Two Algorithms for Lossy Compression of 3D Images

by

Bernard Yiow-Min Chin

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

May 1995

Author .....
Department of Electrical Engineering and Computer Science
May 21, 1995

Certified by .....
Professor David H. Staelin
Assistant Director, MIT Lincoln Laboratory
Thesis Supervisor

Certified by .....
Dr. Keh-Ping Dunn
Group Leader, LL System Testing and Analysis
Thesis Supervisor

Certified by .....
Dr. Dan O'Connor
Staff, LL System Testing and Analysis
Thesis Supervisor

Certified by .....
Professor Frederic R. Morgenthaler
Chairman, Department Committee on Undergraduate Theses

Two Algorithms for Lossy Compression of 3D Images

by
Bernard Yiow-Min Chin

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 1995

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

The amount of space associated with the transmission and storage of 3D images is enormous. This study proposes a system to compress 3D images composed of objects whose surfaces are of prime interest. First, the object of interest is extracted from a 3D data set. Laplacian-based surface extraction is performed to obtain the object's surface. Two different algorithms, the Spherical Mapping and the Ring Peeling methods, are then used to represent the 3D surface in a compact manner. The Spherical Mapping method is a lossy surface representation which applies only to star-shape-convex surfaces. To test its performance, a simulated solid cube and a simulated solid pyramid are generated. Results show that compression ratios of approximately 33% and 25%, respectively, are achieved with visually acceptable reconstructions. The Ring Peeling method is a lossless surface representation method capable of handling nonconvex surfaces. To test its performance, the simulated solid cube, a newly simulated nonconvex C-shape object, and the human head in a real MRI data set are used. Results show that compression ratios of approximately 34%, 34%, and 38%, respectively, are achieved with lossless reconstructions. In general, the Spherical Mapping method is more efficient because the level of compression can be selected, but is computationally slower and is restricted to the class of star-shape-convex objects. The Ring Peeling method can handle nonconvex surfaces and is computationally faster, but its compression level is bounded. The choice of which of the two surface representations to use will largely depend on the application at hand.

Thesis Supervisor: Professor David H. Staelin
Title: Assistant Director, MIT Lincoln Laboratory

Biographical Note

Bernard Chin currently attends the Massachusetts Institute of Technology where he is pursuing the Ph.D. degree in Electrical Engineering and Computer Science. He received the Bachelor's degree in Management Science at MIT in 1994, with a minor in Economics. In 1991, he worked at the MIT Plasma Fusion Center where he co-authored "Construction of a Scanning Two-Dimensional Thomson Scattering System for Alcator C-Mod". He was also appointed Laboratory Teaching Assistant for the course, Structure and Interpretation of Computer Programs (6.001). From 1992 to 1994, he worked in the System Testing and Analysis Group at MIT Lincoln Laboratory. Various projects include "Non-uniform Sampling and the Cramer-Rao Bound" and "Characterization of Optical Wake from TCMP-1B". Research interests include signal processing, image processing, and multimedia applications. Bernard Chin is also a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

Acknowledgments

This work was done at the System Testing and Analysis Group (Group 32) at MIT Lincoln Laboratory, Lexington, MA. The author thanks Professor David Staelin, Dr. Keh-Ping Dunn, and Dr. Dan O'Connor for their encouragement, support, and feedback with regards to developing an algorithm capable of handling nonconvex objects. Richard Mozzicato and Steve Wiener provided invaluable advice on presenting the findings. Michael Jordan has provided support with computer graphics programming. Gavin Steyn and Tom Loden assisted with computing resources.

Table Of Contents

Chapter 1
Introduction

1.0 Background

1.1 Prior Research Efforts

1.2 Applications

1.3 Outline of Thesis

Chapter 1

Introduction

1.0 Background

In ever increasing numbers, data sets that represent three-dimensional, volumetric information are being generated. Three dimensional arrays of digital data representing spatial volumes arise in many scientific applications. Computed tomography (CT) and magnetic resonance imaging (MRI) systems create a volumetric data set by imaging a series of cross sections. These techniques have found extensive use in medicine and non-destructive evaluation (NDE) [1][2][3]. Other precocious noninvasive imaging tools, such as optical tomography [5], augment the growing presence of volumetric data sets. Astrophysical, meteorological, geophysical, and climatological measurements quite naturally generate volumetric datasets. Alternatively, volumetric data sets can be artificially generated rather than empirically measured. Computer simulations such as stress distributions over a mechanical part or pressure distributions within a fluid reservoir generate such data sets via analytically defined models or more commonly, numerical models employing finite difference or finite element techniques. As technology advances in imaging devices and computer processing power, more and more applications will generate volumetric data in the future.

In 3D data sets, objects are often of prime interest. In particular, surfaces are important characteristics of the objects and provide a lot of information. In climatology, isodensity surfaces provide information about cloud shapes and temperature distributions. In medical imaging, three-dimensional surfaces of the anatomy offer a valuable medical tool [4]. Images of these surfaces, constructed from multiple 2D slices of CT, MRI, and single-photon emission computed tomography (SPECT), help physicians to understand the complex anatomy present in the slices and to perform diagnosis.

Associated with 3D data sets are the tremendous storage and transmission requirements. As data set resolution increase and as high bandwidth communications technologies become cheaper and more prevalent, there will be increasing pressure to develop efficient storage techniques. This study proposes a compression system to compress 3D data sets composed of objects whose surfaces are of particular interest. First, an object of interest in a 3D data set is extracted. Laplacian-based surface extraction is performed to obtain the surface of the object. Two different algorithms, the Spherical Mapping and the Ring Peeling methods, are then used to represent the 3D surface in a compact manner. These two algorithms not only provide compact surface representations for applications such as image archiving and telemedicine, but also surface representations conducive for applications such as object recognition and image registration.

1.1 Prior Research Efforts

In the past, a large amount of research effort has been dedicated towards shape compression and shape analysis in one- and two- dimensions. Description of boundary curves is an important problem in image processing and pattern recognition. Boundary representations for two-dimensional boundary contours have been well-developed. Run-length encoding, chain codes, quadtree, spline representation, Hough transforms[7], invariant-moments[8], and Fourier descriptors[6] have been used to compress two-dimensional contours as well as for applications such as detection and recognition of objects in a two-dimensional scene.

Less effort has been devoted towards shape description and compression in three-dimensions. More specifically, there has been substantially less work on shape description and compression of three-dimensional boundary surfaces. One method of shape representation, the 3D octree method, has been developed by Meagher[10][11] to represent arbitrary 3D objects. Essentially, this octree encoding method is an extension of 2D quadtree efforts into 3D and can handle complex objects. The main disadvantage of the encoding technique is the large memory requirement. A proof is given by Meagher[9] that the memory and processing computation required for a 3D object is on the order of the surface area of the object. Depending on the object and the resolution, this can still represent a large storage requirement. Several million bytes to store nodes, the basis of the octree data structure, may be necessary to represent realistic situations. If the object in consideration is a 3D surface, the octree method generates overhead in setting up the tree data structure and in representing non-surface volume elements, or voxels. Likewise, if run-length encoding is used to encode a 3D surface, there is substantial overhead in representing non-surface voxels. Intuitively, it should be more efficient to represent a 3D surface from using and manipulating only the surface voxels.

Another effort to represent and compress 3D structures originated from Burdin *et al* [12][13][14][15] with their work on long cylindrical anatomical objects such as long bones. Their approach is based on the use of a complete and stable set of Fourier descriptors of 2D curves that are used to define invariant features under elementary 3D geometrical transformations. However, the primary shortcoming is that their method does not exploit the correlation between the 2D slices of the 3D data set. Conceptually, we know that there is substantial correlation between contours on adjacent slices -- a contour on one slice has a similar shape to a contour on an adjacent slice. Additionally, their method applies to star-shape-convex objects only and can fail on nonconvex objects. This difficulty arises from their choice of a cylindrical parameterization for 3D objects.

Jankowski and Eichmann have endeavored to develop shape representations for image processing and pattern recognition applications [16][17][18][19][20]. In [17][20], they extended the idea of Fourier shape descriptors for 2D curves into higher dimensions for multi-dimensional surfaces. Essentially, a closed boundary in 3D is decomposed into spherical harmonics that are functions of two variables -- the elevation angle and the azimuth angle. Again, this method also applies to closed star-shape-convex surfaces only since the surfaces are sampled via a spherical parameterization. The advantage that correlation between slices are reflected in the 3D nature of

the spherical harmonics components motivates the development of Algorithm 1, the Spherical Mapping method, for surface compression.

The use of the spherical parameterization results in the inability to represent nonconvex surface sections. To represent nonconvex surfaces, there needs to be an ordered traversal of the 3D surface. While there is a unique ordered traversal of 2D curves, there are infinitely many traversals of a general closed surface. One traversal scheme is introduced in Algorithm 2, the Ring Peeling method, that results in a structure conducive for 3D chain-code compression.

1.2 Applications

There are multiple applications in which this research work can be applied. One of the applications where our two algorithms can be applied is in the area of object and shape recognition. Our algorithms convert a 3D surface into a description in another domain. In the Spherical Mapping method, shape is described by the stronger spherical harmonic components. In the Ring Peeling method, the shape is described by the chain-code links. Our shape representations can be adapted to form possibly translation, rotation, and scale invariant shape descriptors.

Another application is in the compression of data independent of the display device. For instance, our algorithms deal directly with the volume data. They make no assumptions about the display device such as maximum spatial resolution, type of renderer, or type of 3D display (i.e. stereographic vs. holographic). Since our algorithms depend only on the volume data, the compression format is portable from one display platform to another.

In the area of telemedicine, there is the desire to develop means for a physician to perform diagnostic examination and even perform surgery on patients who are located arbitrary far away[21]. For example, it is more desirable to send the 3D object of a patient's heart over telecommunication lines to an expert physician on the other end rather than sending a time-sequenced animation flow of 2D images. There is increased interaction by the expert physician from being able to view the object at viewing angles which he chooses than from being restricted to only viewing a predetermined animation sequence.

As three-dimensional display systems become cheaper and more feasible, the large amount of space associated with storage and transmission of 3D images will motivate further compression research. Scientific visualization generates enormous numbers of 3D data and surfaces are common features that provide researchers with a lot of information. Our algorithms for 3D surfaces are synergistic with the efforts to reduce the storage and transmission requirements.

1.3 Outline of Thesis

In the following chapters, we will explore and discuss the compression of 3D images composed of objects whose surfaces are of particular interest. Chapter 2 presents vocabulary, definitions, and the proposed compression system. Chapter 3 introduces the Laplacian-based method for the extraction of surfaces from objects. Also, the discrete implementation of the 3D Laplacian will be discussed. Once the surface of interest is obtained, surface compression is performed via two developed methods.

The first method, which is presented in Chapter 4, is labelled the Spherical-Mapping method. A 3D surface is mapped into a 2D planar image through the use of a spherical parameterization. Well-developed image compression techniques can then be applied to the 2D image. Specifically, transform coding using the discrete Fourier transform and the discrete cosine transform will be considered. Compression occurs when a subset of all the transform coefficients are retained. Thus, reconstruction entails inverting a 2D transform into a 2D map and then generating the 3D surface from the 2D map using the equations for spherical parameterization.

The second method, which is presented in Chapter 5, is labelled the Ring-Peeling method. Given a starting volume element, a 3D surface is iteratively peeled into adjacent rings. The rings are then processed through a stripping and gluing process to obtain a peel whose elements are as contiguous as possible. Conceptually, this method can be visualized as removing the apple skin of an apple by using a peeler. Ideally, one continuous peel is desired when possible. Because most peel elements are contiguous, the peel structure is conducive to 3D chain-coding compression. Reconstruction entails decoding the chain-coded peel into a peel structure and then generating the 3D surface.

Chapter 6 will compare and discuss the advantages and disadvantages of the Spherical Mapping and the Ring Peeling methods. Computational effort, complexity, reconstruction speed, reconstruction error, and relevant applications will be considered. Chapter 7 summarizes the findings and present future directions of this research project.

Chapter 2
Compression System

2.0 Introduction

2.1 Volumetric Data Sets

2.2 Surfaces

2.3 Compression Diagram

Chapter 2

Compression System

2.0 Introduction

This chapter introduces the basic mathematical definitions to be used throughout the thesis. Basic definitions and vocabulary of volumetric data sets as well as notations are discussed. Also the proposed compression system for compressing 3D images composed of objects whose surfaces are of interest is presented.

2.1 Volumetric Data Sets

In this study, we will focus on volumetric data sets that are scalar fields. In three-space, a scalar field is a single valued function, f, in the three spatial variables — x, y, and z. In other words, only one data value is associated with every (x,y,z) sample point in the volume. The basic element for a 3D volumetric data set is known as a volume element, (x,y,z), commonly referred to as "voxel". The voxel is analogous to the pixel we refer to in two-dimensional images. Figure 2-1 shows a 3D volumetric data set, V, and a representative voxel.

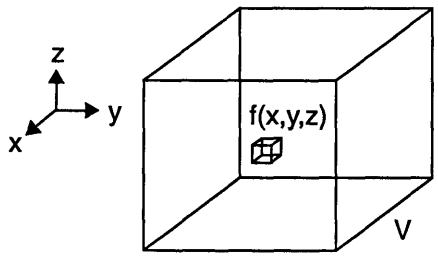


Figure 2-1. A scalar-field volumetric data set and a sample voxel.

Mathematically, we will refer to a discrete volumetric data set, V, defined over the cartesian space as

$$V \triangleq \left\{ f(x,y,z) \in \Re, \quad \begin{array}{l} x = 0,1,\ldots N_x - 1 \\ y = 0,1,\ldots N_y - 1 \\ z = 0,1,\ldots N_z - 1 \end{array} \right\} \tag{2.1}$$

For simplicity, we will assume that the lengths of the dimensions are equal,

$$N = N_x = N_y = N_z. \tag{2.2}$$

2.2 Surfaces

Figure 2-2 illustrates the surface associated with a sample object in a volumetric data set.

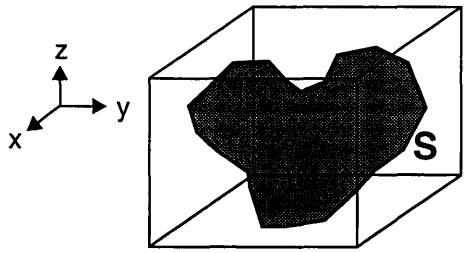


Figure 2-2. A sample surface in a volumetric data set.

Mathematically, we will refer to a surface in a 3D data set as S, the set of all voxels in a volumetric data set that are also in the surface:

$$S = \{ (x, y, z) \in V \mid f(x, y, z) = f_S \}. \tag{2.3}$$

In our study, we restrict our surfaces to isosurfaces where surface voxels are voxels with some constant value, f_S, in the volumetric data set. These chosen surfaces allow us to perform Laplacian-based surface extraction from a 3D data set. As a result, our surface can be rewritten as

$$S = \{ (x, y, z) \in V \mid \nabla^2 f(x, y, z) = 0 \}. \tag{2.4}$$

The surface, S, is then the set of all voxels in a volumetric data set where the Laplacian equals zero.

2.3 Compression Diagram

This section presents the proposed compression system diagram for compressing 3D images composed of objects whose surfaces are of particular interest. Once again, it is important to point out that the surfaces are the features we are interested in compressing. Figure 2-3 shows the system diagram.

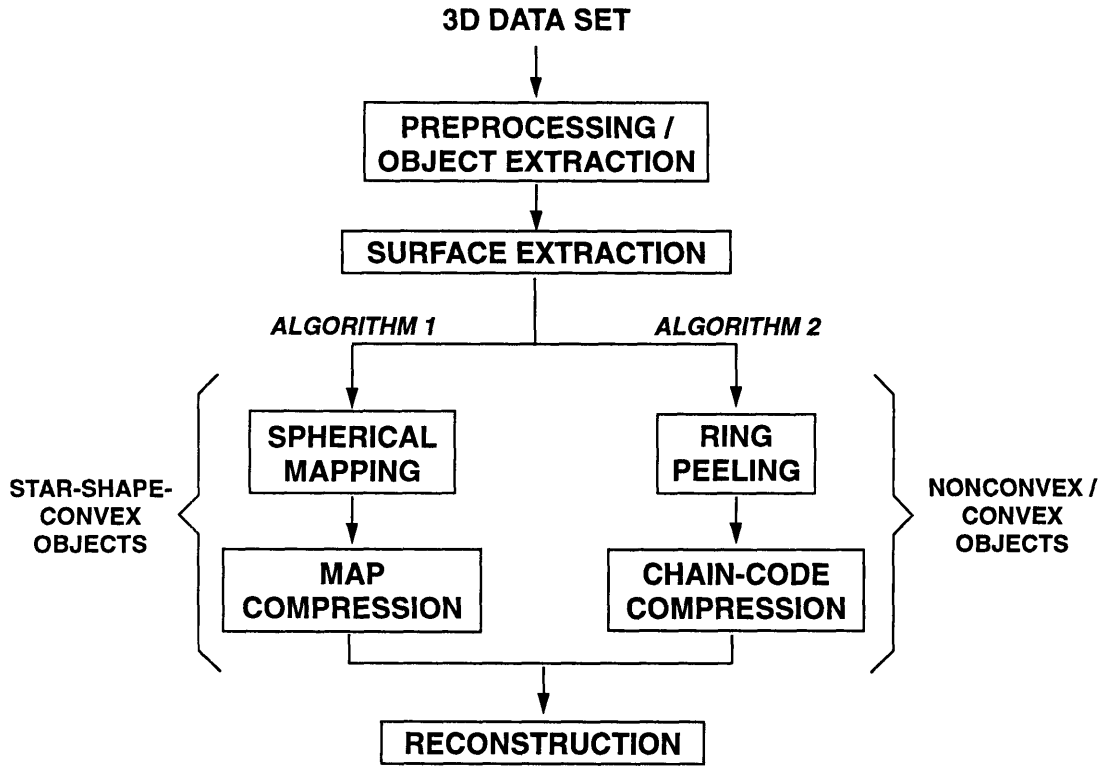


Figure 2-3. System Diagram for Compression.

Given any 3D data set, the first step is pre-processing to clean the data set, if necessary, and object extraction to select the object of interest within the volume. Some common pre-processing operations are median-filtering, smoothing, and interpolation. Once the object of interest is selected, Laplacian-based surface extraction is performed to extract the surface of the object. Now that the surface is obtained, two surface set representation methods can be applied to compress the number of bits needed to represent the surface -- the Spherical Mapping method and the Ring Peeling method.

The first method, Spherical Mapping, takes a 3D surface that is star-shape-convex and closed. Star-shape-convexity is defined in Chapter 4. Using a spherical parameterization, the 3D

surface is essentially mapped into a 2D planar image where well-developed image compression techniques can be performed. In our study, transform coding using the discrete Fourier transform (DFT) and the discrete cosine transform (DCT) will be used. Reconstruction involves inverting a reduced transform into a spherical map which then undergoes the reverse parameterization to yield a 3D surface.

The second method, Ring Peeling, takes a 3D surface that can be convex or nonconvex. The restriction of a closed surface is lifted in this method. Essentially, a 3D surface is decomposed into adjacent rings that undergo a splicing and gluing process to form a continuous peel. In practice, the peel is usually semi-continuous. The peel structure is conducive to a 3D chain-coding scheme which is presented in Chapter 5. Reconstruction involves decoding a chain-coded peel and then generating the 3D surface from the elements in the decoded peel.

Chapter 3
Surface Extraction

Chapter 3

Surface Extraction

3.0 Introduction

This chapter presents the Laplacian-based method of extracting surfaces. This method is an extension of Laplacian-based edge detection used to extract boundary contours in two-dimensional images [22]. The Laplacian-based method can extract surfaces from a 3D binary data set containing a surface as well as selecting a surface in a 3D raw data set that is complete.

3.1 One-Dimensional Edge

In the case of a one-dimensional edge, we have an profile as shown in Figure 3-1. Ideally, a one-dimensional edge is monotonically increasing or decreasing. We have shown only the case of the increasing edge, $f(x)$, in Figure 3-1a since the decreasing edge is analogous. If we take the first derivative of a the profile $f(x)$, we see that the resulting gradient in Figure 3-1b has a local maxima where the edge is rising the fastest. When we take the second derivative of $f(x)$, we find that in Figure 3-1c, there is a local maxima where the gradient rises the fastest and a local minima where the gradient drops the fastest. Also, a zero-crossing occurs where the gradient is maximum or where the original edge $f(x)$ rises the fastest. So we define edges as locations of x where the second derivative of $f(x)$ equals zero.

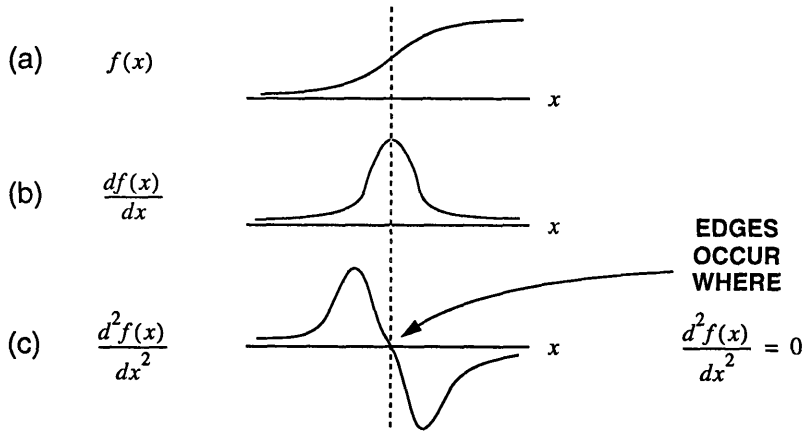


Figure 3-1. (a) An ideal one-dimensional increasing edge f(x). (b) First derivative. (c) Second derivative.

3.3 Three-Dimensional Surface

If we extend the case of the one-dimensional edge into two-dimensions, two-dimensional boundary contours or edges occur where the Laplacian with respect to spatial variables x and y equals zero. In three dimensions, this translates into detecting 3D surfaces. The Laplacian of $f(x,y,z)$ in three dimensions is

$$\nabla^2 f(x, y, z) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \quad , \tag{3.1}$$

and the surface of $f(x,y,z)$, S, is a set of points (x,y,z) where the Laplacian equals zero,

$$S = \{ (x, y, z) \mid \nabla^2 f(x, y, z) = 0 \} . \tag{3.2}$$

In practice, we deal with discrete data that are functions of discrete spatial variables. The 3D Laplacian given above applies to continuous functions so we have to find a discrete approximation. Also, because we deal with discrete spatial variables, it is often that the Laplacian is not identically zero. As a result, we restate the condition that describes surfaces as follows. Surface voxels are those voxels corresponding to a zero-crossing in the discretized Laplacian. Because zero-crossing detection in this manner is sensitive to noise, the local variance for a voxel can also be checked.

3.4 Discretized 3D Laplacian

A discrete approximation of the 3D Laplacian can be obtained if we implement the second derivatives in the 3D Laplacian formula with second-order difference equations. Specifically, we can approximate $\partial f(x,y,z)/\partial x$ with a forward difference equation,

$$\frac{\partial}{\partial x} f(x, y, z) \rightarrow f_x(n_1, n_2, n_3) = f(n_1 + 1, n_2, n_3) - f(n_1, n_2, n_3) . \tag{3.3}$$

We omit the scaling factor since it does not affect zero-crossing points. Since the forward difference in approximating the first partial derivative, we can use the backward difference in approximating $\partial^2 f(x,y,z)/\partial x^2$,

$$\frac{\partial^2}{\partial x^2} f(x, y, z) \rightarrow f_{xx}(n_1, n_2, n_3) = f_x(n_1, n_2, n_3) - f_x(n_1 - 1, n_2, n_3) . \tag{3.4}$$

Substituting equation (3.3) into equation (3.4), we find that,

$$\frac{\partial^2}{\partial x^2} f(x,y,z) \rightarrow f_{xx}(n_1,n_2,n_3) = f(n_1+1,n_2,n_3) - 2f(n_1,n_2,n_3) + f(n_1-1,n_2,n_3). \quad (3.5)$$

From equations (3.1) and similar versions of (3.5) for the other spatial variables, we obtain

$$\begin{aligned}\nabla^2 f(x,y,z) \rightarrow \nabla^2 f(n_1,n_2,n_3) &= f_{xx}+f_{yy}+f_{zz} \\ &= f(n_1+1,n_2,n_3)+f(n_1-1,n_2,n_3) \\ &\quad +f(n_1,n_2+1,n_3)+f(n_1,n_2-1,n_3) \\ &\quad +f(n_1,n_2,n_3+1)+f(n_1,n_2,n_3-1) \\ &\quad -6f(n_1,n_2,n_3). \end{aligned} \quad (3.6)$$

The discrete Laplacian operator, $\nabla^2 f(n_1,n_2,n_3)$, can be represented as a convolution of $f(n_1,n_2,n_3)$ with the Laplacian kernel $h(n_1,n_2,n_3)$ defined in Figure 3-2.

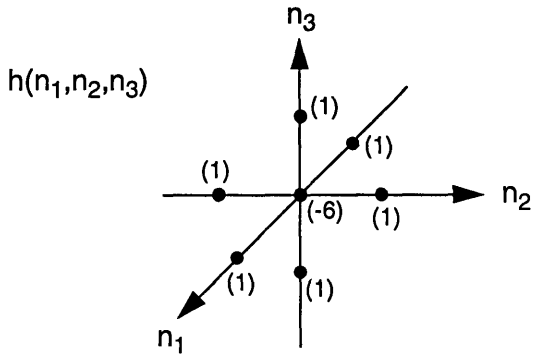


Figure 3-2. The $h(n_1,n_2,n_3)$ used in approximating $\nabla^2 f(n_1,n_2,n_3)$ with $f(n_1,n_2,n_3) * h(n_1,n_2,n_3)$.

Depending on how the second-order differences are approximated, it is possible to derive many other corresponding impulse responses $h(n_1,n_2,n_3)$.

3.4 3D Convolution

As described earlier, $\nabla^2 f(n_1,n_2,n_3)$ is approximated as a convolution of $f(n_1,n_2,n_3)$ with $h(n_1,n_2,n_3)$. Implementing convolution in the spatial domain can be enormously computation intensive. However, Fourier transform techniques can be used to speed up the computation of our Laplacian. Instead of convolving the two 3D sequences in the spatial domain, we find the 3D discrete Fourier transform (DFT) of $f(n_1,n_2,n_3)$ and $h(n_1,n_2,n_3)$, multiply the transforms, and

inverse the resulting transform to obtain the convolution result.

The equation of the 3D DFT for an N_1 by N_2 by N_3 sequence $x(n_1,n_2,n_3)$ is given by

3D Discrete Fourier Transform Pair

$$X(k_1,k_2,k_3) = \begin{cases} \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\sum_{n_3=0}^{N_3-1} x(n_1,n_2,n_3)\, e^{-j\left(\frac{2\pi}{N_1}\right)k_1n_1}\, e^{-j\left(\frac{2\pi}{N_2}\right)k_2n_2}\, e^{-j\left(\frac{2\pi}{N_3}\right)k_3n_3}, \\ \qquad 0 \le k_1 < N_1,\ 0 \le k_2 < N_2,\ 0 \le k_3 < N_3 \\ 0, \qquad \text{otherwise.} \end{cases}$$

$$x(n_1,n_2,n_3) = \begin{cases} \frac{1}{N_1N_2N_3} \times \\ \quad \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\sum_{n_3=0}^{N_3-1} X(k_1,k_2,k_3)\, e^{j\left(\frac{2\pi}{N_1}\right)k_1n_1}\, e^{j\left(\frac{2\pi}{N_2}\right)k_2n_2}\, e^{j\left(\frac{2\pi}{N_3}\right)k_3n_3}, \\ \qquad 0 \le n_1 < N_1,\ 0 \le n_2 < N_2,\ 0 \le n_3 < N_3 \\ 0, \qquad \text{otherwise.} \end{cases} \tag{3.7}$$

In equation (3.7), the sequence $X(k_1,k_2,k_3)$ is called the DFT of $x(n_1,n_2,n_3)$, and $x(n_1,n_2,n_3)$ is called the inverse DFT (IDFT) of $X(k_1,k_2,k_3)$. If $X(k_1,k_2,k_3)$ is computed in a straightforward manner, it can be shown that it requires on the order of

$$N_1^2N_2^2N_3^2 \tag{3.8}$$

additions and multiplications. If dimensional decomposition of the multidimensional DFT is used to decompose the 3D DFT into 1D DFTs and we perform direct computations of 1D DFTs, then it requires on the order of

$$N_1N_2N_3\,(N_1+N_2+N_3) \tag{3.9}$$

additions of multiplications. Furthermore, we can use Fast Fourier Transform algorithms to quickly and efficiently calculate the 1D DFTs, it would require on the order of

$$\frac{N_1N_2N_3}{2}\log_2 N_1N_2N_3 \tag{3.10}$$

multiplications and twice this number of additions to calculate the 3D DFT. Appendix A describes the decomposition process and compares the number of arithmetic operations required to calculate the 3D DFT using the three ways mentioned above.

Chapter 4
Spherical Mapping Method

Chapter 4

Spherical Mapping Method

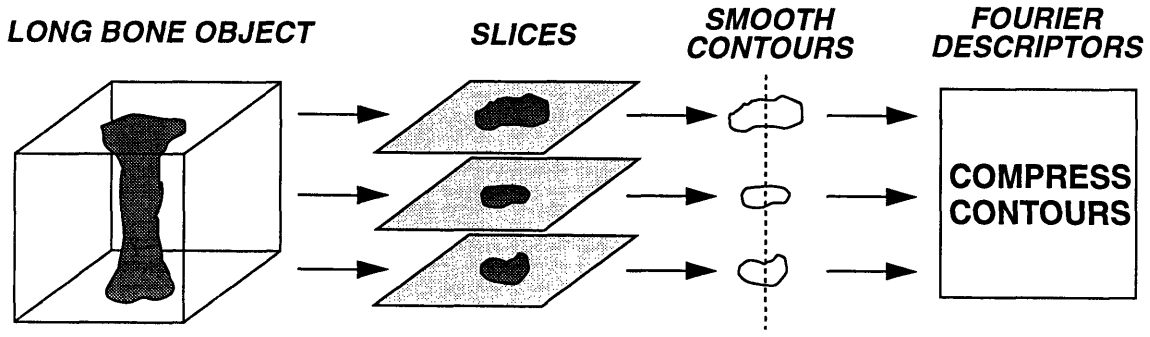
4.0 Introduction

Once the 3D surface, S, is obtained, we can compress the surface and reduce the amount of space needed to represent the surface. This chapter introduces the first of the two methods of surface compression proposed in this thesis. Namely, the Spherical Mapping method maps a 3D star-shape-convex surface S into a two-dimensional planar image or map, $r(\phi,\theta)$. Once this has been accomplished, $r(\phi,\theta)$ can be compressed using two-dimensional image transform coding techniques.

In this chapter, we will present what prior research has been done, discuss the limitations, and move to propose the Spherical Mapping method. The Spherical Mapping method will be explained. Then we will discuss what transform coding is, the DFT and DCT transforms used in converting the 2D map, and what is actually done with the transforms to compress the map. Performance measures will be defined and results from using simulated 3D data will be presented.

4.1 Previous Work Based on Long Bones

Earlier research by Burdin *et al* [12][13][14][15] attempted to compress 3D surfaces by working with contours on individual 2D slices. Figure 4-1 shows an overview of their method.



ALGORITHM:

1) TAKE CROSS SECTIONAL SLICE
2) OBTAIN BOUNDARY CONTOUR
3) INTERPOLATE CONTOUR USING SPLINES
4) FIND 2D FOURIER DESCRIPTORS OF CONTOUR
5) RETAIN SUBSET OF DESCRIPTORS

Figure 4-1. Method used by Burdin *et al* to compress 3D long, cylindrical objects.

Given an MRI data set obtained from imaging many two-dimensional slices of a long cylindrical bone, such as the radius or ulna bones in the arm, they first extract the boundary contour on each slice. For each contour, the curve is then interpolated by piecewise parameterized splines which verify certain continuity conditions. From this continuous description, it is simple to obtain new points which are spaced at constant curvilinear distance along the boundary curve.

The choice of cylindrical parametric form for contour representation is then used because of the morphology of the shape being considered. The surface of the bones is described by the following parametric equations:

$$\left.\begin{aligned} x &= \rho(s,t)\cos[\theta(s,t)] \\ y &= \rho(s,t)\sin[\theta(s,t)] \\ z &= t \end{aligned}\right\} \quad \forall (x,y,z) \in S \tag{4.1}$$

where $\rho(s,t)$ is a radius function which measures the length of the line connecting the boundary of the slice on the plane $z = t$ to the centroid of the boundary, $\theta(s,t)$ is the polar angular function in the sample planes, and s is the normalized arc of the 2D curve. Such a shape parameterization enables a reduced shape representation which is directly related to 2D shape information. Essentially, a slice is represented by its boundary which is itself described by the planar closed curve expressed by x and y.

For the radius function to sufficiently represent each contour, the contour is assumed to be star-shaped with respect to its centroid. The Fourier coefficients of the radius function are then computed. In their study, they further pursue to combine Fourier coefficients to obtain Fourier descriptors that are invariant to rotation, translation, and starting point. Compression is obtained when a subset of the Fourier descriptors are kept and used for reconstruction. Results show that retaining 6 out of 128 desciptors for each contour yielded a reconstruction that contain more than 90% of the original contour's energy.

