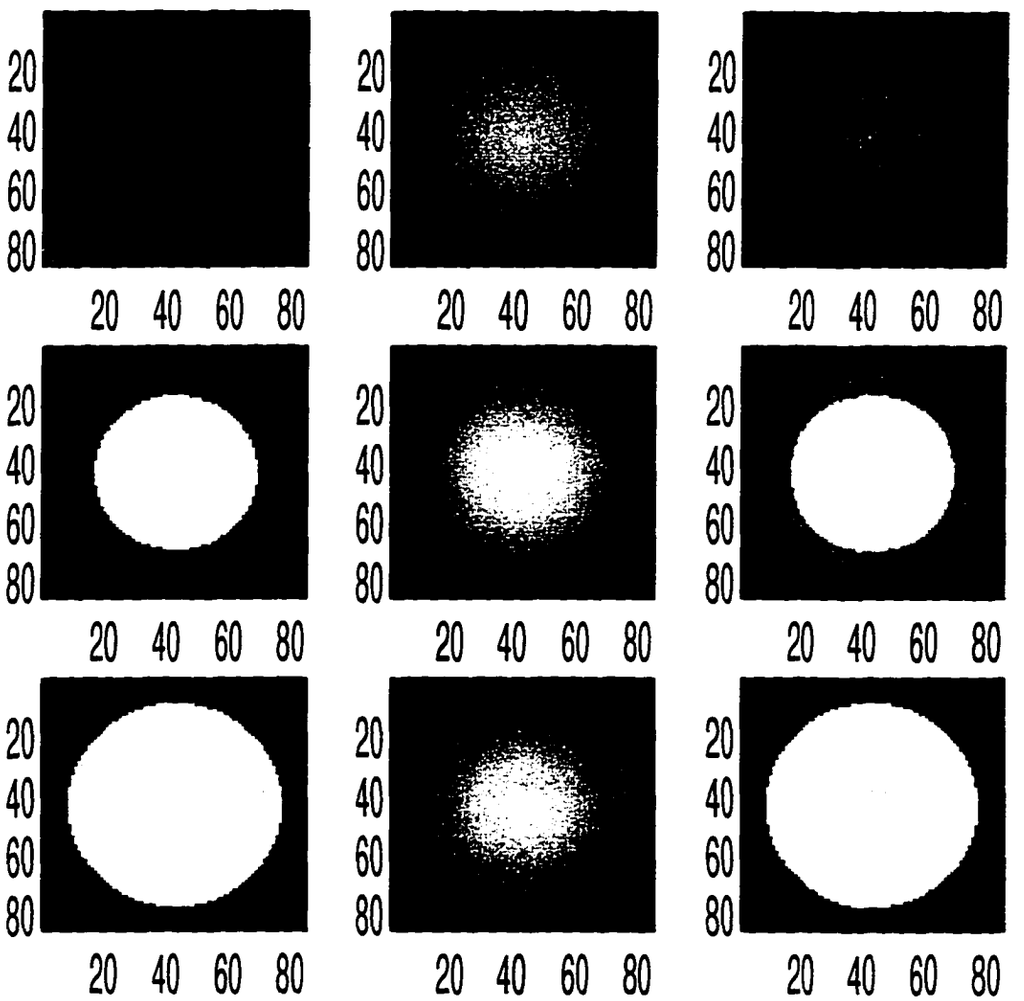
3.4.1 Ideal sphere data

The PCMI algorithm is applied to two sets of data, one is synthetic data and the other one is real microscope data. A series of sections of a sphere is generated for the synthetic data by voxelizing the following equation:

$$f(x,y,z) = \begin{cases} 1, & r \le r_0 \\ 0, & otherwise \end{cases}$$

where $r = \sqrt{x^2+y^2+z^2}$ and r_0 is the radius of the sphere. The radius is chosen to be equal to 40μm, and nine 85x85 sections of the sphere are generated, i.e., the volume is 85x85x9. The distance between each section is 10μm. The generated sections are presented in Fig. 3.2, column (a). Because a sphere is circularly symmetric, only five of the sections are presented in the figure. Each section is blurred as described by Eqn. 3.6, the image formation equation, using six planes above and six planes below the focal plane. The blurred sections are presented in Fig. 3.2 column (b). The axes on the images are

the number of pixels. From Fig. 3.2 column (b), it can be seen that the edges of blurred images (representing the observed images) are smeared, and no clear edges can be observed. The resulting estimated functions of the sphere sections using the PMCI algorithm are presented in Fig. 3.2 column (c). Four planes below and four planes above the focal plane (i.e., M=4) and K=45 iteration are used for the estimation of each function. From Fig. 3.2 column (c), it can be seen that the PMCI algorithm removed a large amount of defocus blur, and edges of sections are clearly visible after the restoration.



(continue on to the next page...)

(Continued from the last page...)

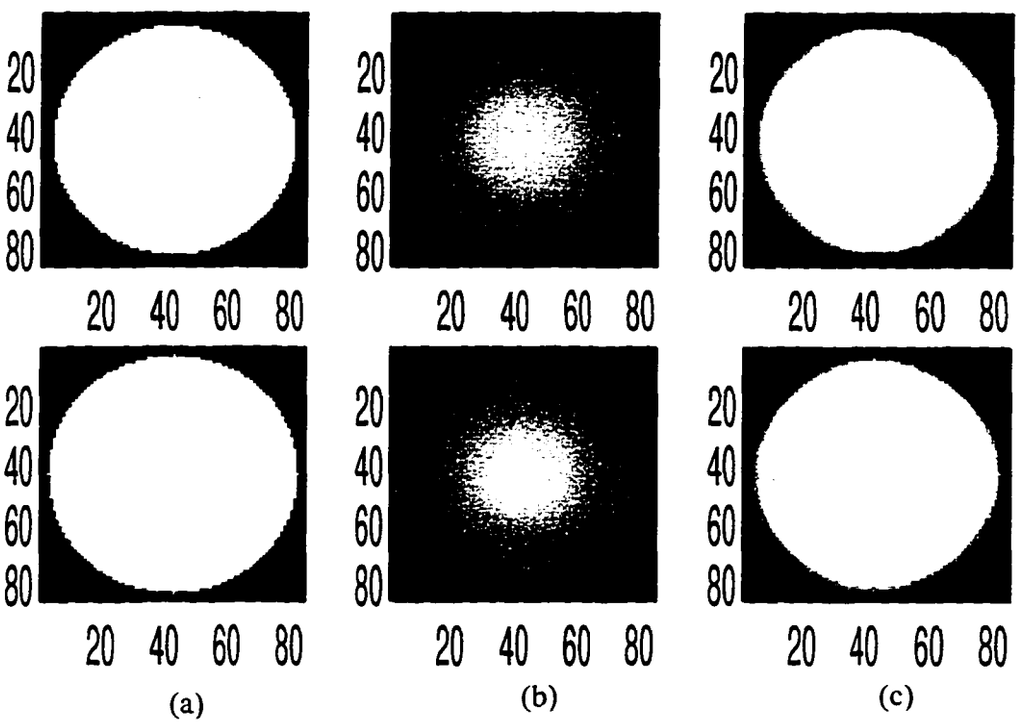


Figure 3.2 The simulation results using a sphere. Column (a) shows the original sphere sections. Column (b) shows the blurred images using the image formation equation where M is chosen to be 6. Column (c) shows the results of the deconvolution using the PMCI algorithm where M is chosen to be 4. The iteration number is 45.

Because the distance between sections is large, 10μm, M equal to four in Eqn. 3.22 is sufficient in image restoration although M is equal to six in the image formation.

In Fig. 3.3, the plots of averaged normal errors E_{an} (See Eqn. 3.25) with respect to the iteration numbers for each sections are presented. From Fig. 3.3, it can be seen

that the errors E_{an} between the observed images and estimated images converge after about 15 iterations for most of the sphere sections. Results for the image corresponding to the top of the sphere converge much slower because the black background dominates the object information.

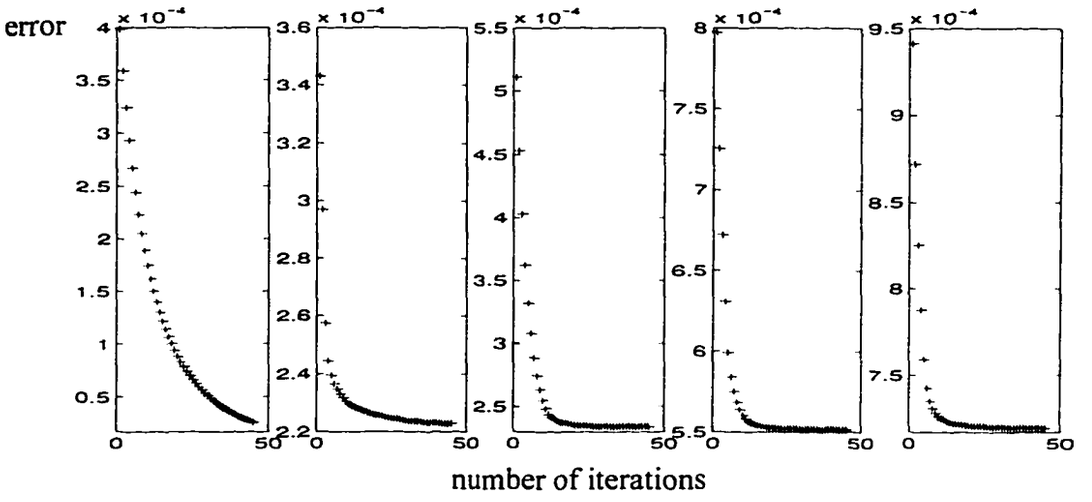
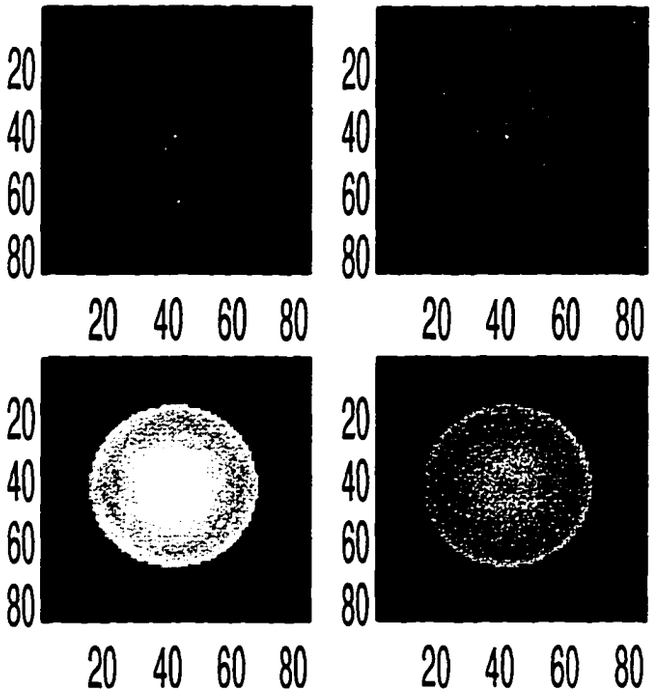


Figure 3.3 The plot of the averaged normal error for each section in Fig. 3.2. As can be seen from the figure, most of the sections converge after about 15 iterations.

In Fig. 3.4, two sets of estimated results are presented using the PMCI algorithm with M=2 and K=45 iterations. The difference in obtaining these two sets of results is the initial estimations used. The left column shows the results using observed images as the initial estimation and the right column shows the results using highpass filtered observed images as the initial estimation. The two sets of results are comparable. The error E_{an} of the third section from the two initials with respect to the iteration numbers are shown in Fig. 3.5. As expected, the PMCI algorithm converges faster using the highpass filtered images as initial estimation (solid line) than using the observed images directly as initial estimation (dashed line) because the highpass filter removes

the excess low frequencies from adjacent defocus planes. From the figure (Fig. 3.5), one can also see that the final E_{an} of the solid line (corresponding to highpass filtered images as initial estimates) is higher than the final E_{an} of the dashed line (corresponding to the observed images as initial estimates). This is because the highpass filter removes the low-frequency information from out-of-focus planes as well as low-frequencies of the in-focus information from the observed images. In the example presented here, the sphere section contains large amount of low-frequencies and they are removed by the highpass filter thus the initial estimation using highpass filtered images causes larger error to the final estimation.



(continue on to the next page...)

(continued from the last page...)

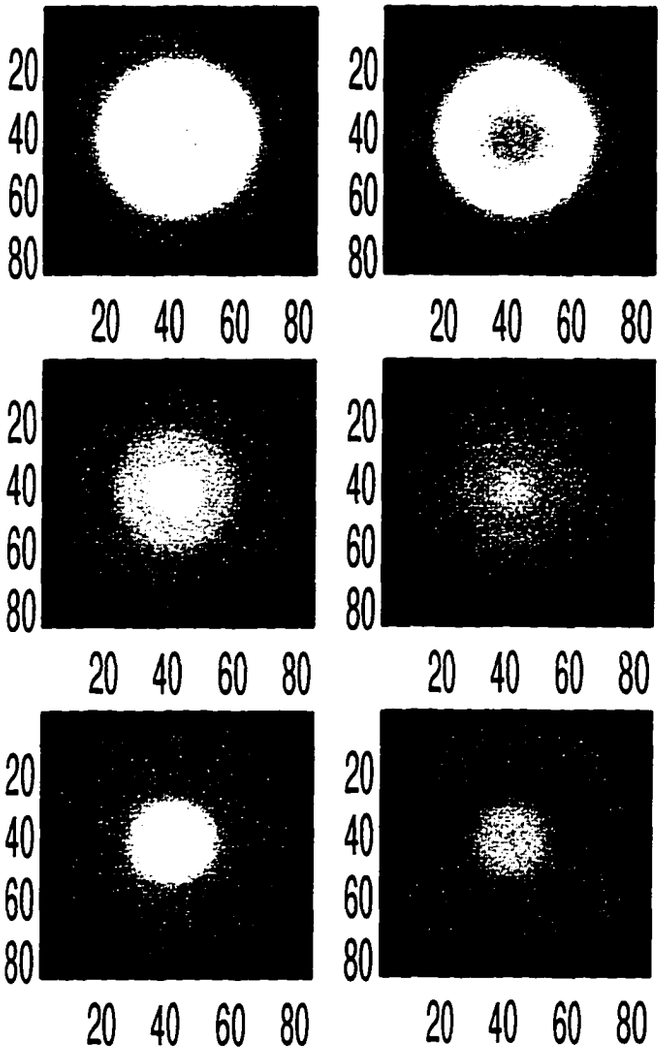


Figure 3.4 The deconvolution results from PMCI algorithm with M=2, K=45. The left column shows the results using the observed images as initial, and the right column shows the results using highpass filtered observed images as initial.

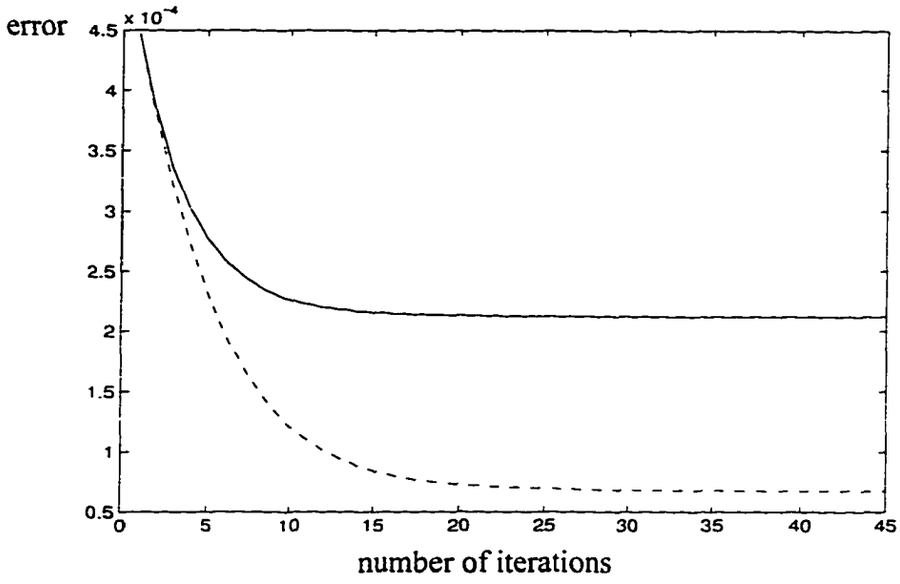


Figure 3.5 Error plot of the third sphere section with respect to the iteration number. The solid line represents the error curve for results using highpass filtered images as initial and the dashed lines represents the error curve for results using observed images as initial. The highpass filtered initial has a fast convergence rate.

In Fig. 3.6, the deconvolution results from the nearest neighbor (left column) and the PMCI with M=1 and K=45 (right column) are presented. The PMCI algorithm uses the same number of planes (M=1) in the deconvolution process as the nearest-neighbor. The computation time for the PMCI algorithm is more than that of the nearest-neighbor algorithm because of the iteration process. However, from Fig. 3.6, it can be seen that the results from PMCI have better restoration results with section edges that are clearer than those from the nearest-neighbor algorithm. Another advantage of the PMCI algorithm over the nearest-neighbor algorithm is that it can estimate the intensity functions for all the observed images, while the nearest-neighbor algorithm does not estimate the

first and last function in the observed image volume.

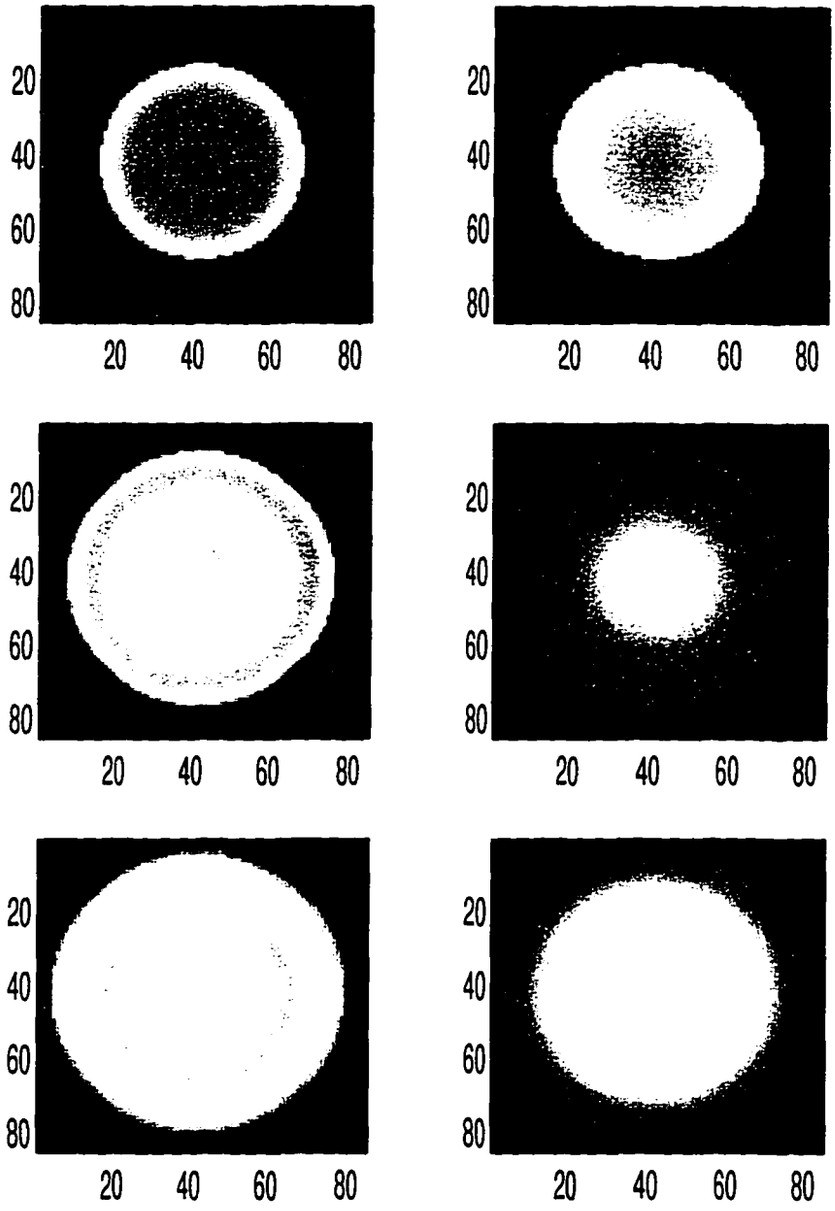


Figure 3.6 The deconvolution results from the nearest-neighbor algorithm (left column) and from the PMCI algorithm with M=1, K=45 (right column).

3.4.2 Pollen images

The PMCI algorithm is also applied to microscope images. Note that for a transmitted light microscope, the image formation model is different from the one given by Eqn. 3.6. Images formed by transmitted light microscopes are the result of subtracting object information from the white background as given by:

$$g_i = B - \left(\sum_{j=1-i}^{N-i} f_{i+j} \otimes h_j \right) \tag{3.28}$$

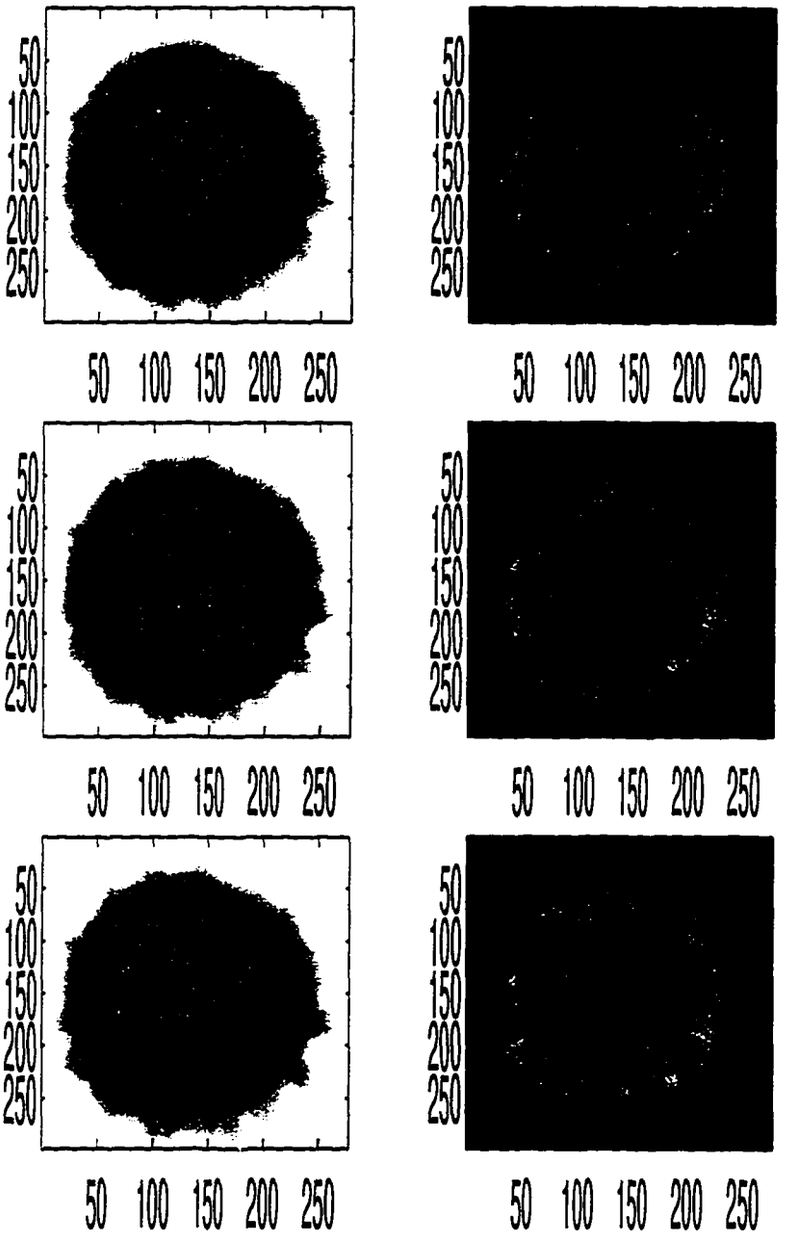
where B is the background. However, Eqn. 3.28 can be transformed to Eqn. 3.29.

$$g'_i = B - g_i = \sum_{j=1-i}^{N-i} f_{i+j} \otimes h_j \tag{3.29}$$

Eqn. 3.29 is actually an inversion of gray levels in an observed image where a white background is transformed into a black background. The transformation of Eqn. 3.29 puts the image formation model of a transmitted light microscope into the same form as image formation models for optical sectioning as Eqn. 3.6. The PMCI algorithm then can be applied to the transformed images for object estimation.

An unstained pollen is used as a specimen, and the transmitted light microscope used in Chapter 2 is used to obtain serial microscope images. The 16x, 0.35 objective lens is used to obtain the pollen images. The defocused PSFs from mathematical models of this objective lens correspond well with those from measurement as presented in Chapter 2. Therefore the defocused PSF of this objective lens from either approach can be used. Each image has the size of 298x276 pixels. The distance between two pixels along the x- and y-axis is 0.58μm where the distance between adjacent images is 4μm. A total of 21 sections are collected for the pollen specimen. In Fig. 3.7, 5 consecutive images from the 21 serial image volume (No. 3 - No. 7) are presented. The left column of Fig. 3.7 presents the original five images digitized from the transmitted light microscope. The right column of Fig. 3.7 presents the corresponding five images after gray

level transformation using Eqn. 3.29, and these transformed images g_i are used as observed images to estimate intensity functions f_i.



(continue to the next page...)

(continued from the last page...)

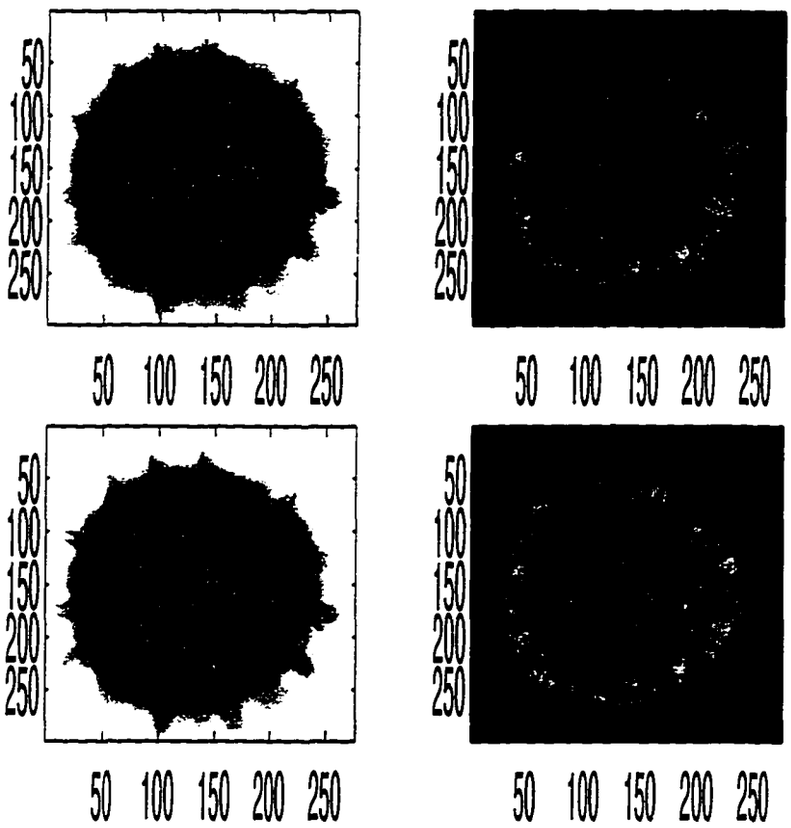
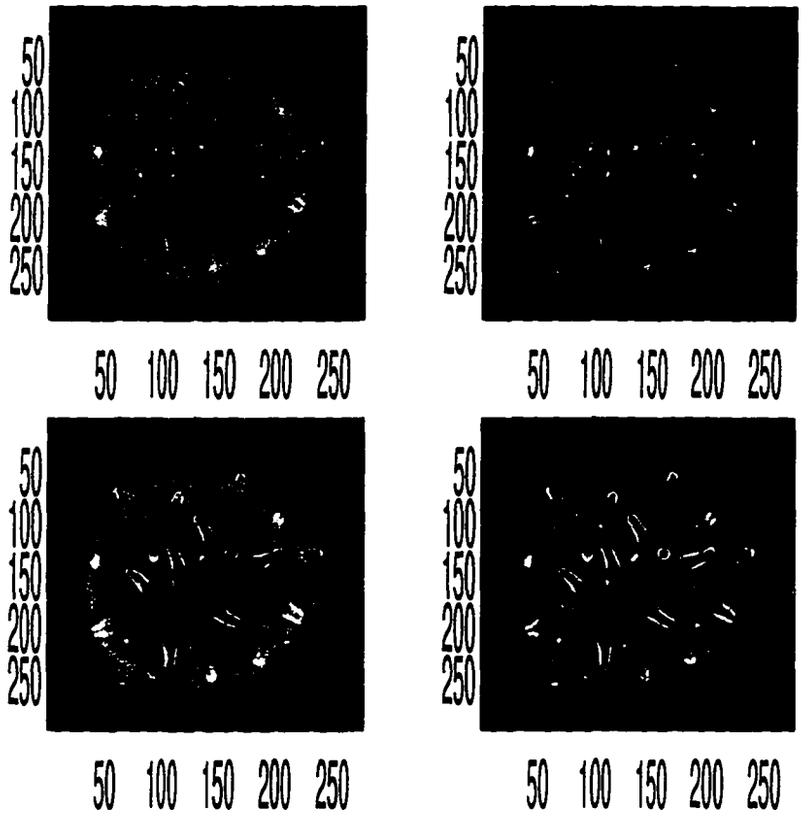


Figure 3.7 The original pollen images obtained from the transmitted light microscope. The left column shows the digitized pollen images from the transmitted light microscope. The right column shows the pollen images after inversion of gray levels using Eqn. 3.29, and these images are used in the PMCI algorithm as observed images.