However, there is a major limitation of their method. Primarily, their method does not exploit the correlation between boundary contours on adjacent slices. In other words, each boundary contour has been compressed individually. Physically, we know that the geometry of the object is such that the boundary contour on a slice is not entirely independent of the boundary contours on adjacent slices. In fact, there is a lot of geometric similarity between contours of adajcent slices. Another limitation is that their method yields different resolutions for contours of different arc length. Their choice of a fix number of samples for each contour results in greater resolution for shorter contours and lower resolution for longer contours. The retention of six descriptors for each contour will result in a 3D reconstruction that can have noticeable variations in resolution from slice to slice.

4.2 Spherical Mapping

To account for the strong correlation between slices, the Spherical Mapping method is developed and is based on Jankowski's and Eichmann's work on 3D Fourier descriptors [17][20]. The method invokes a spherical parameterization to map a 3D star-shape-convex surface into a planar map, compresses the planar map, and then performs the reconstruction. Figure 4-2 shows the section of the compression system diagram corresponding to the Spherical Mapping method.

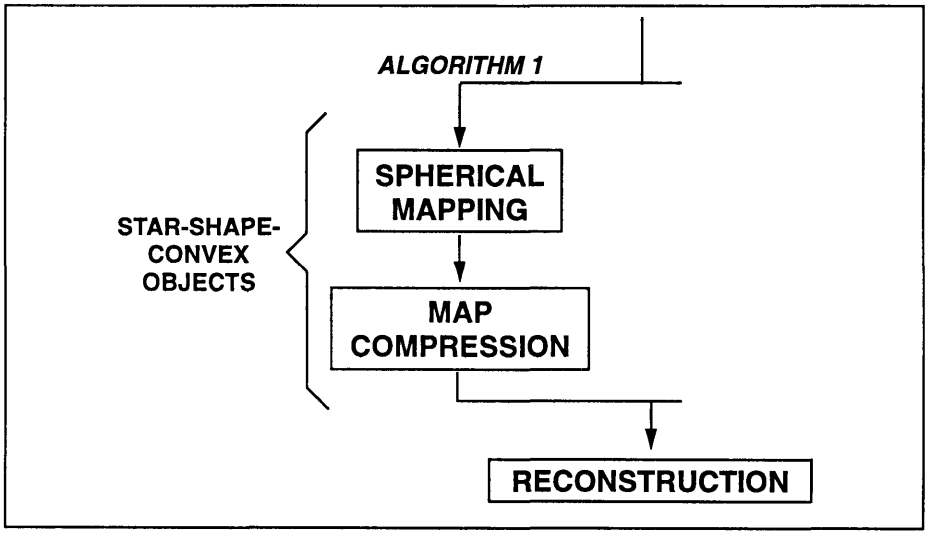


Figure 4-2. Block Diagram of Spherical Mapping method.

4.2.1 Star-Shape-Convexity

Algorithm 1, the Spherical Mapping method, takes as input a 3D surface, S, with certain restrictions. First, S must be a closed surface. Second, S must be star-shape-convex with respect to the centroid. A star-shape-convex surface is a surface whose distance function from surface voxels to the centroid of the surface is single-valued under the spherical parameterization

$$\left.\begin{aligned} x &= r(\phi, \theta)\sin\theta + x_0 \\ y &= r(\phi, \theta)\sin\theta + y_0 \\ z &= r(\phi, \theta)\cos\theta + z_0 \end{aligned}\right\} \quad \forall (x, y, z) \in S \tag{4.2}$$

where (x_0, y_0, z_0) is the centroid of the surface, obtained by

$$x_0, y_0, z_0 = \bar{x}_i, \bar{y}_i, \bar{z}_i \mid (x_i, y_i, z_i) \in S. \tag{4.3}$$

Figure 4-3 shows a visual example of a 3D object whose surface is star-shape-convex with respect to the centroid. The centroid is illustrated along with a sample surface voxel (x_i,y_i,z_i) in S.

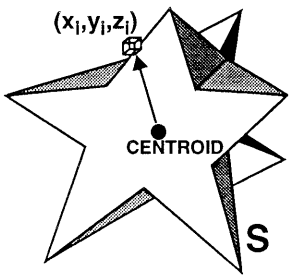


Figure 4-3. A sample star-shape-convex 3D object, along with the centroid and a sample surface voxel (x_i,y_i,z_i).

4.2.2 Sampling via Spherical Parameterization

Given a star-shape-convex surface, we sample the 3D surface, S, via a spherical parameterization described by equation (4.2). This choice of parameterization maps S into the spherical map, $r(\phi,\theta)$. Essentially, the three-dimensional surface function is mapped into a two-dimensional scalar function. Figure 4-4 illustrates the sampling process.

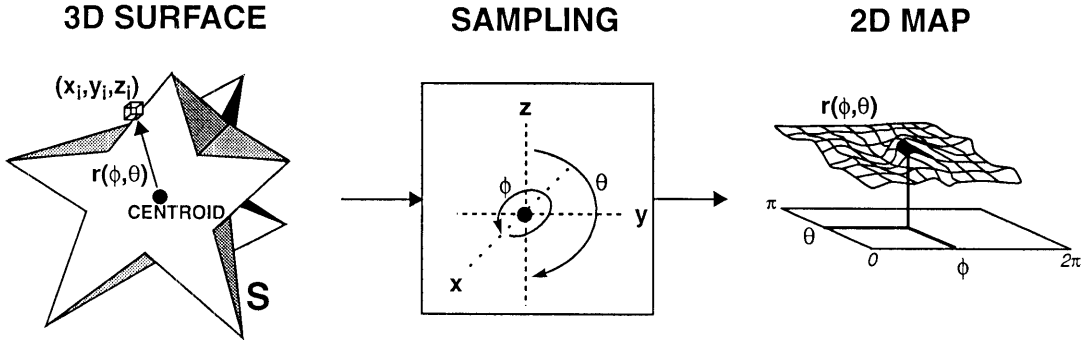


Figure 4-3. Sampling a star-shape-convex surface $s(x,y,z)$ via a spherical parameterization to yield map $r(\phi,\theta)$.

With regards to implementation, we desire a two-dimensional scalar function whose data points are taken on a uniform grid. This would allow us to use image compression methods on uniformly-gridded data later. In our scenario, we would like to sample uniformly in both the ϕ and the θ directions. Starting from the centroid of the star-shape-convex surface, S, we extend a ray radially outwards at a particular direction (ϕ,θ). Where the ray intersects the surface, we record the radius $r(\phi,\theta)$ which is the distance between the surface voxel and the centroid. We continue

associating a $r(\phi,\theta)$ for every (ϕ,θ) pair on our desired uniform 2D grid.

When we implement the ray extension process, we increment the radius outwards discretely. Therefore, we might extend over a surface voxel in certain situations and never find a surface voxel along the ray. In such cases, a nearest-neighbor approach is used. Whenever the ray extension process does not normally detect a surface voxel to generate an associated $r(\phi,\theta)$, we use the average (x,y,z) obtained from the closest neighboring surface voxels at the surface. Figure 4-4 demonstrates how we obtain (x_a,y_a,z_a), the average of the nearest surface voxels, so as to calculate the associated $r(\phi,\theta)$. Note that this sampling ray does not encounter a surface voxel as it extends radially outwards. The x, y, and z values of the three neighboring surface voxels are then averaged to yield (x_a,y_a,z_a). With (x_a,y_a,z_a), $r(\phi,\theta)$ is easily calculated.

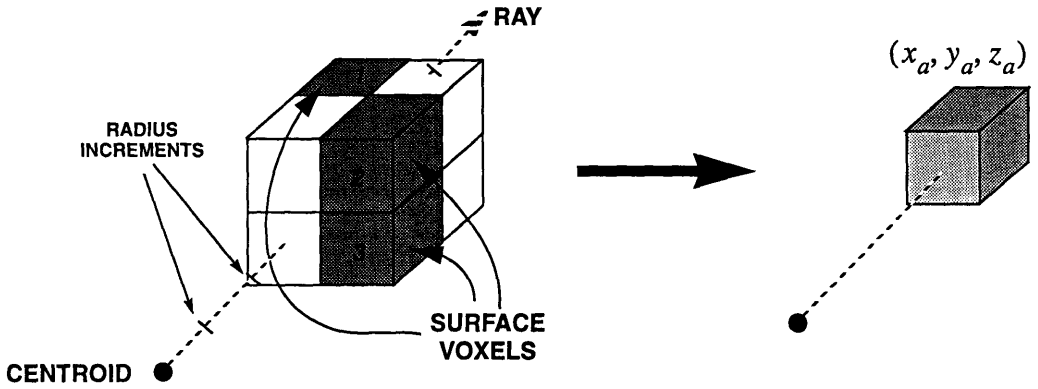


Figure 4-4. Nearest-neighbor averaging to obtain the average surface voxel (x_a,y_a,z_a) used to calculate $r(\phi,\theta)$.

A very important issue in the sampling process is sampling accuracy. If we do not sample the surface sufficiently via the spherical parameterization, we will not capture information on the the fast, varying areas of the surface. Assuming the radius function of the surface is bandlimited, the Nyquist criterion requires the sampling frequency to be at least twice the highest frequency component in the radius function. In our study, we sample a discrete surface. We will assume that Nyquist criterion is satisfied if we sample the surface such that the number of samples is at least twice or three times the number of surface voxels. The samples are located at (ϕ,θ) pairs where

$$\phi \in \{0, \frac{\pi}{M}, \frac{2\pi}{M}, \ldots, \frac{2(M-1)\pi}{M}\}$$

$$\theta \in \{0, \frac{\pi}{M}, \frac{2\pi}{M}, \ldots, \frac{(M-1)\pi}{M}\} \qquad (4.4)$$

M is a positive integer.

This setup results in twice as many sample bins in the azimuth direction than in the elevation direction. By carefully selecting M, we set how finely the surface is sampled in order to for the sampling to sufficiently represent fast, varying surface sections.

4.3 Compression

After the surface, S, has been mapped to $r(\phi,\theta)$, we can treat $r(\phi,\theta)$ as a regular two-dimensional image of radius values. The use of the spherical parameterization when performing sampling produces a $r(\phi,\theta)$ that has characteristics of normal two-dimensional pictures. Namely, in 2D images, we often see a lot of patterns and regularity. Consider a picture of a room with walls and paintings. Walls appear as smooth, regular regions in the image. These regions correspond to low frequency components in the image. On the other hand, paintings often have high details of objects such as people or landscape. These detailed features correspond to high frequency components. Converting an image, one with characteristics just described, into another domain might allow us to represent the same image, but with less information. This is the basis of transform coding which we will use to compress a 2D spherical map $r(\phi,\theta)$.

Transform coding as described in chapter 10 of [22] transforms an image to a domain significantly different from the image intensity domain, and the transform coefficients are then coded. Transform coding techniques to compress images typically perform significantly better than waveform coding techniques with scalar quantization (e.g. pulse code modulation). However, transform coding techniques are more expensive computationally.

Transform coding techniques attempt to reduce the correlation that exists among the image radius values. When the correlation is reduced, redundant information does not have to be coded repeatedly. Important in transform coding techniques is the exploitation of the observation that for typical images, a large amount of energy is concentrated in a small fraction of transform coefficients. This is commonly referred to as the energy compaction property. Because of this property, it is possible to code only a fraction of the transform coefficients and still be able to reconstruct an intelligible and visually acceptable image. Fortunately, our images have been observed to have the energy compaction property. Typically, the selected subset of transform coefficients is coded using quantization techniques followed by codeword assignments. Since the amount of compression obtained from quantization and codeword assignment is very dependent on the data, we will measure compression by the number of bytes needed to store the transform coefficients and their locations. This basis of comparison will also be used later to compare the Spherical Mapping method for compression with the Ring Peeling method for compression.

Two different transforms will be used to transform the 2D map $r(\phi,\theta)$ from the spatial domain into the frequency domain: the discrete Fourier transform (DFT) and the discrete cosine transform (DCT). Since we compute the transform during the compression process and also during the reconstruction process, it is desireable to have efficient ways to compute them. Both the DFT and DCT can use fast Fourier transform (FFT) techniques. They both have separable basis functions so that the 2D forward and inverse transform can be computed by row-column decomposition. As shown in Appendix A, this gives an efficient way of computing multidimensional DFTs. Both the DFT and DCT are energy preserving transforms. A transform coefficient with little energy contributes only a small amount to the signal energy. This can be verified with Parseval's Theorem.

4.3.1 Discrete Fourier Transform

The discrete Fourier transform (DFT) of a N_1 x N_2-point image 2D image $f(n_1,n_2)$ is $F(k_1,k_2)$. The analysis and synthesis equations are given as

2D Discrete Fourier Transform Pair

$$F(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} f(n_1,n_2)\, e^{-j\left(\frac{2\pi}{N_1}\right)k_1 n_1}\, e^{-j\left(\frac{2\pi}{N_2}\right)k_2 n_2}$$

$$f(n_1,n_2) = \frac{1}{N_1 N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} F(k_1,k)\, e^{j\left(\frac{2\pi}{N_1}\right)k_1 n_1}\, e^{j\left(\frac{2\pi}{N_2}\right)k_2 n_2}. \qquad (4.5)$$

The basis functions of the 2D DFT are separable, complex exponentials. The DFT transform has a fixed set of smooth basis functions, efficient algorithms for its computation, and good energy compaction for typical images. In addition, the signal energy in our spherical map $r(\phi,\theta)$ is mostly concentrated in the low-frequency regions. This is so because $r(\phi,\theta)$ typically is made up of some average radius value which represents the average distance between the surface and the centroid. This constant value in the spatial domain shows up as a very strong DC component in the transform domain. In addition to the average radius value, $r(\phi,\theta)$ is also made up of residual displacements when the average radius value is subtracted. Typically, these residual displacements are slowly varying with respect to ϕ and θ, and they together make up the bumps and the impressions on the surface. These slowly-varying displacements are represented as other low frequency components in the frequency domain. As a result, most of the energy is concentrated in the low frequency portions of the transform. Lastly, if we consider each individual frequency coefficient in the DFT transform individually and reconstruct the corresponding spatial functions, we find that the DFT method is essentially decomposing a surface into spherical harmonics. The topic of spherical harmonics will be discussed later when we run the Spherical Mapping method on simulated data.

4.3.2 Discrete Cosine Transform

We can improve the energy compaction property of the DFT without sacrificing other qualities such as the existence of a computationally efficient algorithm. The discrete cosine transform (DCT) has this improved characteristic, is closely related to the DFT, and is the most widely used transform in transform image coding. The N_1 x N_2-point even symmetrical DCT of a N_1 x N_2-point image 2D image $x(n_1,n_2)$ is $C_x(k_1,k_2)$. The analysis and synthesis equations are given as

2D Discrete Cosine Transform Pair

$$C_x(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} 4x(n_1,n_2)\cos\frac{\pi}{2N_1}k_1(2n_1+1)\cos\frac{\pi}{2N_2}k_2(2n_2+1)$$

$$x(n_1,n_2) = \frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} w_1(k_1)\,w_2(k_2)\,C_x(k_1,k_2)\times \cos\frac{\pi}{2N_1}k_1(2n_1+1)\cos\frac{\pi}{2N_2}k_2(2n_2+1) \qquad (4.6)$$

where

$$w(k) = \begin{cases} \frac{1}{2}, & k=0 \\ 1, & 1\le k<N \end{cases} . \qquad (4.7)$$

To illustrate the relationship between the DCT and DFT, and the improvement in energy compaction that the DCT offers, we consider an N-point 1-D sequence $x(n)$ shown in Figure 4-5.

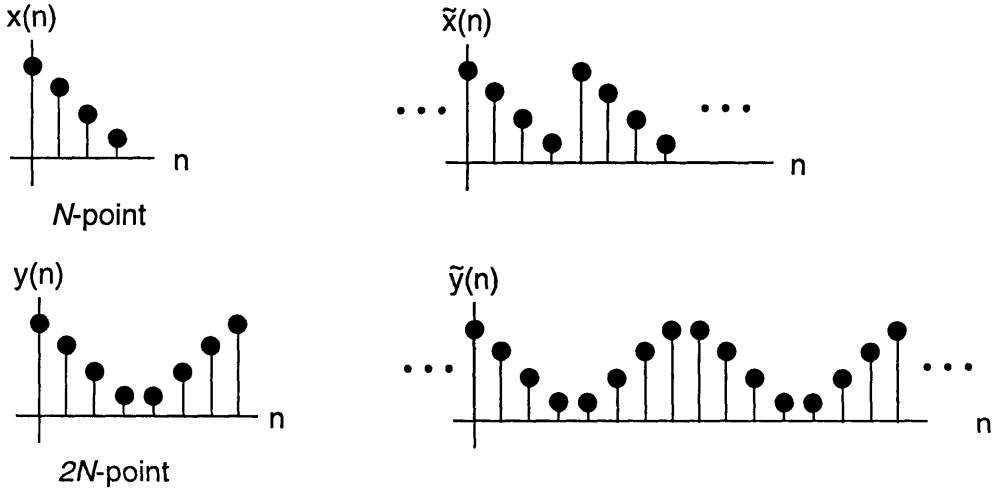


Figure 4-5. Left: y(n) is a extended symmetric version of x(n). Right: Periodic sequences obtained from x(n) and y(n).

The sequence *y(n)* is related to *x(n)*. It is essentially the original *x(n)* extended by a time-reversed copy of *x(n)*. When we take the DFT of *x(n)*, the DFT consists of the discrete Fourier series (DFS) coefficients which are obtained from the periodic sequence $\tilde{x}(n)$. From the figure, we see that this periodic sequence has sharp artificial discontinuities which arise from joining the beginning and end parts of *x(n)*. The discontinuities contribute energy to high frequencies and reduce the DFT's efficiency in energy compaction. To eliminate the artificial discontinuities, we create a *2N*-point sequence *y(n)* from *x(n)* as shown. If we now compute the *2N*-point DFT of *y(n)*, the DFT consists of DFS coefficients obtained from the periodic sequence $\tilde{y}(n)$. The DCT is defined in terms of these DFS coefficients. Clearly, the periodic sequence does not have artificial continuities and the DCT's efficiency in energy compaction is improved.

In our study, our 2D DCT utilizes separable basis functions. These basis functions are real cosine functions that are smooth and continuous. Since we use a separable 2D DCT, we can calculate the 2D DCT efficiently as 1D DCTs using row-column decomposition. Also, because the basis functions of the 1D DCT are real whereas the basis functions of the 1D DFT are complex exponentials, the calculation of one 1D DFT can essentially be adapted to calculate two 1D DCTs.

When we compress our spherical map, $r(\phi,\theta)$, we essentially generate an extended image by extending from $[0,2\pi]$ to $[0,4\pi]$ in the ϕ direction and from $[0,\pi]$ to $[0,2\pi]$ in the θ direction. The even symmetrical choice of extension as outlined above is used as opposed to other possible extensions. By extending the image, we reduce the artificial discontinuities in the azimuth and elevation direction. Since we use a spherical parameterization in our sampling of a closed surface, our sequence is naturally periodic in the ϕ direction. Little reduction in artificial discontinuities is expected. Therefore, most of the reduction in artificial discontinuities come from the smoothing obatined in the θ direction. With regards to compression, there is a greater contribution to energy compaction from the θ direction. This translates into a greater contribution to compression from image extension in the θ direction.

4.4 Reconstruction

In the Spherical Mapping method for surface compression, recontsruction is a straight-forward reverse process. Reconstruction entails taking the reduced 2D transform of a spherical map, inverting the reducing transform to obtain a new spherical map $r'(\phi,\theta)$, and then generating the compressed 3D surface *S'* from $r'(\phi,\theta)$. The most computationally expensive step lies in the computation of the inverse transform.

4.5 Data Analysis

This section will present the results of the Spherical Mapping method on simulated sets. General characteristics about the data sets will be described. Performance measures will also be defined. Also, the performance from using DCT transform coding will be compared to the performance obtained from using DFT transform coding.

4.5.1 General Characteristics of Data Sets

Three simulated volumetric data sets, each containing an object, are generated. The three objects defined are a solid sphere, a solid cube, and a solid square pyramid. Each volumetric data set contains cuberille data, where every voxel is a cube and identical. The resolution of the data set is $N=64$ voxels in each of the three spatial dimensions x, y, and z.

The solid sphere is defined as all the voxels within a radius of 10 voxels from the center of the sphere, which is located at (32,32,32). Because of the low resolution of the discrete data set, the roughness of the surface of the sphere is very noticeable. The solid cube is defined with the faces parallel to the xy-, yz-, and xz-planes. The width of each edge is 21 voxels and the center of the cube is also located at (32,32,32). The solid square pyramid is defined with its base parallel to the xy-plane. The pyramid is a right pyramid with a rising height of 21 voxels.

When we perform sampling via a spherical parameterization, we choose $M=128$. This corresponds to sampling a surface at 32,768 different pairs of (ϕ,θ). For the three simulated data sets, this number of samples is several times the number of surface voxels for the object. By doing this, we assume that we have satisfied an equivalent Nyquist criterion for sampling a cartesian system by using a spherical parameterization in order to resolve the surface fully.

Several graphical methods will be used to render the objects and surface to facilitate visualization of these three-dimensional data sets. The three methods chosen are display of slices, a cuberille representation, and a Marching-Cubes-based method. The first method is the display of 2D cross-sectional slices of a 3D data set. In our study, 2D cross-sectional slices parallel to the xy-plane will be taken along regular intervals on the z-axis. These slices will be arranged to fit in a bigger 2D image in an orderly fashion. Slices taken starting at $z=0$ in a 3D dataset onwards with increasing z are arranged from left to right and top to bottom in the larger 2D image. Thus to visualize what the 3D image looks like, we have to assemble and stack these slices in our mind's eye.

Another set of graphical methods, labelled 3D display methods, aim to show a 3D object whole, rather than breaking it up as in the 2D cross-sectional slicing. One of these methods is the cuberille representation for display. Each voxel in the dataset will be displayed as a miniature, solid 3D cube made up of polygons. Therefore, a solid object is displayed as an assortment of these sugar cubes suspended together in space. Because we are dealing with low resolution at $N=64$, the cube-like nature of the object will be evident. It should be pointed out that this method of visualization is usually not appealing to the eye. As resolution increases, however, each voxel

can perhaps be represented as a dot rather than a cube that is constructed from polygons. Another 3D display method is the Marching Cubes method[4] which ackowledges the fact that 3D volumetric data sets are still relatively low-resolution. This method marches through computational cubes in a 3D data set which a certain charateristic, and outputs a list of polygons which can then be rendered using computer graphics methods. The Marching Cubes method tends to smoothen the look of an object and is more appealing to the eye. However, the caveat is that this method does not work well in certain instances.

All the graphics are created on an SGI Iris Indigo workstation. The 2D cross-sectional slicing display method and the 3D cuberille representation are written using the Iris GL graphic libraries. The Marching Cubes method for display is done by using the isosurface generation and display option in the mathematical software package IDL. This package actually uses a modified version of the Marching Cubes algorithm but conceptually, the idea is similar.

4.5.2 Performance Measures

Four performance measures are defined to compare reconstructed 3D surfaces with the original surface. The first measure is the percentage of voxels missing from the original surface. The second measure is the percentage voxels added to the original surface after reconstruction. These measures are quick calculations which can be used to verify that the compression algorithm generally works properly. The next two performace measures are more commonly used in image processing to gauge the performance of compression algorithms.

The NMSE, or normalized mean-square error, is defined as

$$NMSE\,(in\ \%) \;=\; 100 \times \frac{VAR\,[\hat{x}\,(n_1, n_2, n_3) - x\,(n_1, n_2, n_3)\,]}{VAR\,[x\,(n_1, n_2, n_3)\,]} \quad \% \qquad , \tag{4.8}$$

where $x\,(n_1, n_2, n_3)$ is the original 3D image, $\hat{x}\,(n_1, n_2, n_3)$ is the reconstructed image, and *VAR* is the variance operator. In our study, the original 3D image is the 3D data set that contains the extracted surface. The reconstructed image is the 3D data set that has been generated from a reduced spherical map. The SNR, or signal to noise ratio, is related to the NMSE and is defined as

$$SNR\,(in\ \mathrm{dB}) \;=\; 10\,\log\!\left(\frac{NMSE\ \mathrm{in}\%}{100}\right)^{-1} \quad \mathrm{dB} \qquad . \tag{4.9}$$

In contrast to these quantitative measures of performance, we will also use the visual acceptability of the reconstructions based on the subjective comparison by a human observer as a qualitative measure of performance. Since shape information is important in our study, visual acceptability is necessary and can favor a particular reconstruction in cases where quantitative measures deem otherwise.

4.5.3 Compression and Storage Size

We now address the question: When does the Spherical Mapping method for surface compression outperform conventional methods for storing surfaces? We define N as the number of voxels in each spatial dimension, s as the number of surface voxels in the object of interest, and α as the number of transform coefficients we retain in the transform of the spherical map. To store a raw data set, we have to store the values associated with each of the NxNxN voxels. Thus, to store a raw image, it requires N^3 bytes.

To store only the surface voxels of an object, we require

$$min\left[\frac{N^3}{8}, 3s\right] \tag{4.10}$$

bytes since the surface can be stored in one of the two following conventional ways. First, we can store the surface as bit stream of 0's and 1's, where 0 corresponds to non-surface voxels and 1 corresponds to surface voxels. For example, we can loop through the x dimension, then the y dimension, and lastly the z dimension to obtain the bit stream. Since we traverse N^3 voxels in our search for surface voxels, the bit stream will be N^3 bits long, or $N^3/8$ bytes long. Alternatively, we can directly store the locations of the s surface voxels. For each of the s surface voxels, we require 3 bytes to store the (x,y,z) location. Therefore, it requires $3s$ bytes to store all the surface voxels. Typically, $3s$ is less than $N^3/8$.

To store the surface using the Spherical Mapping method, we require

$$12\,(\alpha + 1) \tag{4.11}$$

bytes to store the α retained transform coefficients. We first record the location of the centroid of the surface. This requires three floats, or 12 bytes. For each of the α transform coefficients we retain, we record the coefficient value and location. Assuming the coefficient value is complex, we require two floats, or 8 bytes. To store the coefficient location, we need another two integers, or 4 bytes. Therefore, to store all α coefficients, we need the number of bytes stated in equation (4.11). Here, we would like to point out that if the DCT transform is used, coefficient values are real and thus require only one float per coefficient value.

In order for the Spherical Mapping method to outperform conventional storage methods, equation (4.11) must be less than equation (4.10). Table 4-1 shows the amount of space required to store the surface of the simulated objects using conventional methods, and how many coefficients the Spherical Mapping can keep while still outperforming conventional storage methods. As mentioned in an earlier section, we have chosen N to be 64 in our analyses. From the column for conventional methods, we find that $3s$ is indeed typically smaller than $N^3/8$. From the column for the Spherical Mapping method, we find that we have to limit the number of transform coefficients we keep, and this number differs depending on the object.

Table 4-1: Conventional vs. Spherical Mapping Method for Surface Storage

OBJECT	CONVENTIONAL $min\left[\frac{N^3}{8}, 3s\right]$ BYTES	SPHERICAL MAPPING $12(\alpha+1)$ BYTES
Sphere	min [32768, 3*978] = 2934	$\alpha < 244$
Cube	min [32768, 3*2402] = 7206	$\alpha < 600$
Square Pyramid	min [32768, 3*5122] = 15366	$\alpha < 1280$

4.5.4 Solid Sphere

Figure 4-6 shows three methods of visualizing the solid sphere data set. In Figure 4-6a, sixty-four cross-sectional slices along the z-axis of the data set have been taken. The filled circles correspond to slices that cut through the solid sphere. In Figure 4-6b, the cuberille representation of the solid sphere is shown. Where the solid sphere is defined, a miniature sugar cube is displayed. The bounding box shows the extent of the data set. Note the "box"-iness of the display that results from the low resolution of the data set. Figure 4-6c shows the smoothened display based on the Marching Cubes algorithm to render the solid sphere. This is visually more appealing than Figure 4-6b, but jaggedness remains again from the low resolution. Once again, it should be stressed that these are just few of many techniques to display 3D data sets. The ones which we use have been chosen to visualize the data enough to make our points regarding surface compression algorithms.

We take the solid sphere data set and perform Laplacian-based surface extraction as described in Chapter 3. Figure 4-7 shows the resulting data set after the surface extraction step. In Figure 4-7a, we see that the cross-sectional slices of the surface yield concentric circles that are hollow. The surface extraction process essentially leaves a shell of the sphere. The 3D displays of Figures 4-7b and 4-7c reveal virtually little difference from Figures 4-6b and 4-6c, respectively, since the 3D display methods do not reveal the insides of the objects. On the other hand, the cross-sectional slices reveal that the surface extraction process works properly.

Given the surface, we sample the surface with respect to its centroid via a spherical parameterization. We obtain a spherical map, $r(\phi,\theta)$, as shown in Figure 4-8a. There are several interesting features to this map. First, the map is basically a constant value with $r(\phi,\theta)$=10. Second, the values vary with a step-like nature around the constant value. This comes from sampling a sphere in a data set which is discrete in space. When we define the sphere, the surface voxels of the sphere are not identically at an integer length of 10 from the center of the sphere. So

some surface voxels are closer than 10 from the center, and $r(\phi,\theta)$ fluctuates below 10. Also, as mentioned earlier, the ray extension process sometimes misses a surface voxel and has to resort to nearest-neighbor averaging. This can result in the fluctuation of $r(\phi,\theta)$ above the defined radius of 10. Lastly, note that the spherical map is periodic in ϕ. Additionally, for the case of the sphere, the spherical map is basically periodic in θ.

Since the spherical map is essentially a constant with relatively small fluctuations, the 2D discrete Fourier transform should also reflect the structure that is strongly present in the map domain. Specifically, the 2D DFT of a constant in map domain translates into a strong DC or 0-th frequency coefficient in transform domain. In Figure 4-8b, the magnitude of the 2D DFT of the spherical map yields a strong DC coefficient with substantially smaller higher frequency components that account for the fluctuations about the constant radius value in the map domain.

Observing such a clean DFT, we should be able to reconstruct the object by keeping relatively few transform coefficients. In the case of the sphere, we should do quite well by retaining only the DC coefficient. Figure 4-9 shows the reconstruction of the sphere surface when only one transform coefficient is kept. Visually, the result is very agreeable with the original surface in Figure 4-7. The sphere is a special case since the sphere object can be virtually represented as one of the basis functions of the 2D DFT. In general, objects require many more than merely one transform coefficient for sufficiently adequate reconstruction.

Figure 4-10 shows the performance of reconstruction when we keep more than one transform coefficient. In each of the four plots, the horizontal axis marks the percent of transform coefficients kept. The values alongside the triangles correspond to the actual number of coefficients kept. The vertical dashed line corresponds to the maximum number of coefficients that the Spherical Mapping method can keep in order to outperform conventional storage techniques. Thus, the regime left of the dashed lines is where we must operate in.

There are several interesting patterns to notice. The percent of voxels missing is zero for all cases. The percent of voxels added is relatively large at 60%. However, the reconstruction is visually acceptable so we will tolerate this high level of added voxels. The upward sloping NMSE tells us that reconstruction of the surface degrades as we retain more of the transform coefficients. The reason for the upward slope is that it takes only one coefficient, the DC coefficient, to represent the surface of a sphere. If we retain more coefficients, we are keeping more of the higher frequency coefficients and reconstructing more of the surface imperfections introduced during the mapping. These high-frequency coefficients in the transform domain translates into contributions of higher-order spherical harmonics in the spatial domain. Appendix B discusses the spherical harmonics associated with the DFT. Another thing to notice is that the plot of the SNR is consistent with the plot of NMSE. Since our reconstruction degrades upon keeping more coefficients, our SNR profile is downward sloping one. Again, the average level of the SNR does not accurately reflect the reconstruction performance. Consider the retention of one coefficient which give us a low SNR of approximately 5.5 dB. The visual acceptability of Figure 4-9, however, is high. The level of SNR can be strongly affected by the number of voxels added or missing to the surface.

4.5.5 Solid Cube

Figure 4-11a shows slices of the defined solid cube. The slices show solid squares since we have taken slices parallel to one of the faces of the cube. After the surface extraction process, we obtain hollow squares in some of the slices as shown in Figure 4-11b. In the two slices corresponding to the top and bottom face of the cube, the slices are solid squares. Figure 4-11c shows the smooth, 3D rendered cube.

Once again, we find the centroid of the surface and sample the surface at 32,768 sample locations, many times more than the 2402 number of surface voxels. The resulting spherical map that is obtained is shown in Figure 4-12a. Note the strong periodicities in both the ϕ and the θ directions. In particular, there are four peaks in the ϕ direction corresponding to the four edges of the cube we pass as we sample from $\phi=0$ to $\phi=2\pi$. In the θ direction, the spherical map has two peaks corresponding to the two edges of the cube when we sample from $\theta=0$ to $\theta=\pi$. This nice structure obtained in the mapping is reflected as large values for a small handful of Fourier coefficients shown in Figure 4-12b. The energy compaction allows us to thus represent the cube with relatively few coefficients.

Figure 4-13a show the reconstruction when we retain only the largest transform coefficient. In this case, the largest transform coefficient happens to be the DC component, which reconstructs to a sphere. When the number of coefficients retained, α, is set to 5, the reconstructed surface in Figure 4-13b begins to resemble more the original cube. By the time $\alpha=300$, we have reconstructed a surface as shown in Figure 4-13f that is visually acceptable. Note that from Table 4-1, we can only keep $\alpha=600$ at most in order for the Spherical Mapping method to perform better than conventional methods. By keeping $\alpha=300$, we are well under the limit and accomplish a compression ratio of 2:1.

In terms of performance, the retention of more transform coefficients allow us to represent the high-frequency surface fluctuations of the sphere. Therefore, the reconstruction performance increases with the number of coefficients retained. Figure 4-14 shows the graphs of the four performance measures. Note that the SNR profile is upward sloping in contrast to the special case of the sphere. By merely keeping 300 of 32,768 coefficients we can perform an excellent reconstruction of the cube.

4.5.6 Solid Square Pyramid

Figure 4-15a shows slices of the defined solid square pyramid. The slices show solid squares since we have taken slices parallel to the base of the pyramid. After the surface extraction process, we obtain hollow squares in some of the slices as shown in Figure 4-15b. However, the slice corresponding to the base of the pyramid contains a solid square. Figure 4-15c shows the smooth, rendered surface of the square pyramid.

As before, we locate the centroid and sample the surface. We take 32,768 samples of the surface which contains 5,122 voxels and obtain the spherical map shown in Figure 4-16a. In contrast to the case of the sphere and cube, the spherical map for the square pyramid has artificial

discontinuities at the edges in the θ direction. Specifically, $r(\phi,\theta)$ wraps back on itself in a continuous manner in the ϕ direction, but not the θ direction. As in the case of the cube, there are four peaks in the ϕ direction corresponding to the choice of a square base for the pyramid. The two peaks in the θ direction correspond to the apex of the pyramid at $\theta=0$ and to the joint between the base and the sloping faces of the pyramid at $\theta=\pi$. The Fourier transform magnitude of the map shown in Figure 4-16b also exhibits quite a bit of energy compaction.

Figure 4-17a shows the reconstructed surface, a sphere, when only the largest transform coefficient is kept. As we retain more coefficients as shown in Figures 4-17b through 4-17f, we reconstruct a surface that resembles more and more like the original square pyramid. By the time we keep 400 coefficients, we obtain a decent pyramid while remaining within the operating regime of less than 1280 coefficients. Also, at 400 coefficients, we achieve a compression ratio of 3.2:1. The slight surface bumps and recesses of Figure 4-17f arise from the imperfect mapping. As the resolution of the data set increases, these surface blemishes should be less conspicuous.

The plots of performance measures are shown in Figure 4-18. The results of the SNR are generally in accord with the visual acceptability. According to the SNR, keeping 300 coefficients give us approximately equal reconstruction performance with keeping 400 coefficients. At 300 coefficients, the compression ratio is approximately 4.3:1. In any case, the number of coefficients retained can be chosen depending on the quality of reconstruction we desire.

4.5.7 Choice of Transform: DFT vs. DCT

As described earlier, the DCT has better energy compaction properties than the DFT since it compacts the high-frequency components of a signal into low-frequency components. With respect to reconstructing surfaces, this means that a surface reconstructed using several DFT basis surfaces can typically be reconstructed using fewer DCT basis surfaces. In practice, this means we expect to achieve better compression using the DCT as the transform of choice, especially for cases where the spherical map has artificial discontinuities at the edges. Appendix B details the spherical harmonics associated with the DCT.

Figures 4-19 through 4-21 show the performance measures for the three objects when the DFT and the DCT is used. The performance of the DFT is shown as the solid line, while the performance of the DCT as the dashed line. For the case of the sphere in Figure 4-19, only one coefficient is needed to represent the sphere so the expected increase in performance from using DCT is irrelevant since the DFT and DCT have the same DC surface harmonic. In fact, from the SNR, we find that the DCT does a poorer job due to the downward sloping nature of the profile. As a rule of thumb, the DCT should perform better for general objects, with the sphere being a special case. Figure 4-20 shows the increased performance when we use the DCT for compressing the cube. Voxels added and missing are substantially lower. The NMSE profile is lower and the SNR profile is higher. The increase in compression is approximately 2:1 when we use the DCT instead of DFT. So we can keep fewer DCT coefficients, around 150, while preserving performance. The resulting compression of the surface is thus 4:1. The increase in performance is

also evident in the case of the square pyramid shown in Figure 4-21.

The quoted overall compression of 4:1 for the case of the cube is still an underestimate of the increased performance of the DCT. Because it takes fewer amount of bytes to store a real DCT coefficient than a complex DFT coefficient, the operating regime when using the DCT is wider. For the cube, this eventually translates in keeping approximately 1.5 times as many DCT coefficients for the same number of DFT coefficients we kept before. Thus for the cube, the quoted 4:1 compression ratio can be adjusted and we obtain an estimated 6:1 compression ratio.

The DFT and DCT transforms are the two transform chosen in this study. Other transforms can be used to transform the spherical map into perhaps a more suitable domain for representation. Because 3D objects tend to have different resolution at different scales and this feature will show up in the spherical map, multiresolutional transforms [23] can potentially have additional compression ability.

4.6 Discussion

The Spherical Mapping method provides an efficient representation of a closed, star-shape-convex surface. The quality of reconstruction can be chosen by deciding the number of coefficients to retain. Therefore, we can always choose a desired compression ratio when using this lossy compresssion algorithm. Even with compression ratios quoted earlier, the ratios measure how many bytes it takes to store the coefficient values and locations. Added compression can be obtained by entropy coding these values and locations. Appendix C briefly discusses how entropy coding can be used with the Spherical Mapping method.

There is a major limitation that arises with the Spherical Mapping method — the restriction of the type of objects it can handle. Particularly, the star-shape-convexity restriction cuts out many physical objects. In Figure 4-22a, a nonconvex C-shape object is defined and surface extraction is performed. The centroid of the surface is located slightly inside of the inner vertical panel of the C-shape. Since the surface has 8,550 voxels, we doubled the sampling in the azimuth and the elevation direction to yield 131,072 samples. If we keep all the transform coefficients and try to reconstruct the nonconvex C-shape, we obtain a reconstructed surface shown in Figure 4-22b. The resulting surface is not close to the original and also has holes in it. The upper lip and lower lip are not reconstructed since the sampling process only detects the closest surface voxel along a sampling ray. Basically, any nonconvex object will be reconstructed as a convex object using the Spherical Mapping method. The difficulty of nonconvex objects is the basis for the developing the Ring Peeling method, which is presented in the next chapter.

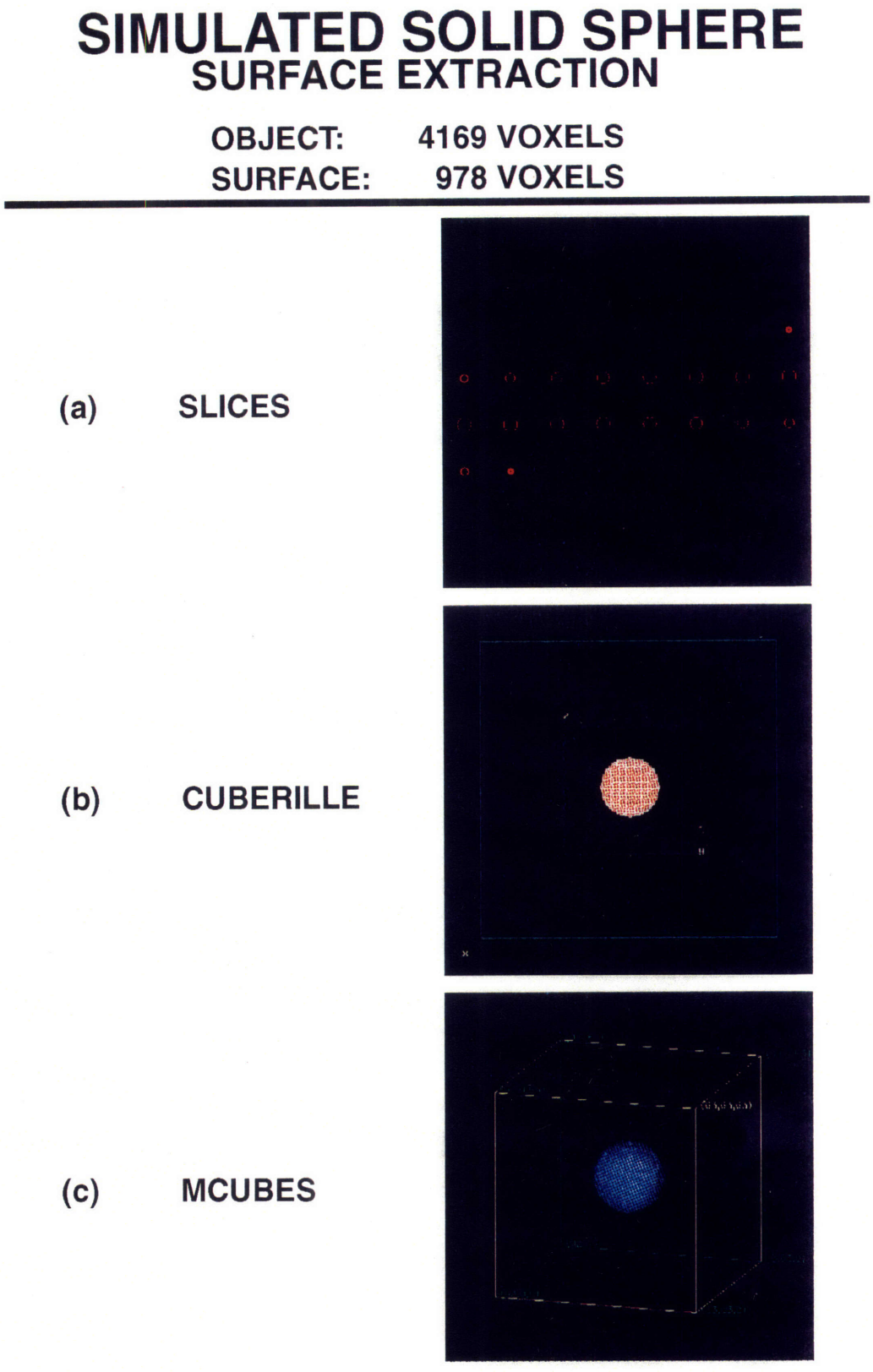
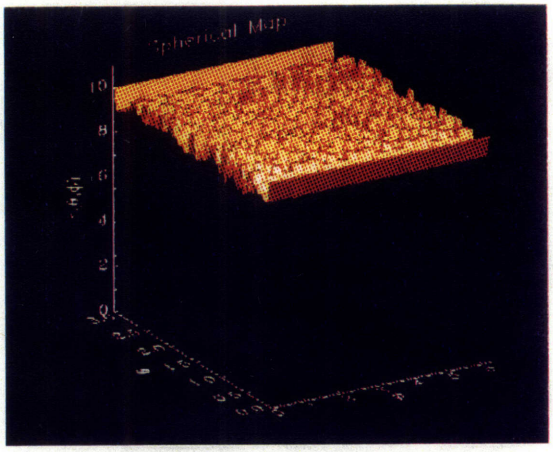


Figure 4-7. Extracted surface of a solid sphere. (a) Cross-sectional slices. (b) Cuberille rendering. (c) Smoothed rendering.

SIMULATED SOLID SPHERE
SPHERICAL PARAMETERIZATION

CENTROID:	(32.0, 32.0, 32.0)	
ϕ RANGE:	0 - 2π (256 BINS)	} 32,768 SAMPLES
θ RANGE:	0 - π (128 BINS)	

(a) SPHERICAL MAP



(b) FOURIER TRANSFORM

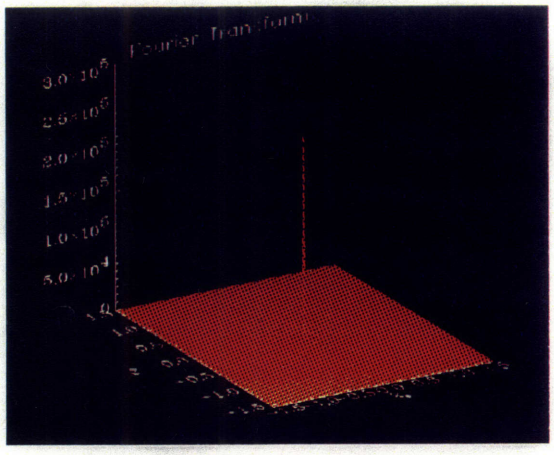


Figure 4-8. **Sampling the surface via spherical parameterization. (a) Spherical map $r(\phi,\theta)$. (b) Fourier transform of map.**

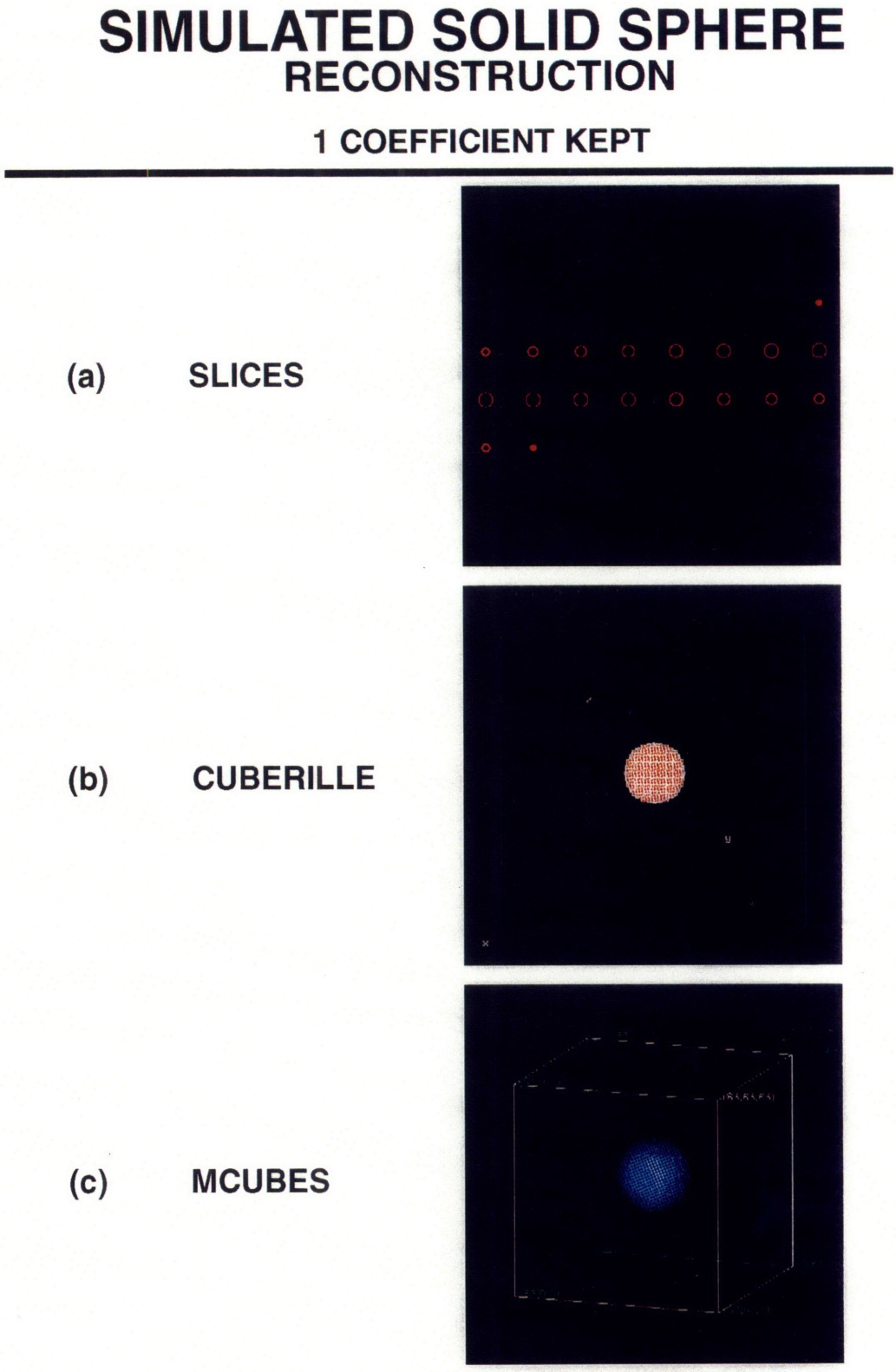


Figure 4-9. Reconstruction of sphere keeping 1 transform coefficient. (a) Cross-sectional slices. (b) Cuberille rendering. (c) Smooth rendering.

SIMULATED SOLID SPHERE
PERFORMANCE RESULTS

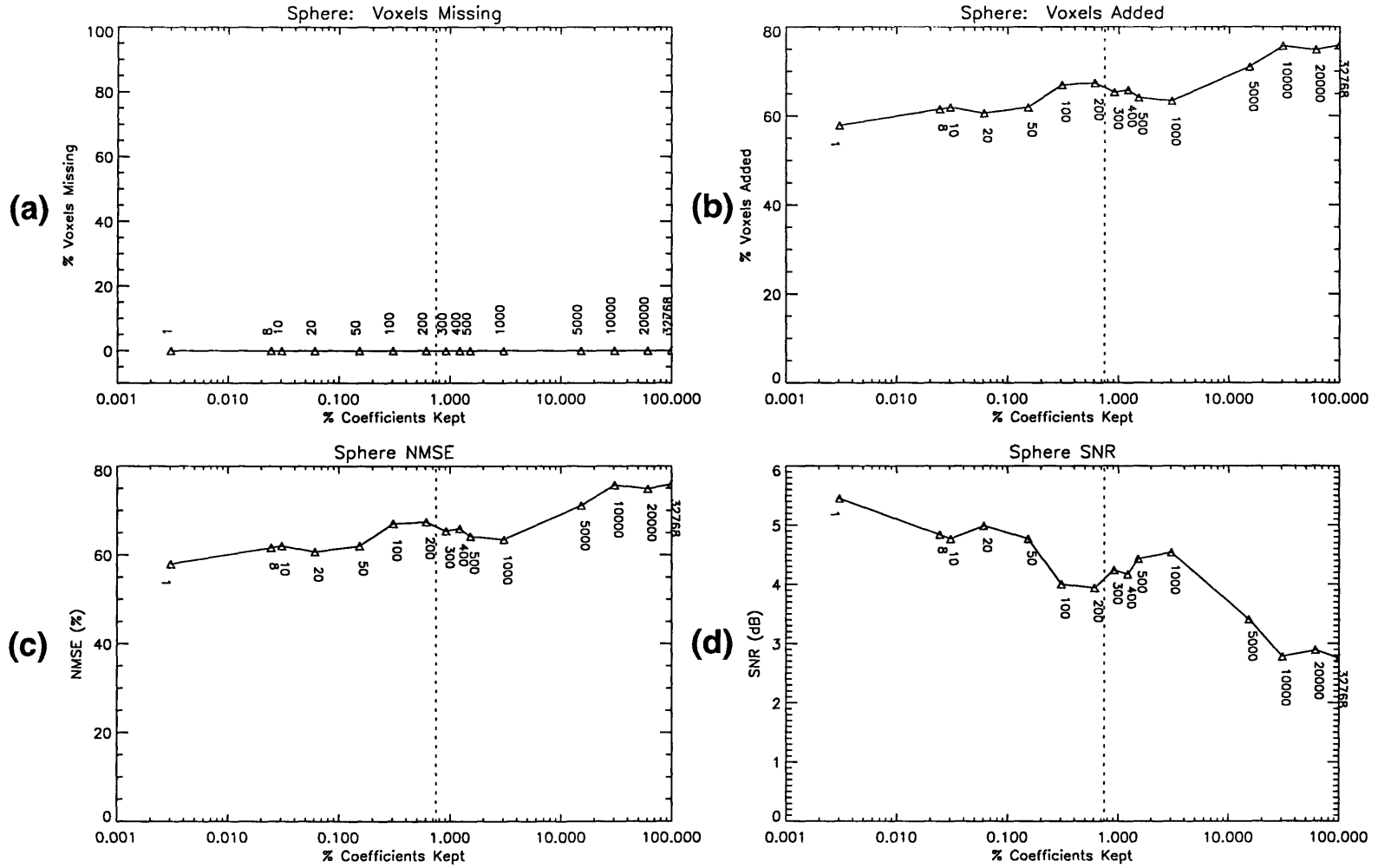


Figure 4-10. **Reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.**

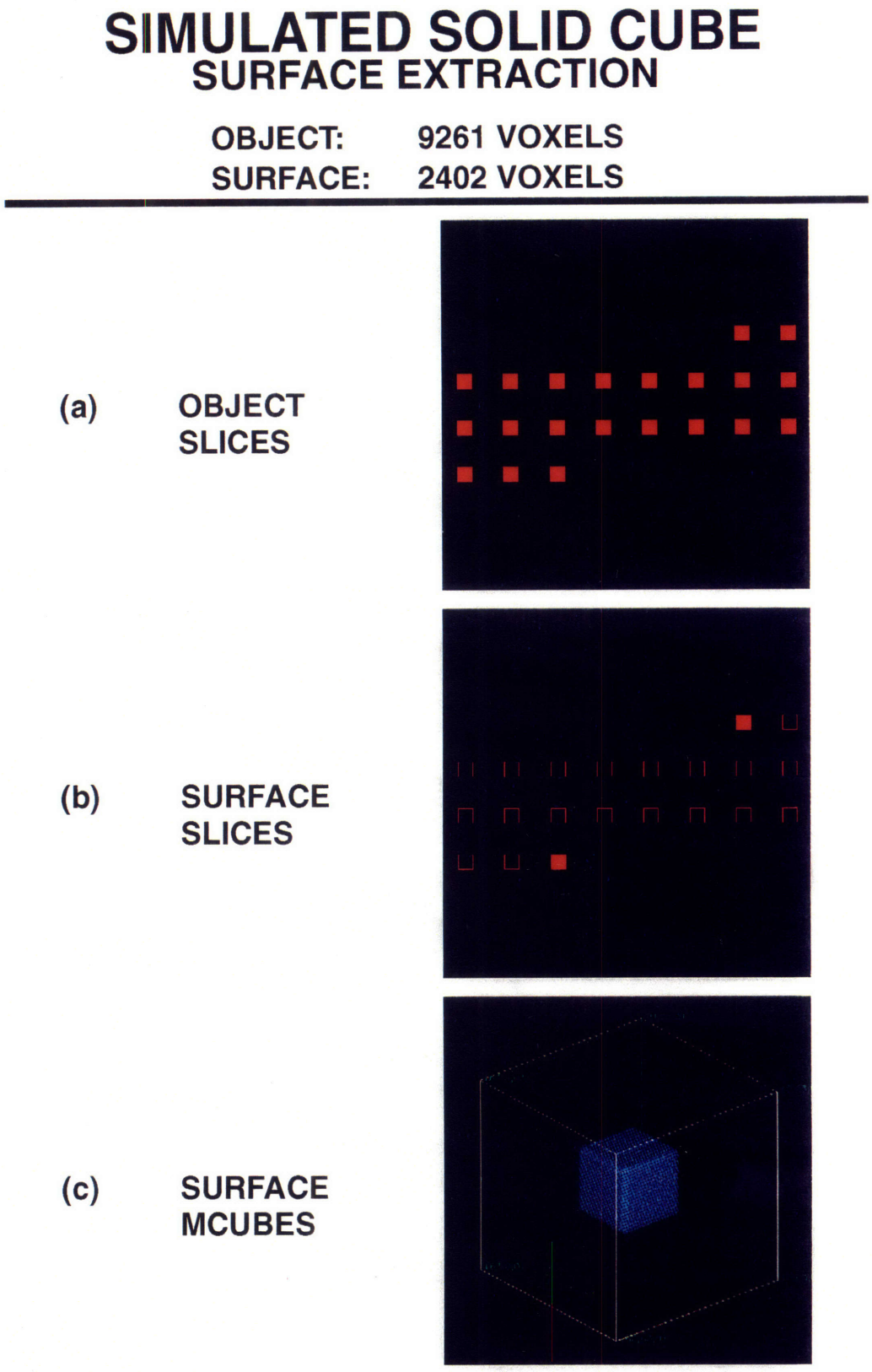
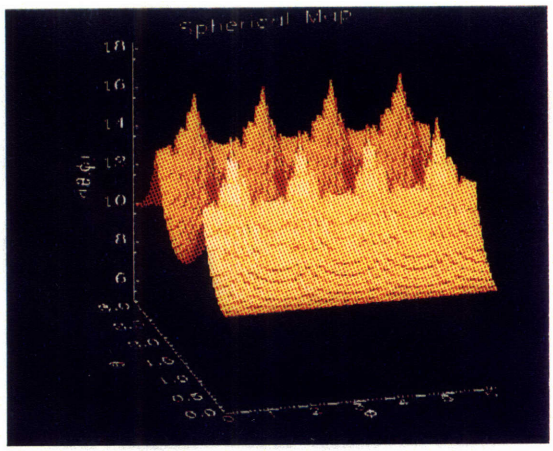


Figure 4-11. (a) Slices of the solid cube. (b) Slices of the extracted surface. (c) Rendered display of the surface.

SIMULATED SOLID CUBE
SPHERICAL PARAMETERIZATION

CENTROID: (32.0, 32.0, 32.0)
ϕ RANGE: 0 - 2π (256 BINS)
θ RANGE: 0 - π (128 BINS)
} 32,768 SAMPLES

(a) SPHERICAL MAP



(b) FOURIER TRANSFORM

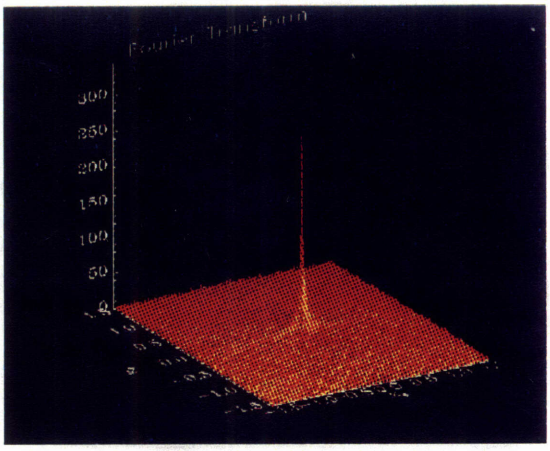


Figure 4-12. Sampling the surface via spherical parameterization. (a) Spherical map $r(\phi,\theta)$. (b) Fourier transform of map.

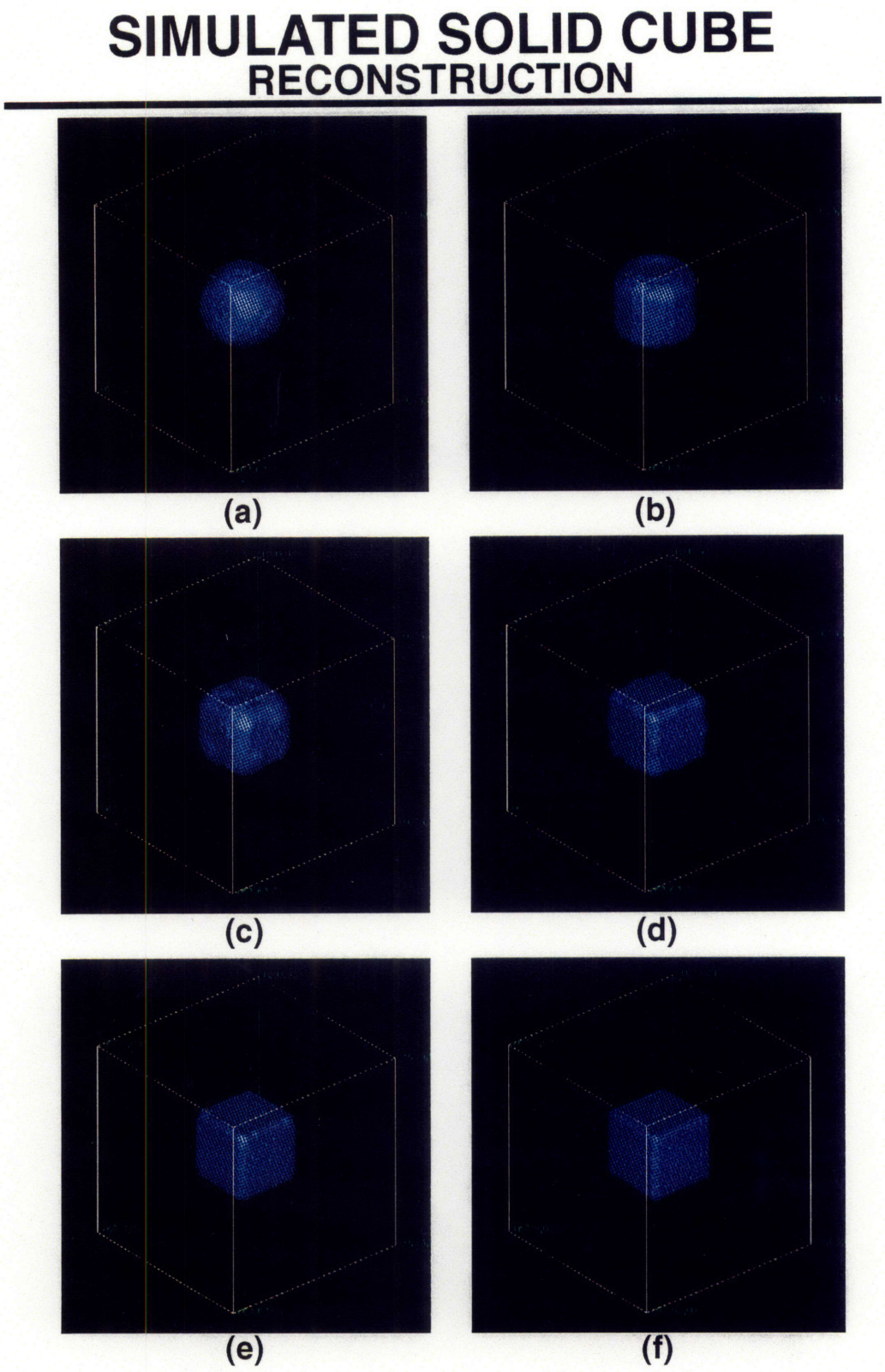


Figure 4-13. Reconstruction when retaining α transform coefficients. α = (a) 1, (b) 5, (c) 10, (d) 50, (e) 100, and (f) 300.

SIMULATED SOLID CUBE
PERFORMANCE RESULTS

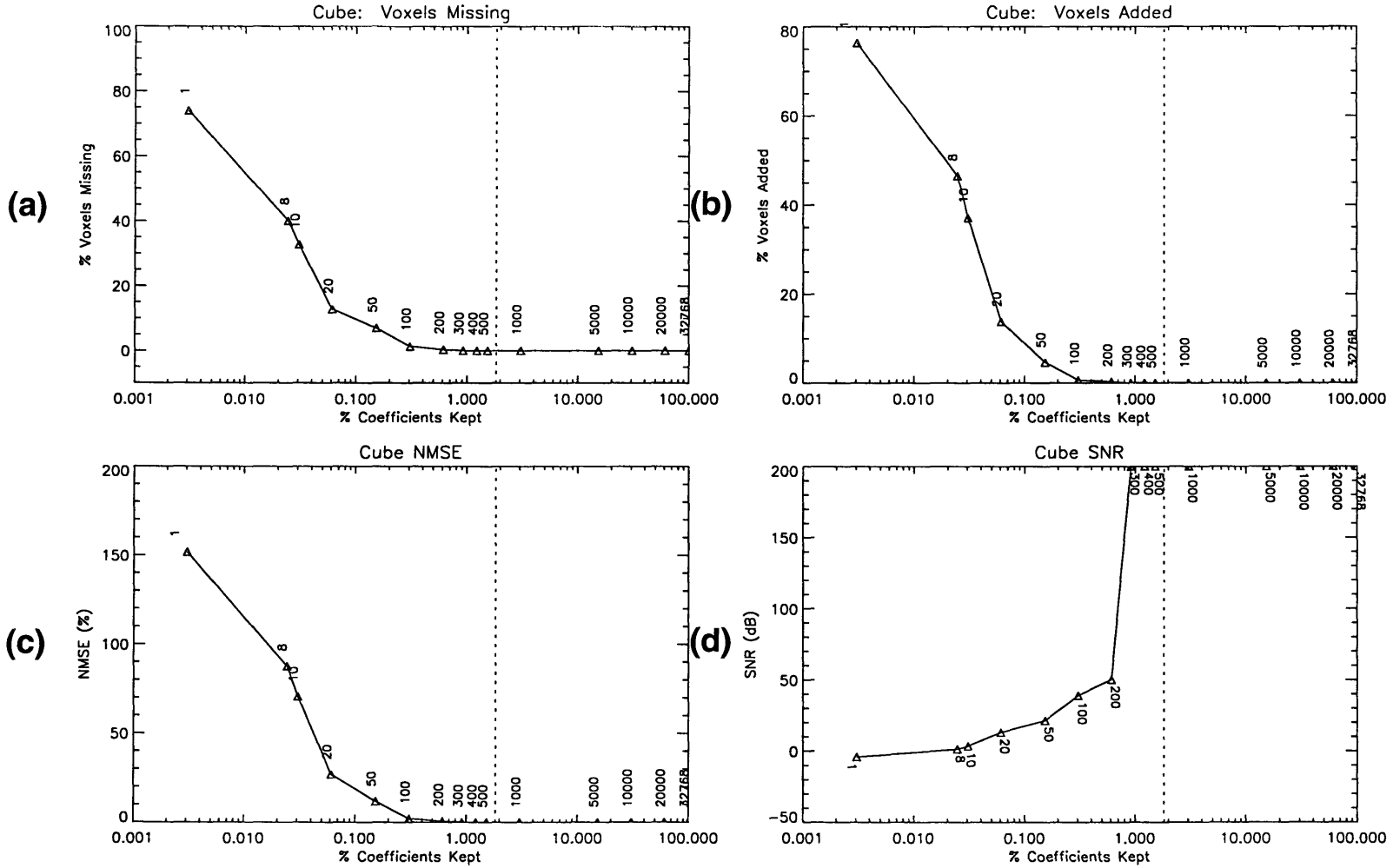


Figure 4-14. Reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.