The estimated intensity functions of the corresponding observed images of Fig. 3.7 using the PMCI algorithm are presented in Fig. 3.8. The left column in Fig. 3.8 shows the results with M=1 and K=45. The right column in Fig. 3.8 shows the results

with M=2 and K=45. Compared with original images in Fig. 3.7, the results presented in Fig. 3.8 show that the PMCI algorithm has removed a reasonable amount of defocus information. It should be noted, however, that some residue of the blurred information still can be observed on the restored images. This is due to the following facts: the specimen is not stained and is quite transparent, which allows more information from adjacent planes to be reflected to the recorded images; the objective lens used has low numerical aperture lens with a large depth-of-field so that a thicker range of object information is in focus.



(continue to the next page...)

(continued from the last page...)

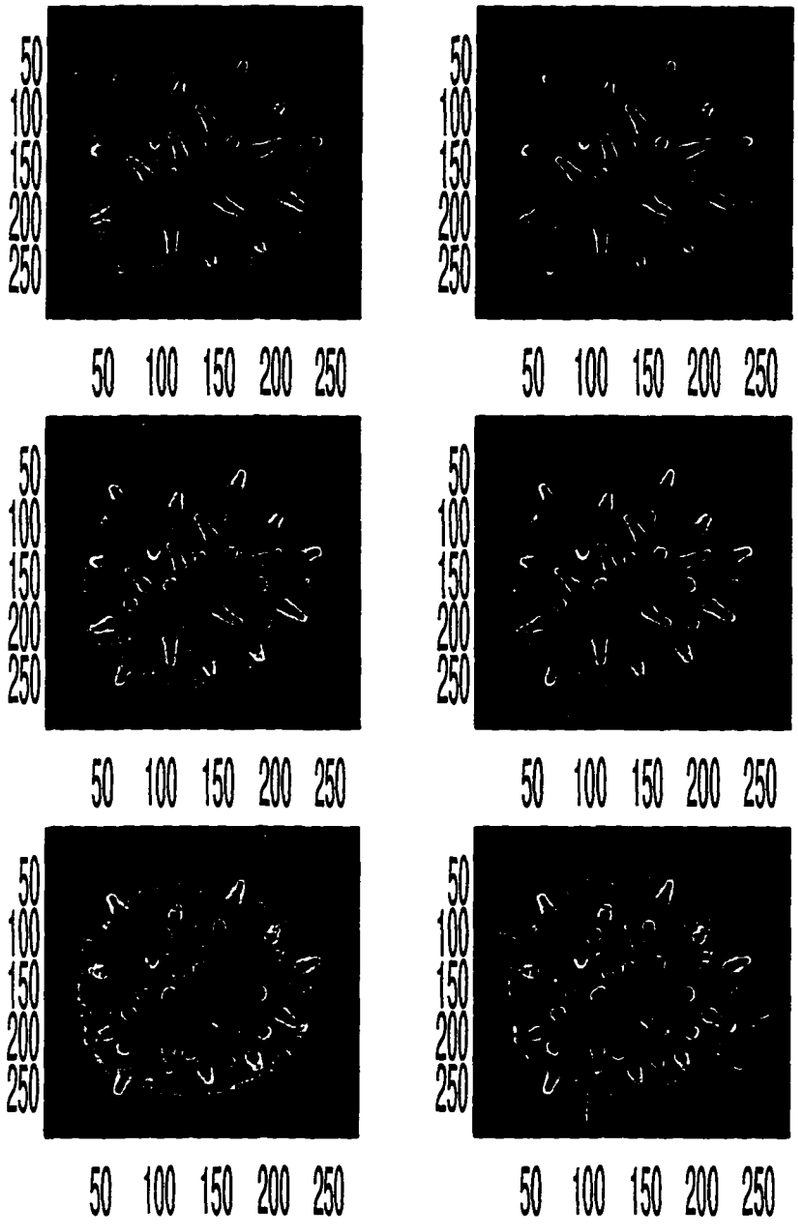


Figure 3.8 The deconvolution results of the pollen section presented in the right column of Fig. 3.7 using the PMCI algorithm. The left column shows the results with M=1 and K=45; The right column shows the results with M=2 and K=45.

The nearest-neighbor algorithm is also applied to the pollen data. Because this set of pollen data has a large spacing (4μm) data, the nearest-algorithm can also produce reasonably good approximations. Two estimated functions from the nearest-neighbor algorithm are presented in the left column of Fig. 3.9. Their corresponding estimated functions by the PMCI algorithm are presented in the right column of Fig. 3.9. The results from the two algorithms may appear close, but a careful examination shows that the results from the PMCI algorithm are clearer and sharper; thus more blur has been removed than from the nearest-neighbor algorithm. This is more evident around the small holes between protrusions on the pollen specimen.

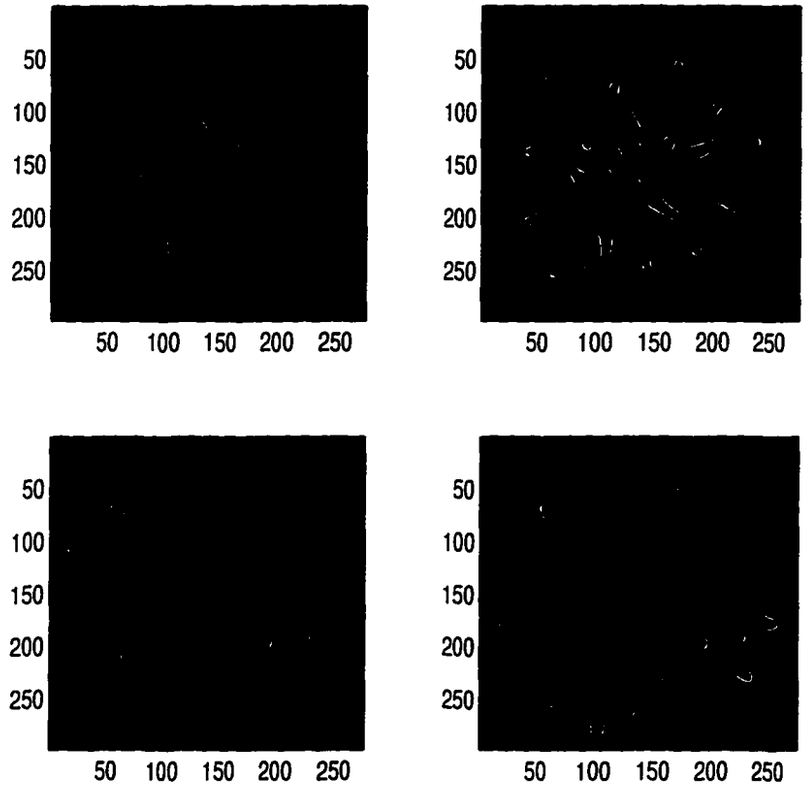


Figure 3.9 The deconvolution results from the nearest-neighbor algorithm (left column) and from PMCI algorithm with M=2 (right column).

3.5 Summary and Conclusion

In this chapter, the principle of optical sectioning has been discussed. Models of image formation, simultaneous equations and 3-D convolution, have been presented, followed by a brief overview of existing optical sectioning algorithms. A new algorithm for optical sectioning, the Partial-Minimization-and-Constraint-Iterative algorithm (PMCI), is proposed. Its principle is to estimate the intensity function on the focal plane by minimizing the error between the observed images and the estimated images. The minimization is accomplished by 2-D VanCitter's constraint-iterative schema. Restoration results for both synthetic and real data show that the PMCI algorithm is effective at removing blurred information from out-of-focus planes. The PMCI algorithm offers several advantages. It breaks the 3-D problem into 2-D problems and can utilize the many of the known functions available in 2-D cases, such as 2-D FFT. It processes images independently, resulting in tremendous reduction of memory requirements and computation time. The algorithm further offers a flexibility to the user to control the computational complexity and the degree of approximation by choosing the number of planes above and below the focal plane involved in the minimization. However, it should be noted that while a reduction in computational complexity in the restoration of 3-D images from volumetric data is the object of this work, image quality from the perspective of the user must remain paramount. Image quality can to some degree be measured quantitatively but there is a large and difficult to define subjective component. The objective of computational efficiency must be balanced against the need for quality and some provision must be made for the user to dictate what is an acceptable price in terms of quality for improvements in computational efficiency.

More complicated constraint-iterative algorithms, such as regularized Tikhonov-Miller filters and especially those that can handle noise problems, can be easily adapted to the PMCI algorithm. This is one of the flexibilities the PMCI algorithm can offer for further development.

Chapter 4

Volume Data Modeling I --- Modeling Algorithms

Serial microscope images, such as those obtained from the last chapter, form volume data when stacked in sequence. Volume modeling distinguishes, extracts and represents objects of interest from acquired volume data. An intuitive and simple example of volume modeling is to extract contours of an object of interest from each slice in the volume data, and to represent the object using meshes of triangles to connect adjacent contours. A volume model should convey as much information as possible, faithfully representing the objects. As volume data usually consist of large amount of data, a volume model should also be efficient in terms of processing time, memory space requirement and the capability for parallel processing. Because of the complexity of real objects, volume modeling is relatively application oriented, with the characteristics of objects of interest taken into consideration. Volume models can be classified according to their applications, or other criteria such as the primitives used [30]. A conventional classification scheme is based on whether an algorithm uses an intermediate representation for the objects of interest [83]. It divides modeling algorithms into two major categories: *surface modeling* and *volume rendering*. Surface modeling uses explicit polygon meshes to represent objects, while volume rendering directly maps the elements of objects to the display screen.

In this thesis, according to the properties of serial microscope images, a high reso-

lution surface generation algorithm, the marching-cube algorithm, is chosen. Two modified algorithms based on the marching-cube algorithm are proposed and are used in the system developed to increase the efficiency.

This chapter presents general volume modeling algorithms and discusses their suitability for serial microscope images. A review of surface modeling is presented in Section 4.1, and volume rendering is presented in Section 4.2. Their comparisons and suitability for serial microscope images are discussed in Section 4.3.

4.1 Surface Modeling Algorithms

Surface modeling algorithms are developed based on the fact that object shapes are specifically defined by their surfaces [36] [52] [86] [91] [95]. The algorithms apply a surface detector to the volume data, extract object surfaces and use geometric primitives, usually polygons, to fit the detected surfaces. The geometric primitives serve as intermediate representations of objects. Techniques that fall into this category differ mainly in the choice of primitives and the scale at which they are defined. The surfaces of objects of interest are formed by properly connecting all the points which have the predefined property, such as intensity. Expressed as a mathematical formula, surface points are:

$$S_\alpha = \{ (x, y, z) \in f(x, y, z) = \alpha \} \tag{4.1}$$

in which $f(x, y, z)$ is the function of the volume data, and S_α is the set of surface points which have the property α of the surface. In most cases, the property of the surface is a threshold value of the volume data function. The threshold value is called an *isovalue*, and surface points are called *isosurface points*. Because all the isosurface points forming surfaces have the same isovalue, surface modeling is also called *isosurface modeling*.

Contour modeling, cuberille modeling and marching-cube modeling are three major surface modeling approaches. Each type of modeling algorithms is discussed in the following sections.

4.1.1 Contour modeling algorithms

Algorithms of this type were used during the early period of scientific visualization. The concepts behind them are straight-forward. The isosurface points, i.e., all the points on surface contours, are either manually traced or automatically extracted from each of the slices. All contours are stacked in sequence and triangular tiles are used to patch each pair of adjacent contours. The modeling algorithm generates triangle meshes to represent the object surfaces under study [27]. These algorithms work well for simple object surfaces whose contours are closed and branch-free curves. That is, a contour may not split into several contours or vice versa. But in most applications, this assumption may not be met, especially when there are several objects of interest. Thus, algorithms of this type often face the branching problem [16] [19]. Another disadvantage of this type of approaches is that whenever a user selects a different object in the volume data, the time-consuming contouring has to be carried over again. However, due to the simplicity and straight-forwardness, these algorithms are still used in many applications. Detailed discussions can be found in [16] [19] [27] [46] [62] [88].

4.1.2 Cuberille modeling algorithms

These types of algorithms assume a constant value within a voxel, a cuberille [6] [26] [36] [37] [38] [73]. After an isovalue is chosen, the volume data is thresholded to produce a 3-D binary volume. The algorithm then distinguishes the isosurface from the background by treating the voxels whose values are below the threshold as background and those above the threshold as the isosurface. For those voxels which are supposed to be the isosurface, their six polygonal faces are generated as primitives and the polygon meshes are then used to represent the surfaces of interest. Cuberille algorithms are regu-

lar and simple, but using orientation-fixed cube surfaces as primitives will produce a quite jaggy object surface [15]. As illustrated in Fig. 4.1, the reconstructed isosurface has many artificial edges although the real object surface just has one real edge. And the angle between the direction of light and the normal to the reconstructed surface can change from small f_1 to large f_2 even when the corresponding angle for the real object surface is constant.

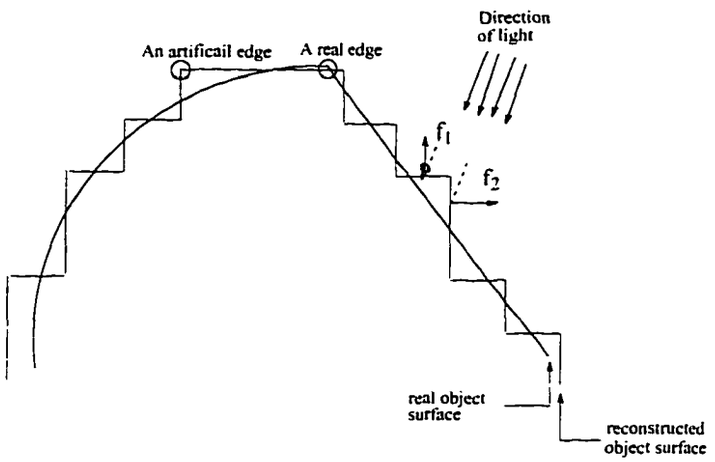


Figure 4.1 A two-dimensional depiction of object surface and the reconstructed surface in the cuberille environment.

Although increasing the resolution of the volume by inserting more points into the volume data by tri-linear interpolating can ease the jagged appearance, it is a very expensive way to overcome the problem. An alternative is to choose a proper shading algorithm. The main parameter to change the shading result is the estimation of the normal of the reconstructed surface to the real object normal. The cuberille algorithms differ from each other mainly in their estimation and use of the object normal [15] [60]. After normal estimations, they all use the following basic equation to calculate illumination:

$$S(P) = \left[\frac{M-L}{2R}(R-d)\,N(\theta) + L\right]_{L}^{M} \tag{4.2}$$

where

$$[v]_L^M = \begin{cases} L, if\ v < L \\ v, if\ L \le v \le M \\ M, if\ M < v \end{cases}$$

In the above equation, d is the distance of P from the source of light, and θ is the angle between the direction of light and the estimated object normal at P. $N(\theta)$ is a function of θ which simulates the properties of a diffuse reflecting surface. M, L, and R are user-defined constants. θ should be close to the real object normal and the neighbor polygons should have close θ values to reduce jaggy appearance.

4.1.3 Marching-cube algorithms

The *marching-cube algorithm* assumes that the scalar function varies tri-linearly within a voxel, and the isosurface, represented by triangles, is located between grid points [52]. The term *cube* is used here to stand for the cubical region bounded by eight grid points, four each from two adjacent slices.

The algorithm determines the relations of cubes and the isosurface as to whether a cube is *inside* or *outside* or *intersecting* the isosurface. In order to obtain such relationships, a threshold value is chosen to correspond to the density value of the isosurface. If the value of a grid point *exceeds* or *equals* the isovalue, the point is positive and is assumed *inside* or *on* the isosurface. If the value is *below* the isovalue, the point is negative and *outside* the isosurface. If all the values of its eight grid points are positive or negative, a cube is inside or outside the isosurface, respectively. A cube is intersected by the isosurface if its two adjacent grid points have different signs. After the algorithm has determined the relationship of a cube with the isosurface, it moves (marches) to the next cube. For those cubes which are determined to be intersected by the isosurface, a set of triangles is created to make up the isosurface of the object of interest.

The isosurface (or triangle) creation is determined by a topology lookup table. Since there are eight vertices in each cube and two statuses, positive or negative, there are $2^8 = 256$ ways a surface can intersect a cube. The *major cases* in which a surface can intersect a cube are shown in Fig. 4.2 after rotation and complementary symmetric cases are excluded. Complementary symmetric cases are those where the statuses of cube vertices are opposite to the statuses of cube vertices in the major cases. The simplest pattern, case 0, occurs if all vertices are above or below the threshold value. This means that the cube is inside or outside the surface, and therefore no triangles are created. These cubes are called *empty cubes*. In case 1, only one vertex is above the isovalue, and a single triangle cutting the three adjacent edges is created. The topology lookup table is produced by permuting these 15 major cases listed in Fig. 4.2.

An index is created based on the statuses of the vertices of a cube as shown in Fig. 4.3. The triangle vertices are calculated by linearly interpolating the values of vertices of the cube along appropriate edges. To produce a smooth shaded image, the normal vectors for each vertex of triangles are obtained by linearly interpolating the gradient vector at each cube vertex along the appropriate cube edge. The unit gradient vector of a cube vertex at *(i, j, k)* is estimated in terms of central differences along the three coordinate axes by:

$$G(i,j,k) = [G_x(i,j,k), G_y(i,j,k), G_z(i,j,k)]$$

$$G_x(i,j,k) = \frac{S(i+1,j,k) - S(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{S(i,j+1,k) - S(i,j-1,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{S(i,j,k+1) - S(i,j,k-1)}{\Delta z}$$

where *S(i, j, k)* is the density at pixel *(i, j)* in slice *k*, and Δx, Δy and Δz are the lengths of the cube edges. Therefore, gradients of all eight vertices of a cube need four consecutive slices. Cubes which are not located at the boundary of the volume only need to be

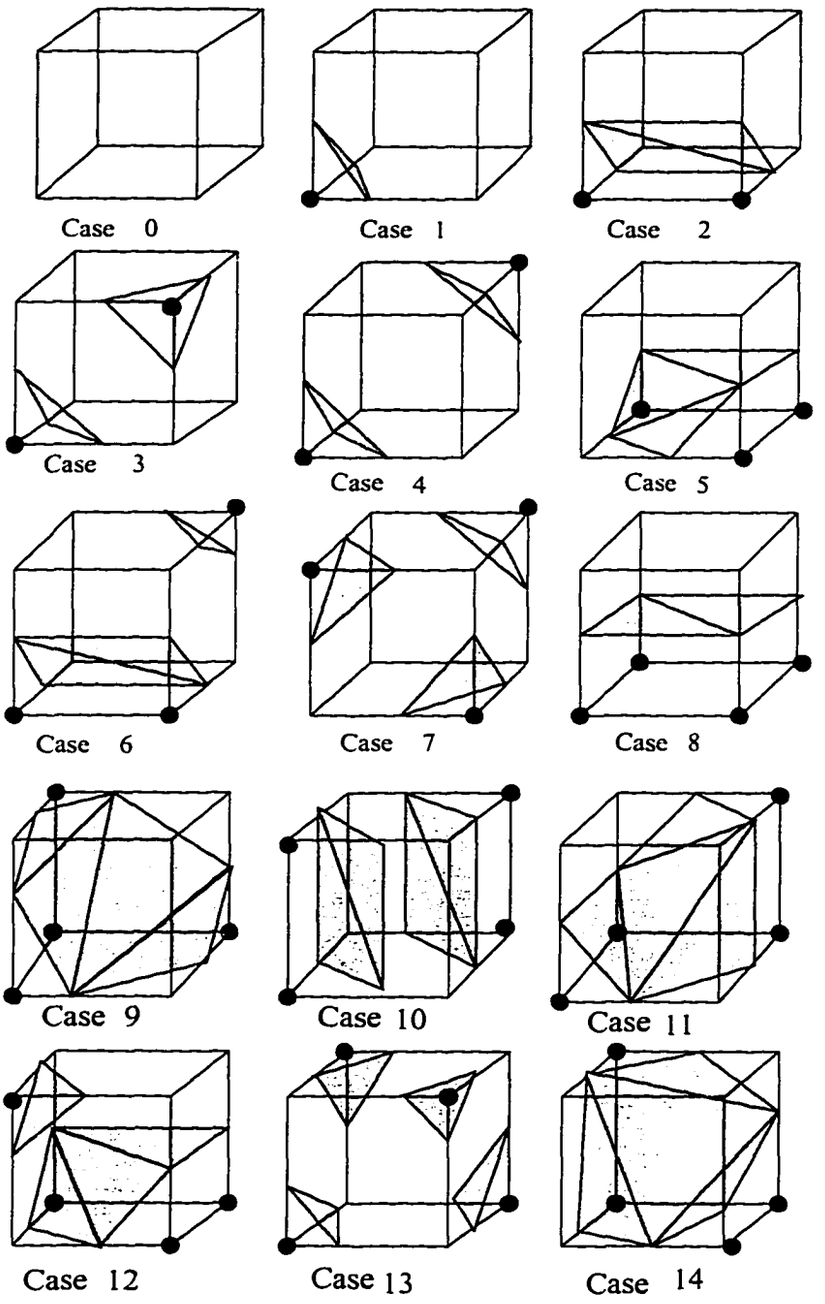


Figure 4.2 Fifteen topologically distinct major cases by which an iso-surface can intersect a cube in the marching-cube algorithm.

interpolated along three new edges, since all other edge values (intersection points and triangle vertex normals) are available from prior interpolations (see Fig. 4.3, where only edges 6, 7 and 12 have to be calculated for the new cube if the surface intersects these edges). After all the cubes in the volume are processed, the triangle mesh and corresponding unit normal vectors are sent to a graphics display processor. One example of an isosurface generated by the marching-cube algorithm is illustrated in Fig. 4.4.

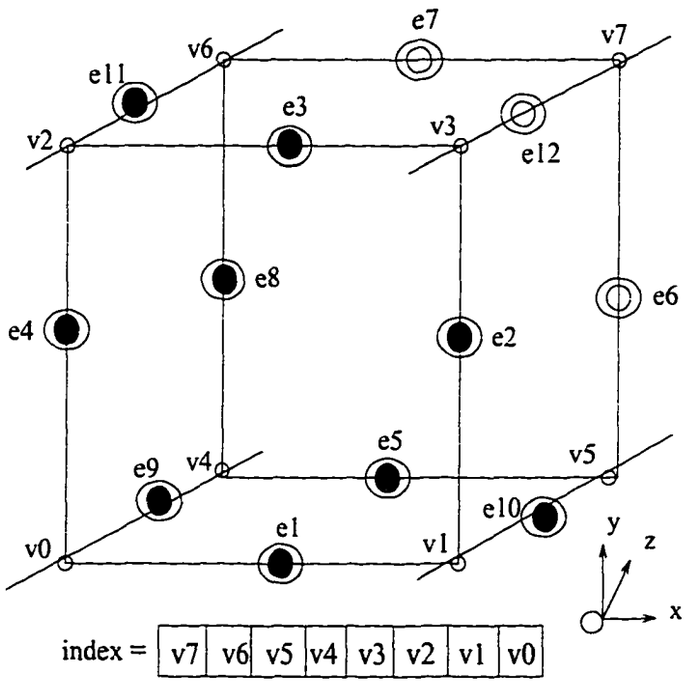


Figure 4.3 Index of a cube in the marching-cube algorithm. The shaded edges are the ones whose information can be reused for the cubes next to them.

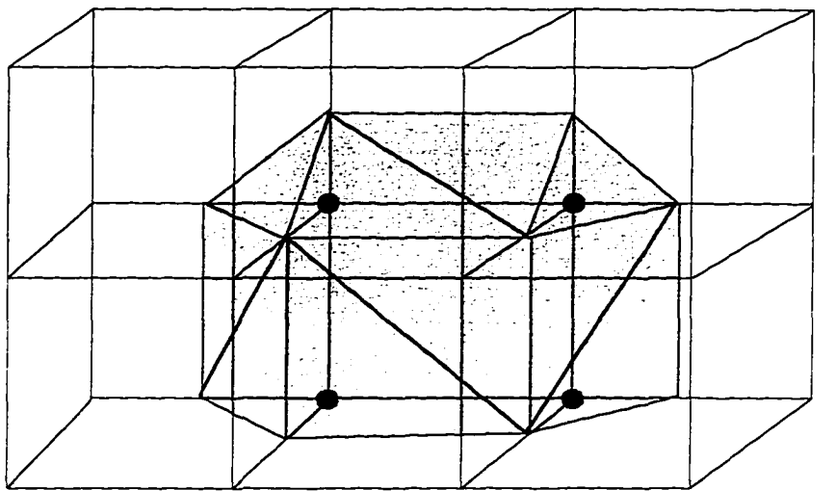


Figure 4.4 An example of an isosurface (gray part) generated by the marching-cube algorithm. The black dots correspond to the vertices with values exceeding the isovalue.

The marching-cube algorithm permits a cutting operation through Boolean operation. Solid modeling uses three notions of *inside, outside* and *on* to create a surface. The indexes of cubes created by the marching-cube algorithm can be put in correspondence with three notions as:

index = 0 for cubes *outside* the surface.
index = 255 for cubes *inside* the surface.
0< index < 255 for cubes *on* the surface.

A cutting plane which has an analytic form of $ax+by+cz-d$ tells where a given point lies with respect to the plane. Let P_{in}, P_{on} and P_{out} represents points that are in, on and outside the surface of a cutting plane, and S_{in}, S_{on} and S_{out} are the points in, on and outside the isosurface. The relations of an isosurface and points of a cutting plane are illustrated in Fig. 4.5.

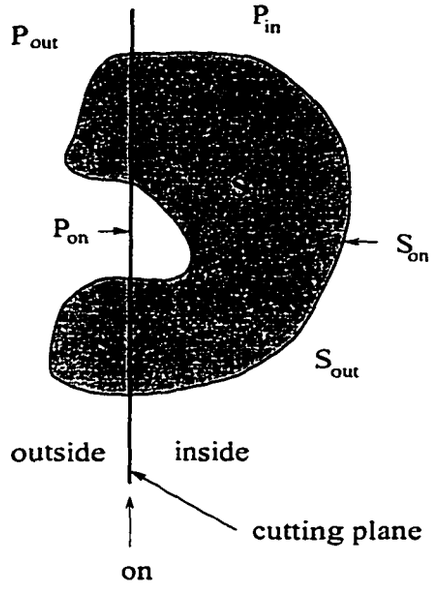


Figure 4.5 Relations of an isosurface and a cutting plane.

Referring to Fig. 4.5, the following observation can be reached. No triangles are generated when a cube is outside the cutting plane or when a cube is outside the isosurface. When a cube is inside the isosurface and also inside the cutting plane, no triangles are generated either. When the cube is on the isosurface but inside the cutting plane, the triangles of the isosurface are generated. When a cube is inside the isosurface but on the cutting plane, the triangles of the cutting plane are generated. When a cube is on the isosurface and also on the cutting plane, each triangle from one surface must clip against each triangle from the other. Each clipping corresponds to two polygons clipping against each other as in computer graphics.The clipping of two polygons can be accomplished by the Sutherland-Hodgeman clipping algorithm [84]. A truth table can be generated as in Table 4.1 from the above statements, with x's representing no operation, S representing the surface from isosurface, P representing the surface from the cutting plane and * representing the surface resulting from the Sutherland-Hodgeman clipping algorithm.

Table 4.1. The truth table for cutting operation.

	P_{in}	P_{out}	P_{on}
S_{in}	x	x	P
S_{out}	x	x	x
S_{on}	S	x	*

The marching-cube algorithm is a high resolution and an effective surface modeling algorithm. However, there are two major concerns. The first one is the ambiguity problem which occurs when a cube face has an intersection point at each of its four edges, such that the positive and negative vertices are diagonally opposite [21], as shown in Fig. 4.6. Thus, it is not clear how to connect these intersection points. An incorrect connection can lead to erroneous topology in the rendered surface and possible discontinuity. Fortunately, in real volume data, the cases of ambiguity do not significantly contribute to the final isosurface. In Fig. 4.2, only cases 3, 6, 7, 10 and 13 have ambiguous faces. These cases do not have high probabilities of appearance, according to previous experiments [93]. The ambiguity problem can be partially solved by methods presented in [61] and [93].