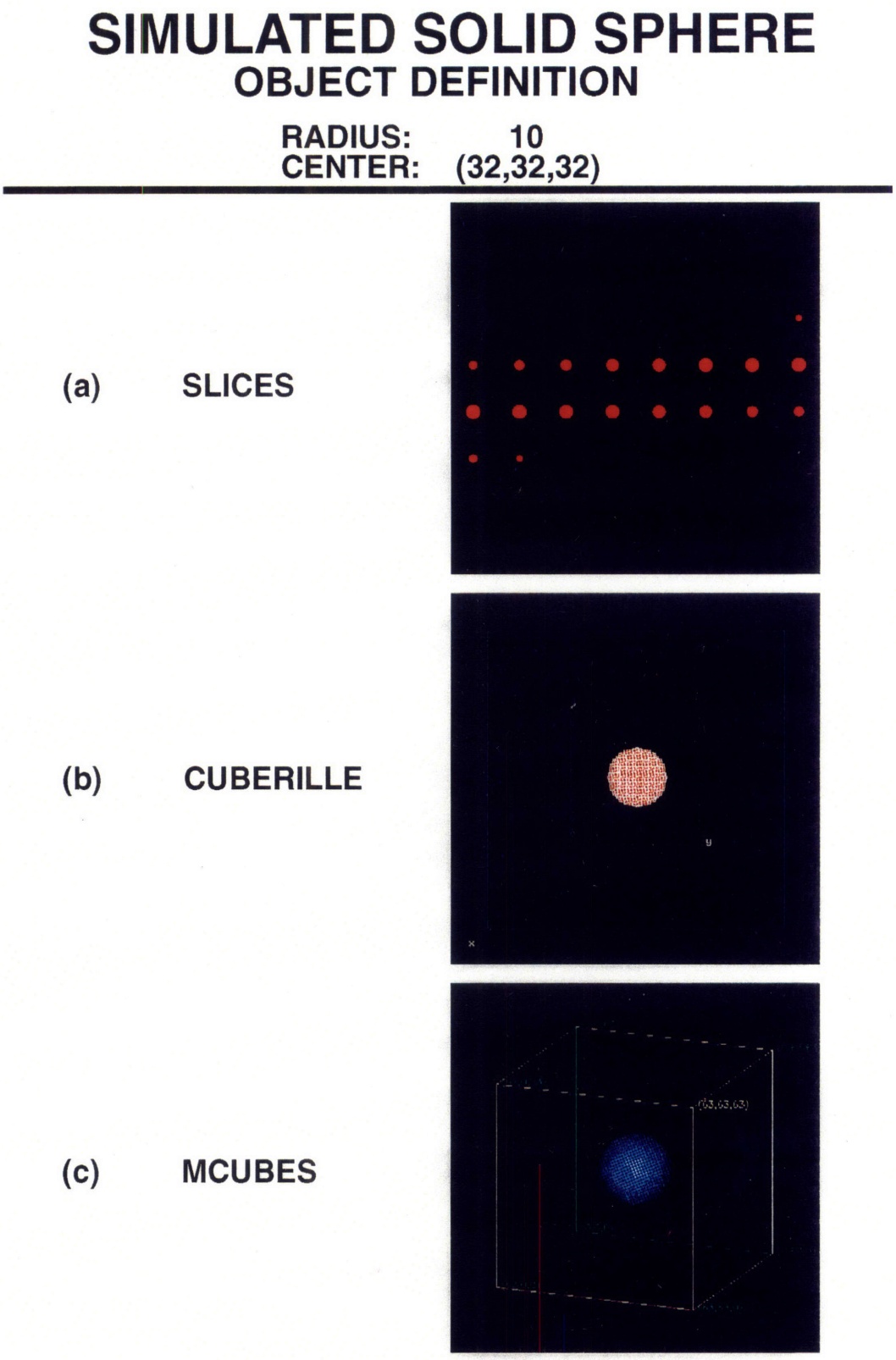


Figure 4-6. Three different ways to visualize a 3D object. (a) Multiple 2D cross-sectional slices. (b) 3D cuberille representation. (c) 3D smoothed representation based on Marching Cubes.

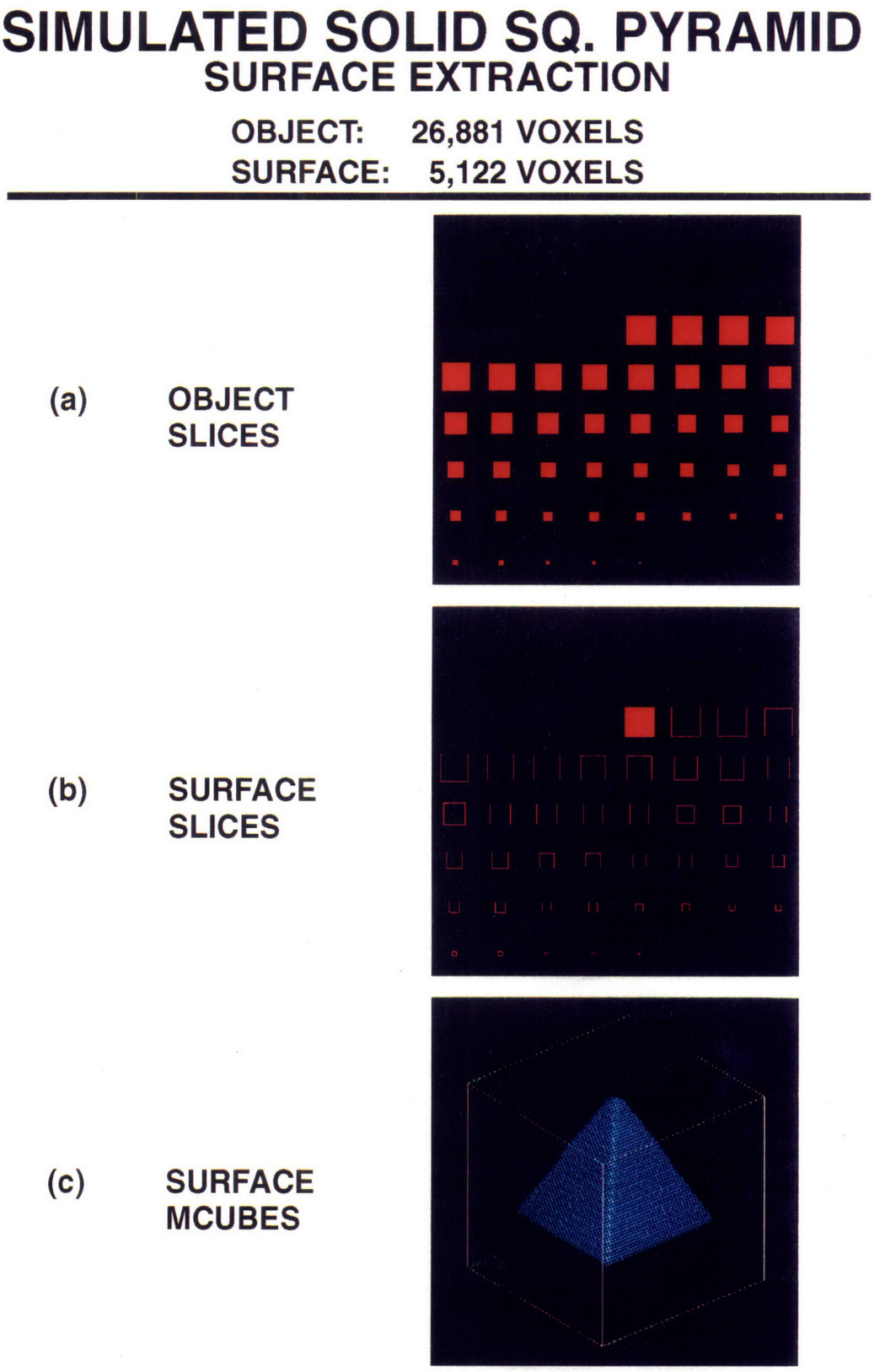
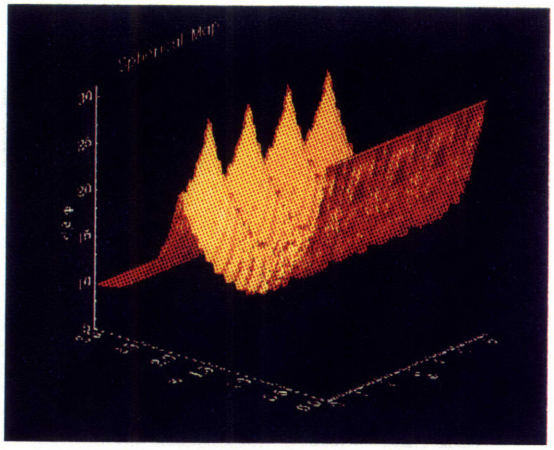


Figure 4-15. (a) Slices of the solid square pyramid. (b) Slices of the extracted surface. (c) Rendered display of the surface.

SIMULATED SOLID SQ. PYRAMID
SPHERICAL PARAMETERIZATION

CENTROID:	(32.0, 32.0, 29.77)	
ϕ RANGE:	0 - 2π (256 BINS)	} 32,768 SAMPLES
θ RANGE:	0 - π (128 BINS)	

(a) SPHERICAL MAP



(b) FOURIER TRANSFORM

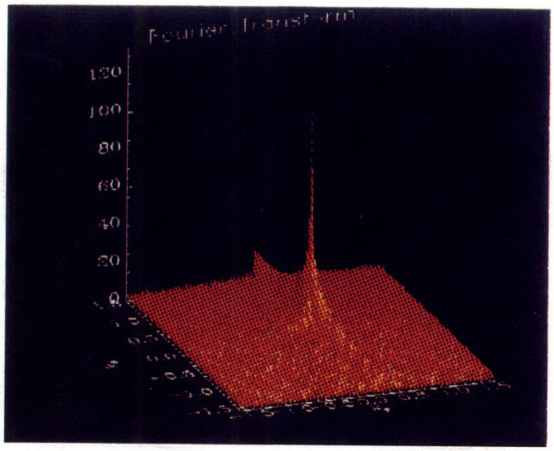


Figure 4-16. Sampling the surface via spherical parameterization. (a) Spherical map $r(\phi,\theta)$. (b) Fourier transform of map.

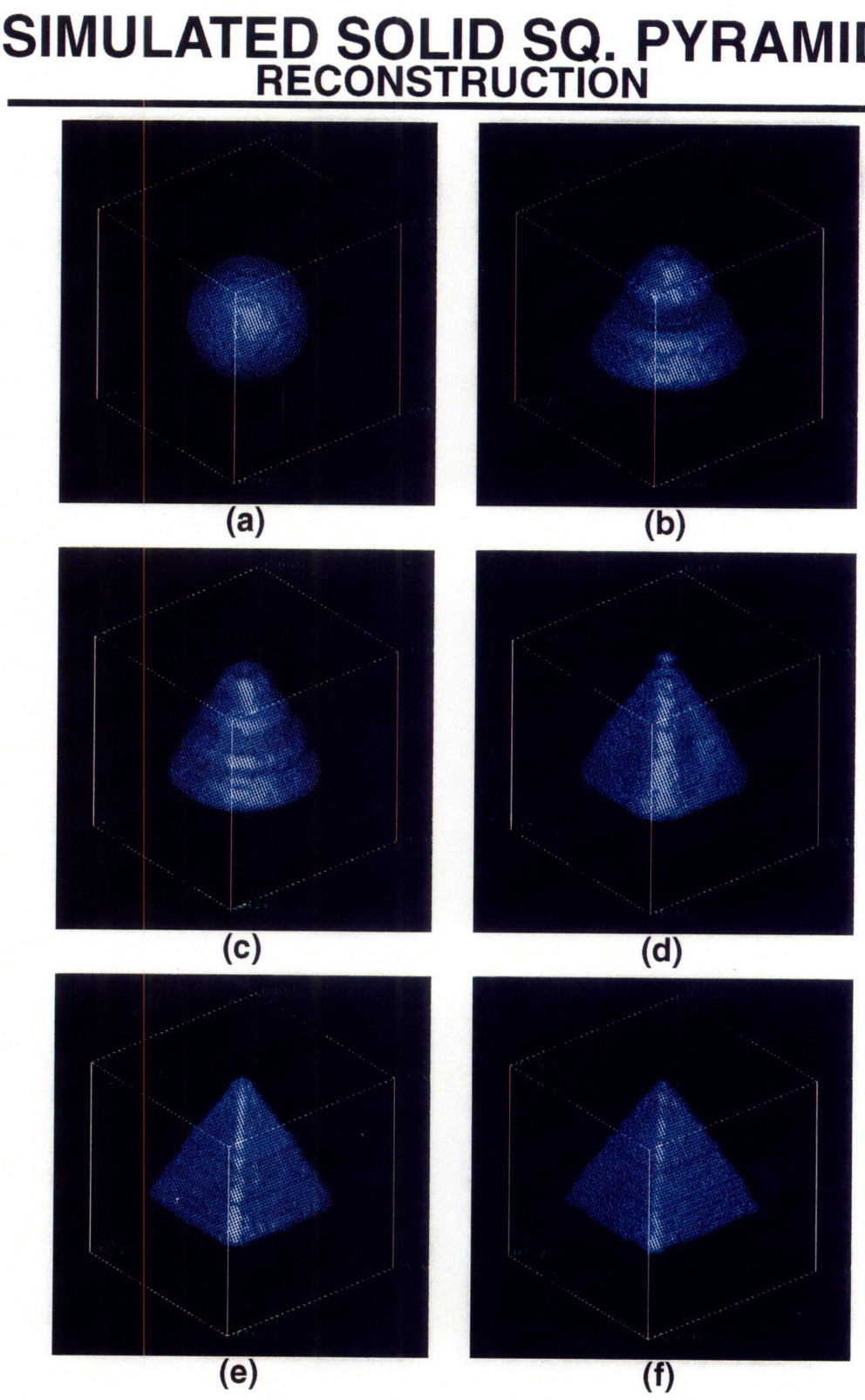


Figure 4-17. Reconstruction when retaining α transform coefficients. α = (a) 1, (b) 5, (c) 10, (d) 50, (e) 200, and (f) 400.

SIMULATED SOLID SQ. PYRAMID
PERFORMANCE RESULTS

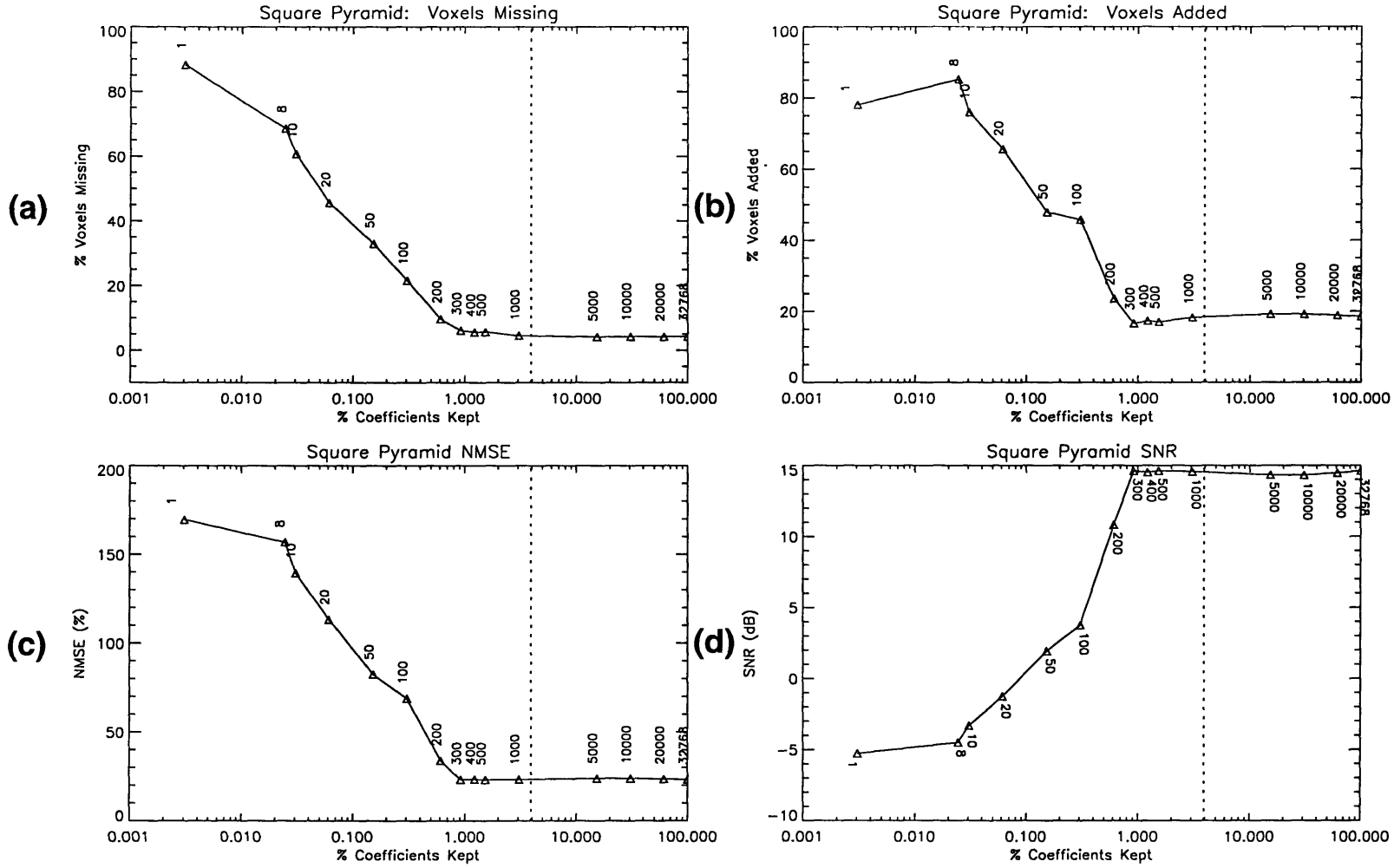


Figure 4-18. **Reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.**

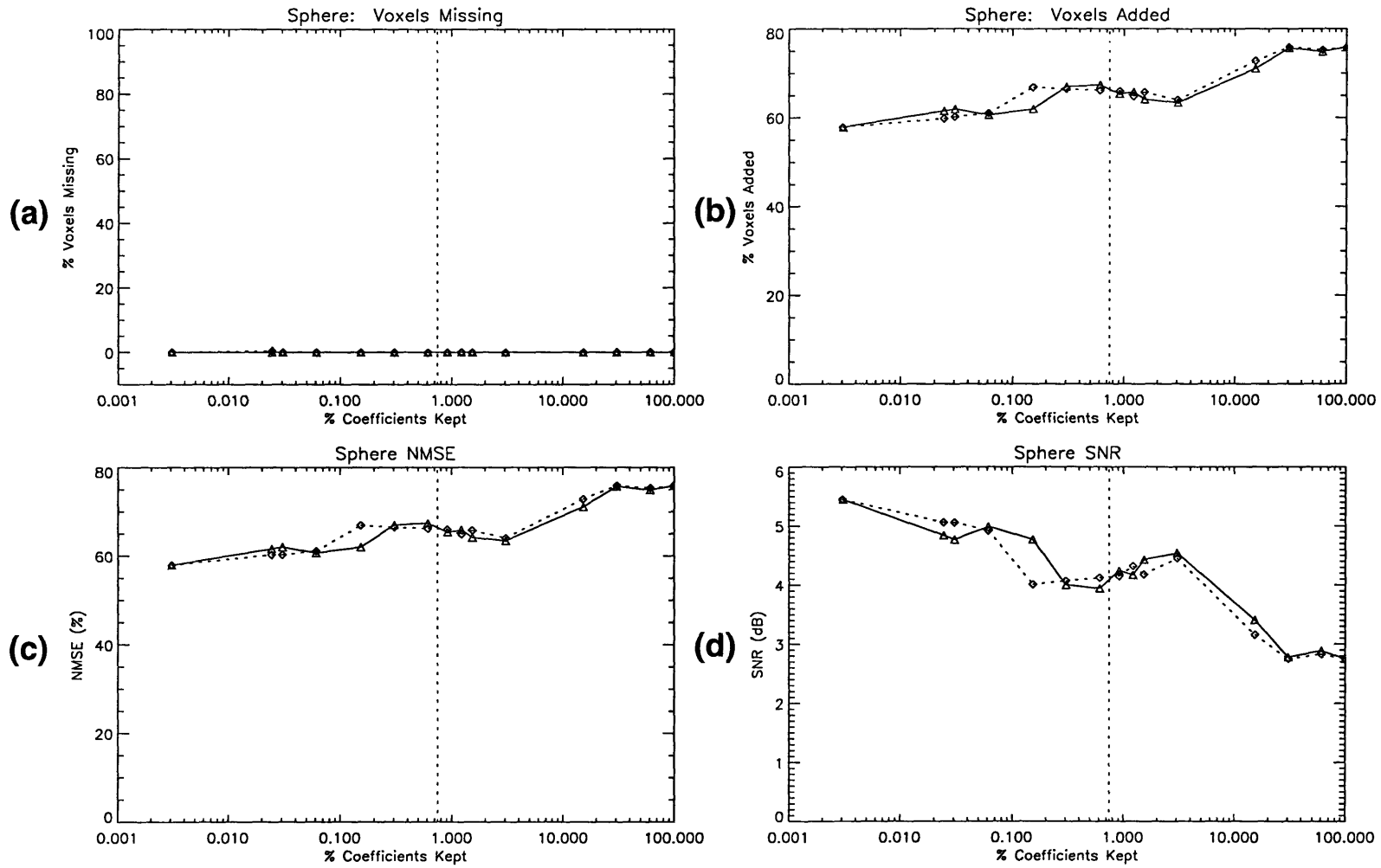


Figure 4-19. **DFT and DCT reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.**

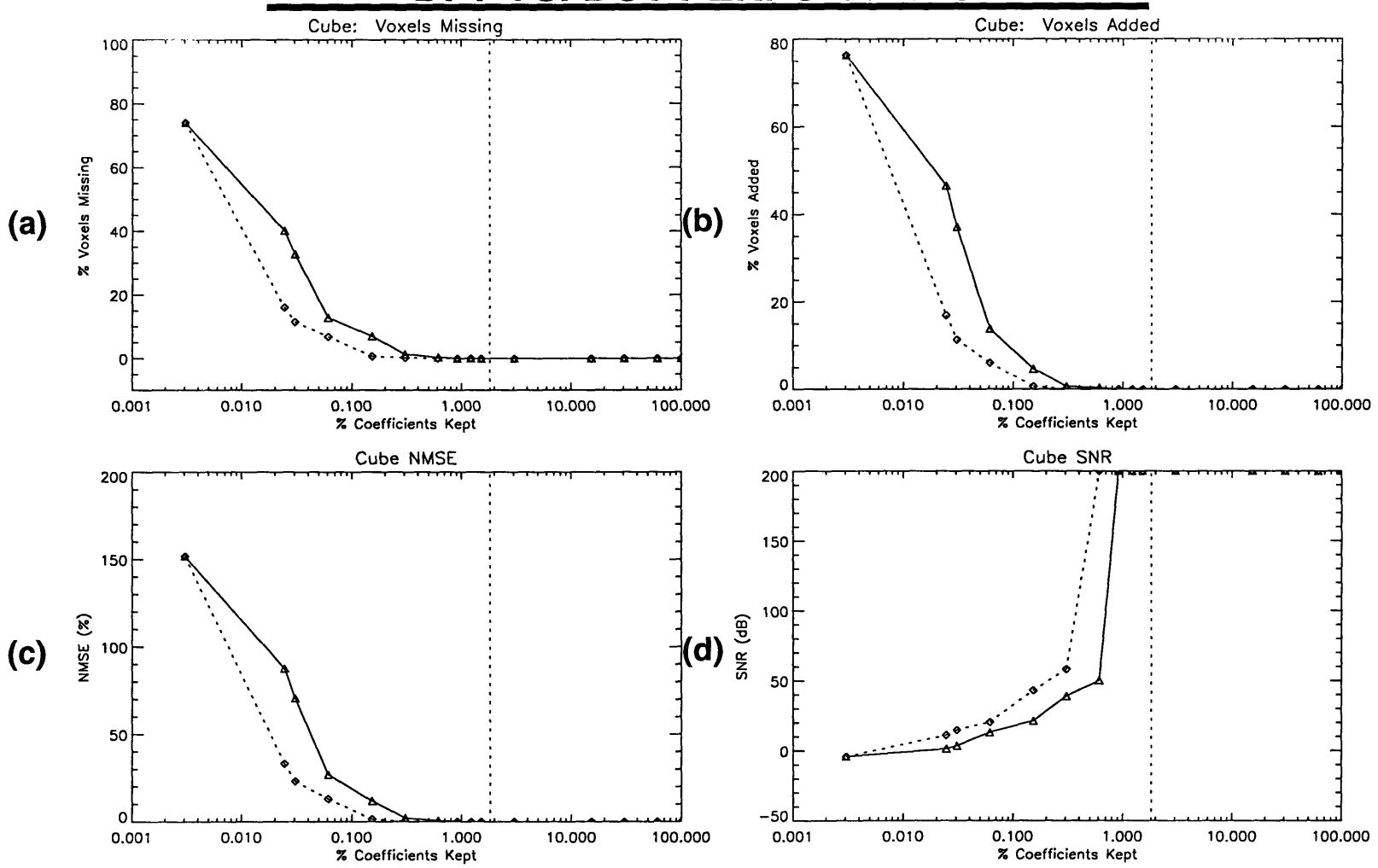


Figure 4-20. **DFT and DCT reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.**

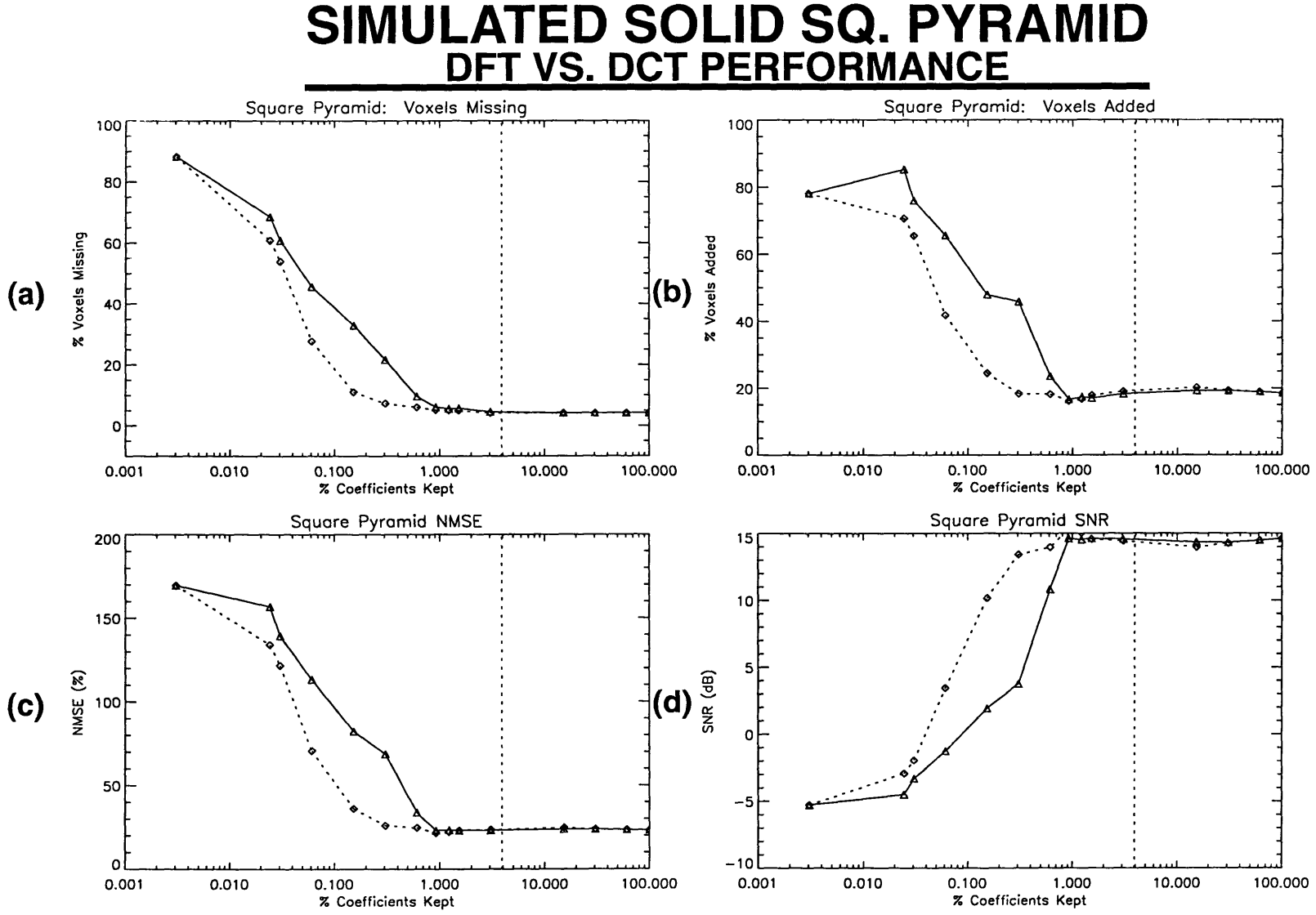


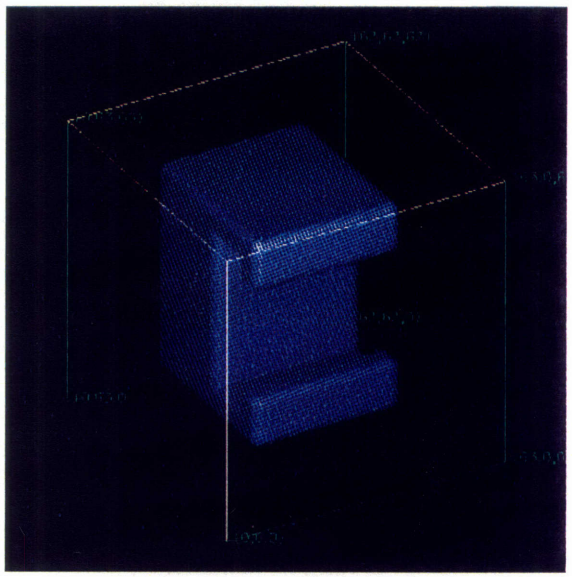
Figure 4-21. DFT and DCT reconstruction performance as a function of percentage ot transform coefficients kept. Percent voxels (a) missing and (b) added to original surface, (c) NMSE, and (d) SNR.

SPHERICAL MAPPING LIMITATIONS
NONCONVEX OBJECTS

C-SHAPE OBJECT

CENTROID:	(31.50, 35.08, 32.00)
SURFACE:	8,550 VOXELS
SPHERICAL MAP:	131,072 SAMPLES (256 x 512)

(a) ORIGINAL SURFACE



(b) RECONSTRUCTED SURFACE

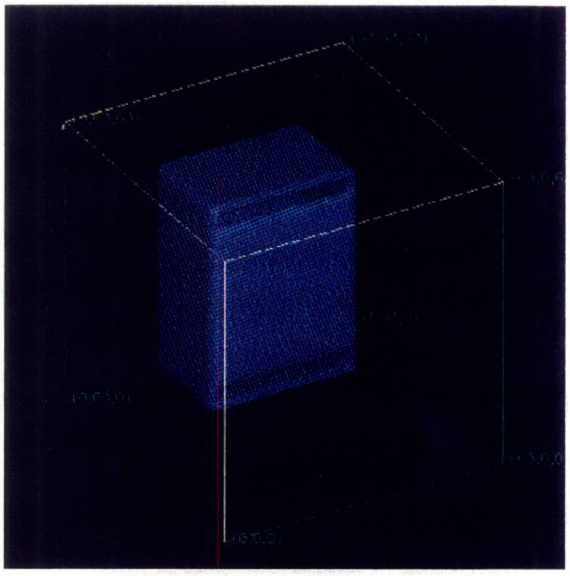


Figure 4-22. Shortcoming of the Spherical Mapping Method.
(a) Original surface of nonconvex C-shape object.
(b) Reconstructed surface keeping all coefficients.

Chapter 5
Ring Peeling Method

Chapter 5

Ring Peeling Method

5.0 Introduction

The inability of the Spherical Mapping method to compress surfaces of nonconvex objects restricts an enormous number of surfaces that can be compressed. This limitation is the impetus for developing the Ring Peeling method. To represent the surface voxels of a nonconvex object, we need to find an ordered traversal of the 3D surface. Unlike in 2D where there is a unique ordered traversal of 2D curves, there are infinitely many traversals of a general closed surface. The Ring Peeling method presents one ordered traversal scheme that results in a data structure conducive to 3D chain-code compression.

Consider, as an example, an object that is an apple. Figure 5-1 shows a cartoon of the Ring Peeling algorithm. The aim of the Ring Peeling method is to obtain the peel of the apple so that the peel can be compressed. Ideally, we would like to retrieve one nice, continuous peel. Practically, as it often happens with a real apple, we often encounter breaks as we peel. As a result, we obtain many peel segments. Nevertheless, once we have the peel that possibly contains several segments, the peel structure is conducive to 3D chain-coding since elements within the peel segments are contiguous and can be related to each other. This relation allows us to represent these elements, or surface voxels, with a chain-code scheme that requires less amount of space than recording the (x,y,z) values for each of these surface voxels.

The mathematics in the Ring Peeling method can be visualized as in the box of Figure 5-1. Given a starting surface voxel for the Ring Peeling algorithm, a surface S is decomposed into adjacent rings through an iterative process labelled ring assignment. Rings are spliced and then connected in such a way to obtain a peel. This gluing process consists of applying adjacency and connectivity rules on the surface voxels in the rings to form the peel. The peel structure is then sent to a 3D chain-code compression scheme.

In this chapter, we will first explain the different parts of the Ring Peeling method – ring assignment, adjacency/connectivity rules in rings-to-peel, and 3D chain-coding. We will then discuss the amount of compression and storage space reduction associated with this method. Simulated data and obtained results will be presented and discussed. Also, a real-world application is presented. The surface of a head extracted from an MRI data set will be compressed using the Ring-Peeling method.

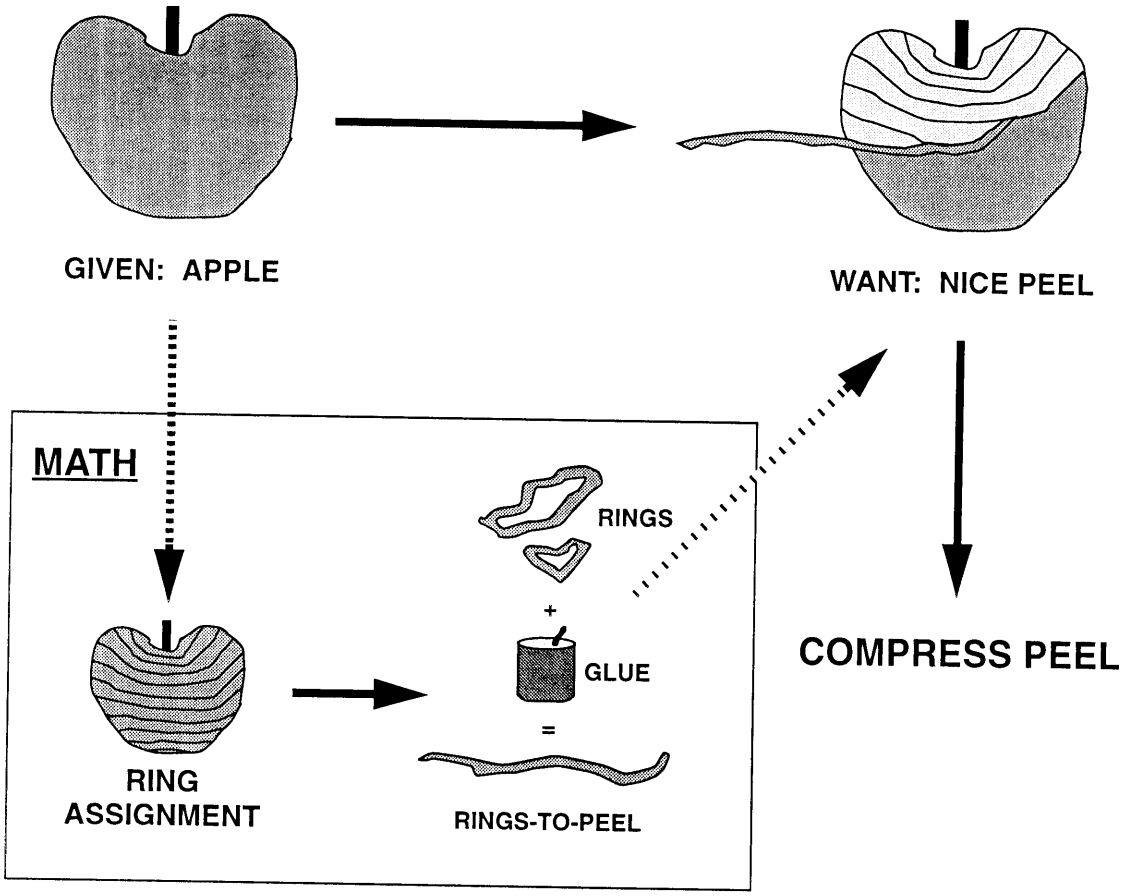


Figure 5-1. Conceptualization of the Ring Peeling method, where surface voxels are organized into a peel structure.

5.1 Ring Peeling

Figure 5-2 shows the section of the compression system diagram shown in Figure 2-3 corresponding to the Ring Peeling method.

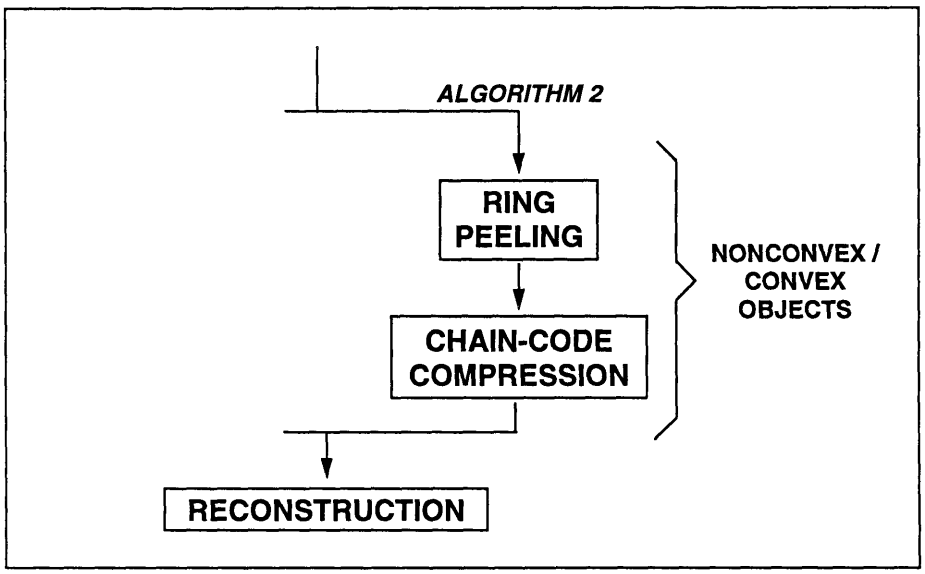


Figure 5-2. Block Diagram of Ring Peeling method.

Algorithm 2, the Ring Peeling method, takes as input a 3D surface, S. In contrast to the Spherical Mapping method, S does not have to be a closed surface, nor does S have to be star-shape-convex. The Ring Peeling method, however, requires a starting surface voxel (x_s,y_s,z_s) for the algorithm to begin.

5.1.1 Ring Assignment

Given a starting surface voxel, the first step in the Ring Peeling method for compression is to decompose the surface into rings. Surface voxels are assigned to different rings through an interative process as outlined in Figure 5-3. The following notation will be used to reference rings and their elements. Mathematically, a ring, R, is defined as a set of surface voxel elements r_n's that have been assigned to the ring:

$$R = \{r_1, r_2, \ldots, r_m\} \tag{5.1}$$

and m is the number of surface voxels in the ring. Since a surface is decomposed into multiple rings, we refer to the k-th ring as R^k, and the n-th element in the k-th ring as r_n^k. The k-th ring, R^k, is then the set of surface voxel elements that have been assigned to that ring:

$$R^k = \{r_1^k, r_2^k, \ldots, r_n^k, \ldots, r_{m_k}^k\}, \qquad k = 1, 2, \ldots, l \tag{5.2}$$

and l is the number of rings which the surface has been decomposed into. Using these notations, the surface S can be expressed as a union of the decomposed rings:

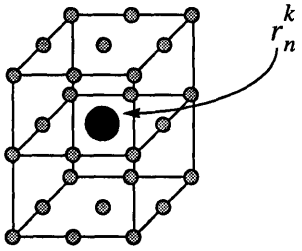
$$S = \bigcup_{k=1}^{l} R^k \quad . \tag{5.3}$$

The step-by-step ring decomposition of a surface can be worded as follows. Take a starting surface voxel and assign it to ring 1. For the single element in ring 1, find all the nearest neighbor voxels that are surface voxels. Since there are only 26 nearest neighbors for every voxel, the number of neighboring surface voxels is always less than 26. If these special neighbors have not been assigned to a ring, assign it to ring 2. Note that ring 1 will always contain a single element only. Since all surface voxels have not been assigned, we now consider all the elements in ring 2. For each element in ring 2, find the nearest neighbors that are surface voxels. Assign these neighbors to the next ring, ring 3, if they have not been earlier assigned. Continue this process until all the surface voxels have been accounted for in this assignment process.

STEP 1: PICK A STARTING VOXEL. ASSIGN IT TO RING 1 (k=1).

STEP 2: FOR EACH ELEMENT r_n^k IN RING R^k

i. FIND THE NEAREST NEIGHBORS THAT ARE SURFACE VOXELS



ii. IF THESE NEIGHBORS HAVE NOT BEEN ASSIGNED TO A RING, ASSIGN THEM TO RING R^{k+1}

STEP 3: IF ALL SURFACE VOXELS HAVE BEEN ASSIGNED, END. OTHERWISE, INCREMENT k AND GO TO STEP 2.

Figure 5-3. Algorithm for assigning surface voxels to rings.

5.1.2 Adjacency/Connectivity Rules

Once the rings of a surface have been obtained, the elements within the rings and between the rings are related after the application of adjacency and connectivity rules. We find the relation between elements within the rings and also across rings. With these relations, we form an ordered set of ring elements which we call the peel data structure *P*.

The rearranging and ordering of the elements in the rings R^1 through R^l is best described as follows. We first take the only element in R^1 and place it as the first element in an ordered set *P*. Since there are no remaining elements in R^1, we consider the next ring R^2. We find the element in R^2 which has the closest distance in space to the last element in the set *P*. This element is moved from R^2 and placed in the next position in *P*. We continue to move the remaining elements in this fashion from the current ring R^2 to the ordered set *P* until no element remains in R^2 . When this occurs, the next ring R^3 is then considered and the process of transferring elements to *P* continues. The entire process stops when all the rings, or equivalently, all the surface voxels, have been accounted for.

Figure 5-4 shows an example of the reordering process and the generation of the peel structure *P*. Sample rings 1 and 2 are shown with their assigned elements obtained from the ring assignment process. The arrows show the stepwise ordering of the elements in the rings, beginning with the starting voxel in ring 1. The type of arrow tells us the spatial relation of two ring elements. Arrows with solid lines indicate that the two voxels are adjacent in space, whereas arrows whose lines are interrupted by a pair of parallel lines indicate that the voxels are not adjacent in space. The peel structure *P* is obtained essentially from following the arrows in the ring diagram. Note that the peel structure consists of the same elements contained within the rings, except the elements have been ordered. The elements in the peel structure are categorized as either segment leaders or segment chain elements. A segment leader is a peel element whose previous peel element is not adjacent in space. Since the first peel element contains no prior peel element, we also call it a segment leader. A segment chain element is a peel element whose previous peel element is adjacent in space. Together, a segment leader and the segment chain elements that follow before the next segment leader is encountered forms a peel segment. Therefore, the peel structure contains several peel segments. We refer to the number of peel segments in the peel structure as *p*.

In each peel segment, every segment chain element is adjacent to the previous element which is another segment chain element or a segment leader. Traditionally, we would record the *(x,y,z)* location for every surface voxel. However, because segment chain elements are related to previous elements, we only need to record the *(x,y,z)* location for the first element in the segment, the segment leader. Every segment chain element can then be represented with a short code representing its relation to the previous element. If the chosen code requires less space than recording the *(x,y,z)* for the segment chain element, then we can use the code to achieve compression. One code that incorporates the relation of any surface voxel to a neighboring surface voxel is the 3D chain-code.

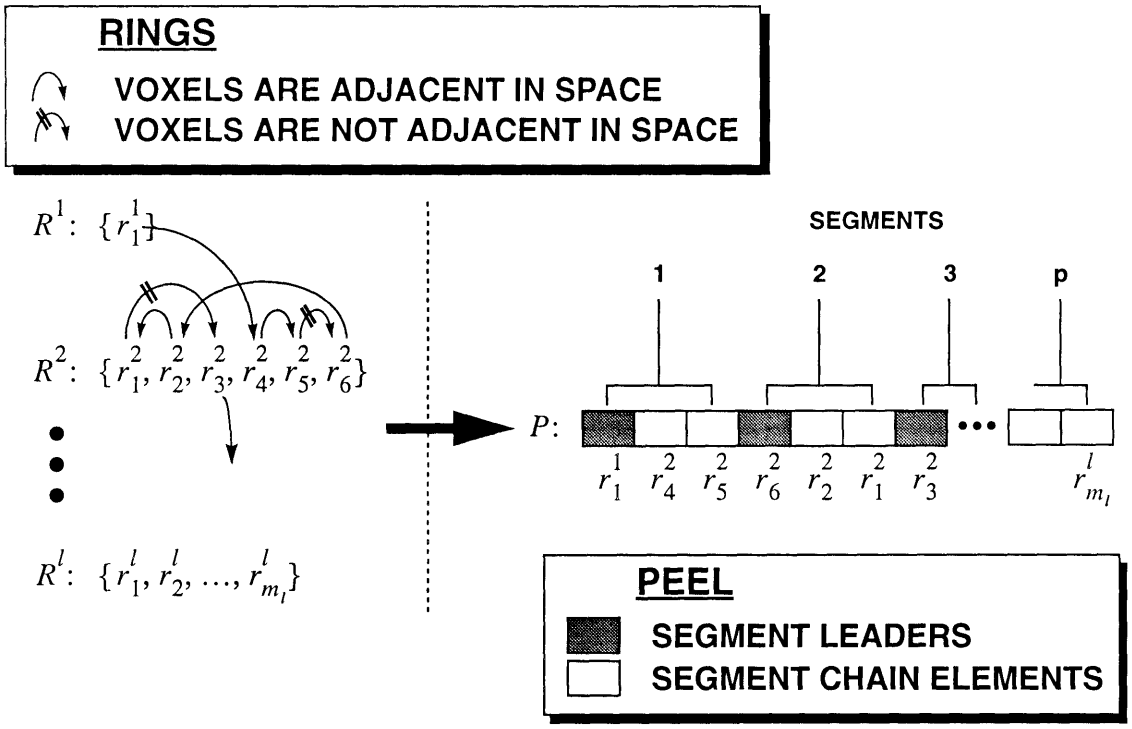


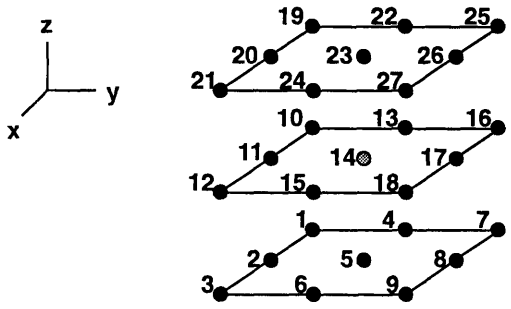
Figure 5-4. Conversion of rings into peel structure.

5.2 Compression

Once a surface has been iteratively decomposed into adjacent rings, and after the rings have been spliced and ring elements reordered into a peel structure, we can compress the peel structure efficiently using a 3D chain code. The basis for this method for compressing the amount of space needed to represent the elements in the peel structure, thus the surface, is that any segment chain element is always adjacent in space to its previous element. This particular nature of the peel structure is conducive to chain-coding via the 26-neighbor connectivity code. The resulting chain-encoded peel structure is very compact and gives an accurate representation of the surface.

5.2.1 3D Chain Coding

The 3D chain code is essentially an extension of chain coding in 2D. Figure 5-5 shows the 26-neighbor connectivity code that relates a segment chain element to the previous element in the peel structure. This code tells us how one voxel is related to an adjacent voxel in space.



CODE = 9 (Δz + 1) + 3 (Δy + 1) + (Δx + 1) + 1

RANGE: [1,27] EXCEPT 14

Figure 5-5. 26-Neighbor connectivity code for 3D chain-coding.

The code is calculate by taking the differences in x, y, and z between the two voxels in concern, and then the differences are multiplied by a factor, slightly modified, and finally summed. We have chosen such a definition such that the range of the code lies between integers 1 and 27, inclusive. Note that there exists only 26 possible directions for which two voxels adjacent in space are related — hence the name, 26-neighbor connectivity. Code 14 cannot exist because two distinct voxels cannot have the same (x,y,z).

5.2.1 Encoding the Peel

Using the 3D chain code, we encode the peel structure in Figure 5-4 as follows. Every segment leader is still represented by recording the (x,y,z) location. Each segment chain element, on the other hand, is represented by recording the assigned 3D chain-code value. The peel structure, both before and after encryption, is shown in Figure 5-6. We denote the unencoded peel

structure as *P*, and the encoded peel strucutre as *E(P)*.

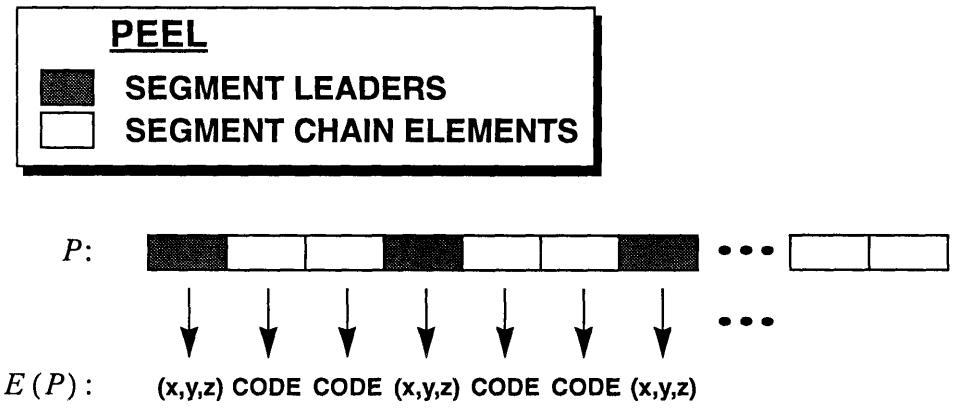


Figure 5-6. Peel structure before and after encryption using the 26-neighbor connectivity code.

We denote the unencoded peel structure as *P*, and the encoded peel structure as *E(P)*.

5.3 Reconstruction

In the Ring Peeling method for surface compression, reconstruction is also a straightforward reverse process. Reconstruction entails taking the encoded peel structure *E(P)*, decoding it to form a regular peel structure *P*, and then simply generating the surface *S'* from *P*. Note that the generated surface *S'* is identically the same as the original surface *S* since the Ring Peeling method we have outlined is virtually lossless. We can, however, make the Ring Peeling method a lossy compression algorithm by throwing out short segments within the peel structure, thus throwing out few surface voxels. It is also worthwhile to point out that with respect to computation expense, the decding process in the Ring Peeling method essentially involves indexing a lookup table, which can be done rather fast.

5.4 Data Analysis

This section will present the results of the Ring Peeling method on simulated data sets as well as on a real MRI data set of a human head. Compression will be defined by the number of bytes required to store the encoded peel data structure associated with a surface. Since the Ring Peeling method is a lossless surface compression algorithm, we need not use the performance measures defined in the analysis of the Spherical Mapping method.

5.4.1 Characteristics of Data Sets

The two simulated data sets of a solid cube and a solid C-shape nonconvex object , both previously defined in the analysis of the Spherical Mapping method, are used here to analyze the performance of the Ring Peeling method. Additionally, a third data set containing real MRI data of a human head is also used in our analysis. Particularly, the surface corresponding to the outer skin of the head is our chosen surface of interest. Again, for simplicity, the three spatial dimensions for any data set has been set to be of equal length.

Graphical visualization of the surfaces is accomplished with the same methods described in the Spherical Mapping method. The choice of 2D and 3D display methods will be used selectively in order for us to best illustrate our point.

5.4.2 Compression and Storage Size

As in the Spherical Mapping method, we address the similar question: When does the Ring Peeling method for surface compression outperform conventional methods for storing surfaces? We define N as the number of voxels in each spatial dimension, and s as the number of surface voxels in the object of interest. In terms of the peel structure, p is the number of peel segments in the encoded peel $E(P)$, and m_k is the number of elements in the k-th peel segment.

If we are to store the surface voxels of an object using conventional methods, we require

$$min\left[\frac{N^3}{8}, 3s\right] \tag{5.4}$$

bytes. To store the surface using the Ring Peeling method, we require

$$\sum_{k=1}^{p} 6 + (m_k - 1) + 1 = s + 6p \tag{5.5}$$

bytes. This formula is obtained by noting the fact that for each peel segment k, we store the (x,y,z) location of the segment leader, chain codes for m_k -1 number of segment elements, and a header or trailer to demarcate a segment. We assume that the (x,y,z) location of the segment leader are integers, thus requiring 6 bytes. Since the chain code for each segment chain element only ranges

from 1 to 27, we conservatively estimate that it requires one byte for every segment chain elements. Lastly, we allocated one byte for a segment marker. The left-hand side of equation (5.5) can be simplified to yield the right hand side since the summation of the number of elements over all segments gives s, the number of surface voxels in the object of interest.

In order for the Ring Peeling method to outperform conventional storage methods, equation (5.5) must be less than equation (5.4). This occurs when

$$p < \frac{s}{3} \quad , \qquad (5.6)$$

where the number of segments is less than one-third the number of surface voxels. Ideally then, we obtain the best performance when there is only one continuous segment in the peel structure. If we divide equation (5.5) by equation (5.4), the bound on performance is a compression level of approximately 33% over conventional methods, since $6p$ is usually small relative to s.

5.4.3 Solid Cube

Figure 5-7 shows the convex surface of the simulated solid cube introduced earlier. In Figure 5-7a, the surface is displayed with a cuberille representation. In Figure 5-7b, the surface is shown after the ring assignment process is completed. The colors are used to visualize the different rings of the cube. The color scale runs as black-blue-green-yellow-red, where black corresponds to the first rings (or the starting voxel in the assignment process) and latter colors correspond to higher order rings that are later assigned. For this particular case, the starting voxel was picked as the bottom corner in the back of the cube. Voxels of a particular color lie in the same ring. As a result, we see that the rings for the cube slowly envelope the cube from the back and ends in the front, at the corner closest to the reader. Figure 5-8 shows the surface after ring assignment, rotated at different angles about the z-axis.

Individual rings are displayed in Figure 5-9. Note that the elements in a ring are not necessarily coplanar. Instead, these three dimensional rings can be complex contours in space. The lower level rings contain the voxels that are assigned in the early stages of the iterative ring assignment process while higher level rings contain the voxels later assigned.

After the rings are obtained, adjacency and connectivity rules are applied to form the peel structure. The images in Figure 5-10 show the peel structure after different number of rings have been processed. In Figure 5-10a, two rings have been processed. The green voxel marks the first element while the red voxel marks the last element in the segment. Essentially, a segment graphically consists of a green voxel, followed by connected lines that depict the direction of the next voxel, and lastly, a red voxel. From Figure 5-10c, ten rings have been processed but we continue to maintain only one segment in the peel structure. However, in Figure 5-10d, we see multiple red-green voxel pairs corresponding to different segments within the peel. In Figure 5-10f, all forty rings of the cube have been processed and the peel structure shown accounts for all the surface voxels.

Figure 5-11 shows the compression results using the Ring Peeling method. For the cube, the compression ratio is 33.75%, close to the upper performance bound of 33%. Figure 5-11a shows the number of surface voxels assigned to the forty separate rings of the cube. Figure 5-11b shows the statistics for the resulting peel structure. Forty rings have been processed to form a peel structure consisting of only five peel segments. Note, in particular, that segments 2 through 4 are very inefficient since they contain very few segment chain elements.

5.4.4 Solid NonConvex C-Shape

Figure 5-12a shows the cuberille representation of the surface of the nonconvex solid C-shape which the Spherical Mapping method was incapable of representing accurately. Figure 5-12b shows the same surface but colored to identify the rings obtained after ring assignment. Here, the starting point of the algorithm used for the Ring Peeling method is the center of the lower lip of the C. The rings gradually wrap the lower lip, cover the lower horizontal section, up the vertical bulk, over the upper horizontal section, and lastly, envelop the upper lip. Figure 5-13a through c shows the ring assigned surface at various rotations about the z-axis.

For the C-shape object, the surface is decomposed into 72 rings. Figure 5-14 shows images of different individual rings. Again, these rings are geometrically complicated in space, are adjacent to other rings, and together span the entire surface of the object. Applying adjacency and connectivity rules convert the rings into the progressive peel structures shown in the images of Figure 5-15. Note that, overall, the Ring Peeling method accounts for most voxels with graphical lines, or equivalently chain codes, rather than by their cartesian spatial location as is true for segment leaders.

Statistics for compression using the Ringe Peeling method is shown in Figure 5-16. Again, the 34.08% compression is close to the upper performance bound of 33%. In Figure 5-16b, we again see the inefficiency of short segments in the peel structure. We can, alternatively, make the Ring Peeling method lossy by discarding these segments in the peel structure. Since these segments are short and together comprise a small percentage of the entire surface, the reconstructed surface should be quite visually acceptable. However, this would also introduce holes in the surface.

5.4.5 MRI of Human Head

In medical imaging, surfaces of anatomical structures are often of interest. As an example of a real and practical application, we will compress the surface of a human head by representing the surface using the Ring Peeling method. The resulting representation can be used for complex applications such as registration and recognition, or for simpler purposes such as compressed data transfer and storage.

Figure 5-17 shows a montage of the 57 slices of an MRI data set of a human head. This data set was obtained from the IMSL/IDL mathematical analysis package. Each slice has a resolution of 80 x 100 pixels and is displayed using a pseudocolored red colormap.

The particular surface we chose to extract is the skin of the human head. Preprocessing steps are initially performed as outlined in Figure 5-18. The data set is first given in raw integers ranging from 0 to 255. The entire data set is bilinearly interpolated in all three dimensions to increase the smoothness and resolution. The result is a 160 x 200 x 114 voxels data set. Next, a threshold at 14% of the maximum value is performed to remove random low-valued image artifacts. Borders are then cropped to obtain the head; the lower neck region is eliminated. Laplacian-based surface extraction is then used to extract the surface of the head. Zero-padding extends the surface data set to a uniform 200 x 200 x 200 voxels data set.

Figure 5-19 shows the rendered surface of the outer skin of the human head. Note that the surface is not closed and can be very complex and nonconvex in regions such as the ears. Ring assignment is performed on the surface voxels to generate the following figures. The starting voxel is chosen to be one of the voxels on the top of the head. Ring assignment assigns the surface voxels to 236 rings. Figure 5-20 shows slices 0 through 7 of the surface along the vertical axis (*z*-axis). Each slice is colored using a red-white colormap. Darker colors correspond to voxels assigned to earlier rings, and lighter colors correspond to voxels assigned to later rings. Note there is a lot of inner anatomical structures that have been designated as belonging to the surface. Initially, we had planned to perform more processing to remove these inner surface regions. However, since the Ring Peeling method only assumes that ring voxels are adjacent, we decided to keep these inner structures and expected that the Ring Peeling method should not stumble. Figure 5-21 shows slices 50 through 57 which cut through the ear regions. Figure 5-22 shows slices 60 through 67 which cut through both the eye regions and the upper ear regions. Figure 5-23 shows slices 90 through 97 corresponding to the scalp region.

The progressive reconstruction of the surface can be seen in Figure 5-24. In (a), the reconstruction of the surface after 15 rings have been processed and coded with the peel structure scheme. In (b), 50 rings have been processed and we reconstruct more of the top of the head. In (c), 75 rings have been processed and we reconstruct the eyes and the beginnings of the cheek bones. In (d), 88 rings have been processed and the nose begins to be reconstructed. Also, the complicated ear regions begin to appear. In (e), 100 rings have been processed and the lips and chin are gradually reconstructed. In (f), 150 out of the total of 236 rings have been processed and we obtain a surface visually similar to that in Figure 5-19. As we process more than 150 rings, we start to reconstruct the inner surfaces. Visually, since our rendering method does not involve transparency, reconstructions where more than 150 rings are processed appear the same as in (f).

In terms of compression, if we use conventional surface compression methods, we require 242,844 bytes to store the 80,948 surface voxels. Using Ring Peeling, we require 90,290 bytes of storage and thus achieve 37.18% compression. In the previous cases, the number of rings are merged into a fewer number of peel segments. Here, however, 236 rings are converted into 1557 peel segments. This indicates there are a lot of short, inefficient peel segments. Yet, the number of peel segments is less than a third the number of surface voxels. As a result, the Ring Peeling method applied here must be more efficient than conventional methods.

5.5 Discussion

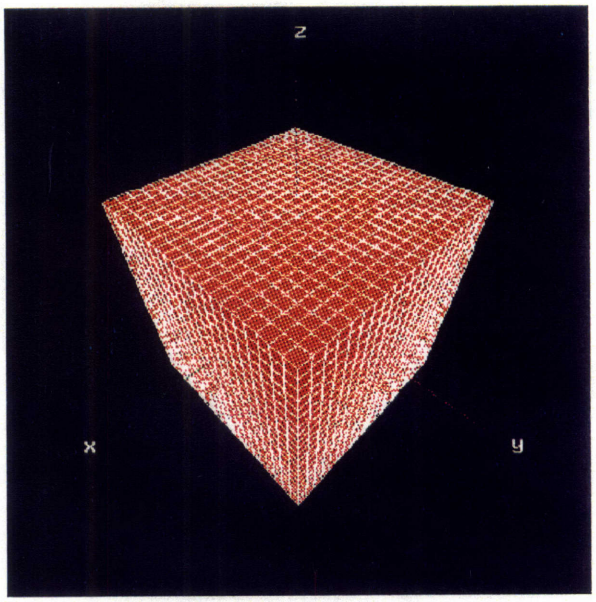
In contrast to the Spherical Mapping method for compression, the Ring Peeling method provides an efficient representation for general nonconvex surfaces. Additionally, the restriction that the surface must be closed in the the Spherical Mapping does not apply to the Ring Peeling method. In essence, the Ring Peeling method is a lossless compression algorithm. However, it can be made lossy by eliminating short peel segments.

The Ring Peeling method outperforms conventional surface compression methods when the number of peel segments is less than a third the number of surface voxels. Ideally, the bound of compression performance is 33% where the peel structure contains only one peel segment. Even this bound is conservatively estimated based on the assumption that we require one byte to code each segment chain element. We can actually do better by realizing that instead of requiring 8 bits to represent each segment chain element, we only require $log_2 27 = 4.76$ bits to represent a number between 1 and 27. This eventually translates into a performance bound of approximately 20% compression. As in the case of the Spherical Mapping method, the encoded peel structure sequence can be further compressed by entropy coding the numeric sequence. Appendix C discusses how entropy coding can be used with the Ring-Peeling method.

There are several key points to note regarding the Ring Peeling method. The number of rings and peel segments obtained is sensitive to the starting point in the algorithm. Ideally, we would like to pick a starting voxel such that the we minimize the number of rings. Also, we would like the number of voxels from one ring to an adjacent ring to change smoothly. The greater the contiguity of elements within a ring, the fewer is the number of peel segments. Lastly, point patches, or peel segments containing only one element, are the most inefficient since they require the full cartesian specification of their location.