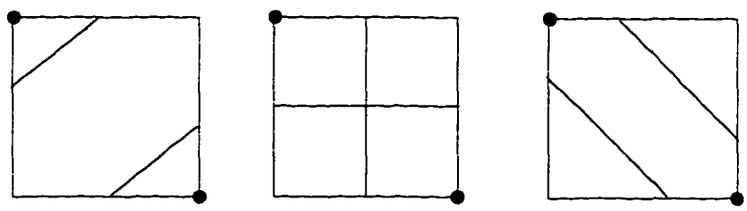


Figure 4.6 Three possible connections in the marching-cube algorithm which cause the ambiguity problem.

The other concern of the marching-cube algorithm, as for all the volume modeling algorithms, is its efficiency. A volume dataset is usually large, and the marching-cube

algorithm produces up to four triangles in a cube thus often producing a large quantity of triangles. Many researchers thus have focused on increasing its efficiency [18] [29] [56] [57] [78] [97]. Methods to increase the efficiency of the marching-cube algorithm include using points as primitives to avoid scan conversion; reducing data traversal time by using a hierarchical database to avoid traversal of empty cubes; and reducing computing and memory space by reducing the amount of triangles generated. In this thesis, two methods (removing redundant polygons and reducing computing time by using a middle-point algorithm to avoid linear interpolation) for increasing the efficiency are proposed and will be discussed in Chapter 5.

Dividing Cubes is an alternative to the marching-cubes algorithm, in which points are the primitives rather than triangles [18]. Hence, this method can eliminate the scan conversion step of the polygonal display and is more efficient in terms of memory space and time. The algorithm subdivides the voxels into small cubes. The voxel scale is chosen to make the small cubes equal to the pixel size on the raster display. It calculates an index for each small cube as in the marching-cube algorithm. For those small cubes that intersect the surface, the gradient vector at each voxel vertex is linearly interpolated to the center of these small cubes to estimate the surface normal vector as illustrated in Fig. 4.7. Both the surface points and normal vectors are transformed and projected onto the viewing plane. The dividing cube algorithm reduces storage requirement and speeds up the computation time by using integers rather than floating points for polygons. However, it does not have the advantage of polygonal algorithms (such as the marching-cube algorithm) to use available computer graphics packages.

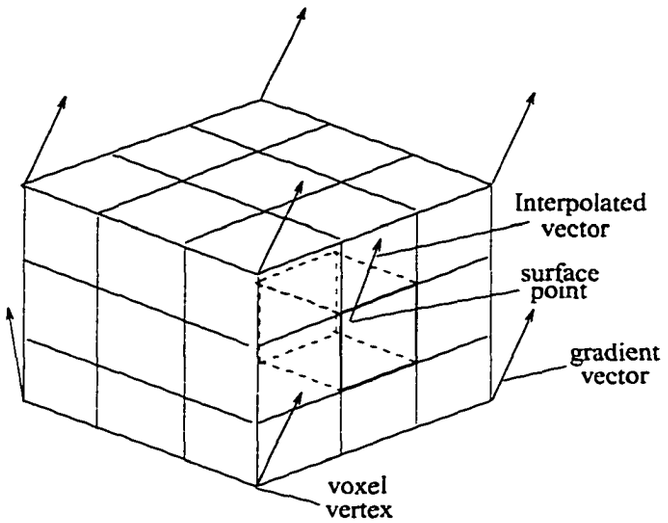


Figure 4.7 Subdividing a voxel into small cubes for a dividing algorithm.

Hierarchical data structures can be used to reduce computation time for almost all the volume modeling algorithms at a cost of increasing memory space [29] [56] [77]. This method is known as *octree traversal* algorithm. Wilhelms and Gelder find that between 30% and 70% of the time of the isosurface-finding phase in the marching-cube algorithm is spent on examining empty cubes [92]. In addition, to find the isosurface for a new isovalue, the whole isosurface-finding phase has to be repeated; there is no carry-on information, and therefore it will suffer a time penalty if the user interactively change the isovalue. The octree-traversal algorithm overcomes the above problems by employing a setup phase that creates an octree before the surface generation phase. An octree is a hierarchical data structure based on recursive decomposition of space into (normally) eight sub-volumes. It is analogous to a quadtree in two-dimensions. The root of the octree represents the entire volume. Each node of the octree contains the maximum and minimum data values found in that node's sub-tree. The leaf nodes of the octree are up to eight vertices of a cube, and each vertex has a single data pointing into the data array. In the surface-finding phase, the octree is traversed with a

particular threshold, with only those branches that contain isosurface are being explored. A node whose maximum is below the threshold or whose minimum is above the threshold is exited without traversing its children. When a leaf node that contains an isosurface is visited, each of the (normally 8) cubes that it covers is visited, and polygons are generated as in the marching-cube algorithm.

Yun and Park proposed a method known as *primitive-polygon* algorithm [97]. The algorithm discards a few major cases (cases 1, 2, 3, 4, 5, 6 and 10 as in Fig. 4.2) which are assumed to be not significant for the final image, and it then classifies other major cases into five polygonal primitives as illustrated in Fig. 4.8 (shaded faces). Each primitive has an orientation number associated with it. The 256 cases can then be categorized into these five primitives and their according orientations. One of the five basic primitives is chosen for a cuberille of the volume data according to the configuration and normal direction estimated from the gradient operator. Although discarding a few major cases would possibly leave some small features out or leave a hole in the shaded image, this method is quite efficient. It is suitable for modeling volume data that contains objects without too many small features.

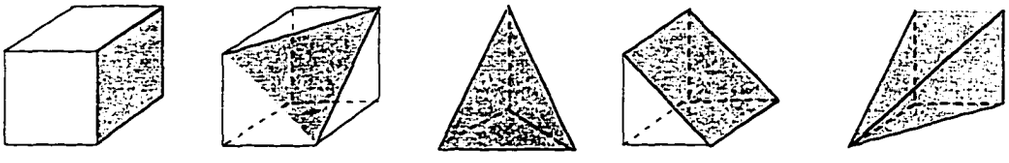


Figure 4.8 Five polygon primitives classified by major cases in the marching-cubes. The shaded polygons are the primary faces which are used to represent object surfaces.

Another alternative is to apply *polygon mesh deduction* algorithms available in computer graphics to the triangle mesh created by the marching-cube algorithm. One typical example is presented in [78]. The algorithm makes multiple passes over an

existing triangle mesh, using local geometry and topology to remove vertices that meet certain criteria. The algorithm is a three-step process. The first step is to characterize the local vertex geometry and topology into one of the three categories, simple, boundary and complex, as in Fig. 4.9. A complex vertex is not a candidate to be removed from the mesh. The second step is to evaluate the decimation criteria. The two criteria, plane and edge, used in the algorithm are illustrated as in Fig. 4.10. A simple vertex uses the plane distance criterion: if $|d| \leq d_s$, where d_s is the pre-set value, the vertex is removed. The boundary criterion uses the edge distance. The third step is to triangulate the resulting hole, i.e., triangulate the loop associated with the deleted vertex. Triangulation is implemented by a loop splitting procedure. A loop is divided into two halves until only three vertices remain in a divided loop to form a triangle. From the Euler relation [64], removing a simple vertex reduces the mesh by two triangles and the boundary vertex by one. The algorithm makes multiple passes of the existing new triangle meshes until a satisfied reduction result is reached.

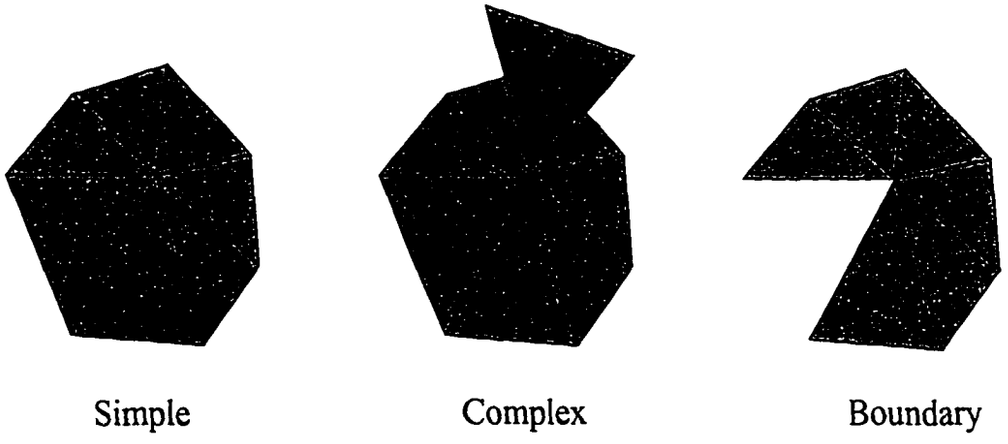


Figure 4.9 Categories of a vertex classification.

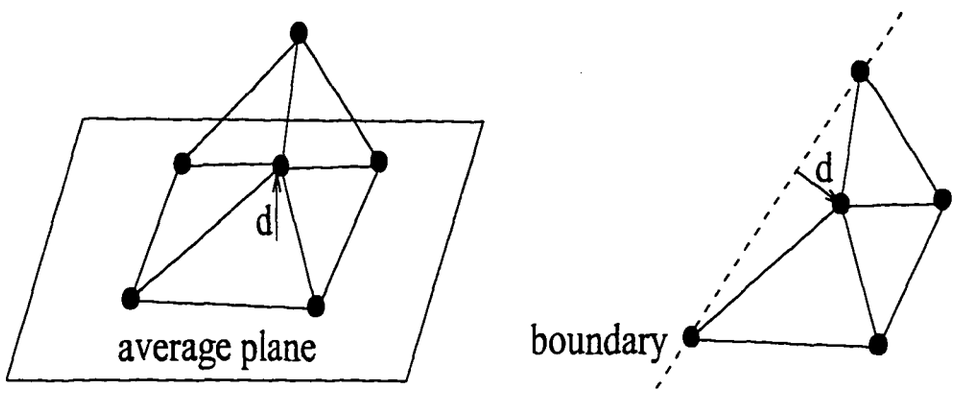


Figure 4.10 Two criteria used in decimation. The left one is the plane distance criterion and the right one is the edge distance criterion.

In summary, contour modeling, cuberille modeling and marching-cube modeling are common isosurface modeling algorithms. The main feature of the isosurface modeling algorithm is that it produces an intermediate representation of the surfaces of objects of interest. The intermediate representation usually consists of polygon meshes. The alternatives among different isosurface modeling algorithms are determined by the choice of the polygon primitives and the scales at which they are defined. A fundamental goal shared by all alternatives is to increase the accuracy and computational efficiency of the algorithm.

4.2 Volume Rendering Algorithms

Volume rendering refers to the modeling algorithms in which object surfaces are not explicitly generated and all the voxels in the volume dataset are primitives. Surfaces are generated at the display time as the result of volume rendering [30]. A volume rendering algorithm usually includes its own shading and rendering algorithm

because it cannot use the shading and rendering algorithms available in traditional computer graphics packages. For this reason, a volume rendering algorithm with its shading and rendering processes is often called *volume rendering.*

A volume rendering algorithm usually starts with creation of four functions, namely R, G, B - the color component functions, and O - the opacity function, according to the components in the volume data. Shading may be applied to these four-colored, semitransparent volumes.The final image is formed by choosing proper rendering methods which maps the shaded or unshaded voxel elements to the screen. There are two main categories of rendering methods. Ray tracing is the first method; in this method a ray emanates from the viewpoint and passes through a pixel in the view plane. After all the voxels encountered by the view ray are processed, the algorithm moves onto the next ray until all the pixels in the view plane have been evaluated or the pixel value reaches unity whichever happens first [45] [50] [76]. The second rendering method is direct projection [20] [63] [87]; in this method each voxel of the volume is projected onto the screen, and the algorithm processes the pixels of the view plane which the voxel's projection covers. In the following, representative volume rendering algorithms for creating color, opacity volumes and their corresponding shading and rendering algorithms will be briefly discussed.

4.2.1 Drebin's approach

Drebin's approach creates the R, G, B and O volumes with the estimated percentage of each material of interest presented in a voxel [20]. The material percentage estimation can be implemented by using probability classifiers. In general, the probability, $P_i(I)$, that any voxel has value (intensity) I is given by:

$$P(I) = \sum_{i=i}^{n} p_i P_i(I)$$

where n is the number of materials presenting in the volume data and p_i is the percent-

age of material *i* in a given voxel; $P_i(I)$ can be obtained by studying the histogram of the input images. A hypothetical histogram and a result of classification functions for a medical image are shown in Fig. 4.11.

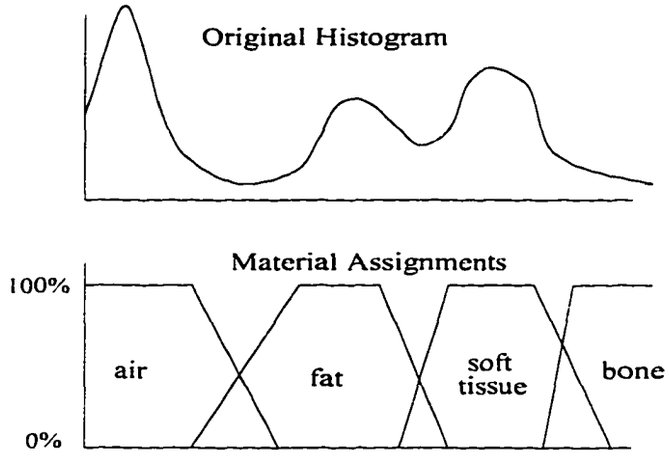


Figure 4.11 A hypothetical histogram and resulting classification functions in Drebin's volume rendering approach.

Each material of interest is assigned to a color and an opacity. Once the percentage volume is available, then R, G, B and O values for a voxel are determined by summing the production of the percentage of the material in the voxel with the assigned color and opacity value as given below:

$$C = \sum_{i=1}^{n} p_i C_i$$

where $C_i = (\alpha_i R_i, \alpha_i G_i, \alpha_i B_i)$, and (R_i, G_i, B_i) is the color and α_i is the opacity associated with material *i* respectively. In order to estimate the surface normal, Drebin, *et al.* assigned another parameter called density ρ_i to each material. A surface occurs when two or more materials of different ρ value meet. The density value for a voxel is the

product of the percentage of each material in the voxel and the assigned material density value ρ, and is calculated in a similar way to the color and opacity values:

$$D = \sum_{i=1}^{n} p_i \rho_i$$

A surface normal is the gradient of the density volume. The normalized gradient is stored as a surface normal volume.

The shading is modeled as follows: if an incoming light with intensity I enters a voxel, a different intensity light will exit from the voxel as the result of two factors. The first is the translucence of the material in the voxel, and the second is the surface or particle scattering that may attenuate and reflect the incoming light. So the resulting light would be

$$I_{out} = C + (1 - \alpha_c) I_{in} \triangleq C \quad over \quad I_{in} \tag{4.3}$$

The first term models the emitting light and the second term models the absorption of incoming light. α_c is the alpha component of the color. For surface shading, a voxel is subdivided into two regions as in Fig. 4.12, a front region and a back region of a thin surface. The intensity of a ray entering and that of a ray going through a voxel are related as follows:

$$I_{out,\lambda} = (C_F over (C_s over (C_B over I)))$$

in which the *over* operation is as defined in Eqn. 4.3 and the reflected surface color is a function of the surface normal, the strength of the surface, the diffuse color of the surface, the direction and color of lights, and the eye position, similar to the Phong illumination (to be discussed in Section 6.2). After the illumination is calculated for each voxel, the final image is formed by direct projection.

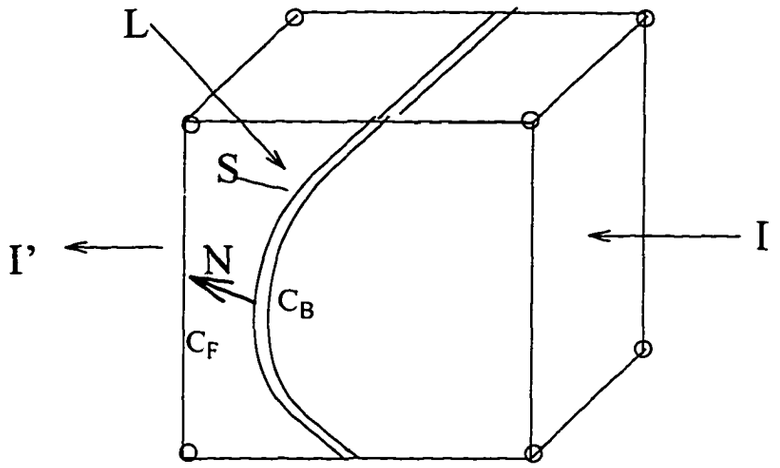


Figure 4.12 Voxel shading model for Drebin's volume rendering. A voxel is divided into two regions: the region in front and a thin surface region behind.

4.2.2 V-buffer

In the V-buffer approach, the 3-D sampled array is treated as a scalar field with a scalar value at each vertex of the cube [87]. The scalar function varies tri-linearly within the cube as:

$$S(x, y, z) = a_1 + a_2x + a_3y + a_4z + a_5xy + a_6xz + a_7yz + a_8xyz$$

The colors and opacities are defined as transfer functions of the scalar function: *O(S(x, y, z)), R(S(x, y, z)), G(S(x, y, z)), B(S(x, y, z))*. Certain features of the object can be enhanced via a variation in opacity and colors by appropriately defining the transfer functions of R, G, B, and O. Two examples of the transfer functions of *S(x, y, z)* are illustrated in Fig. 4.13.

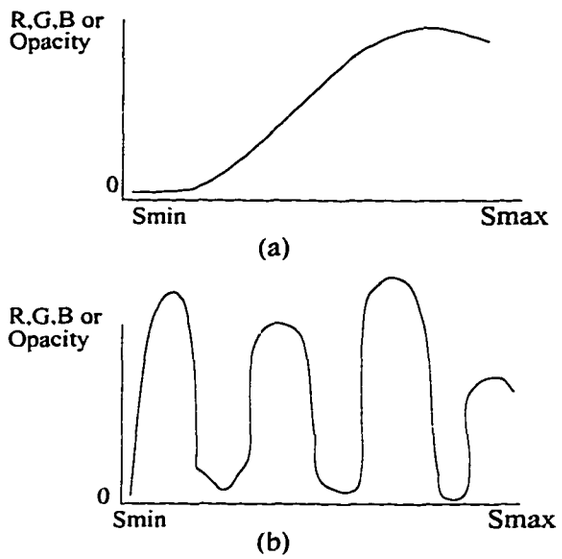


Figure 4.13 Examples of R, G, B and opacity transfer functions in the v-buffer volume rendering approach.

After the color and opacity volumes have been created, shading is applied to all the voxels. Rendering is applied to the color and shaded volumes to produce realistic images. In [87], both ray tracing and direct projection rendering algorithms were presented. One difference between the direct projection rendering method presented in the algorithm and the one commonly used is that the cell processing order is determined by the distance from the viewpoint to the cell; the shorter distance is given the higher priority.

4.2.3 Levoy's approach

In Levoy's approach, only opacity volume is created [49]. He assumes that objects of interest can be put in order by density functions and that each type of object touches only the other types adjacent to it in the order. That is, if there are N object types associated with the scalar function values S_n, $n=1,\ldots, N$, $N \geq 1$, such that $S_m < S_{m+1}$, $m=1,\ldots,$

N-1, then no object of density S_{n1} touches any tissue of density S_{n2}, where $|n_1 - n_2| > 1$. An opacity volume is created in such a way that an opacity value α_n is assigned to the scalar value S_n, α_{n+1} is assigned to S_{n+1}, and an intermediate opacity α_i between α_n and α_{n+1} is assigned to an intermediate scalar value $S_n < S_i < S_{n+1}$ by linear interpolation. In this manner, all voxels are typically mapped onto some nonzero opacity. This ensures that thin regions of tissue will also appear in the final generated image. An example of his opacity function is illustrated in Fig. 4.14.

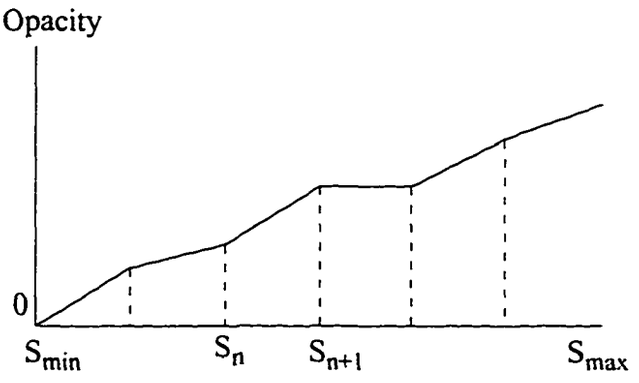


Figure 4.14 An example of an opacity function in Levoy's volume rendering approach.

Levoy's approach doesn't create color volumes. This means that the shading is independent of classifications so that it leads to a robust shading. Meanwhile, shading can be parallelly processed with the creation of the opacity volume.

Volume rendering is usually associated with its shading and image generation procedure because it does not create an intermediate representation for objects. The differences among the algorithms of volume modeling can be found in their approaches to the classification and assignment of objects, and in their rendering procedures. Because

every voxel participates in the modeling and rendering process, the computation for volume rendering is high; creating multiple volumes requires vast memory space. Thus different modifications of volume rendering algorithms are all aimed at an increase of the computation efficiency.

4.3 Surface Modeling vs. Volume Rendering for Serial Microscope Images

One distinct strength of isosurface modeling is its geometric primitives. An isosurface, represented by geometric primitives, takes less storage and, most likely, less computational time because an object can be described by fewer surface elements than by volume elements. Isosurface algorithms can be quickly adapted to the advances in standard techniques of image synthesis and image realism in computer graphics because the rendering of geometric primitives is traditionally involved in these areas. Nevertheless, the isosurface techniques suffer from the common problem of having to make a binary classification which is a high-order non-linear operation thus causing two main side-effects. One side-effect is that small or poorly defined features of an object, whose value is somewhat between none-or-all, would be lost due to binary classification. The other is that virtual artifacts may appear in a generated image when classification error occurs; specifically a spurious surface (false positive) or erroneous holes in surface (false negative) may appear.

Volume rendering algorithms possess some fundamental advantages. By eliminating binary classification, no data is deducted, and this enables displaying small or poorly defined features. Thus, it is more attractive to applications with fine details. By omitting the intermediate geometric representation, surface formation is never explicitly represented. All voxels making up a 3-D scene are accessed at the display time. This

enables fewer interpretation errors of the surface. Further, the relative simplicity of the structure of volume rendering algorithms make them attractive for hardware implementation as well as for an efficient software implementation. However, volume rendering suffers from a number of problems. Foremost is the computational expense, because all voxels participate in the generation of an image. Rendering time grows linearly with the size of the data set. For a sampled original array, in order to produce a 3-D image, other volumes (R, G, B and O etc.) need to be created, and the storage for these volumes is tremendous. Furthermore, the omission of an intermediate geometric representation makes selection of appropriate shading parameters critical to the effectiveness of the visualization. Slight changes in opacity ramps or interpolation will radically alter the features of the objects thus degrading the overall quality of the image. This makes it difficult to create a user-friendly visualization system. Volume rendering algorithms are also very sensitive to artifacts in the data acquisition process due to the omission of an intermediate geometric representation. Any noise from original data can be mapped to a small feature in the final image, and slight misalignment between adjacent slices can produce strong stripes in the final reconstruction image.

If the surfaces of materials are of the primary interest and no small features are presented, surface modeling is likely to be more suitable. If the objective is to understand a volume with fine and multiple surfaces of different properties, volume rendering may be a better candidate. The most important issue is how an algorithm works so that the relevant information for a given application is revealed. Thus, choosing a modeling algorithm depends on the requirement and characteristics of an application.

For serial microscope image visualization especially in cell biology, the main interest is in cell shapes and their spatial relationships. Cell shapes are determined by their cell walls which are coincident with surfaces. These surfaces possess a very small amount of voxels in the whole volume as indicated by an example shown in Fig. 4.15. Unlike medical images which include various tissues and organs such as fat, liver, kidney, bone, air etc., cell walls have relatively consistent gray levels in their images.

Even inner features of the cells have small range of grey levels. Internal structures can be revealed by virtually cutting the volume data at a desired plane. Therefore, a surface model is more suitable. Another advantage of surface modeling for serial microscope image visualization is that the model is independent of viewing and rendering parameters. Thus, when the viewing position is changed, only the rendering process has to be carried out again. In volume rendering, the whole expensive process except the classification has to be repeated. Considering all these factors, surface modeling is better suited for serial microscope image visualization of cell surfaces.

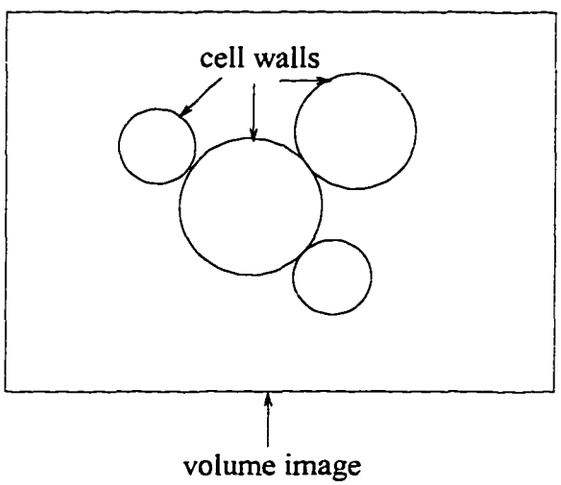


Figure 4.15 Cell walls in a volume image. The pixels occupied by cell walls are only small percentage to the overall image pixels.

The marching-cube algorithm is a *high resolution* and more advanced surface modeling algorithm. It can deal with complex objects since it does not have the branch problem associated with the contouring algorithms. It also produces a smoother surface due to its high resolution and non-fixed orientation of the polygons. Therefore, an efficiency enhanced version of marching-cube algorithm is used to visualize serial microscope images in this thesis.

4.4 Summary and Conclusion

Volume modeling is the process of extracting and representing the objects of interest from the volume data acquired in data acquisition. The modeling algorithms are commonly classified in two categories: surface modeling and volume rendering. Surface modeling takes less storage and computation time because it uses surface primitives rather than volume elements. For the applications of visualizing serial microscope images especially for cell biology, cell shapes and their spatial relationships are of primary interest. Therefore, surface modeling is more suitable in terms of efficiency and effectiveness as cell surfaces are a small percentage of the total volume data. In this thesis, a modified version of an isosurface modeling - marching-cube algorithm - is used for serial microscope image visualization of cell structures. The modified version is more computationally efficient and is discussed in the next chapter.

Chapter 5

Volume Data Modeling II --- Efficiency Enhancements

The efficiency of volume modeling is a primary focus of a modeling process because of the large amount of data involved in volume visualization [18] [29] [56] [57] [78] [97]. Efficiency is related to computation time, memory space requirements, and the capability for parallel processing [56] [57]. The marching-cube algorithm is very attractive for implementation of parallel processing because it processes four images at one time and a volume data can be properly decomposed and sent to several processors. In this thesis, two modifications to enhance efficiency of the marching-cube algorithm in terms of the computation time and memory space (related to the number of polygons generated) are proposed. One modification is to use the *middle-point* to locate a polygon vertex instead of linear interpolation used in the original marching-cube algorithm. The use of the middle-point avoids linear interpolation and thus saves the computation time without a visible degradation in the quality of reconstructed objects. Furthermore, the use of the middle-point, in many cases, leads to co-plane triangles which are merged to a single polygon and thus fewer polygons are generated. The other modification is related to the redundancy problem in the original marching-cube algorithm. The proposed *redundancy removal* algorithm detects and prevents the generation of redundant polygons. In the original marching-cube algorithm, the redundancy problem occurs when sample points are assumed *on* the isosurface and triangles col-

lapse to points and lines. These points and lines are redundant because they do not contribute to the reconstructed isosurface. Nevertheless they use computer resources. These two proposed modifications lead to a more efficient algorithm. The modified algorithm leads to a significant reduction in number of polygons generated and thus to a significant increase in efficiency in terms of memory space and computational time required. The effect of efficiency enhancements by the proposed algorithm is confirmed by experiments using CT and other volume data.

5.1 The Middle-point Algorithm

In the marching-cube algorithm, a cube consists of eight pixels, four from each of two consecutive images [52]. If any of the two adjacent vertices of a cube have different status symbols, a cube is intersected by an isosurface and up to four triangles are generated. A triangle vertex is located by linearly interpolating the values of the two pixels along the proper edges where the triangle vertex intersects as

$$\begin{aligned} x &= x_a + \Delta x \cdot \frac{s_0 - s_a}{s_b - s_a} \\ y &= y_a + \Delta y \cdot \frac{s_0 - s_a}{s_b - s_a} \\ z &= z_a + \Delta z \cdot \frac{s_0 - s_a}{s_b - s_a} \end{aligned} \tag{5.1}$$

where Δx, Δy and Δz are the unit distances of two sample points along the x, y and z dimensions respectively; x_a, y_a and z_a are the x, y and z coordinate components of the vertex a; s_a and s_b are the values of vertex a and b; s_0 is the chosen iso-value. All the parameters are as illustrated in Fig. 5.1.

Since a cube may produce up to four polygons, each polygon created is very small, and some of them are even smaller than a pixel in the final reconstructed image [18]. Thus a slight position change of the polygon vertex within a cube should not affect the final image significantly. In Fig. 5.2, three triangles of different sizes in a cube are presented. The first triangle corresponds to the smallest triangle. In this case, the vertex value of a cube is very close to the iso-value chosen, i.e., (s_0 - s_a) is small; The second one is the middle-point triangle whose three vertices are the middle-points of the edges between two cube vertices. The last one corresponds to the largest triangle. In this case the values of the three cube vertices neighboring to the cube vertex a are very close to the iso-value chosen, i.e. $\frac{s_0 - s_a}{s_b - s_a}$ is close to one. The first and last triangles are the two extremes of triangles produced by a given cube.

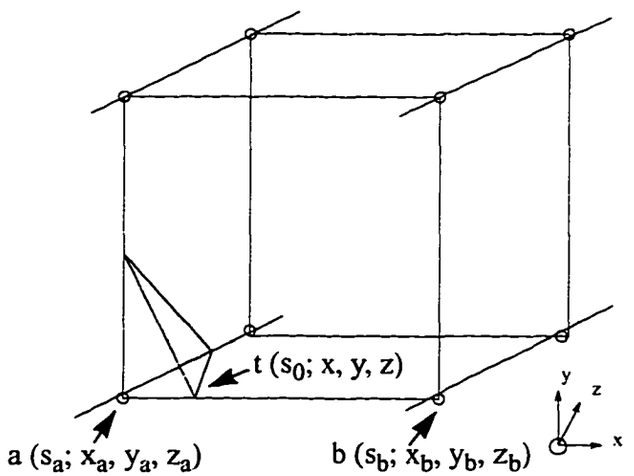


Figure 5.1 Interpolation parameters of the marching-cube algorithm.

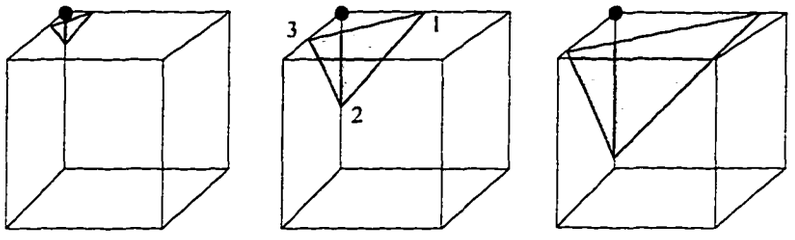


Figure 5.2 Three triangles of different sizes in a cube. The first and last triangle are the smallest and largest one respectively.

Although they are visually different as seen in Fig. 5.2, these three triangles actually have the same scale as a pixel. This is because a cube has the scale of a pixel. The two extreme cases produce a surface with little difference. Based on this assumption, the middle-point algorithm is proposed.

In the middle-point algorithm, a vertex of a triangle along an edge is in the middle of the edge instead of at the point located by linearly interpolating the two sample points as in the original algorithm. The use of the middle-point can reduce computation time by avoiding linear interpolation. The maximum error using the middle-point is half of the distance of two adjacent pixels. For a single vertex, each linear interpolation takes five arithmetic operations: one multiplication, one division, two subtractions and one summation as can be seen in Eqn. 5.1. Even if the factor $\frac{s-s_a}{s_b-s_a}$ is fixed in the equation, it takes a total of nine operations to locate a triangle vertex in the original marching-cube algorithm. In the middle-point algorithm, to locate a vertex requires only three operations, six fewer operations than the linear interpolation. Therefore, for a triangle which has three vertices, a total of eighteen calculations can be saved. In most applications where a large amount of data is involved, the middle-point algorithm can lead to significant saving in computational time.

Another advantage of the middle-point algorithm is that the total number of poly-

gons within a cube is reduced. For example, for case 2 and case 8 in Fig. 5.3, the two triangles produced by both cases will be on the same plane if the locations of triangle vertices are fixed; thus the two triangles can be merged to a quadrilateral. Case 2 and case 8 will create only one polygon instead of producing two polygons. For case 9, all the four triangles will be on the same plane; only one polygon will be output instead of four as in the original algorithm. Fewer polygons generated means less memory space and less computational time for the rendering process. All the 15 major cases of the middle-point algorithm are as illustrated in Fig. 5.4.

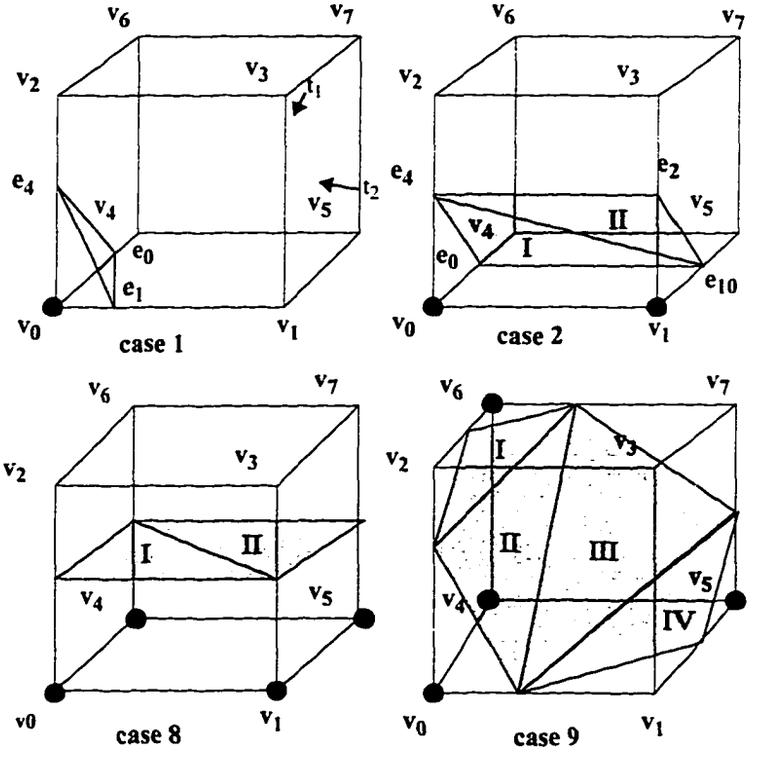


Figure 5.3 Example cases of the marching-cube algorithm.

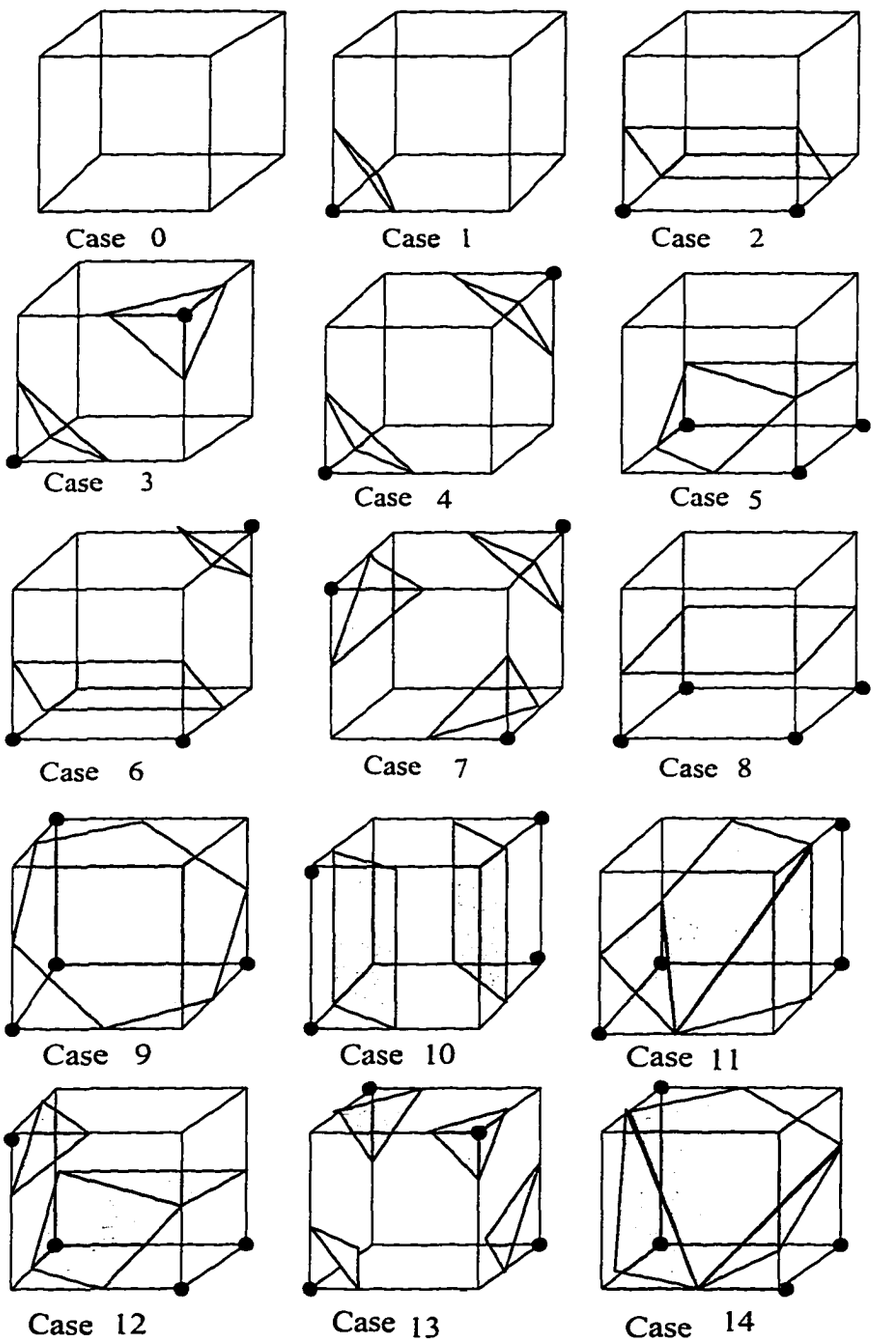


Figure 5.4 Major cases for the middle-point algorithm. Fewer polygons are generated.

5.2 The Redundancy Removal Algorithm

The redundancy problem is due to the fact that the original algorithm treats grid points *on* the isosurface the same as those *inside* the isosurface. This was first presented in [51] and discussed here in more detail. When a grid point is on the isosurface, i.e., a sample value of a cube vertex is equal to the chosen iso-value, triangles produced by the original algorithm may collapse to points and lines which are redundant and do not contribute to the reconstructed images. The two examples shown in Fig. 5.3 illustrates the redundancy problem. For case 1, suppose the sample value of the vertex zero (v_0) of the cube is equal to the iso-value s_0 chosen and thus is on the isosurface, and the rest of the other seven vertices are below the iso-value. According to the figure, one triangle will be produced. Since the sample value of v_0 is equal to the iso-value chosen, i.e., $s_a = s_0$ in Eq. 5.1, all of the three vertices of the triangle produced will merge to the same point, vertex zero v_0, according to the linear interpolation, and a *point* will be produced instead of a triangle. In case 2, suppose the value of vertex zero v_0 is equal to the iso-value s_0, and the value of v_2 is greater than s_0. Then vertices t_1 and t_2 will merge to v_0, and the triangle I collapse to a *line* instead of a triangle. Furthermore, if a rendering program reads only vertices of triangles and computes the surface normal automatically, errors will result because there is no way to calculate the surface normal for a point or a line. However, not every vertex of the major cases shown in Fig. 4.2 has a redundancy problem. In some cases, such as vertex v_4 in case 9 in Fig. 5.3, there is no redundancy since all the edges associated with the v_4 vertex do not intersect with the isosurface. The proposed algorithm will detect the vertices that may cause a redundancy problem, and will offer a solution to prevent the generation of redundant points and lines.

As defined before, a cube has eight vertices, and each vertex has three edges linked to it. An edge is intersected by an isosurface only if two vertices of the edge have a different status after thresholding. An *n-intersect-edge-vertex* is defined if its value is

equal to the iso-value and *n* of its edges are *intersected* by the isosurface. A vertex can be at most a three-intersect-edge-vertex because a vertex has at most three edges linked to it. Suppose the values of all the following example vertices are equal to the iso-value chosen, then vertex 0 (*v0*) of case 1 in Fig. 5.3 is a three-intersect-edge-vertex because all of its three edges, e_0, e_4 and e_{10}, are intersected by the isosurface; *v1* in case 2 of Fig. 5.3 is a two-intersect-edge-vertex because only e_0 and e_{10} are intersected by the isosurface; By the same token, v_5 in case 8 is a one-intersect-edge-vertex; v_4 of case 9 is a zero-intersect-edge-vertex.

A vertex on the isosurface is classified into n-intersect-edge-vertex. It is obvious that a triangle associated with the vertex will become a point or a line if it is a three- or a two-intersect-edge-vertex. Thus, the redundancy problem can be detected in the following way. If a cube vertex is on the isosurface and more than one of its edges is intersected by the isosurface, then this vertex will cause a redundancy problem. The redundancy problem can then be prevented by changing the status of the cube vertex on the isosurface from one to zero in that particular cube while for neighboring cubes the status of the cube vertex and sample value remain the same. As a result, the edges in this particular cube linked to the cube vertex will not be intersected in the cube since its status has changed to zero. At the same time, the isosurface may pass through the cube vertex since the status and value of the cube vertex in neighboring cubes remain unchanged. By changing the status of the vertex on the isosurface from one to zero, re-indexing the cube, and applying the original algorithm to it, the generation of a redundant point or line can be prevented without changing the topology of the isosurface.

An example of redundancy removal algorithm is illustrated in Fig. 5.5 using vertex v_2. In Fig. 5.5 (a), all the values of the vertices with dots are greater than the isovalue chosen, and the resulting isosurface topology is as illustrated in Fig. 5.5. If the value of vertex v_2 is equal to the isovalue, and v_2 is assumed to be *on* the isosurface. The triangle II in cube 2 will collapse into a line along edge I. Thus the original march-

ing-cube algorithm results in an isosurface as in Fig. 5.5 (b) with a redundant line generated along edge I. For redundancy removal, the algorithm examines the vertex v_2. In Cube 1, v_2 is a *one-intersect-edge-vertex*, thus no redundant polygons will be generated. However, in Cube 2, v2 is a *two-intersect-edge-vertex* and redundancy will occur. Thus the redundancy removal algorithm will change the status of v_2 in cube 2 from one to zero. The resulting isosurface is the same as that from the original algorithm except the redundant line (along edge I) is removed.

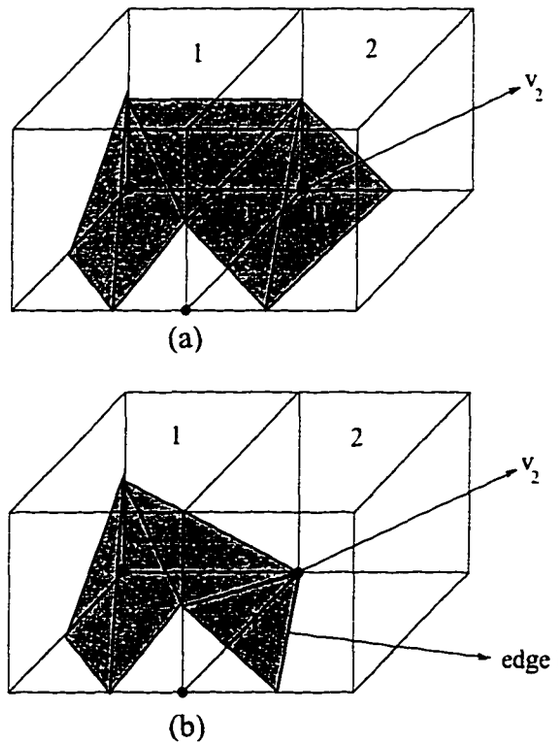


Figure 5.5 An example of an isosurface. (a) The topology of the isosurface when v_2 is not equal to the isovalue. (b) The topology of the isosurface when v_2 is equal to the isovalue. The topology of the isosurface after redundancy removal is the same as in the original one except the redundant edge I is removed.

5.3 Modified Marching-cube Algorithm

The two efficiency enhancements proposed in this thesis have been incorporated into the original marching-cube algorithm. It is clear that triangle vertices are not associated with the sample values of the cube vertices when the middle-point algorithm is used and no triangles will collapse to a point or a line. This means that no redundancy problem will occur if only the middle-point algorithm is used. However, some of triangles generated by the middle-point algorithm correspond to the redundant points and lines which would have been generated by the original algorithm. These triangles are not necessary for the isosurface and they are still redundant and should be removed. In order to discard the redundant polygons, one can first check the redundant vertices and properly index a cube according to the proposition of the redundancy removal algorithm, and then use the middle-point to generate the isosurface. This way the redundant triangles are detected and removed. The inclusion of the redundancy removal and the middle-point algorithm in the original marching-cube algorithm leads to the following modified marching-cube algorithm and can be summarized as follows:

- Visit and index a cube as in the original marching-cube algorithm.
- Check all the values of a cube to determine if there is any cube vertex whose value is equal to the iso-value chosen; if so, determine n, the number-of-intersect-edge-vertex. If n is greater than one, then change the status of the cube vertex from one to zero, and re-index the cube.
- Using the index as a pointer to the topology look-up-table, do all remaining steps as in the middle-point algorithm.

5.4 Experimental Results

The degree of improvement in the performance of the modified marching-cube algorithm over that of the original one depends on the objects used and particularly on the following factors: the percentage of cubes containing the isosurface, the percentage of cubes containing multiple polygons, the percentage of cubes containing sample points whose values are equal to the isovalue, the percentage of vertices causing a redundancy problem.

A mathematically created 16x16x16 cube and a 33x33x33 sphere, and a 256x256x20 CT[1] (Computer-aided Tomography) volume data are used as test objects for the modified marching-cube algorithms. The two mathematically created datasets have only two values, one corresponding to background and the other one corresponding to objects. The threshold values chosen for these two datasets are equal to the value corresponding to the objects. The CT data are 8 bit gray level images originally converted from 16 bit images. In Table 5.1., some statistics from the use of the original marching-cube algorithm on these test objects are listed. These statistics include the percentage of cubes containing isosurface; the number of polygons generated by the algorithm; the time for isosurface generation and rendering; and the total execution time which is the sum of the isosurface generation and rendering time. Time is measured as CPU time which is the sum of the number of seconds of the CPU devoted to the process and the number of the seconds of the CPU consumed by the kernel on behalf of the process. These statistics are listed, in subsequent sections, also for the proposed modifications of the marching-cube algorithms. All the statistics are carried on a Sun Sparc4 workstation [2]. In all the statistics, the same threshold value is used and the unit for the time is second.

---

1. From SoftLab Software Systems Laboratory, University of North Carolina.
2. Sun workstation is a registered trademark of Sun Microsystems, Inc.

Table 5.1.Statistics on the original marching-cube algorithm

Volume Data	16x16x16 cube	33x33x33 sphere	256x256x20 CT
isosurface/cube (%)	43.50%	14.53%	23.64%
polygons created	2,929	9,576	168,027
surface generation time	5.5	20.2	374.4
rendering time	16.6	38.9	778.0
total execution time	22.1	59.1	1152.4

5.4.1 Results of the redundancy removal algorithm

In Table 5.2., the statistics are listed for the redundancy removal algorithm on the same objects as in Table. 5.1. The last two rows list changes in the number of polygons produced and time spent by the modified algorithm as a percentage of the original value. The results show that for the 33x33x33 sphere, the redundancy removal algorithm produces 46.26% less triangles, and thus saves 52.62% execution time. For a real CT data, the algorithm reduces 11.33% triangles, compared with the results by the original algorithm.

Table 5.2.Statistics on the redundancy-removal algorithm.

Volume Data	16x16x16 cube	33x33x33 sphere	256x256x20 CT
isosurface/cube (%)	39.35%	9.74%	23.47%
polygons created	2,657	5,146	148,979
surface generation time	2.5	7.4	320.3
rendering time	13.5	20.6	784.1
total execution time	16.0	28.0	1104.4
Δ polygons / original	9.28%	46.26%	11.33%
Δ total time / original	27.60%	52.62%	4.16%

5.4.2 Results of the middle-point improvement algorithm

In Table 5.3., the same statistics are listed for the middle-point algorithm on the same objects as in the previous tables. The middle-point algorithm does particularly well for the CT data in that it produced 34.39% less triangles, almost 60,000 triangles in this case. The algorithm works well for the other two objects as well.

Table 5.3.Statistics on the middle-point algorithm.

Volume Data	16x16x16 cube	33x33x33 sphere	256x256x20 CT
isosurface/cube (%)	43.50%	14.53%	23.64%
polygons created	1,468	5,617	110,229
surface generation time	3.2	14.9	277.4
rendering time	12.8	28.2	464.3
total execution time	16.0	43.1	741.7
Δ polygons / original	49.88%	41.34%	34.39%
Δ total time / original	27.60%	27.07%	35.63%

Fig. 5.6 and Fig. 5.7 are the 3-D visualization results of the two sets of mathematically created data. In each figure, the visualization results using the original and the middle-point algorithm are presented. The visualization results of the two approaches are visually identical, which indicates that the modification does not cause any visible degradation in the generation of the reconstruction.

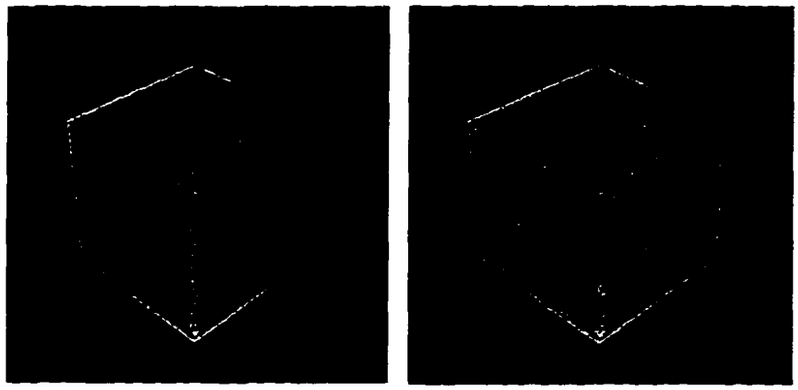


Figure 5.6 The reconstructed cube using the original and the middle-point algorithms. Left: original; right: the middle-point.

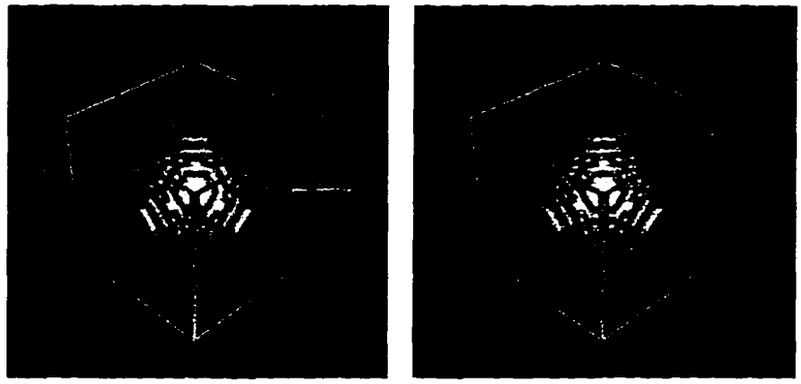


Figure 5.7 The reconstructed sphere using the original and the middle-point algorithms. Left: original; right: the middle-point.

5.4.3 Results of the modified marching-cube algorithm

In Table 5.4., the statistics are listed for the modified algorithm including the redundancy removal and the middle-point algorithm. The high percentages listed in the

last two rows for all the three objects used indicates that the modified algorithm is much more efficient than the original marching-cube algorithm, especially when large datasets are involved. The modified algorithm saves a great amount of memory space by producing even fewer polygons, and saves a great amount of computational time.

Table 5.4.Statistics on the integration algorithm.

Volume Data	**16x16x16 cube**	**33x33x33 sphere**	**256x256x20 CT**
isosurface/cube (%)	39.35%	9.74%	23.4%
polygons created	1328	3192	92,786
surface generation time	2.9	8.2	262.2
rendering time	12.2	16.0	406.9
total execution time	15.1	24.2	669.1
Δ polygons / original	54.66%	66.66%	44.77%
Δ total time / original	31.67%	59.05%	41.93%

5.5 Summary and Conclusion

Efficiency is an important issue in volume modeling because large amounts of data are involved. Two efficiency improvements of the marching-cube modeling algorithm have been proposed in this thesis. One modification deals with the redundancy problem, i.e., generating redundant points and lines, existing in the original marching-cube algorithm. It is shown that the grid points on the isosurface have to be treated differently from those inside the isosurface to prevent the redundancy problem. The other modification uses middle-points to avoid linear interpolation to locate the vertex of a polygon, and thus to increase computational efficiency. The use of middle-point algorithm also allows co-planer triangles to merge to polygons. The two proposed modifications have been integrated into a modified marching-cube algorithm such that it first

detects and prevents the generation of redundant polygons, and then uses the middle-point algorithm for isosurface generation. Results show that the modified marching-cube algorithm significantly reduces the number of polygons generated and thus increases the computation efficiency for surface generation and rendering. Since volume data are usually large datasets, this modeling algorithm with increased efficiency will be a further step in the direction of real-time implementation of surface generation algorithms.

Chapter 6

Object Rendering

Creating a realistically looking image of polygon meshes generated by a surface modeling algorithm, as discussed in the last two chapters, is the task of object rendering. Rendering simulates the interaction of lights with objects represented by polygon meshes.

Rendering starts by traversing the database produced from the modeling process and ends with the generation of a 2-D image [25]. It usually involves the following steps, where the order may differ from one rendering algorithm to another, depending on what algorithm is employed for shading [33] [71], surface determination [6] [14] [75] and scan conversion [90].

- *Data Traversals*: The modeling process produces a polygon mesh database. Data traversal is a process of going through the database to guarantee that all the primitives in the database will be fed into the remainder of the rendering pipeline.
- *Coordinate Transformations:* All the primitives generated by the volume model are in the modeling coordinate system. Coordinate transformation is a process of converting coordinates of polygon meshes from the modeling system to the world coordinates by transforming the vertices of each polygon with a transformation matrix.

- *Trivial Accept/Reject Classification:* This stage is a process of rejecting all primitives, which are wholly outside the view volume, from the rest of the rendering pipeline, since they will not contribute to the final image but take processing energy.

- *Lighting:* Lighting is a process of assigning color values to primitives according to the properties of object surfaces and light sources in the scene. It actually involves illumination which assigns a color to a point on the object surface, and shading which assigns color values to the whole primitive.

- *Projection from 3-D to 2-D:* The purpose of projection is to project, either perspectively or parallelly, the primitives in 3-D space to a projection plane. This projection actually will transform primitives from the world coordinate system to the screen coordinate system.

- *Visible Surface Determination:* When viewing an object from a certain position, a viewer will only see an object partially or not at all if it is obscured by other objects. Visible surface determination determines which part of an object is visible from a defined camera position. It is usually carried out during the scan conversion process.

- *Scan Conversion:* This stage is a process of converting polygons represented by vertices to component pixels into frame buffer for display. On each scan line, the pixels falling within a polygon are evaluated.

- *Display:* This is a process of displaying the colors stored in the frame buffer. If the results are not satisfactory, a user can change either the properties of objects or the viewing position and carry out the whole rendering process again until a desired result is reached.

Using polygon meshes to create complex objects is widely used in computer

graphics. Thus, there are many software packages and hardware implementations available for polygon mesh rendering. In this thesis, the rendering task is accomplished by programming a popular computer graphics C library named SIPP. SIPP[1] stands for *SImple Polygon Processor.* It is a C library for 3-D scene creation and rendering. The fundamental primitives in SIPP are polygons and polygon meshes. A 3-D scene is built up of objects whose surfaces are a number of connected polygons. Polygons are represented by their vertices. Vertices must be given counter-clock-wise when looking at the front side of a polygon. Polygons can be rendered with Phong, Gouraud or flat shading which will be discussed in Section 6.3. Polygons can be rendered as a line drawing without shading at all. A scan-line z-buffer algorithm is used for surface determination and scan conversion. Using SIPP functions to write a program, a user will define the following parameters: objects which are polygons meshes generated from the modeling process; properties of object surfaces, including their colors, orientations, etc.; shading mode such as Phong shading, Gouraud shading, or flat shading, or a wire-frame without any shading; a camera position and a file format that the rendered image writes into. After the program is compiled, then SIPP can carry out the rendering process. The output file format of SIPP is usually Portable Pixel Map (*ppm*) or Portable Binary Map (*pbm*), or users can define their own file format. For *ppm* or *pbm* format files, a special program called *xv*[2] can be used to display the rendered images.