SIMULATED SOLID CUBE
RING ASSIGNMENT

(a) OBJECT CUBERILLE



(b) RINGS CUBERILLE

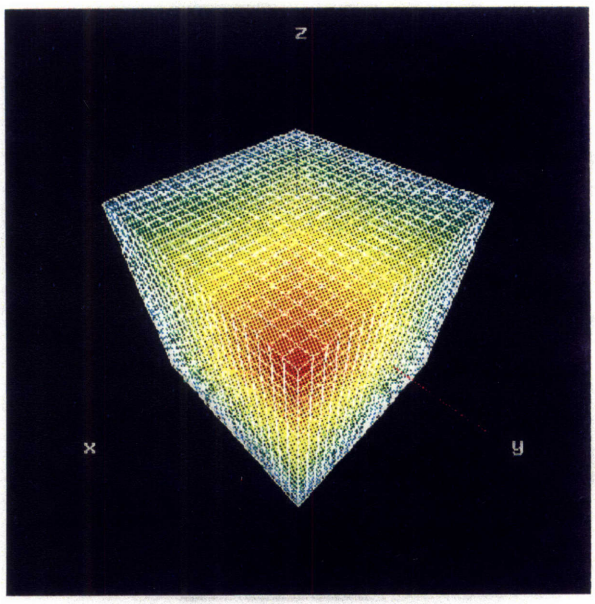
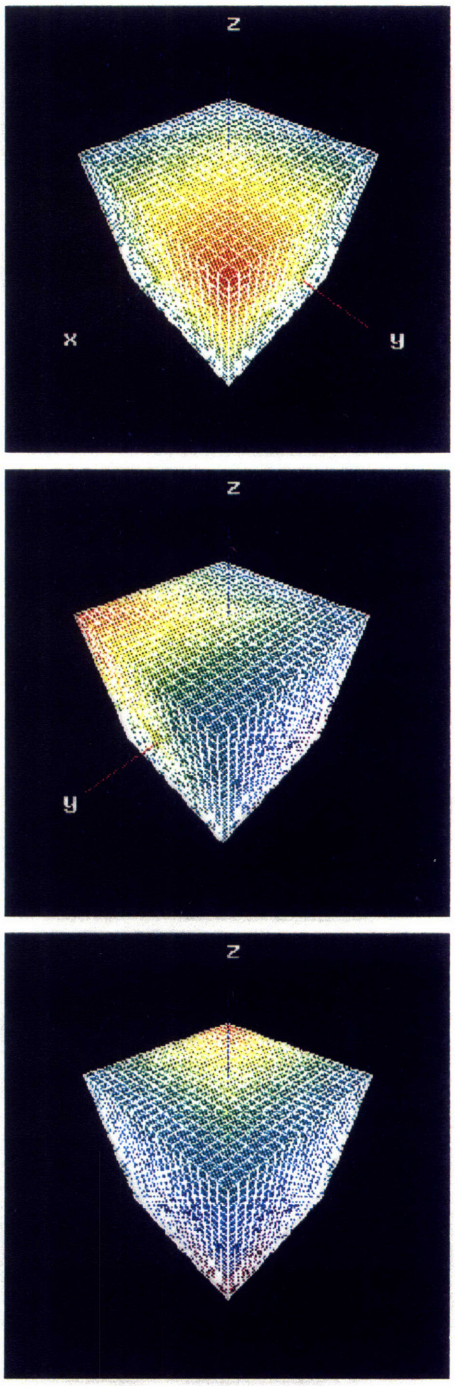


Figure 5-7. Ring assignment of the surface voxels of a solid cube. (a) Original surface. (b) Surface after ring assignment.

SIMULATED SOLID CUBE
RING ASSIGNMENT



(a) 45° CW ABOUT Z-AXIS

(b) 135° CW ABOUT Z-AXIS

(c) 225° CW ABOUT Z-AXIS

Figure 5-8. Different views of the ring assigned surface takenat (a) 45°, (b) 135°, and (c) 225° counter-clockwise about z-axis.

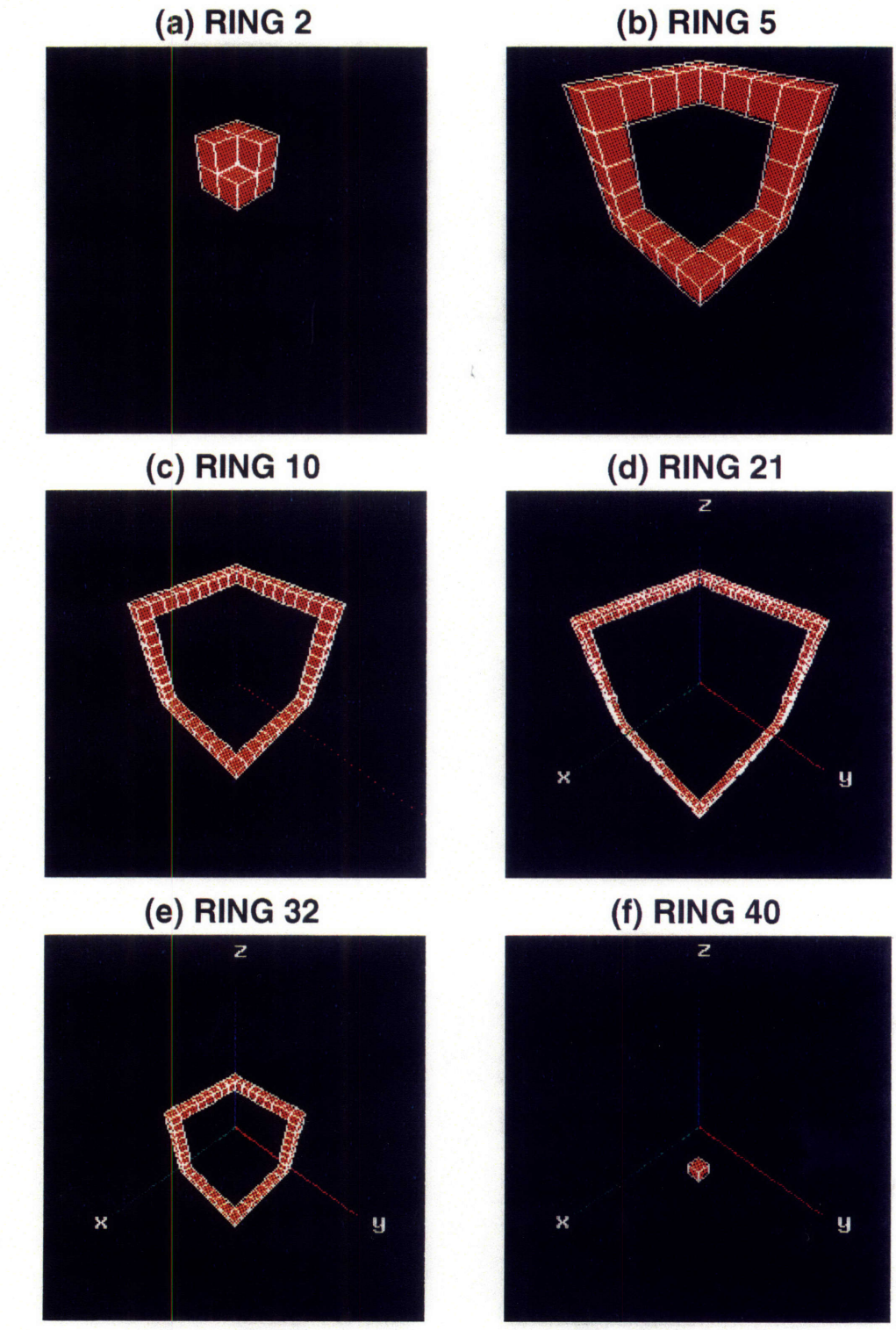


Figure 5-9. Individual rings of the surface after ring assignment at level (a) 2, (b) 5, (c) 10, (d) 21, (e) 32, and (f) 40.

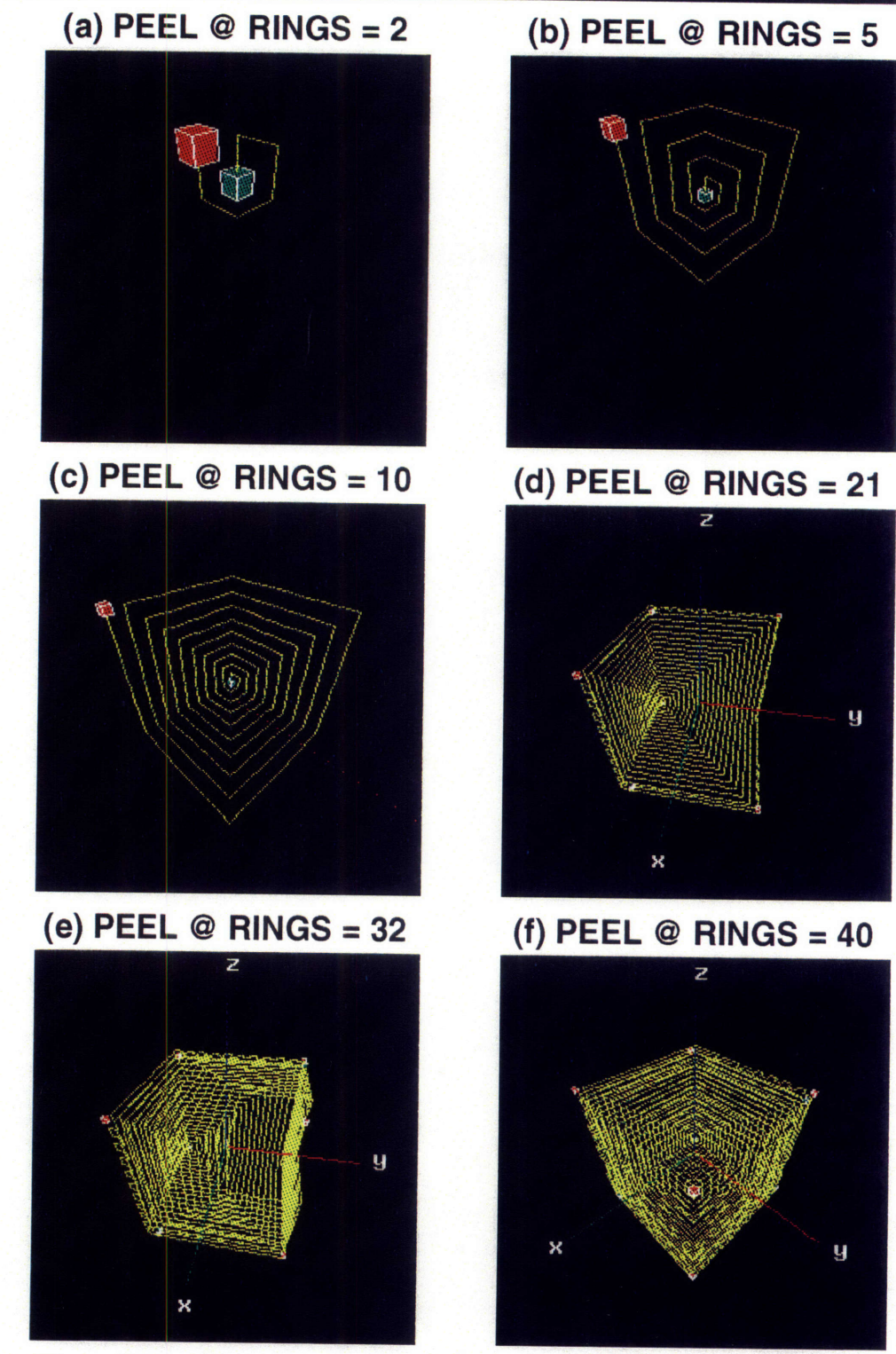


Figure 5-10. Peel structure after merging (a) 2, (b) 5, (c) 10, (d) 21, (e) 32, and (f) 40 rings.

SIMULATED SOLID CUBE
COMPRESSION RESULTS

METHOD

CONVENTIONAL:	7206 BYTES	}	33.75%
RING PEELING:	2432 BYTES		

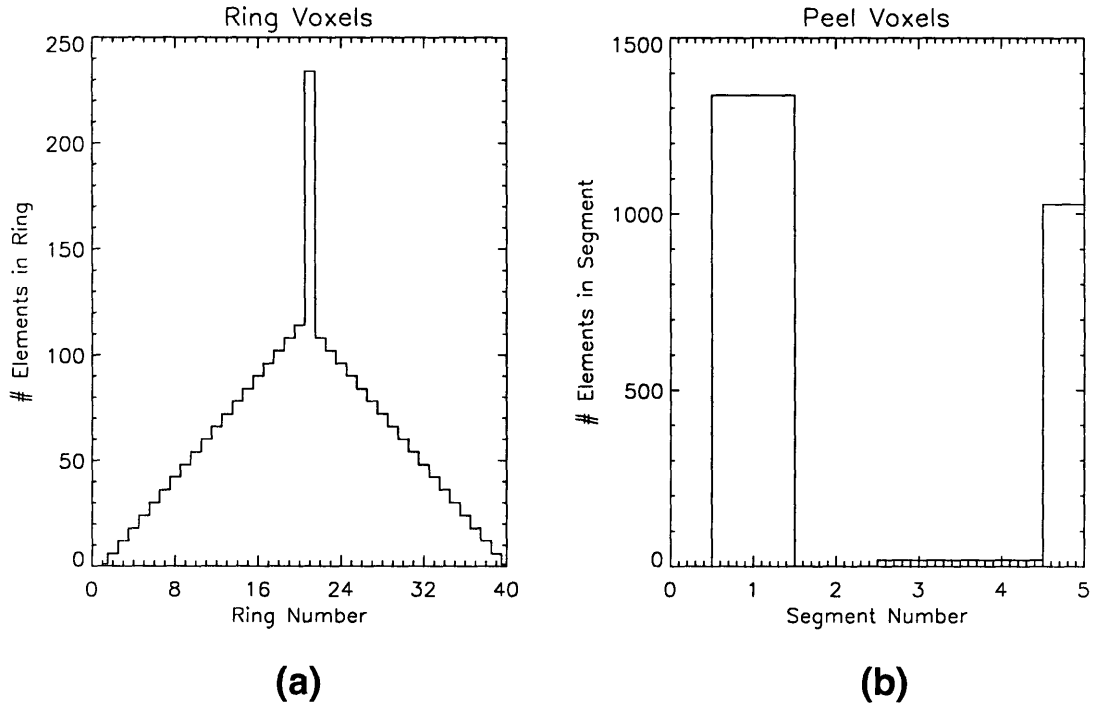
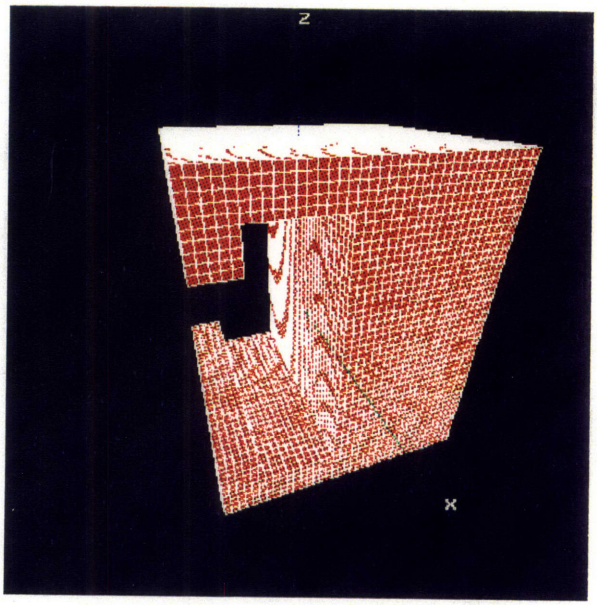


Figure 5-11. **Compression of 33.75% using Ring Peeling method. Histogram of the surface voxels grouped into (a) rings and (b) peel segments.**

NONCONVEX C-SHAPE
RING ASSIGNMENT

(a) OBJECT CUBERILLE



(b) RINGS CUBERILLE

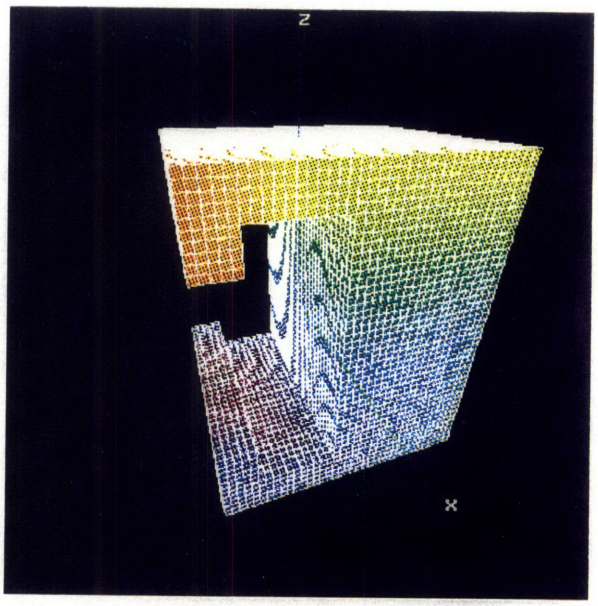


Figure 5-12. Ring assignment of the surface of a nonconvex C-shape. (a) Original surface. (b) Surface after ring assignment.

NONCONVEX C-SHAPE
RING ASSIGNMENT

(a) -45° CW ABOUT Z-AXIS

(b) 45° CW ABOUT Z-AXIS

(c) 135° CW ABOUT Z-AXIS

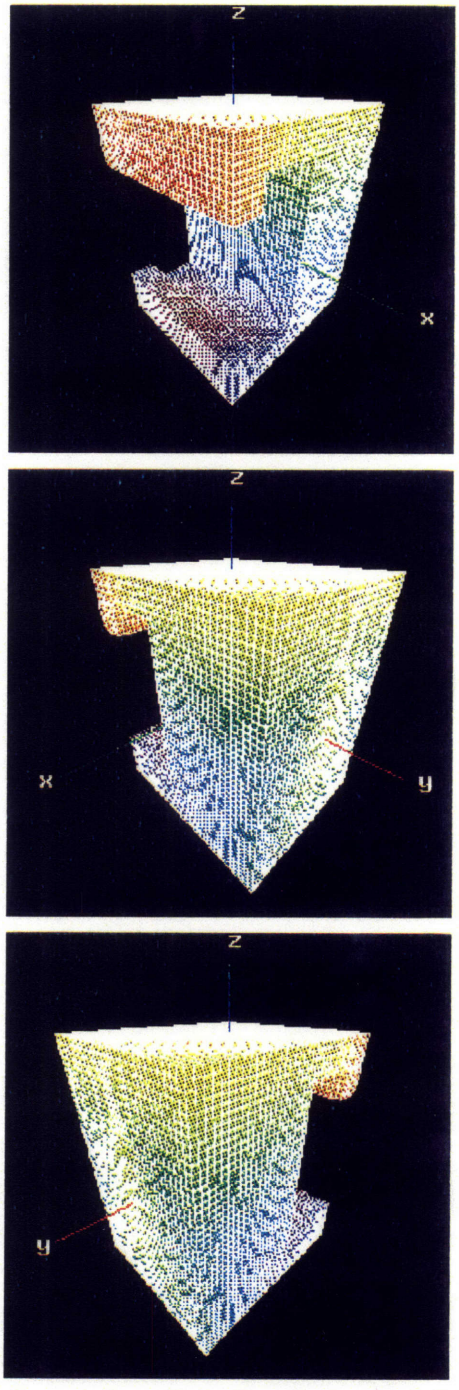


Figure 5-13. Different views of the ring assigned surface takenat (a) -45°, (b) 45°, and (c) 135° counter-clockwise about z-axis.

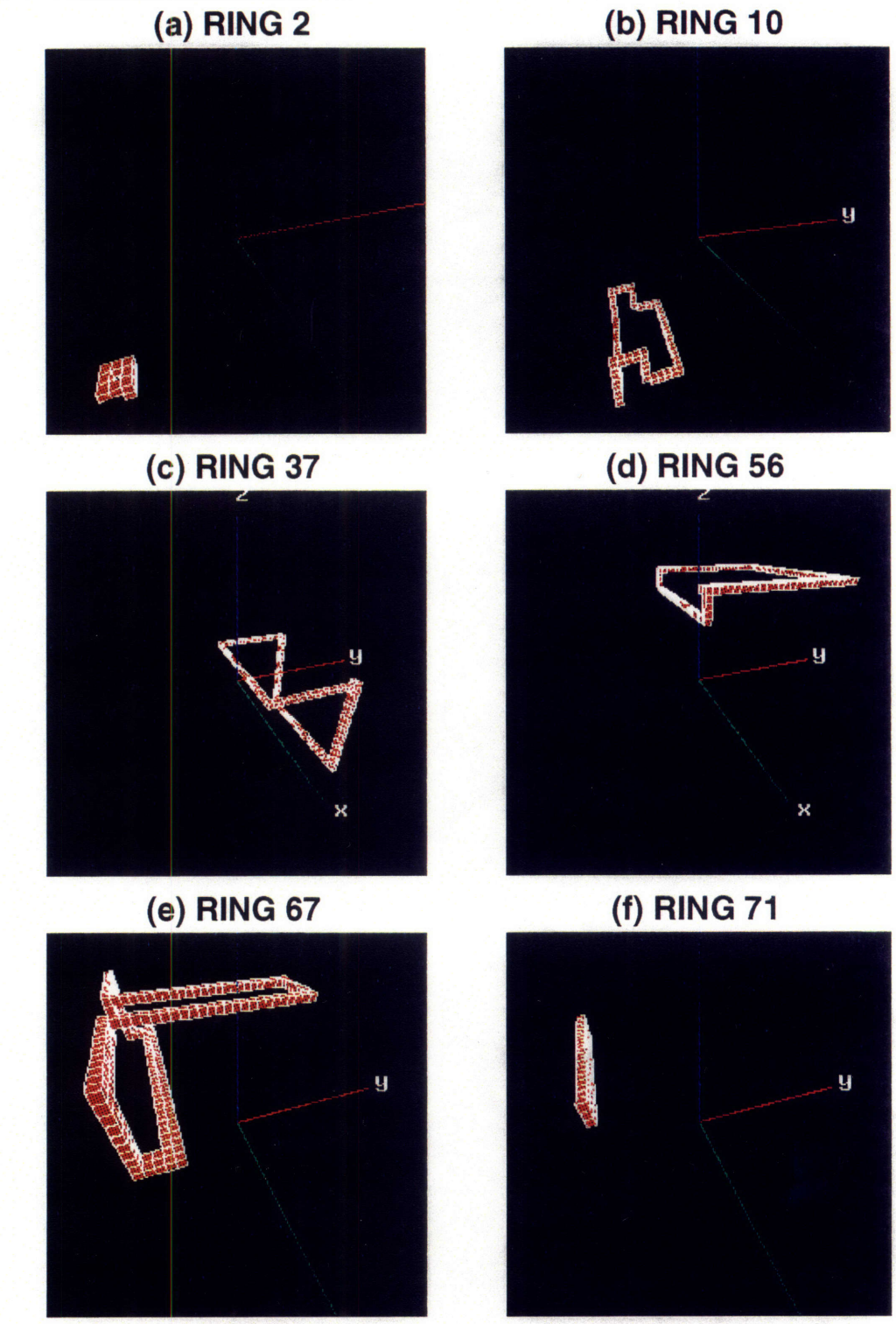


Figure 5-14. Individual rings of the surface after ring assignment at level (a) 2, (b) 10, (c) 37, (d) 56, (e) 67, and (f) 71.

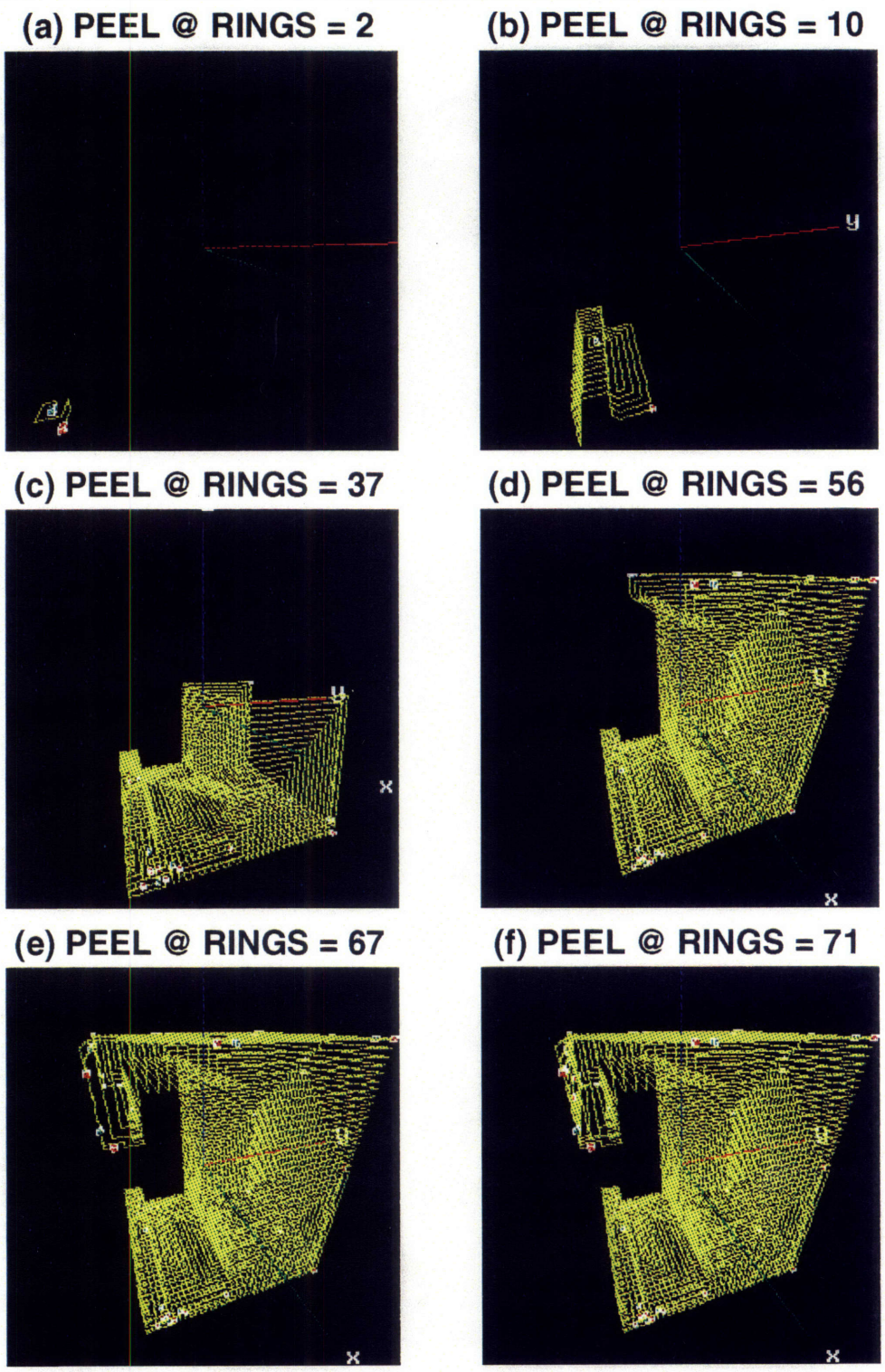
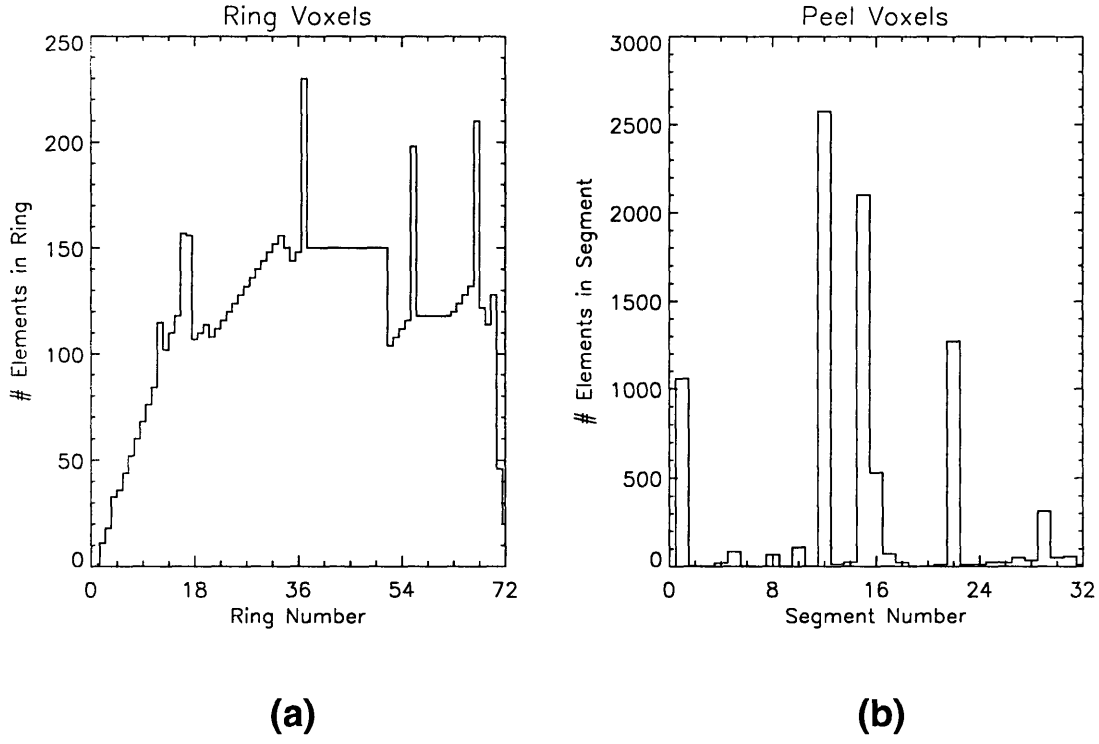


Figure 5-15. Peel structure after merging (a) 2, (b) 10, (c) 37, (d) 56, (e) 67, and (f) 71 rings.

NONCONVEX C-SHAPE
COMPRESSION RESULTS

METHOD

CONVENTIONAL: 25,650 BYTES
RING PEELING: 8,742 BYTES
} 34.08%



(a) (b)

Figure 5-16. Compression of 34.08% using Ring Peeling method. Histogram of the surface voxels grouped into (a) rings and (b) peel segments.

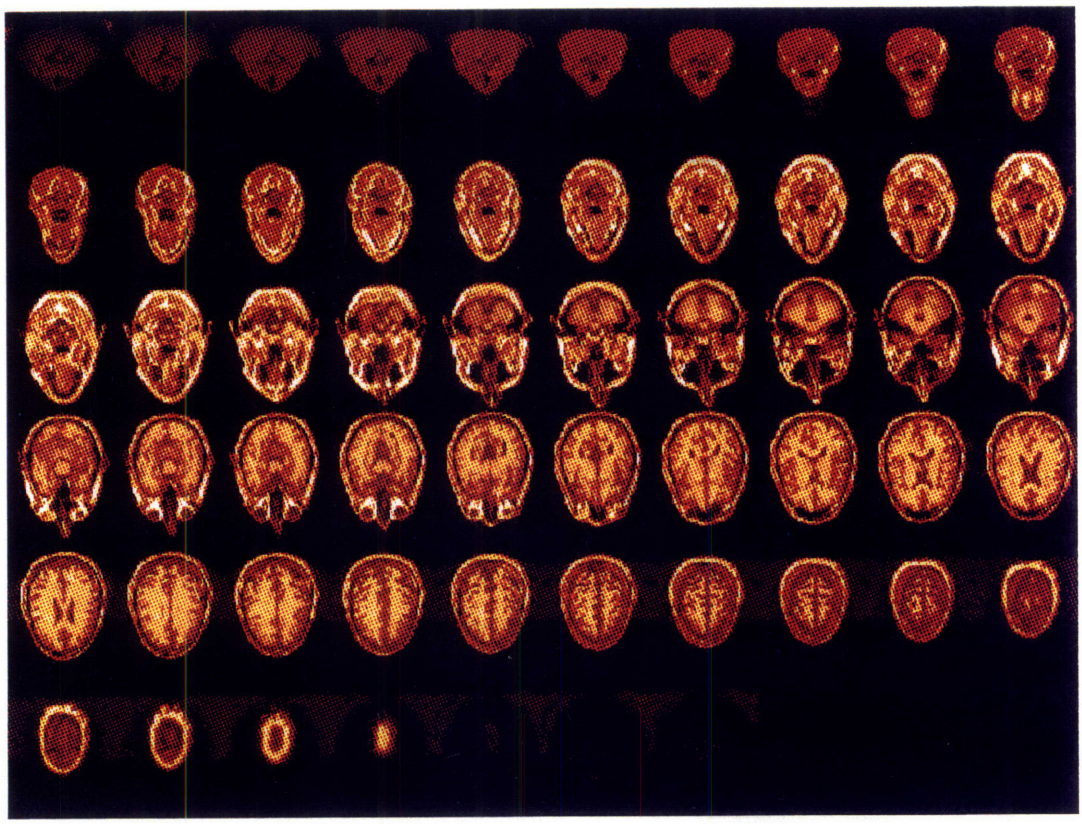


Figure 5-17. Pseudo-colored slices from an MRI data set of a human head.