In the present program written in SIPP, users only need to supply parameters such as surface properties, camera position, shading mode and light sources to create a realistic looking image. In this chapter, the discussions will be focused on the rendering stages involving these parameters. Viewing specifications for a camera position will be

---

1. SIPP -- Copyright of Equivalent Software HB, Linkoping Institute of Technology, Sweden.
2. *xv* is a public domain program. Copyright of University of Pennsylvania.

discussed briefly in Section 6.1. For the lighting stage, the illumination models and polygon mesh shading models will be discussed in section 6.2 and 6.3. The rendering pipeline used for typical renderings will be in Section 6.4 followed by a summary of this chapter.

6.1 Viewing Specifications for Projections

Projection from 3-D to 2-D is the process of converting 3-D objects to a 2-D screen for display [54] [64]. Projection depends on the viewing position and the viewing volume. In computer graphics, one way to specify viewing parameters for a perspective projection is illustrated in Fig. 6.1. Six parameters are needed: *viewing from* is the position of a camera; *view at* is where the camera focuses at; *view up* is the up direction vector; *view plane normal* is parallel to and defined by the vector from *view from* to *view at.*; *W* and *H* are the width and the height of the viewing window respectively. In SIPP, the viewing window width *W* and the height *H* are equal. One parameter called *focal* is used instead of *W* or *H*. *focal* is the ratio between half of the view plane height or width and the distance from viewpoint to the view plane. Overall, the SIPP package uses only four parameters to specify a viewing position, namely *view from, view at, view up* and *focal*. The projection from 3-D to 2-D can be converted to a 4x4 matrix as in any projection. Details of projection matrixes can be found in [25].

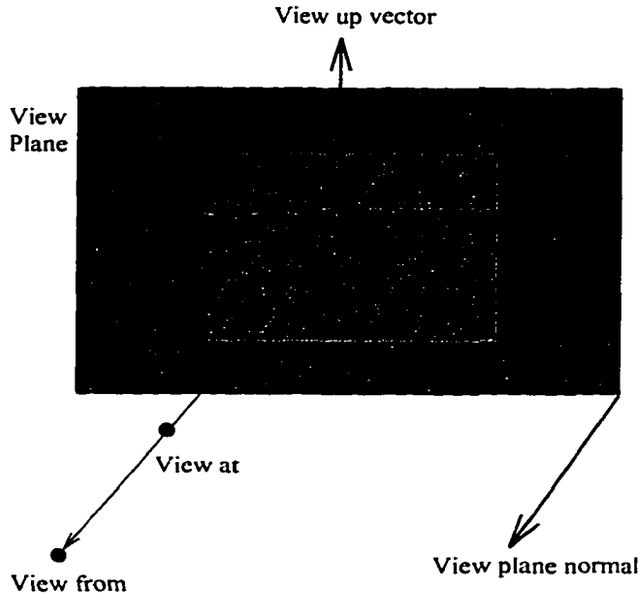


Figure 6.1 One way to define perspective viewing parameters.

6.2 Illuminations

Illumination is a local shading which determines the color of a point on an object in terms of the object's surface properties, position, orientation and the light sources involved [25]. Properties of an object's surface include its original color, shininess, transparency etc. Colors of object surfaces may differ from each other even if they have the same characteristics, because they are located at different positions and thus exposed to the light sources differently. A light source can be a point light from which rays emanate from a single point such as a light bulb, or it can be a directional source from which rays all come from the same direction such as the light from the sun. In order to consider all these factors, the *Phong illumination model* is proposed [70] as:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} \left[k_d O_{d\lambda} \cos\theta + W(\theta) \cos^n \alpha \right] \tag{6.1}$$

in which

I_λ is the intensity at a certain wavelength.

$I_{a\lambda}$ is the light source's ambient intensity at a certain wavelength.

k_a is the ambient reflection coefficient of an object, ranging from zero to one. A higher value of K_a corresponds to more ambient light being reflected from the object. Ambient reflection represents a diffuse, non-directional source of light existing in the 3-D environment which emanates equally from all surfaces from all the directions.

$O_{d\lambda}$ is the diffuse color of an object at a certain wavelength.

$I_{p\lambda}$ is the point light source's intensity.

k_d is the object's diffuse reflection coefficient which is a constant between zero and one. Dull matter surfaces produce diffuse reflection. Diffuse reflection is independent of viewing position because it reflects light with equal intensity in all directions. It only depends on θ.

θ is the angle between the light and the object surface normal at that point. If $\bar{N}$ represents the surface normal at the point and $\bar{L}$ represents the vector from that point to the point light source, then

$$\cos\theta = \bar{N} \bullet \bar{L}.$$

$W(\theta)$ is typically set to a constant k_s, the material's specular-reflection coefficient, which ranges from zero to one. Shiny objects such as glass or metal exhibit specular reflection. A high k_s indicates more specular reflection of light from the object. The value of k_s is selected experimentally to produce pleasing results.

$\cos^n\alpha$ is used to describe specular reflection. It is assumed that maximum specular reflection occurs when α is zero and it falls off sharply as α increases. This rapid

fall-off is approximated by $\cos^n\alpha$, where n is the material's specular reflection exponent. α is actually the angle between vector $\bar{R}$, namely the mirror vector of $\bar{L}$, and the vector $\bar{V}$ which is from the point to the viewer, as illustrated in Fig. 6.2.

f_{att} is a light-source attenuation factor which simulates the attenuation when an object is farther away from a light source. If d_L is the distance that the light travels from the point light source to the surface, then the f_{att} can be either one of those following:

$$f_{att} = \frac{1}{d_L^2}$$

or

$$f_{att} = max(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1)$$

in which c_1, c_2 and c_3 are user-defined constants associated with the light source. All the vectors illustrated in Fig. 6.2 considered, Eqn. 6.1 can be rewritten as:

$$I_\lambda = I_{a\lambda} k_a O_{a\lambda} + f_{att} I_{p\lambda} \left[k_d O_{d\lambda} (\bar{N} \cdot \bar{L}) + k_s (\bar{R} \cdot \bar{V})^n \right] \tag{6.2}$$

If the light source is not a single colored light source, Eqn. 6.1 and Eqn. 6.2 will be applied to each wavelength in the light source, and the color assigned to the point P will be the summation of all the contributions of each wavelength. If multiple light sources are involved in a scene, the final color assigned to the point will be the summation of all the light sources involved, i.e.,

$$I = \sum_{light} \sum_{\lambda} I_\lambda$$

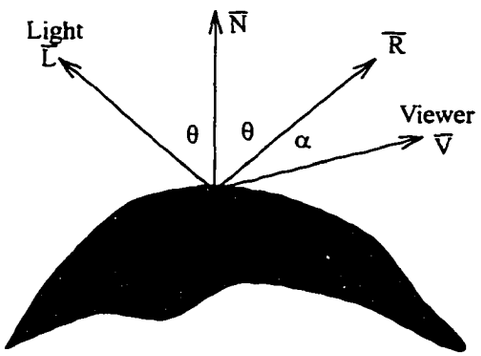


Figure 6.2 The vectors used in the Phong illumination model. $\bar{L}$ is the vector from point P to the point light source. $\bar{N}$ is the normal of the surface at point P. $\bar{R}$ is the mirror reflection vector of $\bar{L}$. $\bar{V}$ is the vector from the viewer to the point P.

It should be noted that the Phong illumination model is a rather general one. Its simplified form leads to other simpler illumination models. One example is obtained by ignoring the last two terms in both Eqn. 6.1 and Eqn. 6.2; the result, as in Eqn. 6.3, is called the *ambient illumination model,* which describes the effect of ambient light. The ambient light can only produce an even intensity over an object and therefore makes the object look flat.

$$I_\lambda = I_{a\lambda} k_a O_{a\lambda} \tag{6.3}$$

Another example is obtained by ignoring only the last term in Eqn. 6.1 and Eqn. 6.2; this yields the so-called *diffuse illumination model* as in Eqn. 6.4 which can be used to produce a realistic image if only the dull objects are presented in the scene.

$$I_\lambda = I_a k_a O_{d\lambda} + f_{att} I_{p\lambda} k_d O_{d\lambda} (\bar{N} \bullet \bar{L}) \tag{6.4}$$

Since the Phong illumination model is more general, it is commonly used for illumination.

6.3 Polygon Mesh Shading

The shading process assigns colors to a more general frame, not only to a point as in the illumination process. In principle, any surface can be shaded by calculating the surface normal at each visible point and by applying the desired illumination model at that point, but this is certainly a very expensive way to shade an object. An object represented by polygons and polygon meshes can take advantage of the intrinsic properties of polygons to increase the efficiency and reduce the cost in shading objects, and this makes the modeling using polygon mesh representation more attractive. The following are commonly used shading modes for polygon meshes.

6.3.1 Constant shading:

This process calls one of the illumination models to calculate the color of a polygon using the normal of the polygon and assigns the color to the entire polygon. Since shading one polygon calls the illumination model only once, it is a very effective approach [25]. However, constant shading is only correct if the polygon represents the actual object surface. For a polygon mesh which approximates a curved surface, if each polygon is shaded by constant shading, it is easy to distinguish neighbors whose orientation is different. The different orientation of neighbors, in general, will produce a *faceted* appearance after shading as a result of the different intensities along their borders. This faceted appearance will make the dark facet look darker and the light one look lighter if the diffuse model or the specular reflective illumination model is applied. The constant shading is correct for polygon mesh shading if both of the following two assumptions stand:

- The light source is at infinity, so that $\overline{N} \bullet \overline{L}$ is constant in Eqn. 6.2 across the polygon surface.

- The viewer is at infinity, so that $\bar{N} \bullet \bar{V}$ is constant in Eqn. 6.2 across the polygon face.

6.3.2 Gouraud shading:

Gouraud shading is an interpolation shading method which can reduce the rigid appearance of an object resulting from constant shading [33]. Gouraud shading is also called intensity interpolation shading or color interpolation shading. The basic idea of Gouraud shading is the determination of vertex colors for polygon meshes. The color of a point in a polygon is interpolated between two vertices along a polygon edge and between two edge points on the scan line as illustrated in Fig. 6.3. The normal for a vertex can be obtained either from the analytical description of a surface or by approximating it from the average of the surface normals of all polygonal facets sharing the vertex as illustrated in Fig.6.4, where the averaged normal $\bar{N}_v = \frac{1}{n}\left(\sum_{1 \le i \le n} \bar{N}_i \right)$.

6.3.3 Phong shading:

Phong shading is also known as normal-vector interpolation shading [70]. It interpolates the surface normal vector $\bar{N}$ instead of the intensity as in Gourard shading. The interpolated normal is normalized, and a new intensity is calculated by calling a desired illumination model. For Gouraud shading, the interpolated intensity is always equal to or smaller than the original intensities which leads to the effect that no highlight would occur inside an polygon. Phong shading can produce more faithful specular-reflection effects because it only interpolates normals on visible points. However, since Phong shading interpolates the normal, normalizes each interpolated normal, and calls an illumination model at each scan line point, it is more time consuming and expensive.

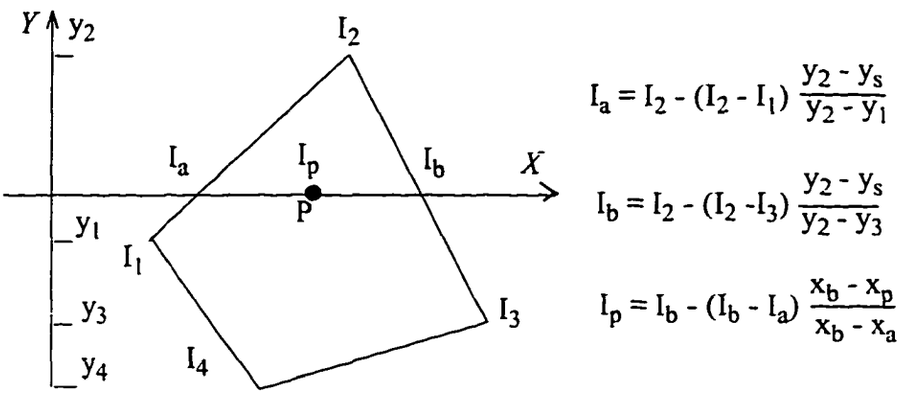


Figure 6.3 Color interpolation along polygon edges and scan lines. The color at point P is interpolated between two edge points I_a and I_b. I_a is interpolated by I_2 and I_4; I_b by I_2 and I_3.

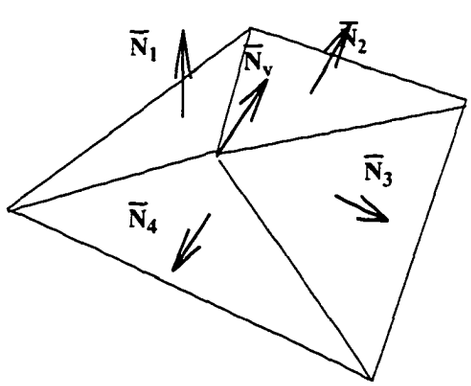


Figure 6.4 Normal calculation for a vertex shared by polygons. The normal at the shared vertex is the average of the polygon normal sharing that vertex.

6.4 Rendering Pipelines

As mentioned earlier, a conceptual rendering pipeline depends on how an algorithm uses shading, visible surface determination and scan conversions. The following are the two typical rendering pipelines.

6.4.1 Gouraud and flat shading with z-buffer

The rendering pipeline for Gouraud and flat shading with z-buffer visible surface determination is shown in Fig. 6.5.

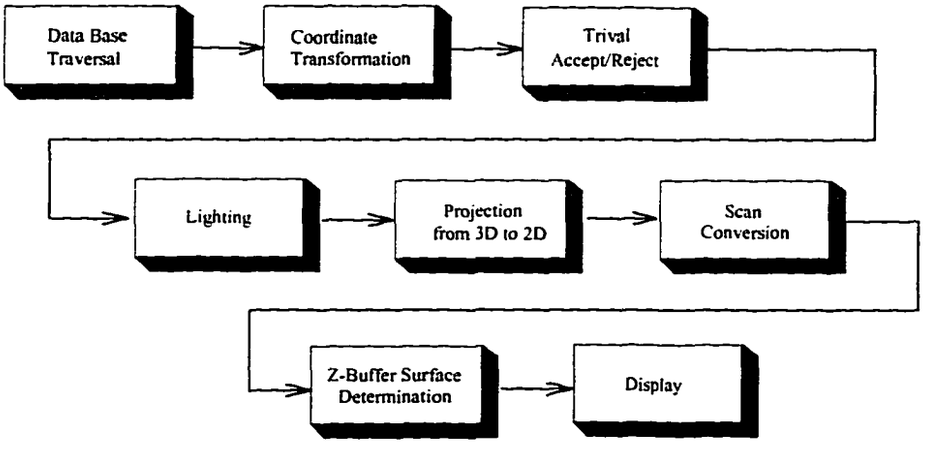


Figure 6.5 The rendering pipeline for the Gouraud, flat shading and z-buffer visible surface determination.

6.4.2 Phong shading with z-buffer

The pipeline for Phong shading is almost the same as the previous one except that the lighting process is carried out at the scan conversion stage instead of before the projection stage. This difference occurs because Phong shading interpolates the normal

instead of the intensity. Every point is needed in order to apply the illumination model to calculate a color. Thus only visible points during the z-buffer surface determination are evaluated.

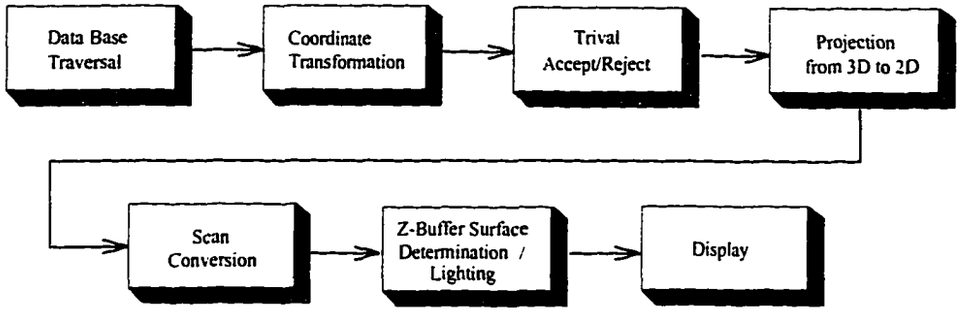


Figure 6.6 The rendering pipeline for the Phong shading with the z-buffer visible surface determination.

6.5 Summary and Conclusion

In summary, a rendering process produces a realistic image of 3-D objects from volume modeling. Rendering involves data traversal, coordinate transformation, trivial acceptance, projection from 3-D to 2-D, scan conversion, lighting, visible surface determination and display. Using polygon meshes in object representation can take advantage of the rapid development of computer graphics in software and hardware. In this thesis, a C rendering library called SIPP is used as part of rendering algorithm implementation. Users can choose different shading models, to define object properties, the light sources involved, and the camera position. The application of SIPP in this thesis allows the whole visualization system to be built efficiently.

Chapter 7

User-interface Design And Implementation

The preceding chapters have focused on each process stage of the 3-D visualization system, namely data acquisition, modeling, and rendering. In this thesis, each of the process is implemented as an individual module for easy debugging and maintenance. Each module, especially for modeling and rendering, needs a large set of parameters and options to define objects and the rendering environment. Mistakes may occur if these parameters are defined using command lines. This is especially true for users without intensive experience in visualization. A graphical user interface can hide the operation details from the user, allowing a new user to learn basic commands fast [25] [24] [55] [72]. A user-interface, MicroVisual, thus is designed and implemented which integrates the modeling and the rendering process. The data acquisition process is not included in the user interface for two reasons. The first reason is that the MicroVisual can be used as a general volume data visualization system dealing with any kind of volume data such as from medical imaging applications. The other reason is that the data acquisition software can be also used as a tool for microscope image quality enhancement and not only for visualization purpose.

The development of MicroVisual involves two steps: the design and the implementation [23] [55]. The design step involves identifying the user requirements and determining the functions needed to specify the parameters and commands required for visualization. Possible inputs, outputs and error during execution for these functions have to be also determined. Implementation involves choosing a proper development

tool to implement the user interface. In section 7.1, the design of MicroVisual will be discussed and the functions will be presented in section 7.2. In section 7.3, the reasons for choosing Tcl/Tk[1] as an implementation tool are discussed. The last section summarizes the MicroVisual design and implementation.

7.1 The Design of MicroVisual

In the design of a graphical user-interface like MicroVisual, the following considerations are important [25] [24] [72]. Firstly, the system should be simple and intuitive, hiding operating details from users as much as possible. A user-interface thereby allows a user, with or without extensive computer knowledge, to master the system quickly and easily. Secondly, the system should provide an on-line *Help* to guide a user when necessary. Thirdly, the system should be able to recover from user mistakes which are unavoidable during the system operation. Finally, the system should provide default values for some of the options and parameters to minimize the start-up time.

The system described here is used for volume data visualization which involves images as input and output, selection of various of parameters and options for modeling and rendering, and visualization commands. Three groups of functions have been defined: *parameter and option entry functions*, *file manipulation functions.* and *visualization command functions*.

The first group, *parameter and option entry functions*, is used to enter parameters and options required for modeling (such as isovalue, volume size etc.), rendering (such as lighting, object color, camera positions etc.) and options (such as inversion of vol-

1. Tcl/Tk is a public software developed by Dept. of Electrical Engineering and Computer Science, University of California at Berkeley.

ume gray values, cutting the volume etc.). The second group, *file manipulation functions*, is used to manage (such as choose, save, open etc.) two kinds of files. One kind of files are MicroVisual files which store parameter and options information for a particular application. The other kind of files are data files, i.e., image files. The third group of functions, *visualization command functions*, are used to execute visualization commands which include basic visualization (modeling and rendering) and *partial* visualization (such as only rendering by omitting modeling or visualizing only the volume box).

7.2 Description of MicroVisual

MicroVisual starts with a main menu window presenting three options to a user, as shown in Fig. 7.1.

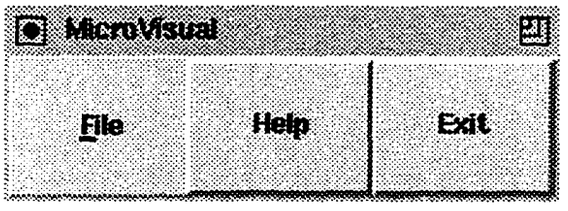


Figure 7.1 The MicroVisual start window

The *Help* option, if clicked, starts a help window in which explanations are given about what the parameters are, how to choose the parameter values, and how to operate MicroVisual. The contents of the *Help* window are listed in the Appendix.

The *Exit* option enables the user to exit the MicroVisual at any time.

If *File* is clicked, a menu is brought up as shown in Fig. 7.2. The options in the menu can be categorized into three groups: *Create* for parameter and option entry func-

tions; *Open*, *Save*, *Save As* and *Select Images* for file manipulation functions; and *Visual Bounding Box*, *Visual Images* and *Rendering Only* for visualization command functions.

The last function, *Quit*, provides the user the option to exit the program from the command window at any time.

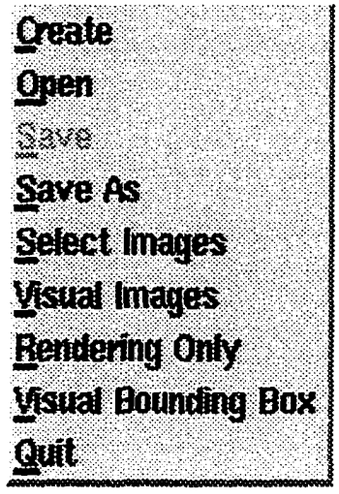


Figure 7.2 MicroVisual command menu.

7.2.1 Parameter and option entry windows

If *Create* is chosen, three parameter windows are brought up: image and shading parameter window, lighting window and cutting plane parameter window.

7.2.1.1 Image and shading parameter window

The image and shading parameter window is shown in Fig. 7.3 and contains three sets of parameters and options: image parameters, options and shading parameters.

Image Parameters: They contain information about the input volume data and

are used in the surface generation process. They include a threshold value for the isosurface, the size of the input volumetric dataset (i.e. its dimensions in x, y, z axis, *xdim, ydim and zdim*) and the distance between two consecutive slices (*dz*).

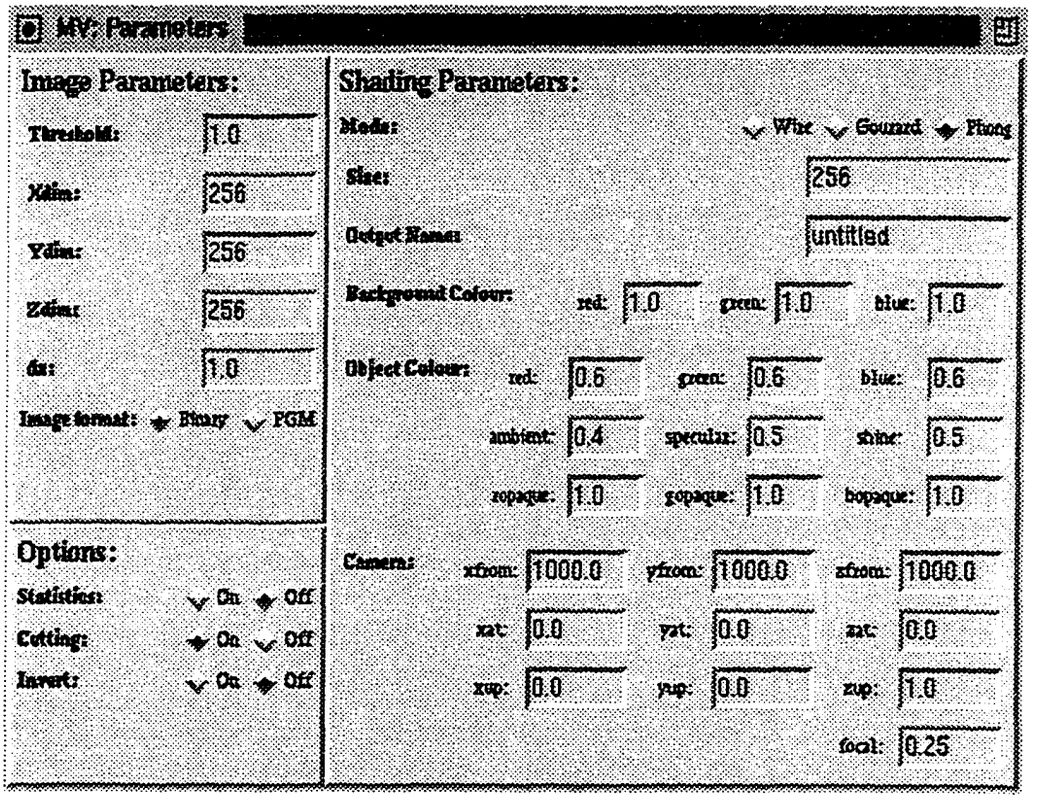


Figure 7.3 The image and shading parameter window.

Options: A user can choose the following options to be either *on* or *off*. The *Statistics* option can be chosen when the user wants to know the information of generated isosurface which includes the percentage of voxels in the volume intersected by the isosurface, the number of polygons generated and so on. The *Cutting* option allows the user to define a cutting plane and to cut through the input volume to reveal inner structures of the volume data. This option will be discussed in detail later in this section. The

Invert option is used to invert the input images. This feature enables the system to consistently assume that the background of an image corresponds to black and that the objects correspond to non-zero gray color levels.

Shading parameters: They are used in the shading and rendering process of polygon meshes generated by the modeling process. The user can choose a shading *mode*, namely the wire-frame, Gouraud or Phong shading, and the *size* (resolution) of the reconstructed image; can choose a desired *background color*, and choose desired properties of the *objects* including their colors, shine and opaqueness; and, can choose the *camera position*. i.e., viewing specification which have been discussed in Section 6.1 of Chapter 6.

To facilitate the difficult task of defining a camera position in 3-D space on a 2-D screen, three supporting features have been added. First, a default value for each of the parameters and options is supplied for the user to start the work; Secondly, a *Help* menu is provided in which instructions are given on how to choose proper values for the parameters. Thirdly, a quick rendering process is available using the *Visual bounding Box* visualization command. This command generates a view of the box surrounding the volume data using the current shading parameters and allows the user to quickly decide if the current parameters are appropriate to generate a desired view. After some sessions using MicroVisual, the user should be confident to choose appropriate shading parameters.

7.2.1.2 Lighting parameter window

The lighting window, as shown in Fig. 7.4, is used to specify lighting parameters used in rendering. Using a separate window for the lighting parameters will allow the dynamic expansion of the numbers of light sources involved in the rendering. Any light in the light parameter window can be deleted by the *Delete* button. The color of a light is defined by three numbers corresponding to its red, green and blue components.

The position of a light is defined by its *x*, *y*, *z* coordinates.

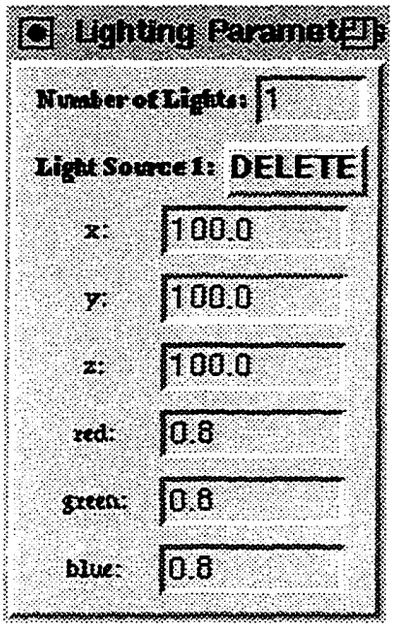


Figure 7.4 The lighting parameter window.

7.2.1.3 The cutting parameter window

The cutting operation allows the user to section the volume along a desired plane and to generate the view along a cutting plane so that it can reveal internal structures of a dataset, which is impossible to do with physical sectioning. The cutting operation is one of the most useful features in the visualization system.

To define a plane, three points on the plane are needed. A point can be defined in two ways: through the keyboard or through a canvas presented on the screen. To define a point in 3-D space on a 2-D display, one possible method is to use two canvases, the *x*-*y* plane canvas and the *y*-*z* plane canvas, as shown in Fig. 7.5. This method corresponds to the projection of a point onto the *x*-*y* plane and then to the *y*-*z* plane.

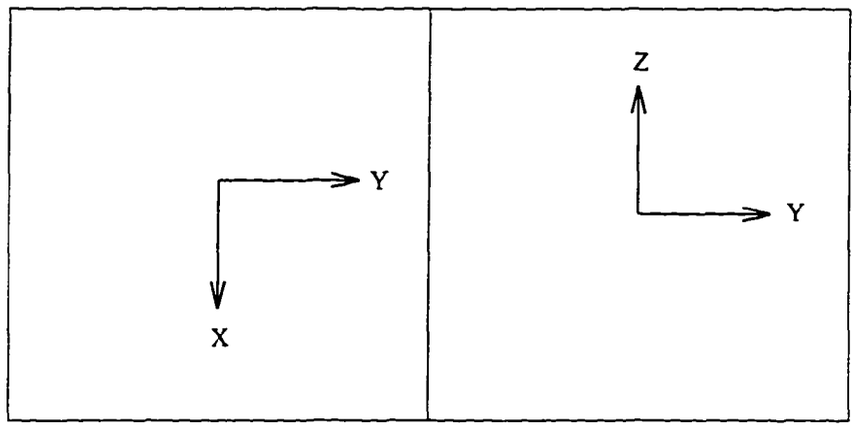


Figure 7.5 Two 2-D canvases used to define a cutting point.

In one canvas, the user can define the *x*, *y* values of the point by clicking the mouse in the *x*-*y* canvas. The user defines the *z* value and/or the *y* value of the point by clicking the mouse in the *y*-*z* canvas. Any *y* value defined or changed in one canvas would be reflected in the other one. A point in 3-D space is then determined when its (x, y, z) values are defined. The system uses a mouse with three buttons, and each of the buttons is used to define one of the three points respectively. Three different colors are used to distinguish the three points. Although using two canvases can uniquely define a point in 3-D space, this method doesn't reveal the spatial relationship among the three points and the relation of the defined cutting plane to the input volume. A 3-D canvas with a wire frame box representing the input volume is introduced as shown in Fig. 7.6. The three points chosen from the two 2-D canvases will be automatically transformed to the 3-D canvas. A triangle will be formed to represent the cutting plane and will be shown in the 3-D canvas.

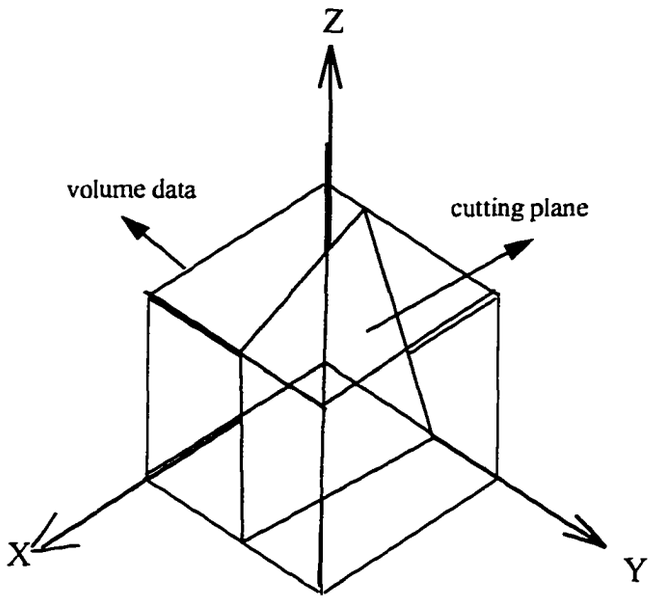


Figure 7.6 A 3-D canvas to define a cutting plane.

Through the 3-D canvas, the user can see the shape of the input volume, the defined points, and the cutting plane relative to the volume. Further, this 3-D canvas is not only a display tool for the two 2-D canvases, but can also be used as an alternative in defining the cutting plane. This second technique to define the cutting plane is based on defining three points on the edges of the volume as the three points on the cutting plane. The user can click along the volume edges to determine a cutting plane point. The point then will be transformed and displayed in the two 2-D canvases. Any changes in one of the three canvases will be reflected in the other two canvases. Thus the over-all cutting plane definition window has three canvases as illustrated in Fig. 7.7. An example of a cutting plane is also illustrated in Fig. 7.7. The Redraw button clears all the points and plane on the three canvases and allows the user to redefine the cutting plane.

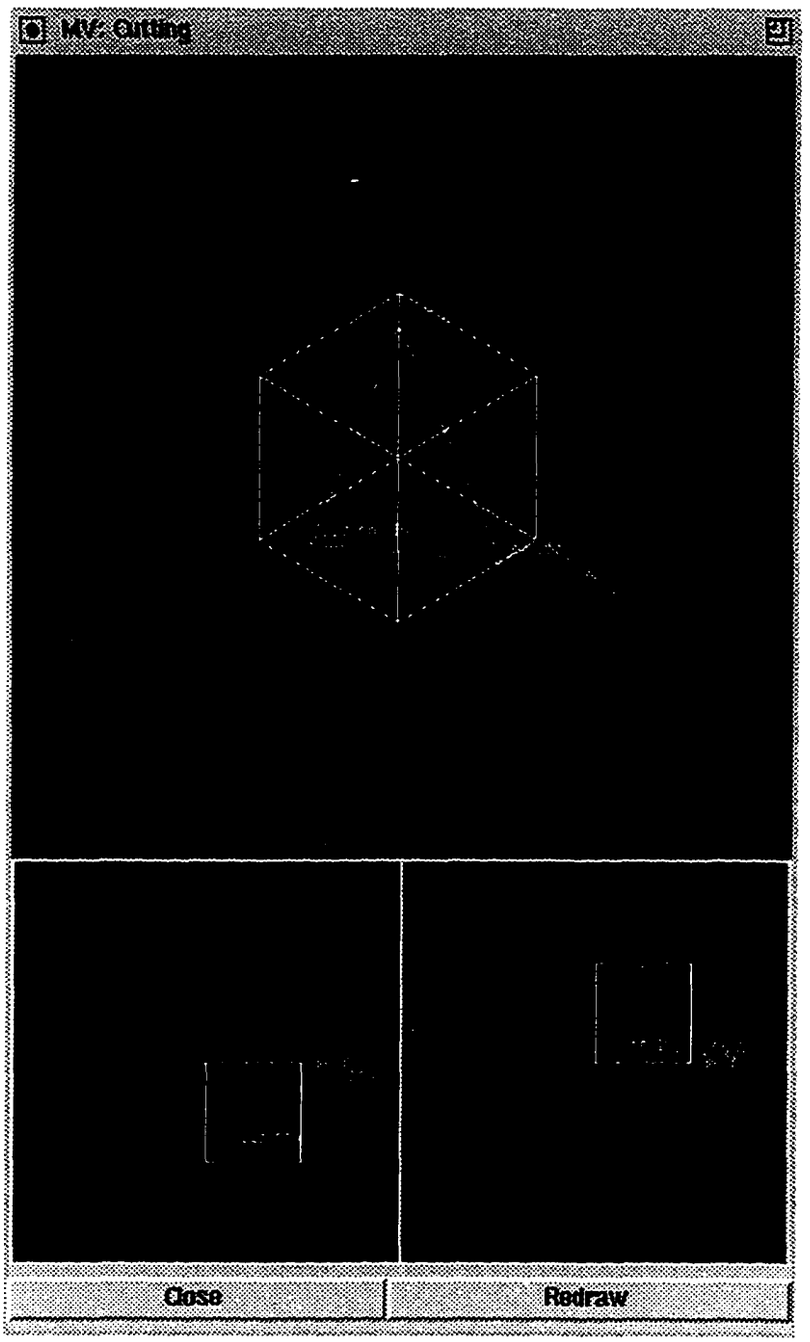


Figure 7.7 The cutting parameter window.

7.2.2 File manipulation functions

The file manipulation functions manage two kinds of files, input image files and special MicroVisual files which contain saved parameters and options. The *Select Images* command is used to handle image files, and, if selected, it brings up a window as shown in Fig. 7.8.

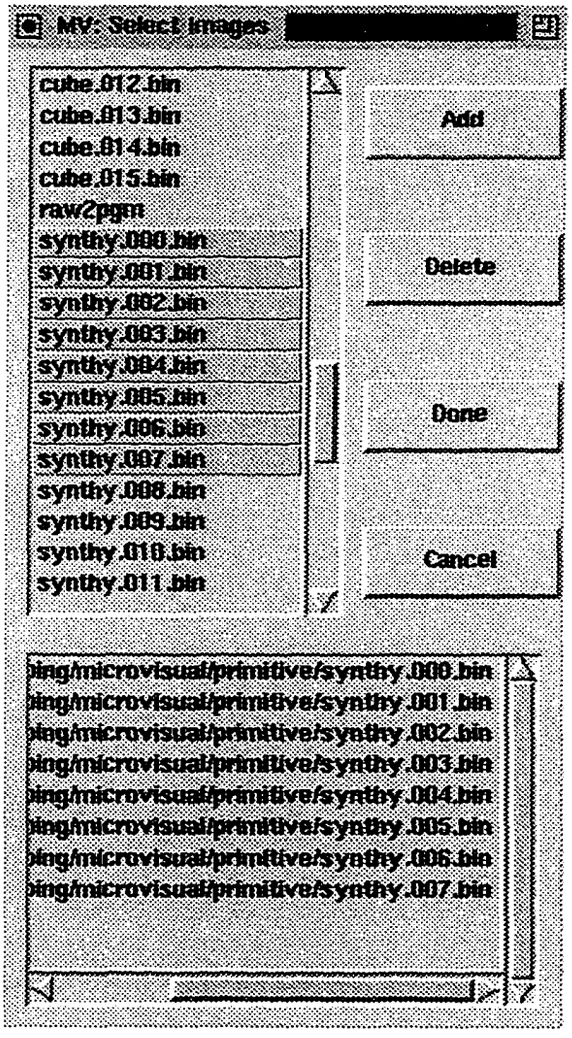


Figure 7.8 Image selection window.

The bottom box in the window is the list of the selected images which, stacked together, form the volume data. Initially it is an empty list and the user can add or delete images from the image list in three ways. One way is to highlight the desired images by clicking on the image file name, then clicking the *Add* or *Delete* button on the right side of the window. The second way is to double click the image file name to add it to the list. The third way is to drag the mouse through the image file names, and then click on the *Add* or *Delete*. The third method is especially useful when a large number of images have to be selected, and the images are listed consecutively. The user can also change directories by double clicking on a directory name. A *Cancel* button will allow the user to quit the select image process. After the images have been selected, a *Done* button will allow the user to exit the window with selected images. The system will check the size of the selected images with that listed in the parameter and option window. If there is a mismatch, an error message window will pop up to direct the user to correct the error.

The other kind of files the file manipulation functions handle is special MicroVisual files. These files are used when a user wishes to save the parameters in the parameter and option windows for further study. The *Save As* command allows the user to save all current parameters and options into a file specified in the bottom of the file selection window as in Fig. 7.9. When these parameters are requested again, clicking on the *Open* command as in Fig. 7.2 brings up the file selection window. The saved parameters file can then be opened again choosing the desired file and clicking on the *Open* button on the left. The image and shading parameter window, the light window and the cutting window pop up just as when choosing the *Create* command and all parameters are assigned the values saved previously in the file. The *Save* command saves the parameters to a default file .mvp if no previous *Open* function is chosen, or to the opened file name if an *Open* function has been invoked previously.

Only one set of parameter and option windows is allowed during the activation of MicroVisual. If the user tries to open or create a new set of parameter windows while

there is one already active, the system asks the user whether to close the old one and open a new one, or to cancel the command.

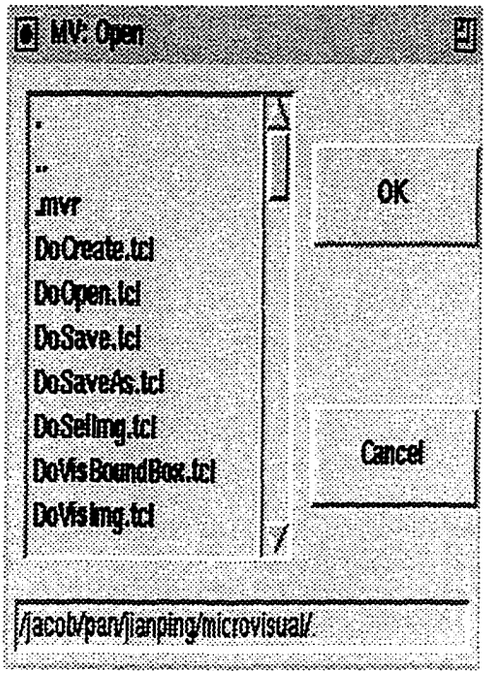


Figure 7.9 File manipulation window.

7.2.3 Visualization commands

Once all the parameters, options and images are specified and selected, the user can visualize the selected volume using the *Visual Images* command. If the user tries to visualize objects before selecting images, an error message window pops up to notify the user to select images first. The *Visual Images* function calls the surface generation routine to generate a polygon mesh and then calls the shading and rendering routine to render the polygon mesh, and finally invokes the *xv* program to display the reconstructed 3-D image.

A user often needs to change shading parameters to get a satisfactory result or multiple views of the objects need to be generated. In such cases, only rendering needs to be carried out again. The command *Rendering Only* is built for this purpose, omitting the surface generation, thereby speeding up the whole process. Whenever this command is chosen, a warning message window, as in Fig. 7.10, pops up to remind the user that this command should be used only if shading parameters have been changed.

Another visualization command is the *Visual Bounding Box* command as described in section 7.2.1.1. Since only twelve edges of the input volume (wireframe) are visualized, this command quickly gives the user a view of how the volume is related to the camera before the actual volume data is visualized. The user can change the camera position accordingly until a desirable position is reached.

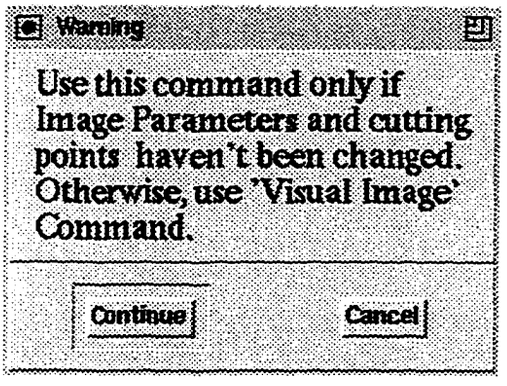


Figure 7.10 The warning message window for the Rendering Only Command.

When any of the above three visualization commands is chosen with the Cutting option *on*, then a window, as shown in Fig. 7.11, will pop up to guide the user to choose proper camera positions so that the user looks straightly at the cutting plane.

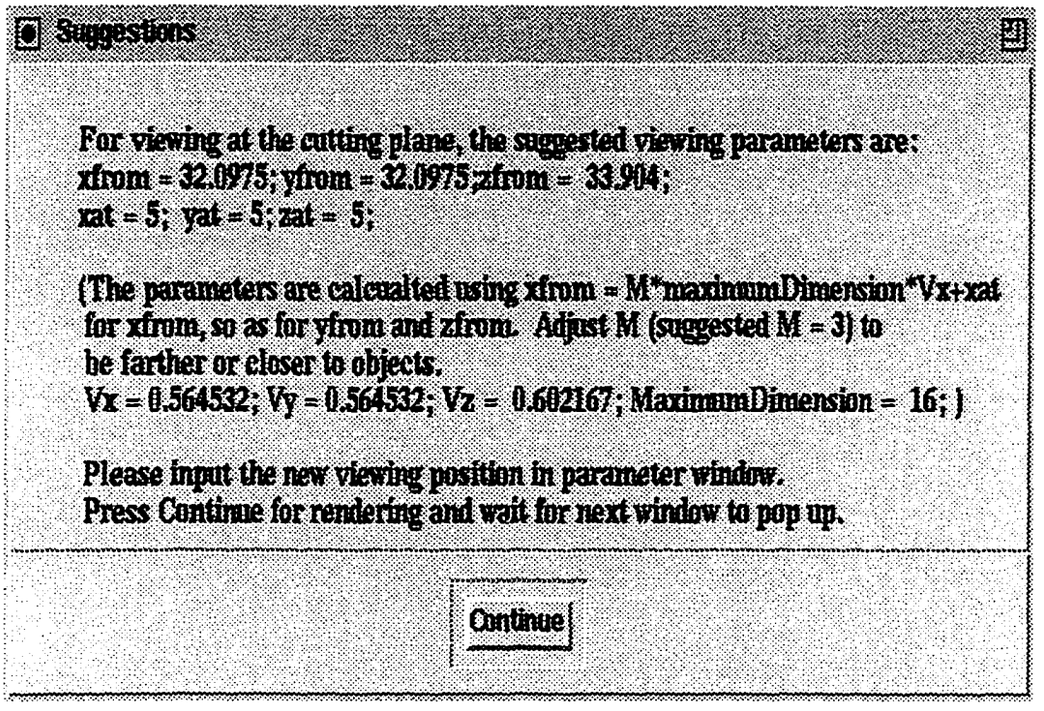


Figure 7.11 Suggestion window for shading parameters if cutting option is chosen.

7.3 The Implementation of the MicroVisual

A properly designed user interface is not be adequate if it lacks good implementation [25] [69]. The critical point in implementation is the choice of the right tool for software development. A software tool with the following features is the most desirable one to be used for the implementation: It should be easy to use on the computer system that MicroVisual is operated on; it should have good portability; it should be easy to maintain after the implementation.

Since the MicroVisual is to be run on a Sun workstation, the first option is to implement the package using *C* and the *Xlib* object library provided with the X-win-

dow systems[1]. Code developed with these tools is likely to be portable since it can be compiled with very little change for any Unix[2] machine with a *C* complier and the X library. However, it requires a significant effort since programming with *Xlib* is rather complicated, and so is the debugging during implementation and maintenance.

There is a commercial package called *DevGuide* [3], which can be used to implement MicroVisual. DevGuide provides a graphical interface for developing user-interface packages. Objects such as windows, menus, text widgets can be selected and manipulated from the icons in DevGuide, and a C code can be generated from the selected and manipulated icons. Implementing MicroVisual is certainly easy using this package, but it presents three problems. The most serious problem is that the code produced by DevGuide generates a user interface with the OpenLook[4] standard which will soon be replaced by the more popular OSF/MOTIF (same as footnote 3) standard. Also, the OpenLook standard is specific to Sun workstations. Thus if DevGuide is used to implement MicroVisual, its appearance is not up-to-date, and it is only useful on a Sun platform. Furthermore, the code produced by DevGuide is not very efficient. Since it is a commercial package, the distributions of MicroVisual would be also complicated.

A more popular tool for implementing the user interface is Tcl/Tk. Tcl stands for Tool Command Language and Tk stands for Toolkit. Tcl/Tk is a public domain software which allows MicroVisual, when implemented, to be distributed freely. This tool is also available on a variety of platforms (e.g. IBM [5], HP [6], SGI[7] and SUN), and hence

---

1. The X-window system is a trademark of Massachusetts Institute of Technology.
2. Unix is a registered trademark of AT&T.
3. DevGuide is a registered trademark of Sun Microsystems, Inc.
4. OSF/MOTIF is a trademark of the Open Software Foundation, Inc.
5. IBM is a registered trademark of International Business Machine Co.
6. HP is a registered trademark of Hewlett-Packard Co.
7. SGI is a registered trademark of Silicon Graphics, Inc.

any code produced with it is portable to those same platforms. Tcl is an interpretive script language like a Unix shell. It supports the use of variables and allows the user to define procedures with arguments and various strings. Thus, it is able to handle files and manipulate mathematical formulas. Each function in MicroVisual can be written in Tcl scripts. The Tcl toolkit, TK, contains modules for developing graphical user interfaces for the X-window systems. These modules can be used for setting up windows, and defining text widgets, scrollbars, menu hierarchies and other basic graphical user interface elements. Tcl/Tk allows a user to implement the MicroVisual more easily. The disadvantage of using Tcl/Tk is that it is an interpretive language, and thus it is not as efficient as writing the interface in a language such as C. But this disadvantage can be compensated by the fact that the Tcl scripts can be converted to C code using a special tool provided with the Tcl/Tk package.

The feature of easy implementation, portability and availability made Tcl/Tk the proper choice to implement MicroVisual. An early version of the MicroVisual was implemented as an undergraduate project [65]. This version is expanded and modified to the current version presented in this section.

7.4 Summary

A graphical user interface package, called MicroVisual is designed to integrate the surface generation and rendering process for the visualization system. The user-interface system hides operating details from users and allows them to operate the system quickly. Excluding the data acquisition, MicroVisual is a general visualization system which can be used to visualize different kinds of volumetric datasets, such as CT and MRI medical data. Meanwhile, the data acquisition process can be used as a tool for microscope image enhancement.

MicroVisual provides an on-line Help menu, undo features and default values for parameters to help the user to use the system easily. Three graphical canvases are also provided for the user to define a cutting plane. The cutting operation is an important feature of the system that can be used to reveal the internal structure of data volumes. With its portability, availability and easy for implementation, Tcl/Tk tool is chosen to implement MicroVisual.

Chapter 8

Results of 3-D Visualization

The system developed in this thesis can be used to visualize volume data in many application areas. Visualization of serial microscope images obtained by optical sectioning is one of the applications. Visualization of medical images obtained from CT or MRI or physical sectioning is another. Visualization of volume data obtained by a confocal microscope is the third application area. In all these applications, the means for data acquisition are different, but once a series of images is available, it can be visualized using the interactive visualization system developed in the thesis. Volume data modeling used in the visualization system is carried out by an isosurface algorithm, and a modified version of the marching-cube algorithm. The rendering process is carried out by using the polygon rendering C library SIPP. The user-interface, MicroVisual, facilitates the use of modeling and rendering. The 3-D visualization system offers functions such as rotation, zooming and cutting for users to examine the objects under study.

In this chapter, some representative examples obtained using the 3-D visualization system are presented. Simple objects such as cubes and spheres, whose volume data are created by voxelizing their mathematic formulae, are used to demonstrate the functionality of the interactive visualization system. Examples of 3-D visualization of simple objects are presented in Section 8.1. Volume data from medical imaging show that the interactive visualization system has the capability to generate realistic 3-D views from real data. Three-dimensional visualization results of volume data for medi-

cal applications, such as CT and MRI images are presented in Section 8.2. Results of 3-D visualization of serial microscopy images are presented in Section 8.3 and it is shown that these 3-D views can be used as a tool to visualize the 3-D relationships of microscopic structures. It will help users to gain an understanding of the objects of interest and allow users to examine the objects in new and novel ways.

8.1 Volume Data Generated From Mathematical Formulae

Volume data of simple objects, such as cubes and spheres, are generated by voxelizing their mathematical formula. These simple objects are used to test and to demonstrate the functionality of the visualization system. Two simple objects are generated, namely cubes and spheres. Combinations of cubes and spheres can yield more complicated synthetic objects.

The volume data of a cube is generated by voxelizing the following formula:

$$f(x,y,z) = \begin{cases} 1, & \begin{cases} x = x_a or x = x_b, y_a \le y \le y_b, z_a \le y \le z_b \\ y = y_a or y = y_b, x_a \le x \le x_b, z_a \le z \le z_b \\ z = z_a or z = z_b, x_a \le x \le x_b, y_a \le y \le y_b \end{cases} \\ 0, & otherwise \end{cases} \tag{8.1}$$

where x_a, y_a, z_a and x_b, y_b, z_b are the two boundaries, i.e., walls, of the cube on the three axes respectively. On the same boundary, one parameter minus the other such as (x_{a2}-x_{a1}) is the thickness of the wall.

The volume data of a sphere is generated by voxelizing the following formula:

$$f(x,y,z) = \begin{cases} 1, & r_a \le r \le r_b \\ 0, & otherwise \end{cases}$$

where $r = \sqrt{x^2 + y^2 + z^2}$ and r_a and r_b are the inner and outer radii of the sphere respectively. When r_a is equal to zero, the sphere is a solid.

In Fig. 8.1, the results of 3-D visualization of a cube, with a length of 14 pixels in a 16x16x16 volume dataset are presented. Each pixel in the volume data is 8 bits. The size of the reconstructed image is 256x256. In this chapter, all the volume data are 8 bits per pixel, and the reconstructed images are 256x256 unless otherwise indicated. The outside frame represents the size of the volume with the red line representing the x, the green for the y and the blue for the z axis. This frame of volume boundary and the colored three primary axes can gives the user an understanding of the orientation of the volume relative to the viewing position.

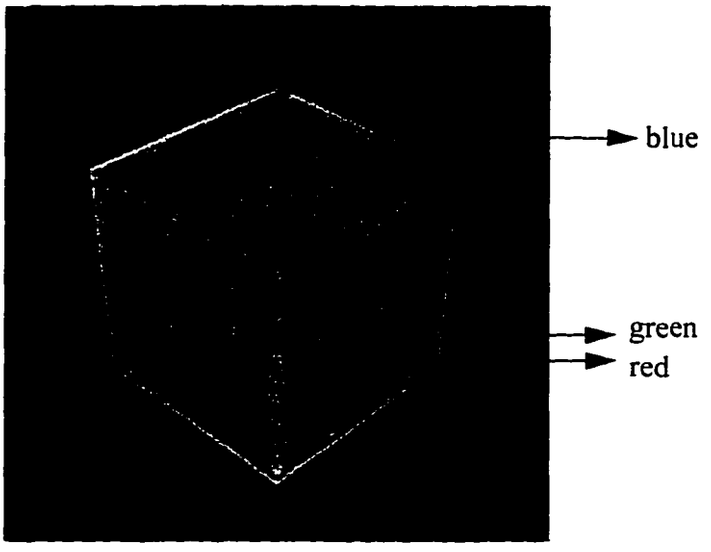


Figure 8.1 A result of 3-D visualization of a 16x16x16 volume data with a cube inside. The outside box represents the size of the volume, and the actual size of the cube is 14.

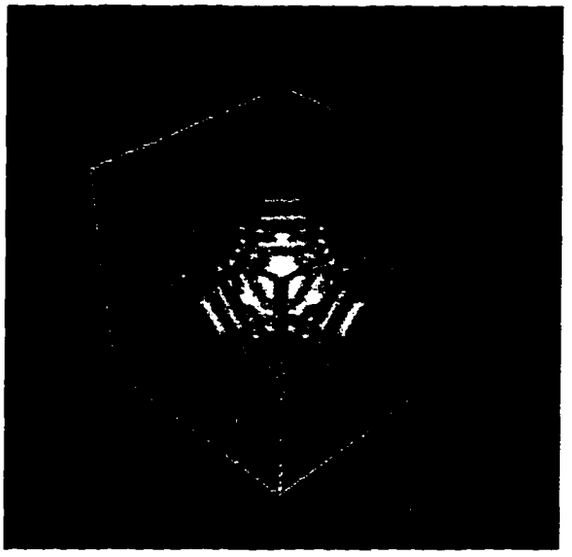


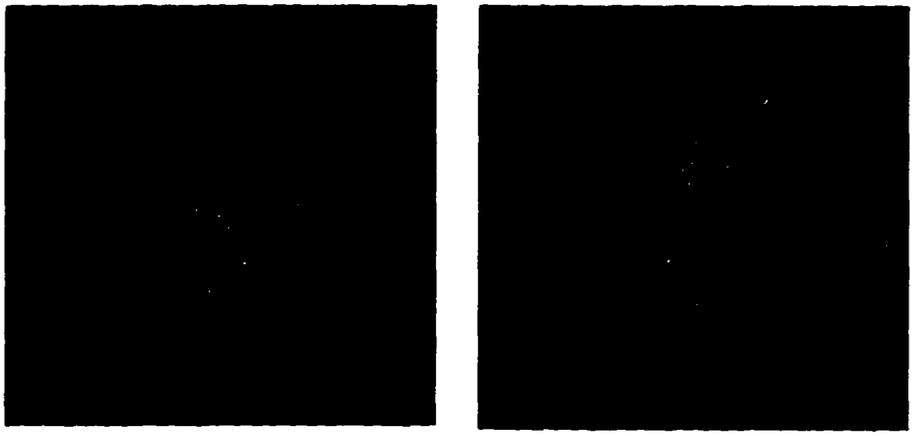
Figure 8.2 A result of 3-D visualization of a 33x33x33 volume with a sphere inside. The radius of the sphere is 14.

In Fig. 8.2, the result of 3-D visualization is presented for a 33x33x33 sphere of radius of 14. The structure shown on the surface of the reconstructed sphere is due to the low resolution in the volume data generated. The surface will become smoother as the resolution increases.

The results of 3-D visualization shown in Fig. 8.1 and Fig. 8.2 indicate that the system established can accurately reconstruct objects of interest. The system also provides functions such as rotation, zooming and cutting to allow users to gain more precise insights of the objects under study. These functions are demonstrated as below.

Rotation: The ability for a visualization system to change the viewing perspective and thus rotate objects gives users more precise information about the objects under study. In Fig. 8.3, two perspectives of 3-D visualization of a 16x16x16 volume are shown. One perspective with viewing direction along the (1,1,1) vector in the volume coordinate produces a perfect cube as shown in Fig. 8.3 (a). However, in the other

perspective with viewing position along the (-1, -1, -1) vector, one can see that the cube is not an exact cube and one wall is missing as shown in Fig. 8.3 (b).