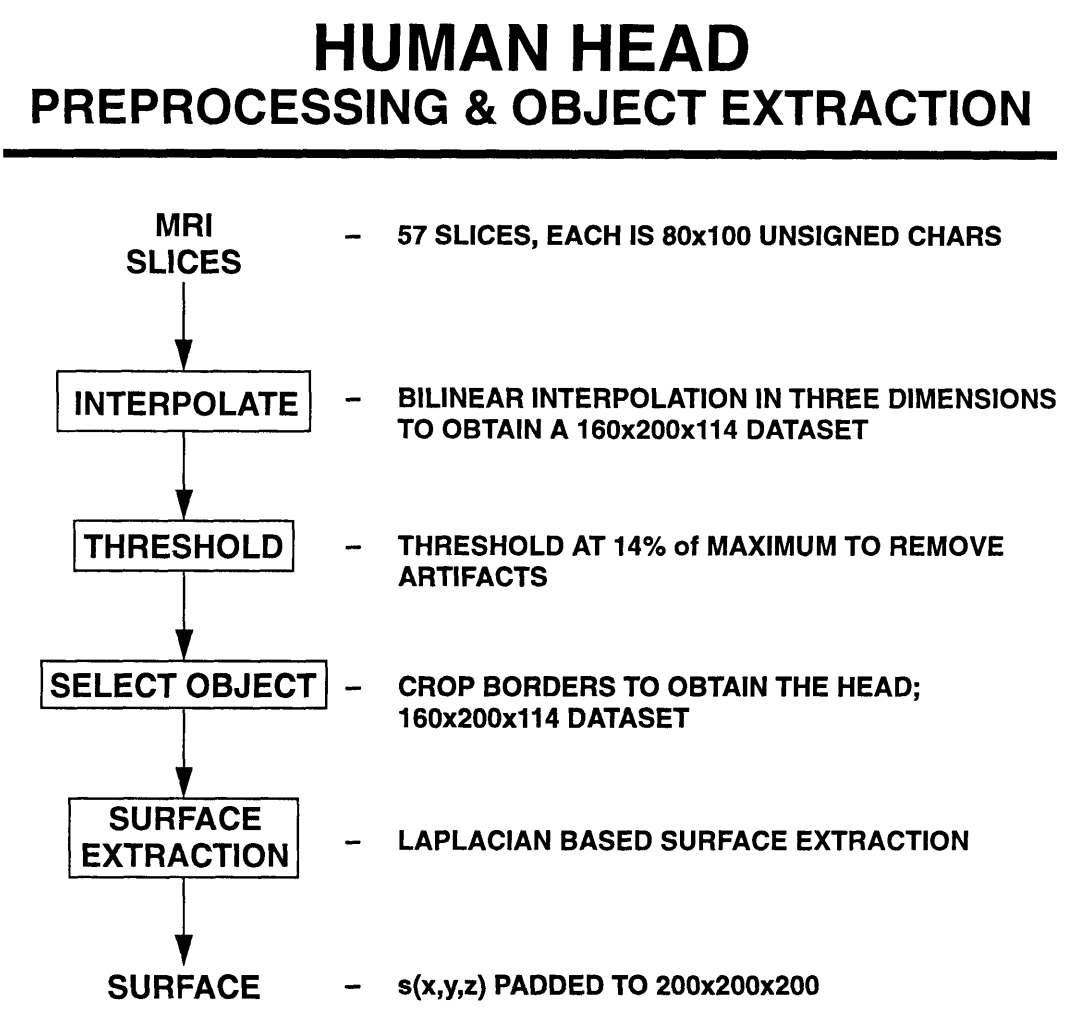


Figure 5-18. Block diagram of the steps involved in preprocessing and object extraction for this particular MRI data set.

HUMAN HEAD
RENDERED SURFACE

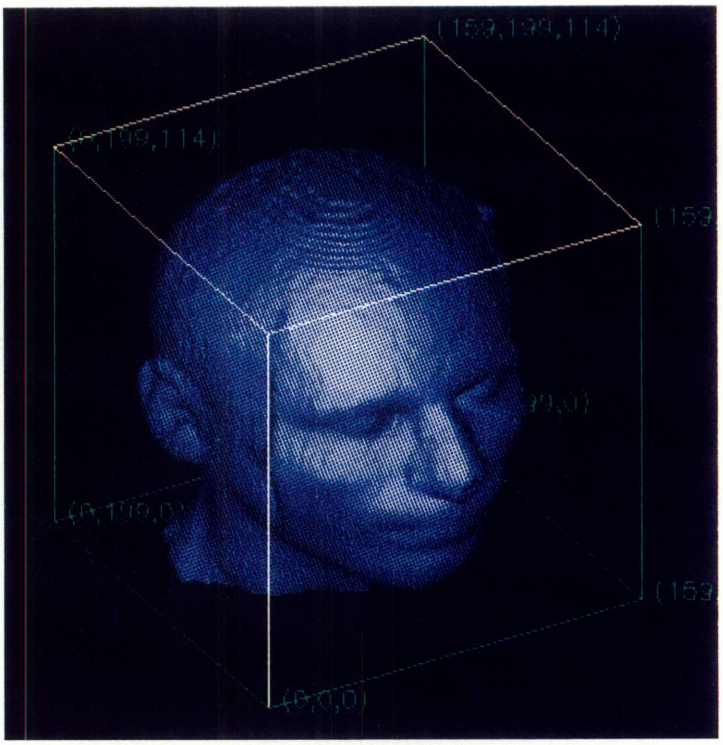


Figure 5-19. Rendered display of the outer surface of the human head.

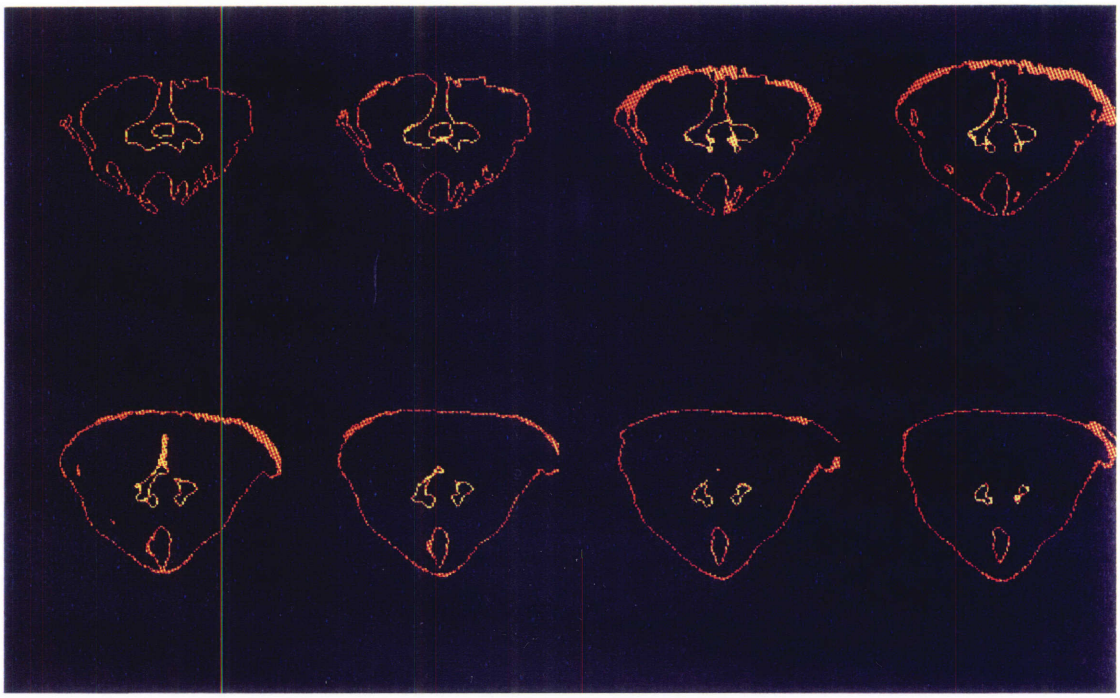


Figure 5-20. Slices $z = 0$ to $z = 7$ of the outer surface of the human head.

HUMAN HEAD
SURFACE SLICES

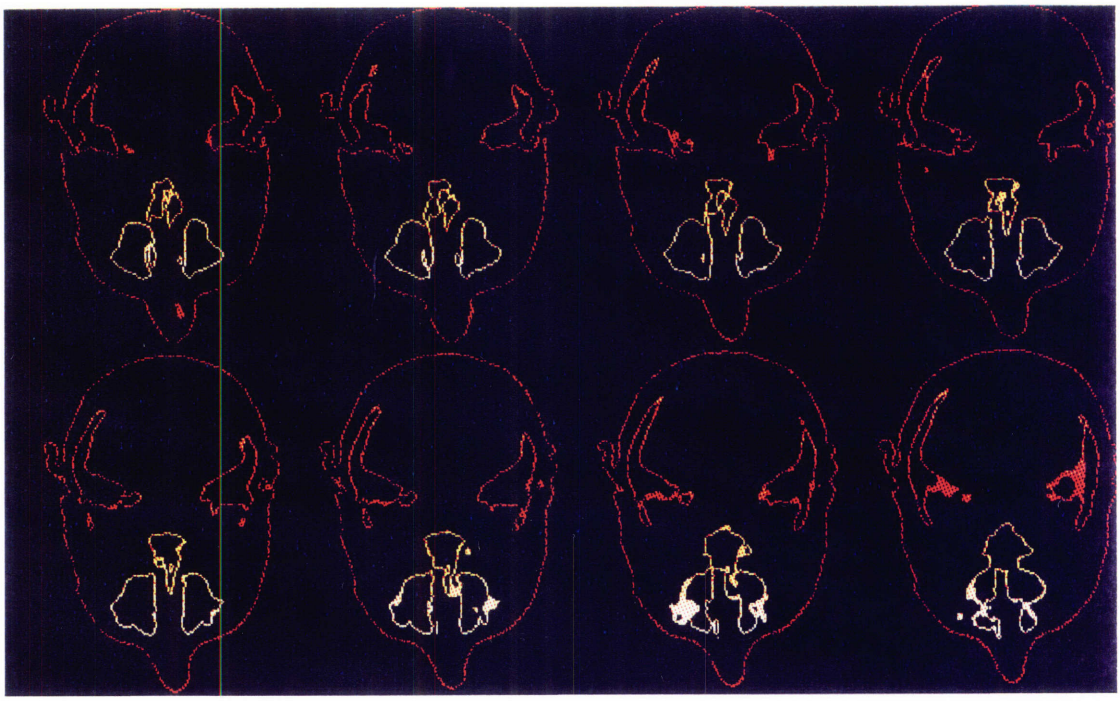


Figure 5-21. Slices $z = 50$ to $z = 57$ of the outer surface of the human head.

HUMAN HEAD
SURFACE SLICES

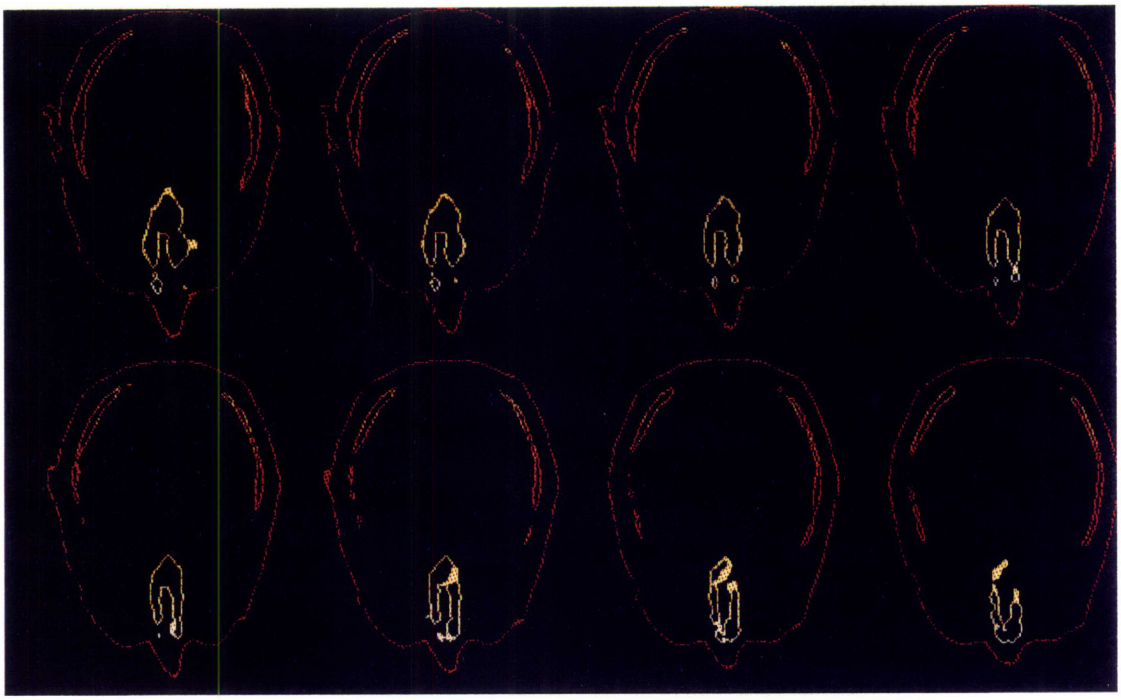


Figure 5-22. Slices $z = 60$ to $z = 67$ of the outer surface of the human head.

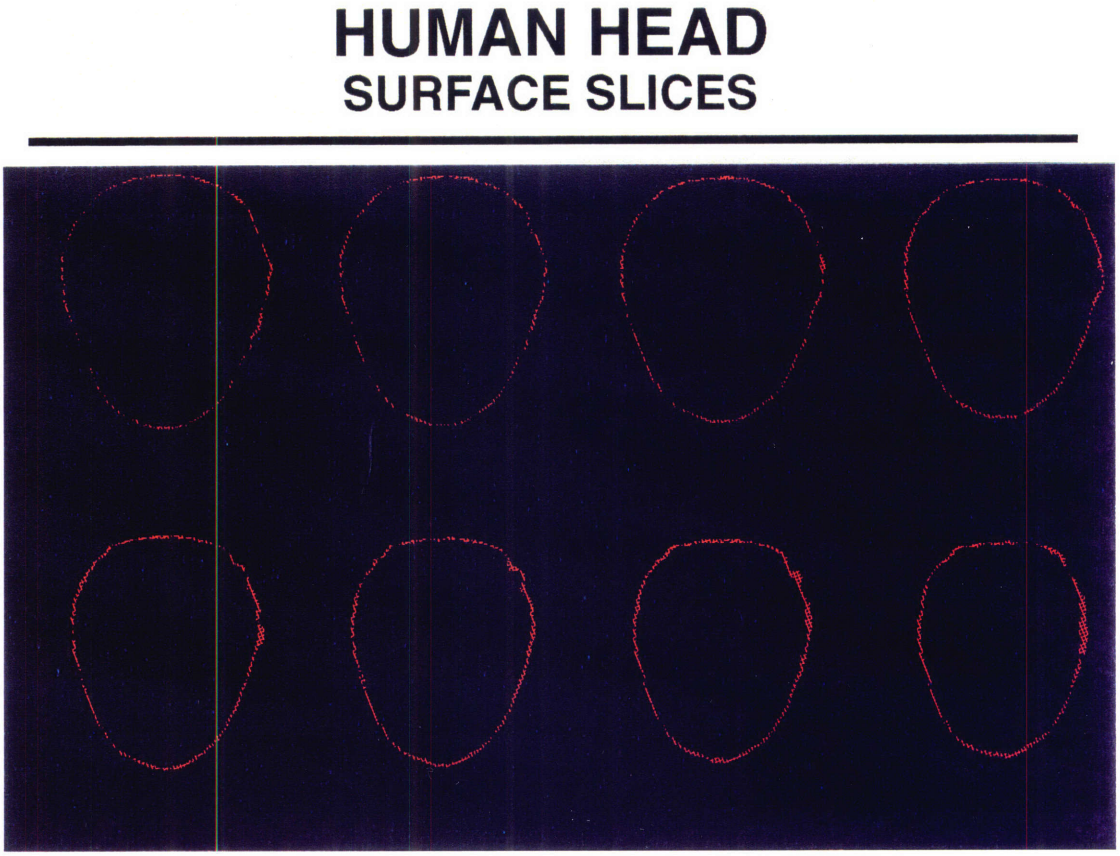


Figure 5-23. Slices $z = 90$ to $z = 97$ of the outer surface of the human head.

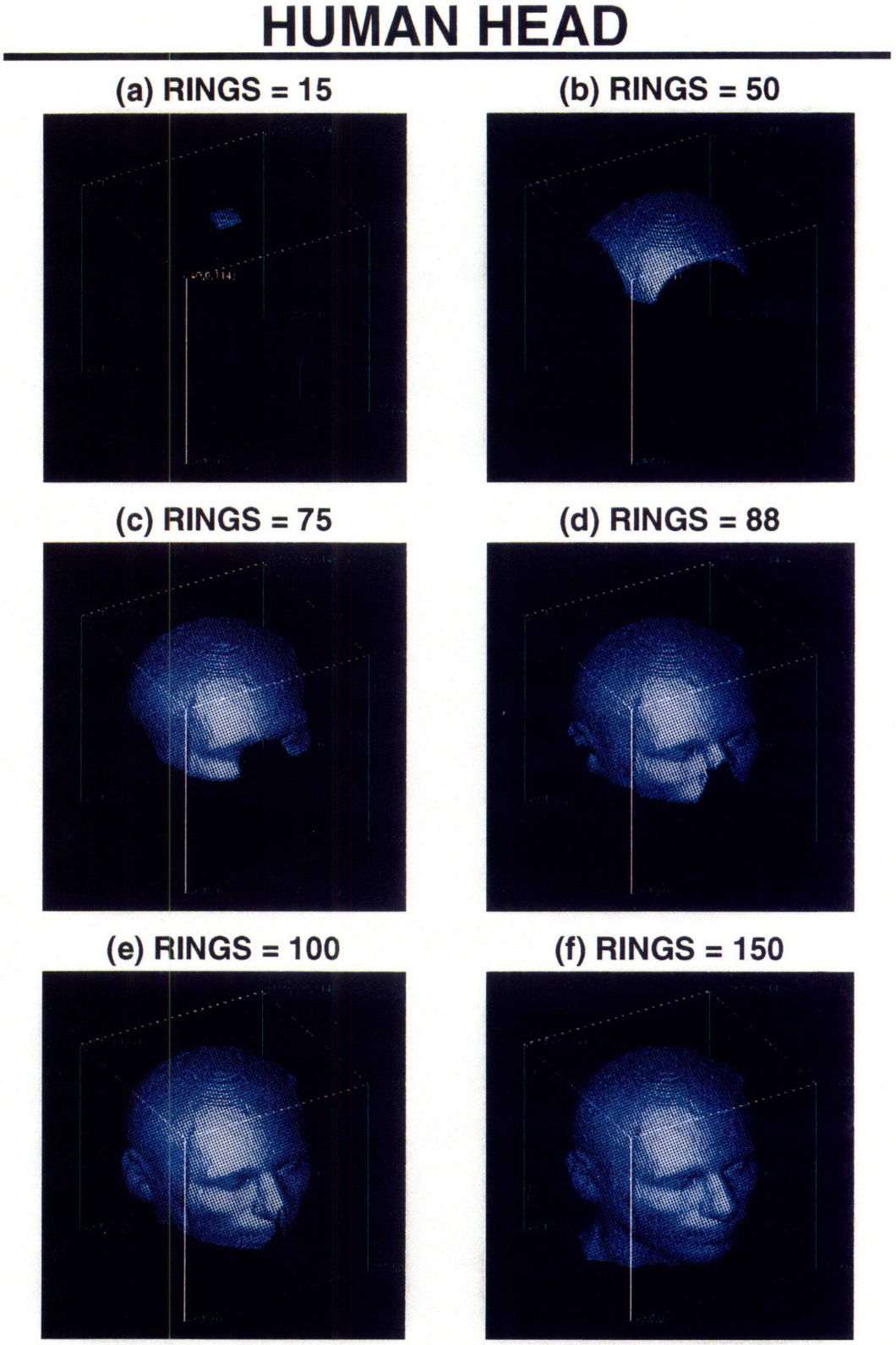


Figure 5-24. Progressive reconstruction of the outer surface of human head.

Chapter 6
Discussion and Comparison

6.0 Compression Comparison

6.1 Geometric Considerations

6.2 Computation Expense

6.3 Reconstruction Comparison

6.4 Applications

Chapter 6

Discussion and Comparison

6.0 Compression Comparison

The Spherical Mapping method for surface compression is inherently a lossy compression algorithm. The surface in cartesian space is described by the radial basis functions located at the centroid of the surface. The compression limit is chosen based on how acceptable the reconstructed surface must be. This acceptability can be subjective and based on human visual judgment, or objective such as utilitizing error norms.

The Ring Peeling method, on the other hand, is a lossless compression algorithm which can be adapted to become lossy by selectively removing inefficient peel segments within the peel structure. The compression limit for the lossless version is approximately 20% when compared to conventional surface representation methods.

As N, the length of each dimension, increases or equivalently, as the resolution increases, the achieved compression using the Spherical Mapping method remains more or less constant since we retain transform coefficients. For example, if a surface is represented by 5 low order radial basis functions, then if we increase the resolution by 2 in each dimension, we should still be able to represent the surface with 5 low order basis functions. However, in the Ring Peeling method, as we increase resolution, we increase the number of voxels so compression is a function of the number of surface voxels. For example, if a surface contains 10,000 surface voxels, then after increasing resolution by 2 in each dimension, we should have at most 80,000 surface voxels. Since the storage space required is a linear function of the number of surface voxels for the Ring Peeling method, as resolution increases, compression is thus a function of the number of surface voxels.

6.1 Geometric Considerations

For the Spherical Mapping method, only star-shape-convex objects can be compressed. Inherent to this method is also the assumption that there is only one object to be processed. For example, if there are two exclusive spheres in a data set, then the Spherical Mapping method will fail, most likely because of nonconvexity with respect to the centroid of the two spheres. In addition, the extracted surface in the Spherical Mapping method must be closed and contain no holes.

The Ring Peeling method lifts the restriction of convexity on object type. It is capable of handling geometrically complex surfaces that can be either open or closed. Since the Ring Peeling method works primarily with the relationship between surface voxels, it can handle multiple objects present in the same dataset. Take, for example, again the case of the two exclusive spheres.

We can concatenate the rings from the first sphere with the rings of the second sphere, and then continue to generate the peel structure.

6.2 Computation Expense

The brunt of the computational effort involved in the Spherical Mapping method lies in the calculation of two-dimensional transforms. After the transforms are calculated, then there is also the process of locating and retaining the best coefficients given a desired compression ratio. In the transform calculations, the use of the 2D DFT will involve complex numbers whereas the use of the 2D DCT involves only real numbers. In either case, floating point numeric representation is needed.

In the Ring Peeling method, the ring assignment process is a quick iterative process involving indices. The next step is to convert rings into a peel structure where surface voxels that neighbor each other are linked. This step involves calculating spatial distances, and several loops of comparing these distances for each ring. Overall, this analysis process requires computational effort that can be quite intensive as the number of rings increases, and as the number of elements within a ring also increases. The concomitant synthesis process, on the other hand, is very fast.

6.3 Reconstruction Comparison

Reconstruction in the Spherical Mapping method is slower when compared to the Ring Peeling method since it involves a costly inversion of a 2D transform rather than indexing entries in a lookup table, or codebook. In the Spherical Mapping method, perfect reconstruction can yield errors because of the ray-extension problems that arise in the mapping stage. For the Ring Peeling method, there is no reconstruction errors in the lossless version of the algorithm.

6.4 Applications

The choice of which surface representation to use can highly depend on the intended application. For performing volume and surface area measurements, calculations based on the Spherical Mapping or the Ring Peeling representation are relatively simple. For object recognition purposes, there is a greater disparity in ease. In the Spherical Mapping method, two objects will yield two spherical maps. These maps or their transforms can be compared in an element-by-element manner. Even if the maps are of different size, we can always compute equal size transforms and compare the transforms. For scale-invariance, we can normalize each of the two map transform before comparison. Likewise, we can find rotation-invariant maps. Translation-invariance is already inherent since we essentially use object coordinates when we transfer the

coordinate system to the centroid of the object. In the Ring Peeling method, however, it is very difficult to compare rings of two different objects, and even more so, their peel structures. The reason is that two objects can have different number of rings, and also the rings are very dependent on the choice of the starting voxel. Suppose we can say that one ring of an object correspond to a ring of the other object. Comparing these two rings is already a formidable task since these rings can have different number of elements and very different geometries.

Chapter 7

Conclusion / Future Work

7.0 Conclusion

7.1 Future Work

Chapter 7

Conclusion / Future Work

7.0 Conclusion

In this thesis, we have developed two algorithms to compress objects in 3D images. To extract the surface of objects, Laplacian-based surface detection is first used. Algorithm 1, the Spherical Mapping method is developed to compress star-shape-convex objects and is more efficient as the data set size increases. Algorithm 2, the Ring Peeling method is more suited to handle arbitrarily complex objects, but yields only limited compression. The choice of surface representation to use depends on the intended application and the class of objects involved. Further surface compression can be achieved by performing entropy coding on the sequence of retained transform coefficients in the Spherical Mapping method or the encoded peel structure sequence in the Ring Peeling method.

7.1 Future Work

There are multiple avenues for future research. One problem is to find more efficient ways to obtain rings always containing adjacent ring elements in the Ring Peeling method. The overall compression will be higher. Also, another problem is to develop a hybridization of the Spherical Mapping and the Ring Peeling methods for the lossy compression of general objects. We would like to capture the spatial frequency information which the Spherical Mapping map transforms provide, but not solely restricted to star-shape-convex objects. The goal is then to develop a 2D map with axes, ring element number and ring number. The function value can then be a quaternion expressing the distance to the centroid. This new map can then be compressed using image compression methods modified for quarternions. Lastly, perhaps motion-estimation models of compression can be added to the Spherical Mapping method. For example, the examination of how well a 3D heart pulsating over time can be compressed.

Appendices

Appendix A **Efficient Implementation of 3D DFT**

Appendix B **Spherical Harmonics**

Appendix C **Entropy and Coding**

Appendix A

Efficient Implementation of 3D DFT

A.0 The 3D DFT

The 3D DFT of a complex N_1 x N_2 x N_3 -point sequence $x(n_1,n_2,n_3)$ that is zero outside

$$\begin{aligned} 0 \le n_1 < N_1 \\ 0 \le n_2 < N_2 \\ 0 \le n_3 < N_3 \end{aligned} \tag{A.1}$$

is given by $X(k_1,k_2,k_3)$, where

$$X(k_1, k_2, k_3) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \sum_{n_3=0}^{N_3-1} x(n_1, n_2, n_3)\, e^{-j\left(\frac{2\pi}{N_1}\right)k_1 n_1}\, e^{-j\left(\frac{2\pi}{N_2}\right)k_2 n_2}\, e^{-j\left(\frac{2\pi}{N_3}\right)k_3 n_3}. \tag{A.2}$$

From equation (A.1), directly computing $X(k_1,k_2,k_3)$ for each (k_1,k_2,k_3) requires $N_1N_2N_3 - 1$ additions and $N_1N_2N_3$ multiplications. Since there are $N_1N_2N_3$ different values of (k_1,k_2,k_3), the total number of arithmetic operations required in computing $X(k_1,k_2,k_3)$ from $x(n_1,n_2,n_3)$ is $N_1N_2N_3\,(N_1N_2N_3 - 1)$ additions and $N_1^2N_2^2N_3^2$ multiplications.

A.1 Dimensional Decomposition

The dimensional decomposition method is an extension of the row-column decomposition method presented in chapter 3 of [22]. To develop dimensional decomposition, we rewrite equation (A.1) as follows:

$$X(k_1, k_2, k_3) = \underbrace{\sum_{n_3=0}^{N_3-1} \left(\sum_{n_2=0}^{N_2-1} \left(\overbrace{\sum_{n_1=0}^{N_1-1} x(n_1, n_2, n_3)\, e^{-j\left(\frac{2\pi}{N_1}\right)k_1 n_1}}^{f(k_1, n_2, n_3)} \right) e^{-j\left(\frac{2\pi}{N_2}\right)k_2 n_2} \right) e^{-j\left(\frac{2\pi}{N_3}\right)k_3 n_3}}_{g(k_1, k_2, n_3)} \tag{A.3}$$

We first compute $f(k_1,n_2,n_3)$ from $x(n_1,n_2,n_3)$. Then we compute $g(k_1,k_2,n_3)$ from $f(k_1,n_2,n_3)$. Lastly, $X(k_1,k_2,k_3)$ can be computed from $g(k_1,k_2,n_3)$. For a fixed n_2 and n_3, $f(k_1,n_2,n_3)|_{n_2,n_3}$ is the 1D N_1-point DFT of $x(n_1,n_2,n_3)|_{n_2,n_3}$ with respect to the variable n_1. Since there are N_2 different values of n_2 and N_3 different values of n_3 in $f(k_1,n_2,n_3)$ that are of interest to us, $f(k_1,n_2,n_3)$ can be computed from $x(n_1,n_2,n_3)$ by computing N_2N_3 1D N_1-point DFTs.

Once $f(k_1,n_2,n_3)$ is calculated, our 3D DFT becomes

$$X(k_1, k_2, k_3) = \sum_{n_3=0}^{N_3-1}\left(\sum_{n_2=0}^{N_2-1} f(k_1, n_2, n_3)\, e^{-j\left(\frac{2\pi}{N_2}\right)k_2 n_2}\right) e^{-j\left(\frac{2\pi}{N_3}\right)k_3 n_3} . \quad \text{(A.4)}$$

We can once again apply this idea of computing one-dimensional DFTs in the n_2 dimension to obtain $g(k_1,k_2,n_3)$. Our 3D DFT now is reduced to computations along one-dimension:

$$X(k_1, k_2, k_3) = \sum_{n_3=0}^{N_3-1} g(k_1, k_2, n_3)\, e^{-j\left(\frac{2\pi}{N_3}\right)k_3 n_3} . \quad \text{(A.5)}$$

One more series of applications of 1D DFT in the remaining n_3 dimension will yield $X(k_1,k_2,k_3)$, the 3D DFT of $x(n_1,n_2,n_3)$. This entire process of dimensional decomposition can be visualized as in Figure A-1. Each darker line shows a 1D DFT operation on a 1D array of data.

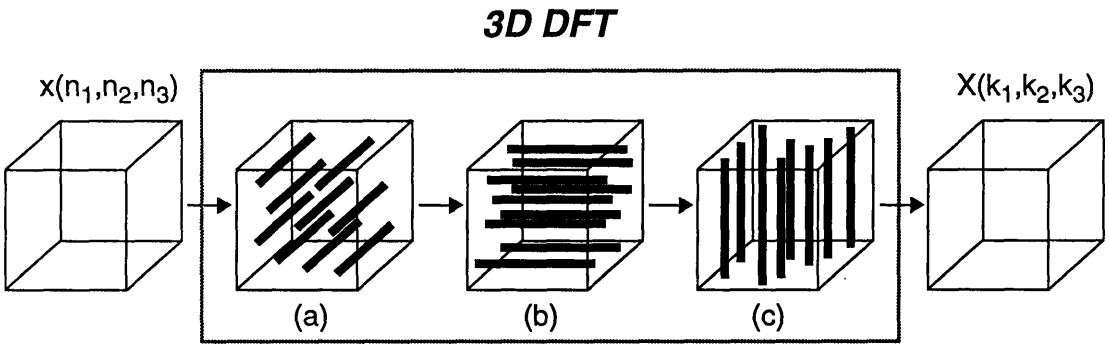


Figure A-1. Decomposition of 3D DFT. Performing 1D DFTs in the (a) n_1 (b) n_2 and (c) n_3 dimensions.

In terms of computations, we perform N_2N_3, then N_1N_3, and N_1N_2 1D DFTs. If we perform direct 1D DFT computations, the total number of arithmetic operations required in computing $X(k_1,k_2,k_3)$ from $x(n_1,n_2,n_3)$ becomes $N_1N_2N_3\,(N_1+N_2+N_3-3)$ addition operations and $N_1N_2N_3\,(N_1+N_2+N_3)$ multiplication operations. If $N_1=N_2=N_3=N$, this represents a decrease of approximately N^2 additions and operations over straightforward computation of the 3D DFT. As N increases, the increase in computation savings becomes more significant.

A.2 The Use of FFTs

Since we have decompose the multidimensional DFT into one-dimensional DFTs, we can further reduce the number of computations by using fast Fourier transform techniques to calculate the 1D DFTs. We can of course use any of a variety of 1D FFT algorithms to compute the 1D

DFTs. When $N = 2^M$, a 1D N-point FFT algorithm based on the Cooley-Tukey approach requires

$$\begin{aligned} & N\log_2 N && \text{additions} \\ & \frac{N}{2}\log_2 N && \text{multiplications.} \end{aligned} \tag{A.6}$$

To compute the 3D DFT using dimensional decomposition and also implementing 1D DFTs using FFTs, it requires a total number of

$$\begin{aligned} & N_1N_2N_3\log_2 N_1N_2N_3 && \text{additions} \\ \text{and} \quad & \frac{N_1N_2N_3}{2}\log_2 N_1N_2N_3 && \text{multiplications.} \end{aligned} \tag{A.7}$$

A.3 Comparison

Table A-1 illustrates the computational savings involved by showing the relative number of computations for the three methods considered. Small constant factors have been left out. The value in the parentheses correspond to the 3D DFT calculation of a data set where $N=N_1=N_2=N_3$ and $N=64$. As N increases, the improvement in performance in quite significant. Consider $N=256$, the third method requires 0.00014% of additions required using direct computation.

A better perspective comes from the absolute number of arithmetic operations. For $N=64$, direct computation requires approximately 68,719 million complex multiplications. Dimensional decomposition using the FFT requires about 2.4 million multiplications and 4.8 million additions. A complex addition requires two regular additions and a complex multiplication requires four regular multiplications and three additions. Thus, the total number of regular arithmetic operations is approximately 26.4 million floating point operations. Given that workstations run at around 1-5 Mflops, the improved 3D DFT still hardly runs at real-time speeds.

Table A-2: Comparison of 3D DFT Implementation Methods

Method	Number of Multiplications	Number of Additions
Direct Computation	$N_1^2N_2^2N_3^2$ (100%)	$N_1^2N_2^2N_3^2$ (100%)
Dimensional Decomposition w/ Direct 1D DFT	$N_1N_2N_3\,(N_1+N_2+N_3)$ (0.07%)	$N_1N_2N_3\,(N_1+N_2+N_3)$ (0.07%)
Dimensional Decomposition w/ 1D FFT Algorithm	$\frac{N_1N_2N_3}{2}\log_2 N_1N_2N_3$ (0.0034%)	$N_1N_2N_3\log_2 N_1N_2N_3$ (0.007%)

Appendix B

Spherical Harmonics

B.0 DFT Spherical Harmonics

When we choose to use the 2D DFT to transform a spherical map, $r(\phi,\theta)$, from the spatial domain into frequency domain, a reconstructed 3D surface can be expressed as a linear sum of spherical harmonics. Spherical harmonics are basis surfaces. Just as a signal can be represented as a linear superposition of complex exponentials, a surface can be represented as a linear superposition of spherical harmonics.