(a) (b)

Figure 8.3 Two perspectives of a 16x16x16 volume data which consist of a cube with five walls. The second view reveals that the cube is not a perfect cube but has one wall missing.

Zooming: This feature allows a user to reconstruct a volume along a desired path and to zoom closely towards the feature the user is interested in, and to create an animation sequence. A 49x49x49 volume is generated to demonstrate the zooming function. The 49x49x49 volume consists of 13 equal sized spheres, symmetrically located about the center of the volume. In Fig. 8.4, the results of 3-D visualization of the volume is shown in sequence. These results, from left to right and top to bottom, represents a zooming effect along the (1, 1, 1) vector towards the center of the volume, aiming at the center sphere. The display program, called *xv*, is modified so that the program can load sequential images and play the sequence forward and backward to create animation using the sequence.

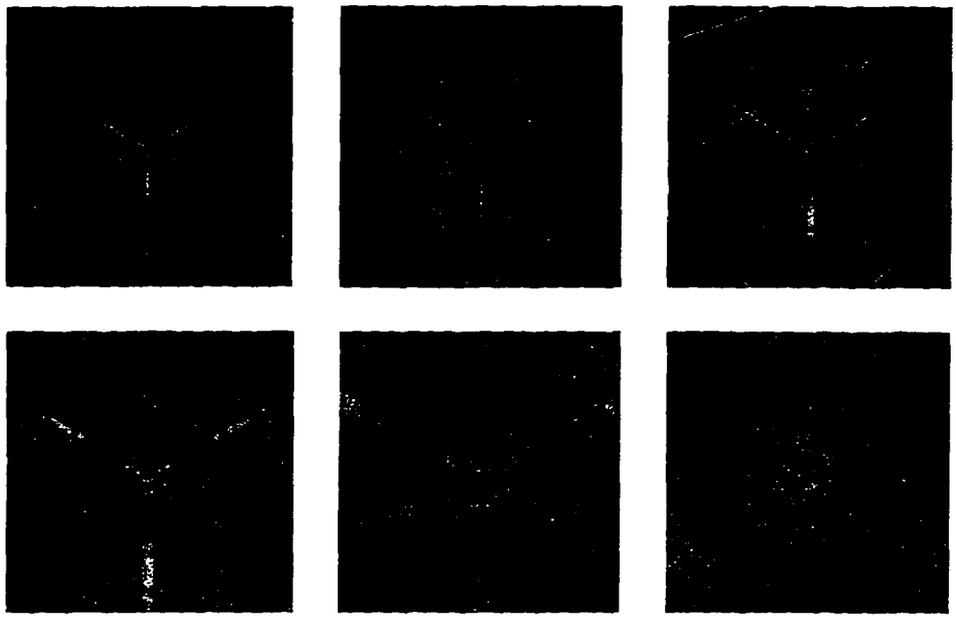


Figure 8.4 The zooming sequence, left to right and top to bottom, of a 49x49x49 volume. This particular example of zooming is aiming towards the center sphere.

Cutting: A very useful operation of the visualization system is the cutting operation. The cutting operation is an operation in which a user defines a cutting plane and cuts the volume into two parts. Interesting features present in an objects are often not guaranteed to be parallel to the physical cutting planes. Thus it is often difficult to interpolate these features. The visualization system developed allows users to define a virtual cutting plane and virtually cut the volume to present the features of interest. A cutting operation is also very useful to reveal internal structures obscured by other parts of the objects. In Fig. 8.6, a sequential examples of cutting operations are presented. The cutting planes are parallel to each other and to the z-axis, starting the far side towards the origin of the volume as shown in Fig. 8.5. Two cutting sequences of a 49x49x49 volume are generated and are shown in Fig. 8.6. The sequence in the left col-

umn represents the view inside the object on one side of the cutting plane, while the corresponding right column the view inside the objects on the opposite side of the cutting plane. The image sequences generated are loaded by the modified *xv* program to create the effect of moving through the objects under study.

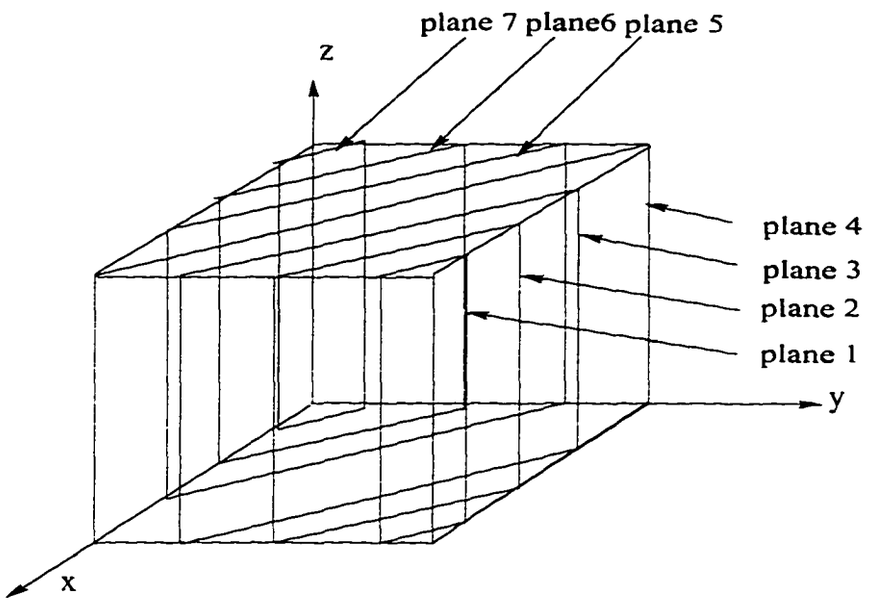
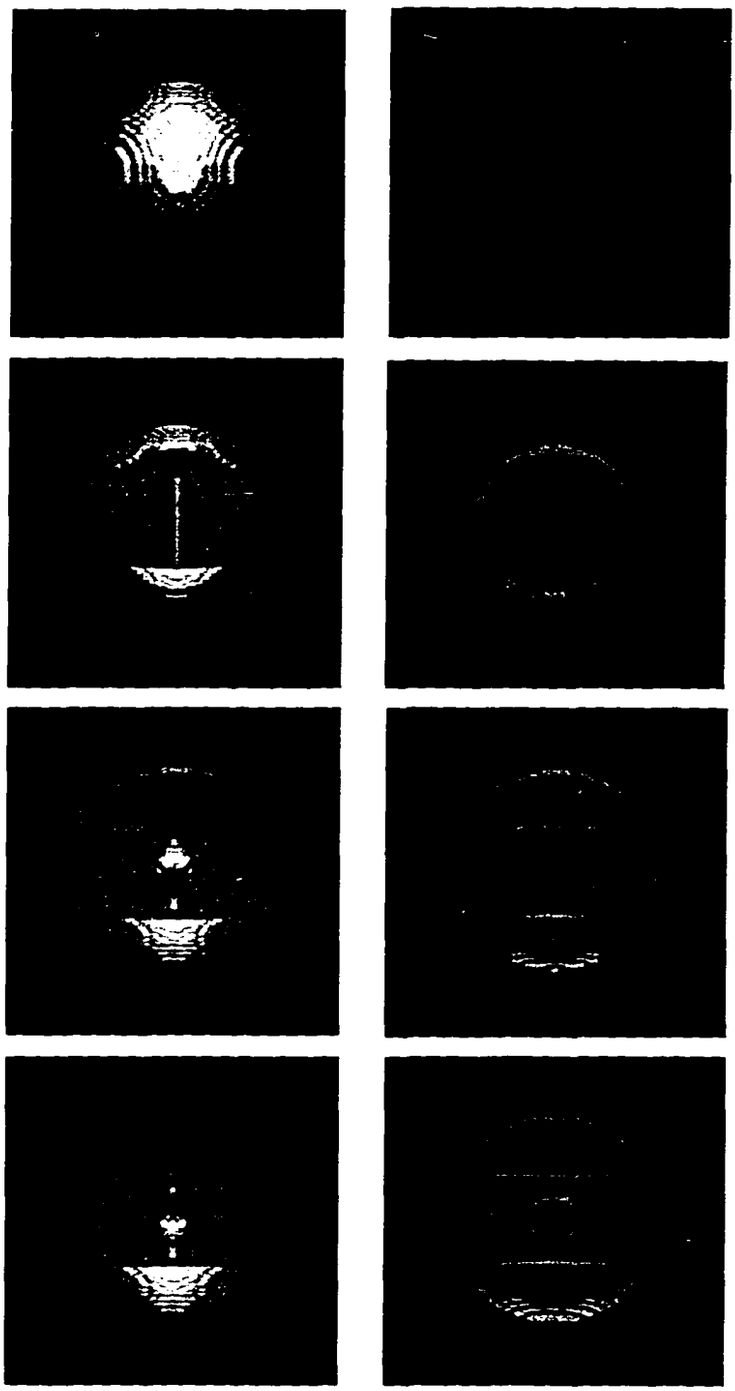


Figure 8.5 The illustration of the cutting planes used for the cutting sequence as shown in Fig. 8.6.



(continue on to the next page....)

(continued from the last page....)

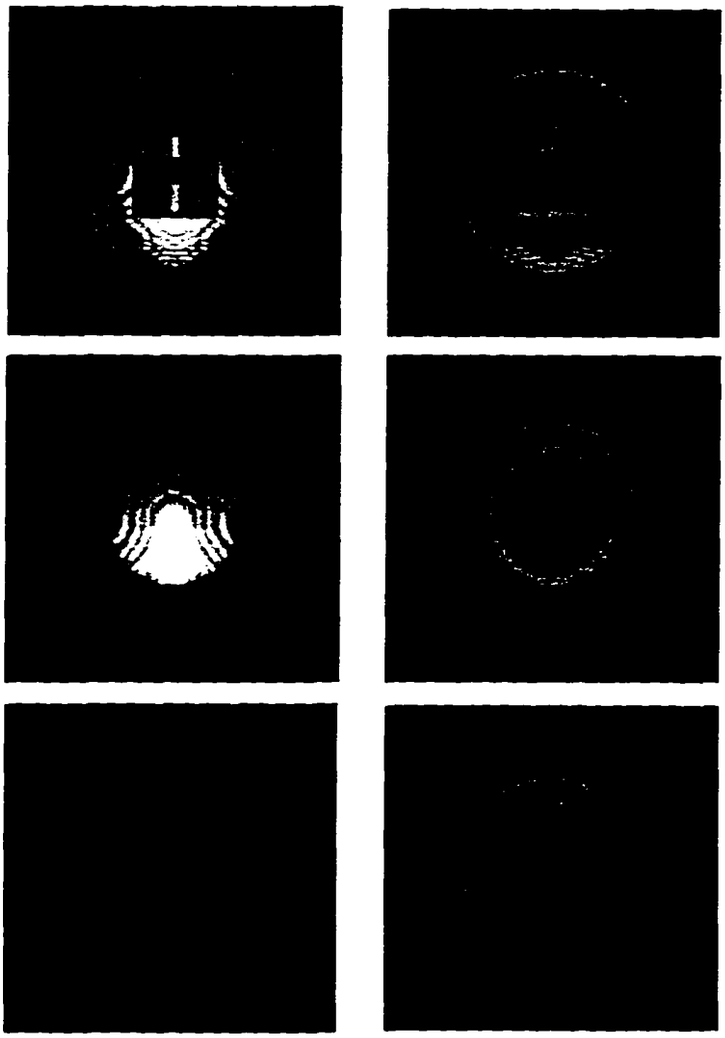


Figure 8.6 The cutting sequence of 49x49x49 volume data. The cutting planes are parallel to each other and to the z-plane. The left column are images from one side of the cutting plane and the right column are images from the opposite side of the cutting plane.

8.2 Medical Images

Three-dimensional visualization of volume data has been used extensively in medical imaging due to its promise as a tool for clinical diagnosis and as a tool for communications between physicians and patients. Visualization of volume data at the anatomy level can serve as an educational aid too. In this section, example results of 3-D visualization from medical data obtained from CT or MRI or physical sectioning are presented to demonstrate the ability of the interactive visualization system to generate realistic images for medical volume data.

8.2.1 Physically sliced images

The data used here are from the Visible Human Project, National Library of Medicine [1]. A frozen cadaver body was physically sliced and serial images of the body are obtained by imaging each individual slice. The volume data of the heart[2], whose size is 47x43x57, are obtained from segmentations of the serial images. Fig. 8.7 shows the result of 3-D visualization of the heart.

1. More information can be found under: http://www.nlm.nih.gov/.
2. Thanks to Mr. Kris Caputa at University of Victoria for providing this volume data.

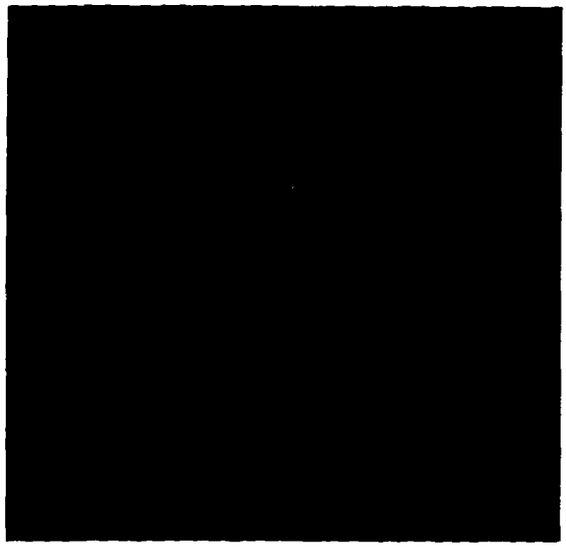


Figure 8.7 The reconstructed result of a human heart. The volume has the size of 43x47x53.

It should be noted that in this application, physical sectioning instead of optical sectioning has been used for data acquisition. The scale of objects involved in medical application, such as organs, is usually large enough so as to introduce marks before physical slicing is applied. The registration among sections can then be carried out after physical sectioning. The volume data of the heart presented here are registered with each other after physical sectioning. For microscope specimens, however, introducing marks before physical sectioning is almost impossible due to their micro scale. This illustrates the difficulty of data acquisition for microscope images and the necessity of introducing techniques of optical sectioning for data acquisition.

8.2.2 CT images

The CT volume data presented here is a cadaver head. The original CT volume data[1] were taken on the General Electric CT Scanner and provided courtesy of North

Carolina Memorial Hospital. In Fig. 8.8, eight images out of 113 in the volume are shown. The size of each image in the volume is 256x256. The distance between consecutive images is estimated as 1.9 mm. Each pixel in the original CT data is 16 bits. The whole volume size, thus, is 14,811,136 bytes, that is, almost 15 MB. The volume has been linearly converted to 8 bits per pixel which cuts down the size to half of the original size, which is still over 7MB. The massive data involved in the visualization indicates that increasing the efficiency of a visualization system is a very important issue. Two perspectives of the reconstructed head are presented in Fig. 8.9 and Fig. 8.10. The results of 3-D images show that there is noise around the mouth area, but this noise actually is due to the noise presented in the original images. The stripes behind the head are visible from the top of the original images as shown in Fig. 8.8.

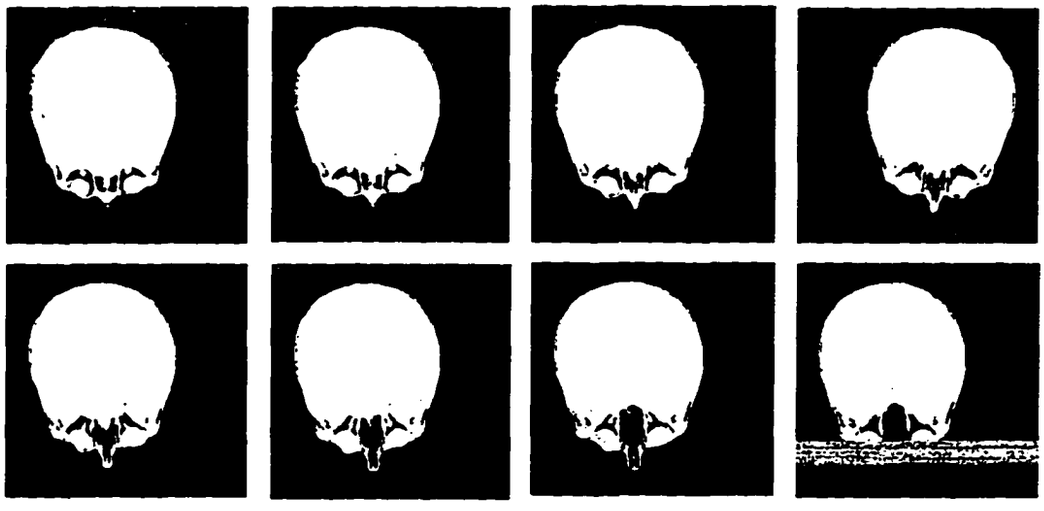


Figure 8.8 Part of CT images from a 256x256x113 cadaver volume.

1. The CT and MRI data are from the Chapel Hill Volume Rendering test dataset, provided by Software Systems Laboratory, University of North Carolina, USA.

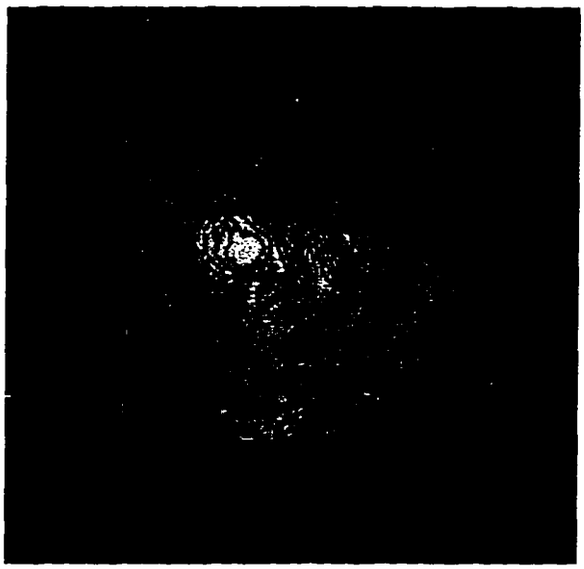


Figure 8.9 Results of 3-D visualization of the CT cadaver head.

Figure 8.10 Another view of the CT reconstructed cadaver head.

8.2.3 MRI images

The MRI volume data presented here is a human head with skull partially removed. The volume data were taken on the Siemens Magnetom and provided courtesy of Siemens Medical Systems, Inc., Iselin, NJ. Data were edited (skull removed) by Dr. Julian Rosenman, North Carolina Memorial Hospital. The volume data have been linearly converted from 16 bits per pixel to 8 bits per pixel. The distance between consecutive images is estimated as 1.6 mm. In Fig. 8.11, eight MRI images from the 256x256x109 volume are shown. Three results of 3-D visualization of the volume data are presented from Fig. 8.12 to Fig. 8.14. The results of 3-D visualization of the MRI images are smoother than those of CT images because of larger dynamic range in the original volume data. The noise around the mouth shown in Fig. 8.13 and Fig. 8.14 is actually from the breath of the patient, which can be identified clearly in Fig. 8.12.

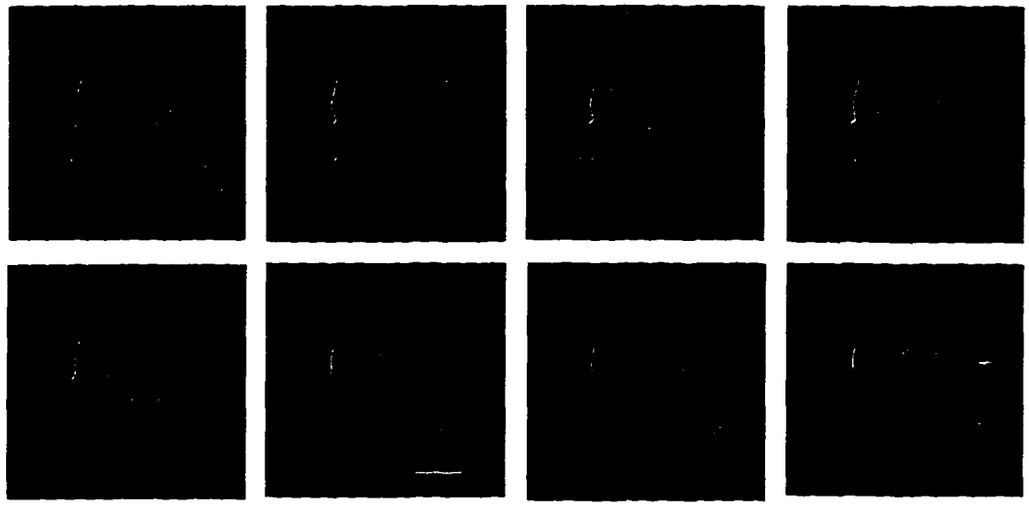


Figure 8.11 Part of the series of MRI volume images.

Figure 8.12 One reconstruction result of a MRI volume data.

Figure 8.13 Another perspective of the serial MRI image Visualization.

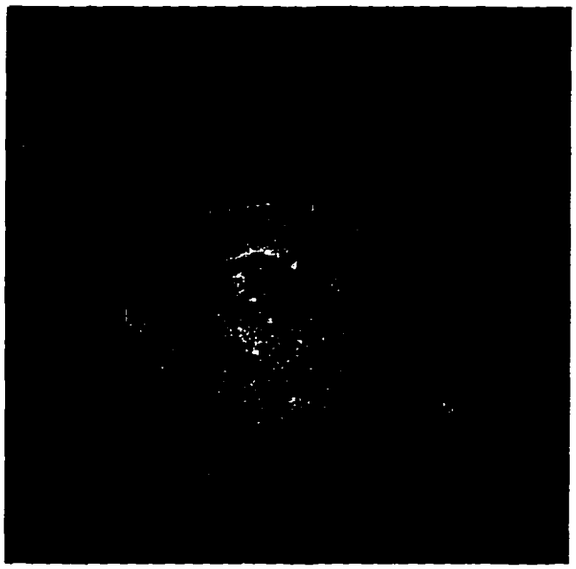


Figure 8.14 Another perspective of the serial MRI image visualization.

Three-dimensional visualization of real medical data using the interactive visualization system developed in this thesis show that this system can reconstruct realistic looking images of the objects of interest form medical volume data such as serial images from CT and MRI. Thus, the interactive visualization system presented is very general and it can be used for other volume data as well as for serial microscope images.

8.3 Serial Microscope Images

8.3.1 Confocal laser scan microscope images

A Confocal Laser Scan Microscope (CLSM) generates volume data of a specimen by collecting image information only on the focal plane of the microscope, and by

moving the focal plane through the specimen. Visualization of serial CLSM images is not the objective in the development of the visualization system. Rather, it is related to the visualization of serial light microscope images. However, visualization of CLSM images can, once again, demonstrate the generality of the interactive visualization system developed. However, it should be pointed out that a CLSM is very expensive equipment in contrast to optical sectioning which is a far more economic way to acquire volume data.

The volume data used are a small portion of a paper fiber[1] with the size of 75x45x93. Fig. 8.15 shows the partial images from the volume data.

Figure 8.15 Partial images in the fiber volume. The volume is 75x45x93. Each image in the figure is 10 images apart.

1. Thanks to Dr. Jim Proven of University of Victoria and Mr. Ran Guin of Pulp and Paper Research Institute of Canada at Vancouver for providing this fiber volume.

In Fig. 8.16 and Fig. 8.17, the two results of 3-D visualization of the paper fiber are presented. The fiber surface is set to be transparent. The bright pipe inside the fiber represents the hollow part inside the fiber. In the first perspective, the node on the top of the fiber can be clearly seen, and in the second perspective, the bent end can be also seen clearly.

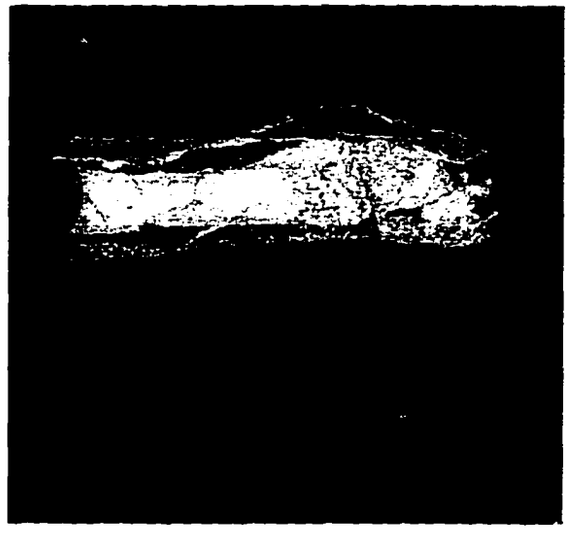


Figure 8.16 Reconstruction result of the fiber volume shown in Fig. 8.15.

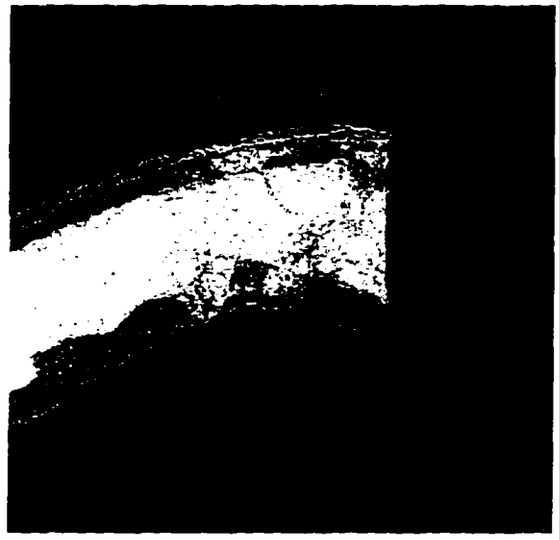


Figure 8.17 Another perspective of the reconstructed fiber volume.

8.3.2 Serial images from a transmitted light microscope

A pollen grain specimen is used as an example of visualizing serial microscope images. Serial microscope images of the pollen are obtained by moving and recording images of the specimen at constant interval (4μm) so that the focal plane of the microscope moves through the top to the bottom of the specimen, resulting in 21 sections. Each image is 298x276 with pixel distance 0.5μm, and it is shown in Fig. 3.7. Each image is processed with the proposed 3-D optical sectioning algorithm, the PMCI algorithm presented in Chapter 3, to remove out-of-focus information. The processed images from optical sectioning as presented in Section 3.4.2 are used as serial microscope images for visualization. In Fig. 8.18 and Fig. 8.19, two reconstruction results of the pollen volume data are shown. The two figures reveal the spatial relationships of the protrusions of the pollen which is difficult to mentally visualize from the serial images shown in Fig. 3.7. This confirms that a 3-D visualization system can help users gain further understanding of the objects under study.

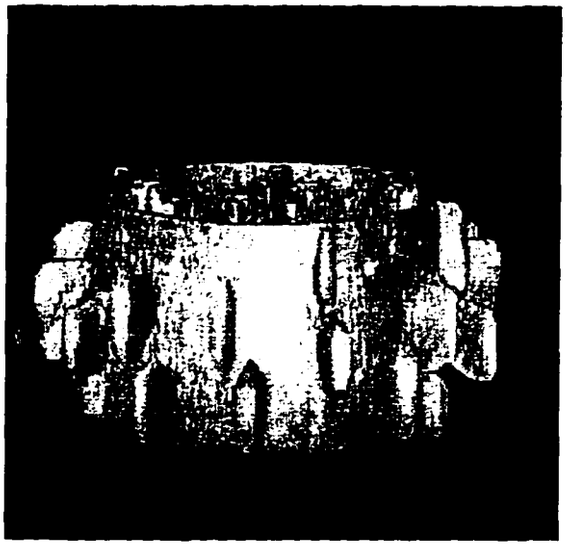


Figure 8.18 The reconstruction result of the pollen volume data acquired by optical sectioning with the PMCI algorithm.

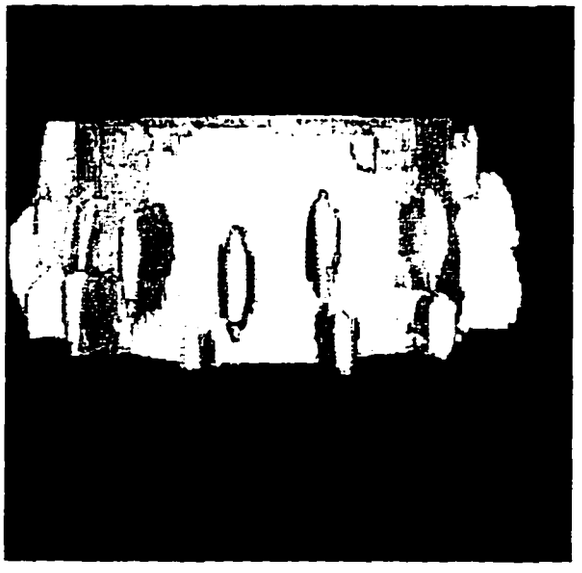


Figure 8.19 Another perspective of the pollen volume data.