We can choose to think about a spherical harmonics as follows. Consider any transform coefficient of a 2D DFT. This particular transform coefficient corresponds to a complex basis function in the spatial domain. Namely, the basis function is a 2D complex exponential. Because we are interested in shape only, if we take only the magnitude of the 2D complex basis function and perform inverse spherical parameterization, we obtain a spherical harmonic. In other words, a particular transform coefficient $R(k_\phi,k_\theta)$ corresponds to a spherical harmonic. Any surface can be constructed from summing different contributing weights of different spherical harmonics. In our study, our surfaces are restricted to star-shape-convex surfaces. Likewise, our spherical harmonics are also star-shape-convex.

Figures B-1 through B-3 shows the spherical harmonics corresponding to different transform coefficients (k_ϕ,k_θ). Note the smooth and periodic nature of the surfaces. Also, higher-frequency transform coefficients correspond to more complex basis surfaces with more fastly-varying surface areas.

B.1 DCT Spherical Harmonics

Instead of using the 2D DFT to transform a spherical map into a frequency representation, we can choose to use the 2D DCT. The spherical harmonics associated with the DCT transform coefficients are shown in Figures B-4 through B-6. We can compare corresponding basis surfaces of the DFT and DCT. The DC component, or the DC basis surface, is the same for both transforms and is a sphere whose radius is determined by the value of the coefficient. If we compare the (1,0) basis surfaces of Figure B-1b and Figure B-4b, for example, we see that the basis surface associated with the DCT is more complex in structure. Consider a complicated surface area that can be partly represented with the contribution of this low-order (1,0) DCT spherical harmonic. If DFT spherical harmonics were used instead, typically more DFT spherical harmonics are necessary to lend to the contribution in the reconstruction of the complicated surface area. As a result, we basically see the better energy compaction of the DCT where a surface can be represented with fewer coefficients.

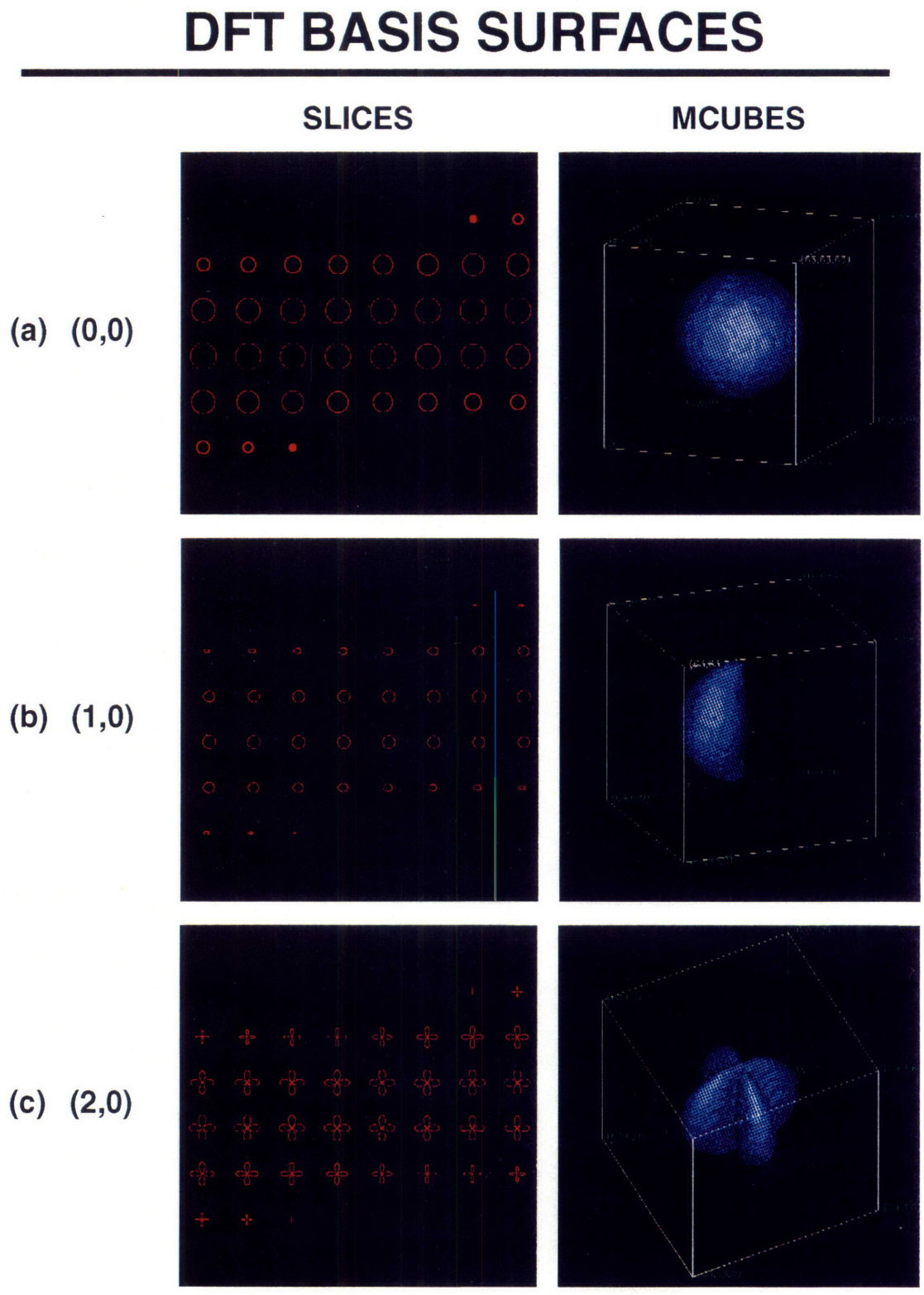


Figure B-1. DFT basis surfaces corresponding to different (k_ϕ, k_θ) transform coefficients. (a) (0,0). (b) (1,0). (c) (2,0).

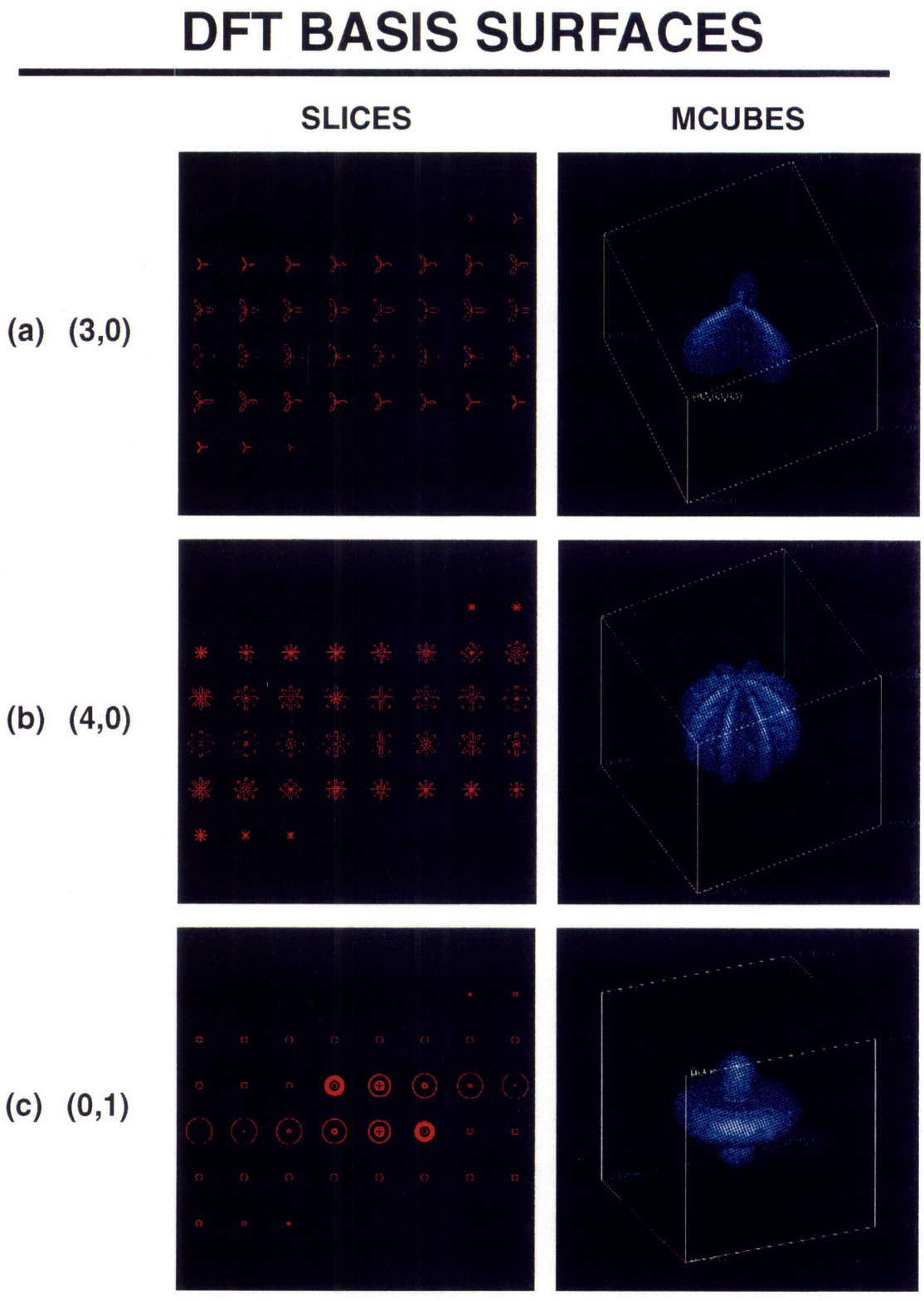


Figure B-2. DFT basis surfaces corresponding to different (k_ϕ, k_θ) transform coefficients. (a) (3,0). (b) (4,0). (c) (0,1).

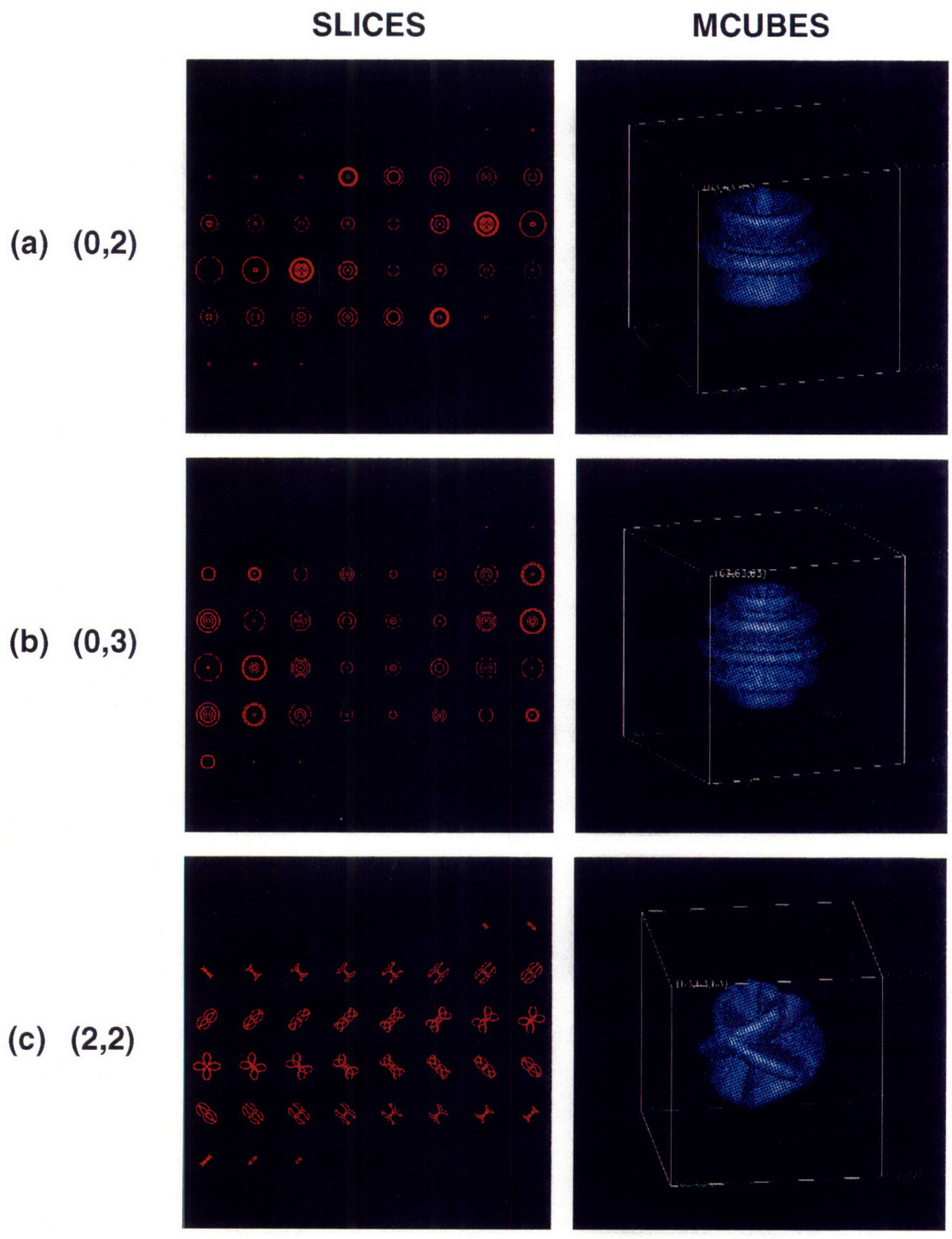


Figure B-3. DFT basis surfaces corresponding to different (k_ϕ, k_θ) transform coefficients. (a) (0,2). (b) (0,3). (c) (2,2).

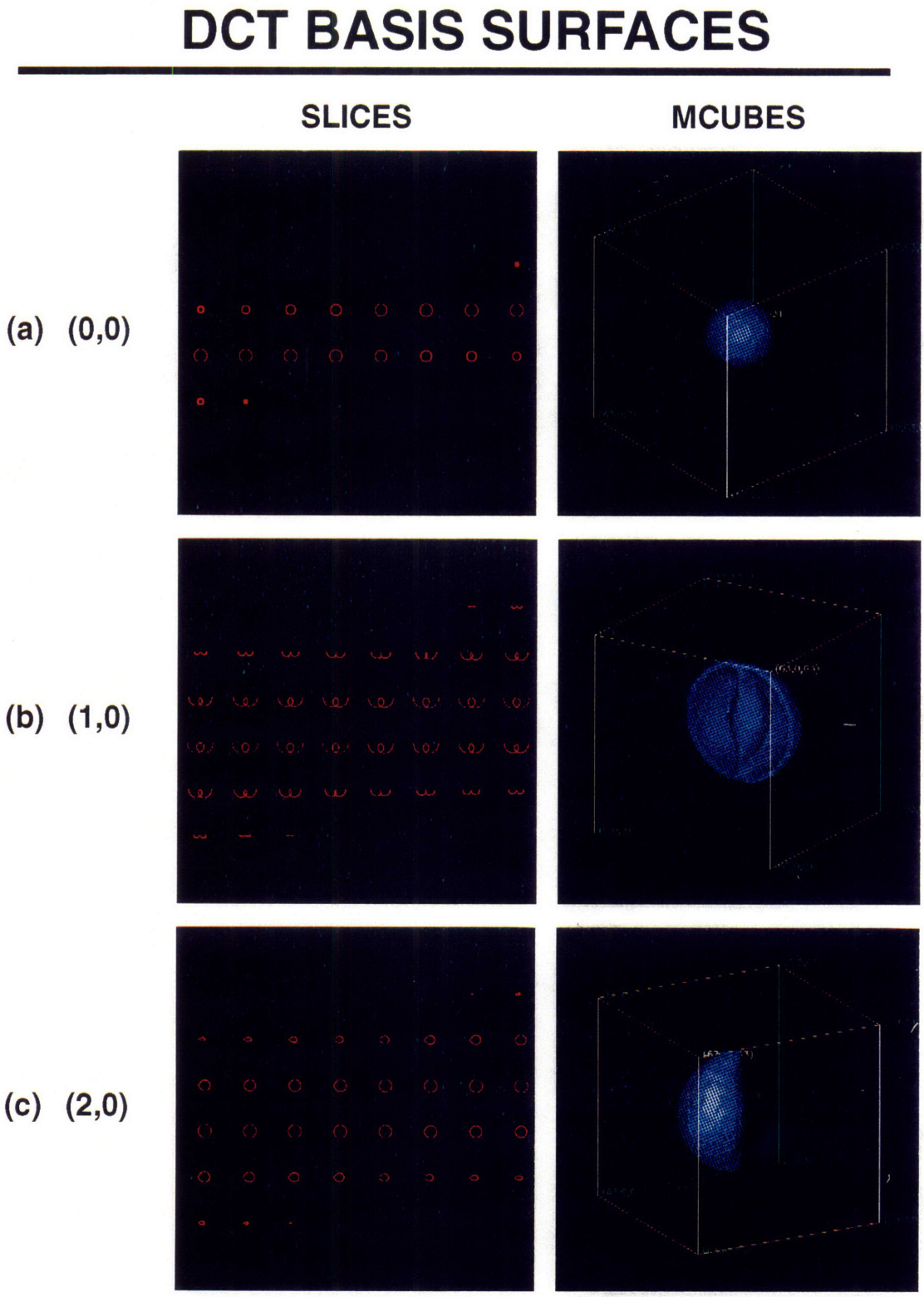


Figure B-4. DCT basis surfaces corresponding to different (k_ϕ,k_θ) transform coefficients. (a) (0,0). (b) (1,0). (c) (2,0).

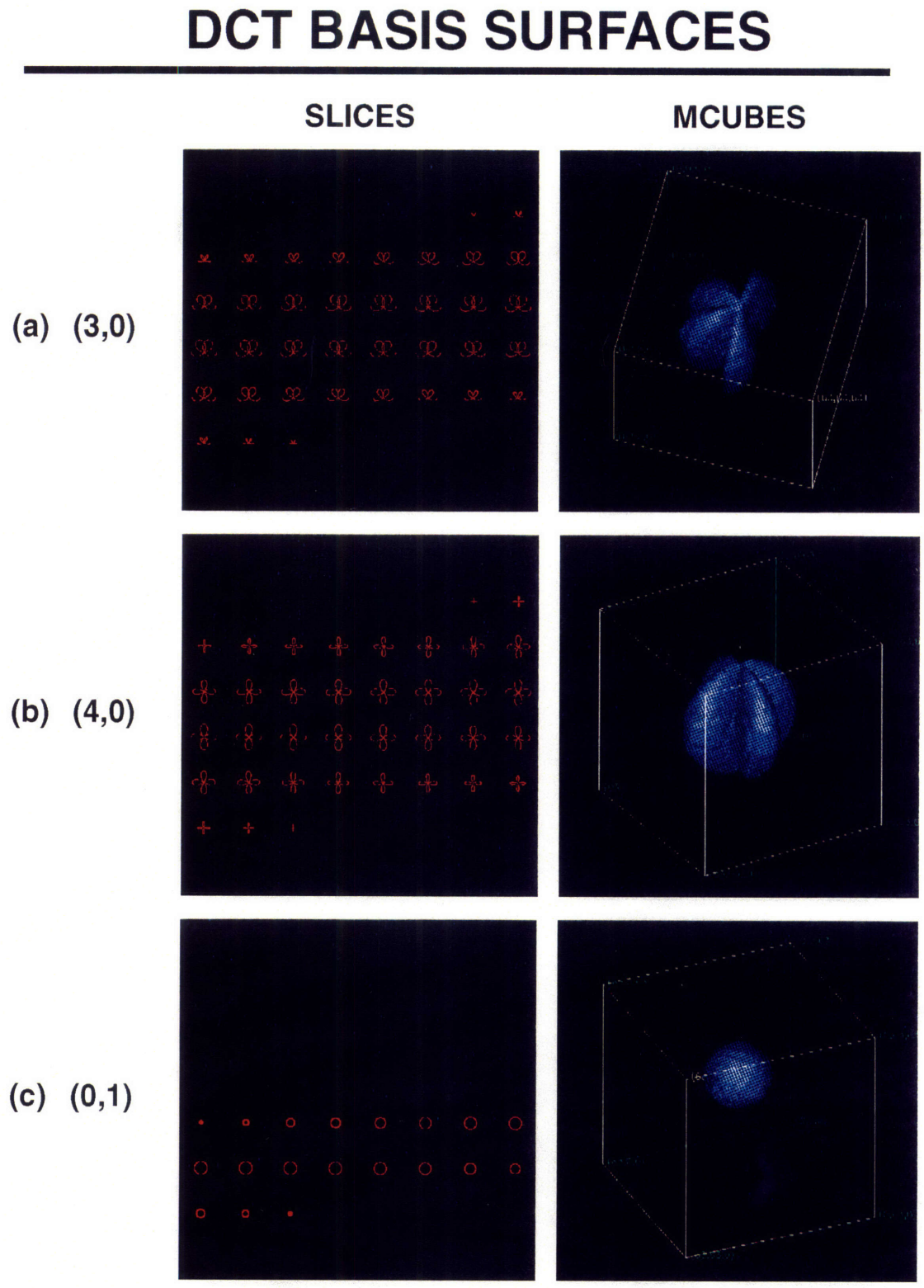


Figure B-5. DCT basis surfaces corresponding to different (k_ϕ,k_θ) transform coefficients. (a) (3,0). (b) (4,0). (c) (0,1).

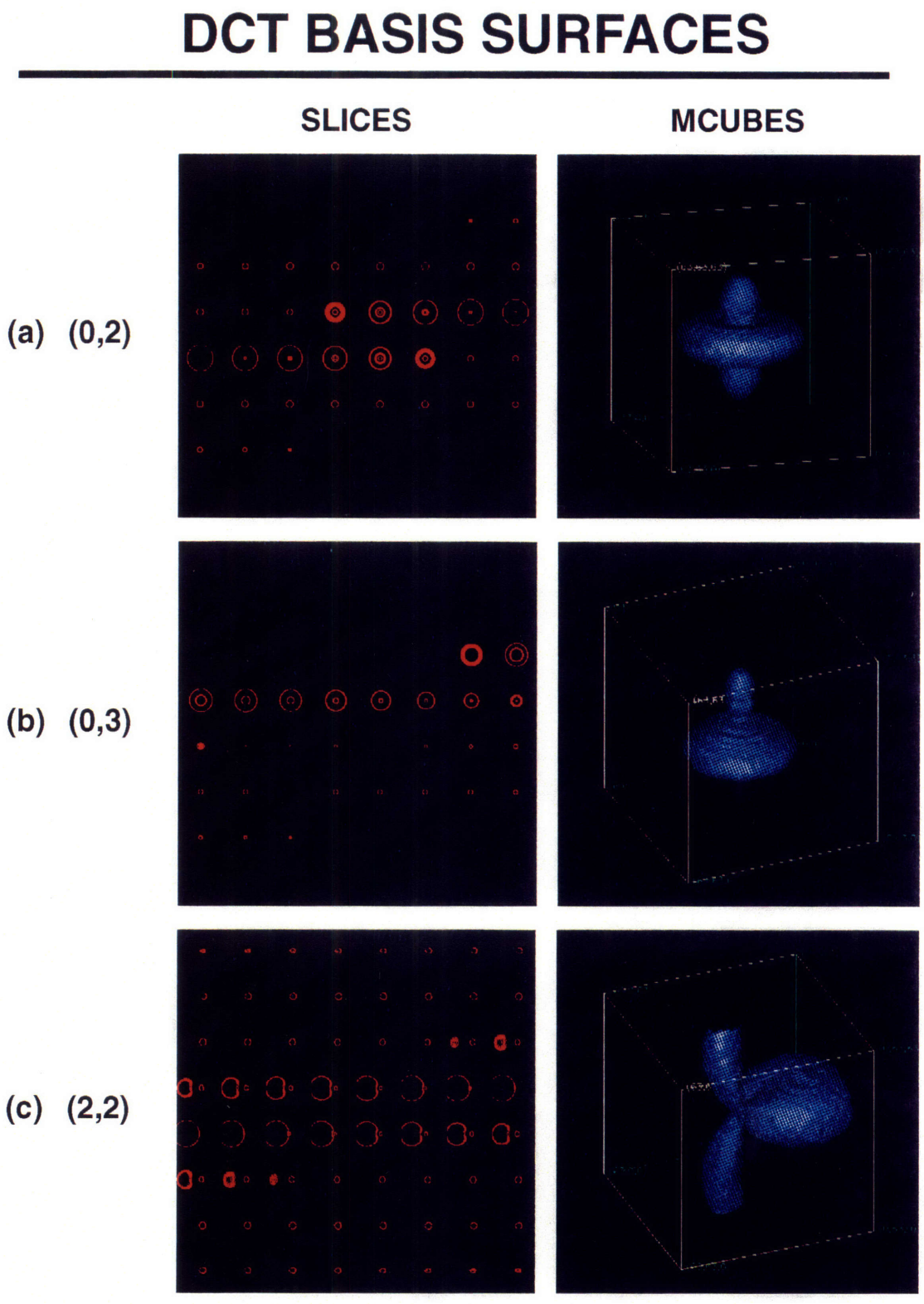


Figure B-6. DCT basis surfaces corresponding to different (k_ϕ, k_θ) transform coefficients. (a) (0,2). (b) (0,3). (c) (2,2).

Appendix C

Entropy and Coding

C.0 Entropy

The term entropy was first used by Clausius in 1864 and first introduced into information theory by Shannon in 1948. As in [22], we define the entropy H by

$$H = -\sum_{i=1}^{L} P_i \log_2 P_i \tag{C.1}$$

where L is is the number of different messages to be represented and P_i is the probability that the message is the *i-th* message. The entropy H can be interpreted as the average amount of information that a message contains. Thus, the units for entropy are bits per mssage. From information theory, the entropy H in equation (C.1) is the minimum possible average bit rate required in coding a message. Though this formula does not specify a means to design codewords for messages, it is still very useful. Suppose that the average bit rate corresponding to the use of a set of designed codewords equals the entropy. This tells us that the codewords are optimal and we neet not search any further.

C.1 Noiseless Coding

We will now discuss codes for noiseless encoding of data. Specifically, we wll discuss two forms of encoding: fixed-length codes and variable-length codes.

C.1.1 Fixed-Length Coding

In fixed-length or uniform-length codes, every message is represented by a code of the same length. For example, if we wanted to represent 8 messages that can occur equally likely, we can design a codebook where the first message is represented by 000, the next message by 001, and so forth until the last message which would be represented by 111. The length of each code is 3 bits and the average bit rate of 3 bits/message. From equation (C.1), we calculate the entropy to be 3 bits/message also. As aforementioned, if the average bit rate equals the entropy, then our codeword assignment is optimal and we need not search any further for a better codebook.

C.1.2 Variable-Length Coding

Often, some message possibilites are more likely to be sent than others. In such scenarios,

fixed-length encoding of messages might not be optimal. However, by assigning shorter codewords to the more probable message possibilities and longer codewords to the less probable message possibilities, it may be possible to reduce the average bit rate and approach the entropy.

One optimal codeword design method which is uniquely decodable and results in the lowest possible average bit rate is Huffman coding. In general, when there are statistical properties to the occurrences of the different messages to be encoded, variable-length codes will result in an average bit rate that is lower than that of uniform-length codes and that approaches the entropy.

C.2 Relation to Spherical-Mapping Method

In the Spherical-Mapping method, we eventually obtain transform coefficients and their locations. If we quantize the possible values that transform coefficients can take, we can then design a variable-length codebook to represent these finite message possibilities. The variable-length encoding of these transform coefficients will thus add an additional level of compression on top of the compression provided by the Spherical-Mapping method.

C.3 Relation to Ring-Peeling Method

In the Ring-Peeling method, we enventually obtain the encoded peel structure. Since the peel segment elements can take only finitie number of values and are plenty, we can design a variable-length codebook to code the peel segment elements.

In Chapter 5, we assumed uniform-length encoding by stating that it will take one byte to represent any peel segment element. This assumption resulted in an overall compression of approximately 33% if we assume that the representation of peel segment headers are negligible. Next, we stated that we do not need all 8 bits to represent a peel segment element since a peel segment element can take on only 27 different possible values. If we use uniform-length encoming again but with codes of length 5 (round up of 4.76) bits, we obtain an overall compression of approximately 21%. However, since the peeling structure is such that we are very likely to peel along a certain direction to obtain rings, it is very likely that some chain codes or equivalently messages will occur more often than others. The use of variable-length encoding will then be very likely to represent the peel structure even more efficiently and with overall compression better than 20%.

References

[1] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering", *Computer Graphics*, Vol 22(4), August 1988, pp 65-74.

[2] P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields", *Computer Graphics*, Vol 22(4), August 1988, pp 51-58.

[3] C. Upson and M. Keeler, "V-BUFFER: Visible Volume Rendering", *Computer Graphics*, Vol 22(4), August 1988, pp 59-64.

[4] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D surface Construction Algorithm", *Computer Graphics*, Volume 21(4), July 1987, pp 163-169.

[5] K. Leutwyler, "Optical Tomography", *Scientific American*, January 1994, pp 147-149.

[6] C.T. Zahn and R.S. Roskies, "Fourier Descriptors for Plane Closed Curves", *IEEE Trans. Computers*, Vol C-21, March 1972, pp 269-281.

[7] P.V.C. Hough, "Method and Means of Recognizing Complex Patterns", U.S. Patent 3,069,654, 1962.

[8] M.K. Hu, "Visual Pattern Recognition by Moments", *IRE Trans. Information Theory*, Vol IT-8, 1962, pp 179-187.

[9] D. Meagher, "Octree Encoding: A New Technique for the Representation, Manipulation, and Display of Arbitrary 3-D Objects by Computer", Technical Report IPL-TR-80-111, Image Processing Laboratory, Rensselaer Polytechnic Institute, October 1980.

[10] D. Meagher, "Geometric Modeling Using Octree Encoding", *Computer Graphics and Image Processing*, Volume 19, June 1982, pp 129-147.

[11] D. Meagher, "Efficient Synthetic Image Generation of Arbitrary 3-D Objects", *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, June 1982, pp 473-478.

[12] V. Burdin, F. Ghorbel, J.L. de Bougrenet de la Tocnaye, and C. Roux, "A Complete and Stable Set of Fourier Descriptors of 2D shapes for Invariant Analysis and Reconstruction of 3D Objects", in L. Torres, E. Masgrau, and M.A. Lagunas (Eds), Signal Processing V: Theories and Applications, Elsevier Science Publishers B.V., North-Holland, Amsterdam, 1990, pp 1655-1658.

[13] V. Burdin, F. Ghorbel, J.L. de Bougrenet de la Tocnaye, and C. Roux, "A Geometrical Analysis for a Data Compression of 3D Anatomical Structures", in P.J. Laurent, A. Le Mehaute, and L.L. Schumaker (Eds), Curves and Surfaces, Academic Press, Boston, 1991, pp 59-65.

[14] V. Burdin, F. Ghorbel, J.L. de Bougrenet de la Tocnaye, and C. Roux, "A Three-Dimensional Primitive Extraction of Long Bones Obtained From Bi-Dimensional Fourier Descriptors", *Pattern Recognition Letters*, Vol 13, March 1992, pp 213-217.

[15] V. Burdin and C. Roux, "Reconstruction Error of 3D Anatomic Structures Represented by 2D Fourier Descriptors using a Dissimilarity Measure", *Proceedings of the 14th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Paris, France, 1992, pp 1944-1945.

[16] G. Eichmann, B.Z. Dong, and M. Jankowski, "Optical Extraction of Transform Shape Descriptors", *Proceedings of the SPIE*, Volume 380, 1983, pp 432-438.

[17] G. Eichmann and M. Jankowski, "Fourier Shape Descriptors for Surfaces of Multi-Dimensional Closed Volumes", *Proceedings of the SPIE*, Volume 579, September 1985, pp 482-487.

[18] G. Eichmann, M. Jankowski, and M. Stojancic, "Shape Description with an Associative Memory", *Proceedings of the SPIE*, Volume 638, April 1986, pp 76-82.

[19] G. Eichmann, C. Lu, M. Jankowski, and R. Tolimieri, "Shape Representation by Gabor Expansion", *Proceedings of the SPIE*, Volume 1297, 1990, pp 86-94.

[20] M. Jankowski and G. Eichmann, "Three-Dimensional Fourier Shape Descriptors", *International Journal of Robotics and Automation*, Volume 8(2), 1993, pp 92-96.

[21] T.B. Sheridan, D. Zeltzer, N. Durlach, D. Miller, M. Picardi, and A.M. Waxman, "Telesystems Study Report", *MIT Lincoln Laboratory Technical Report*, Project Report ACC-8, February 1994.

[22] J.S. Lim, Two-Dimensional Signal and Image Processing, Prentice-Hall, Englewood Cliffs, 1990.

[23] Ruskai *et al.*, Wavelets and Their Applications, Jones and Bartlett Publishers, Boston, 1992.