8.4 Summary and Conclusion

The results of 3-D visualization of a variety of volume data using the interactive visualization system developed are shown in this chapter. These results indicate that the system is fairly accurate in its ability to produce realistic images of the 3-D objects of interest. The functions such as rotation, zooming and cutting of the system provide users a means to closely examine the objects of interest and thus enable the users to gain objective insights of 3-D information of the objects. The results of 3-D visualization of serial microscope images indicate that the optical section algorithm proposed and the modification of the modeling algorithms are effective and efficient. The fact that the interactive system can be used for various volume data, real or synthetic, indicates that the system can be used as a general visualization system.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis deals with 3-D visualization in microscopy, a fast evolving research application area. A 3-D visualization system for serial microscope images has been developed, extending the techniques of volume visualization which has been traditionally active in medical applications into microscopy and cell biology. The 3-D visualization system includes three main processes: data acquisition, volume data modeling and rendering.

The thesis makes contributions in volume data acquisition of collecting serial microscope images which has long been a big burden for 3-D visualization in microscopy. In this thesis, the concept of optical sectioning is used and an algorithm for it, the partial-minimization-and-constrained-iterative algorithm (PMCI) is proposed and tested for volume data acquisition. Optical sectioning turns out to be an efficient and effective technique for collecting and processing serial microscope images. It avoids many problems of physical sectioning such as registration, harming living cells, curling and compression of specimens. Optical sectioning has also advantages over using a CLSM in terms of cost and harming living cells. The proposed PMCI algorithm breaks the 3-D problem into 2-D problems to avoid computationally expensive 3-D convolutions and 3-D Fourier transforms. The new algorithm has been shown to give good results at a fraction of the computation cost of 3-D deconvolution. The PMCI also

offers flexibility to the user to control the computation complexity and the degree of approximation by choosing the number of planes above and below the focal plane involved in the minimization. In addition, the algorithm can serve as a general purpose image restoration algorithm to remove haze from defocus so as to enhance the quality of single microscope images.

This thesis has studied another important issue in data acquisition of collecting serial microscope images, i.e., the defocused imaging properties of transmitted light microscopes. In this thesis, these properties are studied by analyzing its defocus point spread functions and optical transfer functions. These functions are obtained by two approaches: direct measurements and theoretical calculations. The study reveals that a defocused transmitted light microscope tends to discriminate high frequency information and thus introduces haze to images. The maximum intensity of a defocused PSF drops rapidly with increasing defocus. For high numerical aperture objective lenses, the maximum intensity drops dramatically even for a small amount of defocus. An extensive comparison of the results from direct measurements and mathematical models are carried out in the thesis since this issue has not been adequately addressed in the literature. The purpose for the comparison is to verify whether these two approaches lead to similar results. The comparison reveals that for a low magnification and low numerical aperture objective lens of a microscope, the theoretical results match the experimental ones. However, for a high magnification and high numerical aperture lens, these two methods produce different results in terms of the diffraction patterns and sizes. The theoretical approach produces more diffraction rings and larger diffraction patterns, and thus introduces more haze to the final images. Possible causes of these disagreement for objective lenses of high numerical aperture have been discussed.

The modeling of serial microscope images has been considered and several existing volume data modeling algorithms have been studied. It is concluded that isosurface modeling is effective and efficient for modeling serial microscope images. This is in part due to the fact that cell walls possess only a small percentage of voxels in the vol-

ume data. The marching-cube algorithm is considered as a good choice because it offers not only high resolution but also allows the construction of cutting views to reveal internal structures of the volume. A *modified* marching-cube algorithm is proposed with two modifications of the original algorithm. One modification is to detect and prevent the redundancy problem. This is related to the fact that the grid points *on* the isosurface have to be treated differently from those *inside* the isosurface to prevent the generation of redundant points and lines which occur in the original marching-cube algorithm. The other modification is to use the middle-point algorithm to avoid linear interpolation to locate the vertex of a polygon, and thus to increase computational efficiency. The use of the middle point algorithm also allows co-planar triangles to be merged to polygons. Results show that the modified marching-cube algorithm proposed in the thesis can significantly reduce the number of polygons generated and thus increase the computational efficiency for surface generation and rendering. This modeling algorithm with improved efficiency will be a further step in the direction of real-time implementation of surface generation algorithms.

An interactive visualization system, MicroVisual written in Tcl/tk, is developed to facilitate the modeling and rendering processes. The interactive visualization system together with the data acquisition algorithm proposed in the thesis form a 3-D visualization system for serial microscope images. The 3-D visualization system offers functions such as rotation, zooming and cutting for users to examine objects from various views and visualize their internal structures. The results of 3-D visualization presented in Chapter 8 show that the 3-D visualization system has realistically reconstructed objects of interests from serial microscope images as well as from various volume datasets such as CT and MRI medical images. Thus, the 3-D visualization system developed is efficient and effective in many application areas.

9.2 Future Work

The work presented in this thesis indicates that several extensions can be made. They are:

- investigate the cause of the disagreement between direct measurement and mathematical modeling of the defocused PSF and OTF for high numerical aperture lenses. The result of such an investigation will enable users to choose either approach to obtain defocused PSF and OTF for optical sectioning with high numerical aperture lenses.
- apply sophisticated constrained-iterative minimization algorithms, such as regularized Tikhonov-Miller filters, to the existing PMCI algorithm for the minimization of the errors between the observed images and the estimated images so that the PMCI algorithm can produce even better optical sectioning results.
- develop algorithms to further reduce the number of polygons generated and thus to further increase the efficiency of modeling of a 3-D visualization system. The algorithms for polygon reduction include specific algorithms to reduce the number of polygons that the marching-cube algorithm generates, and general algorithms to deduct the number of polygons on polygon meshes as used in many computer graphics applications. A significant efficiency improvement will lead to real time visualization and animation.
- enhance functions of the 3-D visualization system. A measuring function can be added so that the system can give numerical information of objects of interest, such as the size and volume of a cell. Algorithms for multiple thresholds for the marching-cube algorithm can be developed so that multiple surfaces of different objects can be visualized. The algorithms for multiple thresholds will yield similar visualization results to those from volume rendering modeling where multiple objects are visualized.

Bibliography

[1] Aferzon, J., Chau, R. I. and Cowan, D. F., A Microcomputer-based System for Three-Dimensional Reconstructions from Tomographic or Histologic Sections, *Analytical and Quantitative Cytology and Histology*, Vol. 13, No. 2, 1991.

[2] Agard, D. A., Optical Sectioning Microscopy: Cellular Architecture in Three Dimensions, *Annual Review of Biophysics*, Boeing, Vol. 13, pp. 191-219, 1984.

[3] Agard, D. A., Hiraoka, Y., Shaw, P. and Sedat, J. W., Fluorescence Microscopy in Three Dimensions, *Methods in Cell Biology*, Vol. 10, pp. 353-377, 1989.

[4] Agard, D. A., Steinberg, R. A., and Stroud, R. M., Quantitative Analysis Of Electrophoretograms: A Mathematical Approach To Superresolution, *Analytical Biochemistry*, Vol. 111, pp. 257-268, 1981.

[5] Andrews, H. C., and Hunt, B. R., *Digital Image Restoration*, Prentice-Hall, 1977.

[6] Appel, A., The Notion of Quantitative Invisibility and the Machine Rendering of Solids, in *Proceedings of the Association for Computing Machinery National Conference*, pp. 387-393, 1967.

[7] Artzy, E., Frieder, G. and Herman, G. T., The Theory, Design, Implementation and Evaluation of a Three-dimensional Surface Detection Algorithm, *Computer Graphics and Image Processing*, Vol.15, pp.1-24, 1981.

[8] Bates, R. H. T., and McDonnell, M. J., *Image Restoration and Reconstruction*, Oxford Science Publications, 1989.

[9] Born, M., and Wolf, *Principle of Optics*, Pergamon Press, Oxford, 1964.

[10] Briarty, L. G. and Jenkins, P. H., GRIDSS: An Integrated Suite of Microcomputer Programs For Three-dimensional Graphical Reconstruction From Serial Sections, *Journal of Microscopy*, Vol. 134, 1984.

[11] Carrington, W. and Fograty, K., 3D Molecular Distributions of Living Cells by Deconvolution of Optical Sections Using Light Microscopy. *Proceedings of the 13th Annual Northwest Bioengeering Conference* (ed. by K. Foster), pp. 108 - 111. 1987.

[12] Carrington, W. A., Lynch, R. M., Moore, E. D. W., Isenberg, G., Fogarty, K. E. and Fay, F. S., Superresolution Three-Dimensional Images of Fluorescence in Cells With Minimal Light Exposure, *Science*, Vol. 268, pp. 1483 - 1487, 1995.

[13] Castleman, K. R., *Digital Image Processing,* Prentice Hall, Englewood Cliffs, New Jersy, 1979.

[14] Catmull, E., A Subdivision Algorithm for Computer Display of Curved Surfaces, *Ph. D thesis*, University of Utah, 1974.

[15] Chen, L. S., Herman, G.T., Reynolds, R. A.and Upuda, J. K., Surface Shading in the Cuberille Environment, *IEEE Computer Graphics and Applications.* Vol.5, 1985.

[16] Christianser, H. and Sederberg, H. T., Conversion of Complex Contour Line Definition into Polygonal Element Mosaics, *Computer Graphics.* Vol.12, No.3, pp.187-192, 1987.

[17] Clinch, N. F., Daly, C. J., Gordon, J. F., Moss, V. A. and Spurway, N. C., Wide-field Volume Visualization of Thick Microscope Sections by Computed Nearest Neighbor Deconvolutions, *Journal of Physiology.* Vol. 452, p.3, 1992.

[18] Cline, H. E., Lorensen, W. E. and Ludke, S., Two Algorithms for the Three-Dimensional Reconstruction of Tomograms, *Medical Physics.* Vol.15, No.3, pp. 320-327, 1988.

[19] Cook, L., Dwyer, S., Batnitzky, S. and Lee, k., A Three Dimension Display System for Diagnostic Imaging Applications, *IEEE Computer Graphics and Applications*, Vol.3, No.5, pp.13-19, 1983.

[20] Drebin, R. A., Carpenter, L. and Hanrahan, P., Volume Rendering, *Computer Graphics.* Vol.22, No.4, 1988.

[21] Durst, M. J., Letters: Additional Reference to 'Marching Cubes', *Computer Graphics*, Vol. 22, No.2., 1988.

[22] Erhardt, A., Zinser, G., Komitowski, D. and Nille., J., Reconstructing 3D Light-Microscopic Images by Digital Image Processing, *Applied Optics*, Vol. 24, No. 2, pp. 194 - 200, 1985.

[23] Foley, J., Wallace, V., and Chan, P., The Human Factors of Computer Graphics Interactive Techniques, *Computer Graphics and Applications*, Vol. 4, No. 11, pp. 13-48, 1984.

[24] Foley, J., Kim, W., Kovacevic, S., and Murray, K., Defining Interfaces at a High Level of Abstraction, IEEE software, Vol. 6 No. 1, pp. 25-32, 1989.

[25] Foley, J. D., van Dam, A. V., Feiner, S. K. and Hughes, J. F., *Computer Graphics: principles and practice.* Second edition, Addison-Wesley Publishing Company, 1990.

[26] Frieder, G., Gordon, D. and Reynolds, R. A., Back-to-Front Display of Voxel-Based Object, *Computer Graphics and Applications.* Vol.5, No.1, pp. 52-59, 1985.

[27] Fuchs, H., Kedem, Z. M. and Uselton, S. P., Optimal Surface Reconstruction from Planar Contours, *The Communications of Association for Computing Machinery*, Vol. 20, pp.693-702, 1977.

[28] Fuchs, H., Levoy, M., and Pizer, S. M., Interactive Visualization of 3-D Medical Data, *Computer.* Vol.22, No.8, 1989.

[29] Gargantini, I, Walsh, T. R., Wu, O. L., Viewing Transformations of Voxel-based Objects via Linear Octrees, *Computer Graphics and Applications.* Vol.6, 1986.

[30] Goldwasser, S. M., Reynolds, R. A., Talton, D. A. and Walsld, E. S., Techniques for Rapid Display and Manipulation of 3-D Biomedical Data, *Computer Medical Images and Graphics.* Vol.12, No.1, 1988.

[31] Gonzalez, R. and Wintz., P., *Digital Image Processing*, second edition, Addison Wesley publisher, 1987.

[32] Goodman, J. W., *Introduction to Fourier Optics*, McGraw-Hill Book Company, New York, 1968.

[33] Gouraud, H., Continuous Shading of Curved Surfaces, *IEEE Transactions on Computers*, Vol. 20, No. 6, pp. 623-629, 1971.

[34] Hartley, W. G., *Hartley's Microscopy*, Senecio Publishing Company, 1979.

[35] Helms, H. D. and Rabiner, L. R. eds, *Literature in Digital Signal Processing*, IEEE Press, New York, 1973.

[36] Herman, G. T. and Liu, H. K., Three-Dimensional Display of Human Organs from Computer Tomograms, *Computer Graphics and Image processing.* Vol.9, No.1, pp. 1-21, 1979.

[37] Herman, G. T. and Udupa, J. K., Display of Three Dimensional Discrete Surfaces, *Proceedings of SPIE.* Vol. 283, 1981.

[38] Herman, G. T., Reynolds, R. A. and Udupa, J. K., Computer Techniques for the Representation of Three Dimensional Data on a Two-Dimensional Display, *Proceedings of SPIE*, Vol. 367, pp.3-14, 1982.

[39] Hesselink, L., Post, F. H. and vanWijk, J. J., Research Issues in Vector and Tensor Field Visualization, IIII *Computer Graphics and Applications*, Vol. 14, 1994.

[40] Hiraoka, H., Sedat, J. W. and Agard, D.A., Determination of Three-dimensional Imaging Properties of a Light Microscope System: Partial Confocal Behavior in Epifluoresence Microscopy, *Biophysical Journal*, Vol. 57, 1990.

[41] Holmes, T., Image Restoration for 3D Fluorescent Microscopy, *New Dimensions of Visualization in Biomedical Microscopes - 3-D Imaging and Computer Applications*, VCH Verlagsgesellschaft Weinheim, pp. 283 - 327, 1992.

[42] Holmes, T. and Liu, Y., Richardson-Lucky Maximum Likelihood Image Restoration Algorithm for Fluorescence Microscopy Tissue Testing. *Applied Optics*, Vol. 28, pp. 4930 - 4938. 1989.

[43] Hopkins, H. H., The Frequency Response of a Defocused Optical System. Proceeding of Royal Society of London, A231, pp. 91-103, 1955.

[44] Johnson, B. D. and Malafant, K. W. J., 1995 - Visualization Techniques for Geoscience Data and Modelling, *Proceedings of the Third National Conference on the Management of Geoscience Information and Data*, organized by the Australian Mineral Foundation, Adelaide, Australia, pp. 18.1-4, 1995.

[45] Kajiya, J. T., Van Herzen,B.P., Ray Tracing Volume Densities, *Computer Graphics.* Vol.18, No.3, pp.165-174, 1984.

[46] Kepple, E., Approximating Complex Surfaces by Triangulation of Contour Lines, *IBM Journal of Research Development.* Vol.19, No.1, pp. 2-11, 1975.

[47] Lagendijk, R. L. and Biemond, J., *Iterative Identification and Restoration of Images*, Kluwer Academic Publishers, 1991.

[48] Lehmann, W. and Wachtel, A., Numerical Aperture of Light Microscope Objectives, *Journal of Microscopy*, Vol. 169, 1993.

[49] Levoy, M., Display of Surfaces From Volume Data, *IEEE Computer Graphics and Applications.* Vol.8, No.3, 1988.

[50] Levoy, M., Efficient Ray Tracing of Volume Data, *Association for Computing Machinery Transactions on Graphics.* Vol.9, No.3, 1990.

[51] Li, J. and Agathoklis, P., A Case Study of Isosurface Generation In 3D Visualization, *IEEE PacRim Conference on Communications. Computers and Signal Processing Proceeding*, Vol.2, pp. 622-625, 1993.

[52] Lorensen, W. and Cline,H., Marching Cubes: A High Resolution 3-D Surface Construction Algorithms, *Computer Graphics.* Vol.21, No.4, 1987.

[53] Masters, B. R. and Paddock, S. W., Three-dimensional Reconstruction of the Rabbit Cornea by Confocal Scanning Optical Microscopy and Volume Rendering, *Applied Optics*, Vol. 29, pp. 3816-3822, 1990.

[54] Max, N., SIGGRAPH' 84 Call for Ominimax Films, Computer Graphics, Vol 16, No. 4, pp. 208-214, 1982.

[55] Mayhew, D., *Principles and guidelines in User Interface Design*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[56] Meagher, D. J., Geometric Modeling Using Octree Encoding, *Computer Graphics and Image processing.* Vol.19, 1982.

[57] Muller, H. and Stark, M., Adaptive Generation of Surfaces in Volume Data, *The Visual Computer*, Vol.8, No.9, pp. 182-199, 1993.

[58] Müller-Plathe, F. Laaksonen, L. and vanGunsteren, W. F.,Cooperative Effects in the Transport of Small Molecules Through an Amorphous Polymer Matrix. *Journal of Molecular Graphics*, Vol. 11, 1993.

[59] Mundo-Ocampo, M., Greene, M., Flaxman, M. and Baldwin J.G., Morphology of Nurse Cell Nuclei Induced by Meloidodera Floridensis: A Computer Graphics Application, *Journal of Nematology*, Vol. 25, No. 5, 1993.

[60] Nelson, M., Smooth Appearance for Polygonal Surfaces, *The Virtual Computer.* Vol.5, No.3, pp160-173, 1989.

[61] Nielson, G.M. and Hamann B., The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes, *Proceedings of Conference on Visualization*, pp. 83-91, 1991.

[62] Newell, M. E., Newell, R. G. and Sancha, T.L., A Solution to the Hidden Surface Problem, in *Proceedings of the Association for Computing Machinery National Conference*, pp. 443-450, 1972.

[63] Ney, D. R., Fishman, E. K., Magid, D. and Drebin,R.A., Volumetric Rendering of Computed Tomography Data: Principles and Techniques, *Computer Graphics and Applications.* Vol.10, No.2, 1990.

[64] Noll, M., A Computer Techniques for Displaying N-dimensional Hyperobjects, *The Communications of Association for Computing Machinery Computing*, Vol. 10, No.8, pp.469-473, 1967.

[65] Nouredd, B., Undergraduate project, ELEC 499, Dept. of Electrical and Computer Engineering, University of Victoria, 1994.

[66] Prearate, F. P. and Shamos, M.I., *Computational Geometry*, Springer-Verlag, 1985.

[67] Oppenheim, A. W. V. and Schafer, R. W., *Digital Signal Processing*, Prentice-Hall Inc., 1975.

[68] Optikos Corp., How to measure MTF and Other Properties of Lenses. Manual.

[69] Ousterhout, J. K., *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, 1993.

[70] Phong, B., Illumination for Computer Generated Pictures, *The communications of Association for Computing Machinery*, Vol. 18, Nov. 6, pp. 311-317, 1975.

[71] Reilly, P., Data visualization in Archaeology, *IBM Systems Journal*, Vol. 28, pp. 569-579, 1989.

[72] Reisner, P., Further Developments Towards Using Formal Grammar as a Design Tool, in Proceedings of the Human Factors in Computer Systems Conference, pp. 304-308, 1982.

[73] Reynolds, R. A., Gordon, D. and Chen, L., A Dynamic Screen Technique for Shaded Graphics Display of Slice-Represented Objects, *Computer Vision. Graphics and Image Processing.* No.38, pp.275-298, 1987.

[74] Rhyne, T. and Petterson, L., The US Environmental Protection Agency Scientific Visualization Center, *Computer Graphics*, Vol. 26, p.178, 1992.

[75] Roberts, L. G., Machine perception of Three Dimensional Solids, *Lincoln Laboratory, TR 315, MIT*, Cambridge, 1963.

[76] Sabella, P., A Rendering Algorithm for Visualizing 3D Scalar Fields, *Computer Graphics*. Vol.22, No.4, pp. 51-58, 1988.

[77] Samet, H., *Design and Analysis of Spatial Data Structures*. Addison-Wesley Reading, MA, 1990

[78] Schroeder, W. J., Zarge, J. and Lorensen, W. E., Decimation of Triangle Meshes, *Computer Graphics*, Vol. 26, No.2, 1992.

[79] Shantz, M. J., Description and Classification of Neuronal Structure in the Frog Retina, *Ph.D. Thesis*, California Institute of Technology, Pasadena, California, 1976.

[80] Shaw, P. J. and Rawlins, D. J., The Point Spread Function of a Confocal Microscope: its Measurement and use in Deconvolution of 3D Data, *Journal of Microscopy*. Vol. 163, pp.151-165, 1991.

[81] Stokseth, P. A., Properties of a Defocused Optical System, *Journal of Optical Society of American*, Vol. 59, pp.1314-1321, 1969.

[82] Stricker, S. A., Paddock, S. W., and Shatten, G., Laser Scanning Confocal Microscopy of Living Sea Urchin Embryos: 3D reconstruction and Calcium Ion Imaging, *Journal of Cell Biology*, Vol. 111:113a., 1990.

[83] Stytz, M. R., Frieder, G. and Frieder, O., Three-dimensional Medical Imaging: Algorithms and Computer Systems, *Association for Computing Machinery Computing Surveys*, Vol. 23, No. 4, pp. 421-499, 1991.

[84] Sutherland, I.E. and Hodgman,G., Reentrant Polygon Clipping, *Communications of Association for Computing Machinery Computing*, Vol. 17, No.1, pp.32-42, 1974.

[85] Tiede, U., Hoehne, K., Momans, M., Pommert, A., Riemer, M. and Wiebeke,G., Investigation of Medical 3-D Rendering Algorithms, *IEEE Computer Graphics and Applications*, Vol.10, No.2, 1990

[86] Toennies, K. D., Surface Triangulation by Linear Interpolation in Intersecting Planes. *Proceedings of SPIE*, Vol.1137, pp. 98-105, 1989.

[87] Upson, C. and Keeler, M., The V-buffer: Visible Volume Rendering, *Computer Graphics*, Vol.22, No.4, 1988, pp59-64.

[88] Udupa, J. K., Interactive Segmentation and Boundary Surface Formation for 3D Digital Images, *Computer Graphics and Image processing.* Vol.18, pp. 213-235, 1982.

[89] VanDerVoort, H. T. M. and Strasters, K. C., Restoration of Confocal images for Quantitative Image Analysis, *Journal of Microscopy*, Vol. 178, pp. 165 - 181, 1995.

[90] Van Aken, J. R., and Novak, M., Curved-Drawing Algorithms for Raster Displays, *Association for Computing Machinery Transactions on Graphics.* Vol. 4, No. 2, pp. 147-169, 1985.

[91] Weiler, K. and Atherton, P., Hidden Surface Removal Using Polygon Area Sorting, *Proceeding of Special Interest Group on Graphics 77*, pp. 214-222, 1977.

[92] Wilhelms, J. and Gelder, A. V., Octrees for Faster Isosurface Generation Extended Abstract, *Computer Graphics*, Vol.24, No.5, pp. 57-62, 1990.

[93] Wilhelms, J. and Gelder, A. V., Topological Considerations in Isosurface Generation Extended Abstract, *Computer Graphics.* Vol.24, No.5, pp. 79-86, 1990.

[94] Willis, R. B., Turner, J.N. and Holmes, T.J. Iterative, Constrained 3-D Image Reconstruction of Transmitted Light Bright-field Micrographs Based on Maximum likelihood Estimation, *Journal of Microscopy*, Vol. 169, pp. 347 - 361, 1993.

[95] Wyvill, G., McPheeters C. and Wyvill B, Data structure for soft objects, *The visual computer* Vol. 2, pp. 227-234, 1986.

[96] Yaegashi, H., Takahashi, T., and Kawasaki, M., Microcomputer-aided Reconstruction: a System Designed for the Study of 3-D microstructure in histology and histopathology, *Journal of Microscopy*, Vol. 146, 1987.

[97] Yun, H. J. and Park, K. H., Surface Modeling Method by Polygonal Primitives for Visualizing Three-dimensional Volume Data, *The Visual Computer*, Vol. 1, No.8, 1992.

Appendix

Help menu for MicroVisual

QUICK REFERENCE FOR VISUALIZING A VOLUMETRIC DATASET:

1. **Create** or **Open** a parameter file.
2. Set or change the parameters.
3. **Select Images** to visualize.
4. Choose **Visual Images** or **Rendering Only** or **Visual Bounding Box**.
5. Examine the result and adjust the parameters.
6. **Save** parameter window to a file if desired.

TO VISUALIZE A VOLUMETRIC DATA SET:

1. Type 'MicroVisual' to start the program.

   An icon with three sub-icons, namely '**File | Help | Exit**' will appear.

   Help icon is used to open a help window if it is clicked.

   (clicking means one click in this manual unless otherwise specified.)

   Exit icon is used to exit the MicroVisual program if it is clicked.

File icon is used to do most of the operations needed for visualizing a volumetric dataset. By clicking **File**, another icon will appear with the following options to choose: '**Create | Open | Save | Save As | Select Images | Visual Images | Rendering Only | Visual Bounding Box | Quit**'. Move the cursor to the desired operation which will be highlighted, and a click will carry out the highlighted operation. The function of each option is as following:

Create: to create a parameter file by changing the default values with the desired values.

Open: to open a pre-stored parameter file.

Save: to save the parameters to the parameter file in use.

Save as: to save the parameter file to a distinguish file name.

Select Images: to select images for visualizing.

Visual Images: to visualize selected images which including surface generation and surface rendering.

Rendering Only: to render surfaces. This option is used only if image parameters and cutting plane haven't been changed.

Visual Bounding Box: to visualize only the volume under the current camera position. This option is used to have a quick check as to how the camera position is composed.

Quit: is to quit the program.

1. Open or create a parameter file by clicking the 'File' menu and choose **Open** or **Create**. If **Create** is chosen, four windows will appear. If **Open** is chosen, one window will pop up in which all the files and directories in the current directory all listed. Choose the desired file to open by clicking on the file and click **OK** button on the right, or double click the file name. Or click on... to go to the desired directory. After the open file is chosen, also four windows will pop up just like using **Create**.

2. Change parameters in the Parameters and Lighting Parameters windows as desired by clicking in the corresponding box, deleting the old one and input the new value. There are three groups of parameters in the Parameter window: *Image Parameters, Options* and *Shading Parameters*.

Image Parameters:

Threshold: is the value distinct objects of interest from background.

Xdim: is the volumetric dimension along x-axis.

Ydim: is the volumetric dimension along y-axis. Xdim x Ydim is the image size of the volumetric data.

Zdim: is the volumetric dimension along Z-axis. It is the number of slices in the volumetric data.

dz: is the distance between two consecutive slices. It uses the same length unit as that of two pixel distance of the images.

Options:

Statistics: is an option for collecting information about how many cubes contain objects of interest and what percentage of the overall volume is represented by the object over the overall volume.

Cutting: is the option to allow a cutting operation on the volume.

Invert: is used when the background corresponds to white and the object corresponds to black. The program defines background as black.

Shading Parameters and Light Parameters:

Mode: has three options: Wire: for wire frame plot of the reconstructed object; Gourard: for Gourard shading; Phong: for Phong shading. Both Gourard and Phong shading will give a realistic results, and Phong shading is more realistic although it takes longer time to render.

Size: is to define the size of the final reconstructed image.

Output Name: is to define the file name of the reconstructed image.

Background: is to define the background color by assigning intensity values to three colors. Same intensities of the three colors will give a white (1.0) or black (1.0) background.

Object color: is to define the object color properties.

color intensity: red, green and blue. Values are between 0-1.

ambient: is the object intensity without a light, value is between 0-1.

specular: is the reflection ratio of the objects, value is between 0-1.

shine: is the shininess of an object. Value is between 0-1.

Camera: is to define the viewing position.

location: xfrom, yfrom and zfrom, the position the camera view from.

view at: xat, yat and zat, is where the camera is focusing at.

orientation: xup, yup and zup, is to define where is the up axis.

The vector of (view_from - view_at) must NOT be parallel to the view_up vector.

3. If a cutting operation is ON, choose three points, to define the cutting plane, in Cutting window by double clicking the three buttons of the mouse respectively.

4. Save changes to a file either through **Save** which will save the file to the opened parameter file, or **Save As** to a new file. Both **Save** and **Save As** are under **File** menu.

5. Select images by clicking on **Select Image**s under **File** menu.

6. Visualize images by clicking on **Visual Images** under **File** menu. The visual images process includes two main parts: surface generation and rendering.

7. If the visualization result is not satisfactory because of the shading parameters or new results are desired from different shading parameters, change these parameters and use **Rendering Only** to do rendering and skip the surface generation process which will save processing time significantly. **Rendering Only** is also under **File** menu.

8. **Visualize Bounding Box** under **File** menu is used when only a tentative view is desired of how the input volumetric data is related to the camera used. **Cutting** option must be kept OFF if this command is used.

9. **Quit** to quit the program.